University of Southern Queensland
Faculty of Engineering & Surveying

‘Online Learning Environment’ using ASP.NET

A dissertation submitted by

Timothy Ross Vriesema

in fulfilment of the requirements of

ENG4111/2 Research Project

towards the degree of

Bachelor of Engineering (Software Engineering)
Bachelor of Business (Management and Leadership)

Submitted: October, 2004
Abstract

This dissertation focuses on the management, design, implementation, testing, and deployment of a learning environment for educational institutions. The management activities of this project are based on the techniques outlined by Bruegge & Dutoit (2004) and Sommerville (2001), involving the scheduling of project tasks, management of the possible risks, and documentation of the project’s design. The design of the project involves the integration and alteration of documents generated from the management activities, while implementation consists of using the Microsoft .NET Framework class library via the programming languages, ASP.NET, C# .NET, ADO.NET, XML, and SQL. The activities revolving around testing focuses on its documentation via the management activities. As well, a small-scale deployment of this project is introduced into a Microsoft Windows Server 2003 operating system environment.
Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof G Baker
Dean
Faculty of Engineering and Surveying
Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

TIMOTHY ROSS VRIESEMA

Student Number: 0011122446

____________________________
Signature

____________________________
Date
Acknowledgments

I would like to take this opportunity to thank the many people that influenced and encouraged me throughout this project.

Firstly, I would like to thank my loving girlfriend, Cherie, and all the support she has given me to continue the rigorous work required to complete this project. Her undivided attention to my incessant ramblings on the technical development helped me overcome problematic issues.

Secondly, I would like to thank my family for their ongoing support in my University studies.

Thirdly, my supervisor, Dr. Hong Zhou. I would like to thank her for her constant advice, feedback, and experience in our weekly meetings to allow my ideas to flow and not kept inside my head.

Finally, in the research and implementation of this project, I was faced with many programming language issues where I could not find the answers. The Most Valuable Professional (MVP) developers within the Microsoft newsgroups promptly answered my many of my questions. Thank-you to Jon Skeet [C# MVP], David Portas [SQL Server MVP], and Oleg Tkachenko [XML MVP].

Timothy Ross Vriesema

University of Southern Queensland

October 2004
Contents

Abstract

Certification of Dissertation

Acknowledgments

List of Figures

List of Tables

List of Listings

Chapter 1 Introduction

1.1 Structure of the Dissertation

Chapter 2 Methodology and Techniques

2.1 Software Project Development

2.2 Software Project Management

2.2.1 Software Process Models
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.2 Software Project Documentation</td>
<td>8</td>
</tr>
<tr>
<td>2.2.3 Software Project Schedule Management</td>
<td>13</td>
</tr>
<tr>
<td>2.2.4 Software Project Risk Management</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Programming Languages</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1 Microsoft Active Server Pages .NET 1.1</td>
<td>17</td>
</tr>
<tr>
<td>2.3.2 Microsoft C# .NET 1.1</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3 Microsoft Active Data Objects .NET 1.1</td>
<td>21</td>
</tr>
<tr>
<td>2.3.4 Extensible Markup Language</td>
<td>21</td>
</tr>
<tr>
<td>2.3.5 Structured Query Language</td>
<td>22</td>
</tr>
<tr>
<td>2.3.6 Hyper-Text Markup Language</td>
<td>24</td>
</tr>
<tr>
<td>2.3.7 Cascade Style Sheeting</td>
<td>24</td>
</tr>
<tr>
<td>2.4 Development Environments and Tools</td>
<td>24</td>
</tr>
<tr>
<td>2.4.1 Microsoft Visual Studio .NET 2003</td>
<td>24</td>
</tr>
<tr>
<td>2.4.2 Microsoft ASP.NET Web Matrix</td>
<td>25</td>
</tr>
<tr>
<td>2.4.3 Microsoft Windows Server 2003</td>
<td>25</td>
</tr>
<tr>
<td>2.4.4 Microsoft Internet Information Service 6.0</td>
<td>25</td>
</tr>
<tr>
<td>2.4.5 Microsoft SQL Server 2000 and 2005</td>
<td>26</td>
</tr>
<tr>
<td>2.4.6 Microsoft Developer Network</td>
<td>26</td>
</tr>
<tr>
<td>2.5 Software Project Implementations</td>
<td>27</td>
</tr>
<tr>
<td>2.5.1 Web Applications</td>
<td>28</td>
</tr>
</tbody>
</table>
CONTENTS

Chapter 3  Software Design 31

3.1 System Architecture ................................................. 31

3.1.1 System-to-Data Relationship .................................. 33

3.1.2 Data Storage Techniques ....................................... 33

3.2 Database Architecture ............................................... 35

3.3 Design using the Microsoft .NET Framework .................... 46

Chapter 4  Software Implementation 50

4.1 Implementation Issues ................................................ 51

4.1.1 XML in Microsoft SQL Server 2000 ............................. 51

4.1.2 Registry and File System Permissions ........................... 51

4.1.3 Native XML Architecture ........................................ 52

4.2 Implementation using ASP.NET ...................................... 52

4.2.1 Authentication ..................................................... 52

4.2.2 Web Forms and User Controls ................................. 54

4.3 Implementation using XML .......................................... 54

4.3.1 XML Data Structures and Database Compatibility .......... 55

4.3.2 Serialization and Deserialization ............................... 55

4.4 The Incomplete Web Application .................................. 55

Chapter 5  Software Deployment 58
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Configuration of Microsoft Windows Server 2003</td>
<td>58</td>
</tr>
<tr>
<td>5.2</td>
<td>Installation - The Setup Application</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td><strong>Chapter 6 Software Project Management</strong></td>
<td>66</td>
</tr>
<tr>
<td>6.1</td>
<td>Documentation</td>
<td>66</td>
</tr>
<tr>
<td>6.2</td>
<td>Schedule Management</td>
<td>67</td>
</tr>
<tr>
<td>6.3</td>
<td>Risk Management</td>
<td>69</td>
</tr>
<tr>
<td>6.4</td>
<td>Resource Requirements</td>
<td>72</td>
</tr>
<tr>
<td>6.5</td>
<td>Implementation Management using Microsoft Visual Studio .NET</td>
<td>73</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Database Source Code Project</td>
<td>75</td>
</tr>
<tr>
<td>6.5.2</td>
<td>OLEWebSetup Project</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td><strong>Chapter 7 Conclusions</strong></td>
<td>79</td>
</tr>
<tr>
<td>7.1</td>
<td>Further Work</td>
<td>80</td>
</tr>
<tr>
<td>7.2</td>
<td>The Future Project</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td><strong>References</strong></td>
<td>83</td>
</tr>
<tr>
<td></td>
<td><strong>Appendix A Project Specification</strong></td>
<td>88</td>
</tr>
<tr>
<td></td>
<td><strong>Appendix B Requirements Analysis Document</strong></td>
<td>90</td>
</tr>
<tr>
<td></td>
<td><strong>Appendix C System Design Document</strong></td>
<td>122</td>
</tr>
<tr>
<td></td>
<td><strong>Appendix D Object Design Document</strong></td>
<td>135</td>
</tr>
<tr>
<td>Appendix E Test Manual</td>
<td>166</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>Appendix F Web Application Model</td>
<td>186</td>
<td></td>
</tr>
<tr>
<td>Appendix G Project Gannt Chart</td>
<td>188</td>
<td></td>
</tr>
<tr>
<td>Appendix H Source Code</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>H.1 OLEWebApplication Project</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>H.2 OLEBusinessLogic Project</td>
<td>293</td>
<td></td>
</tr>
<tr>
<td>H.3 Database Source Code Project</td>
<td>403</td>
<td></td>
</tr>
</tbody>
</table>
List of Figures

2.1 The Waterfall Process Model ........................................... 7
2.2 The Evolutionary Development Process Model ......................... 8
2.3 The outline of the Requirements Analysis Document (RAD) ........... 10
2.4 The outline of the System Design Document (SDD) .................... 11
2.5 The outline of the Object Design Document (ODD) ................. 12
2.6 The outline of the Test Manual .......................................... 13
2.7 The Microsoft .NET Framework’s components .......................... 16
2.8 The tier application architectural designs ............................ 29
3.1 The three-tier application architecture ................................ 32
3.2 The steps in the Database Design Methodology (Steps 1 to 2) .... 36
4.1 Login page ........................................................................... 56
4.2 Home page for user ‘Jonathan Trinder’ (userID: 26) .................. 56
4.3 ‘Assignment 1’ for ‘Research Project 1’ (courseID: ENG4111) within the Class page ............................................................. 57
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>The Application Server window from the Windows Component Wizard, under Windows Server 2003</td>
<td>59</td>
</tr>
<tr>
<td>5.2</td>
<td>Welcome screen (first step)</td>
<td>60</td>
</tr>
<tr>
<td>5.3</td>
<td>SQL Server name setup (second step)</td>
<td>61</td>
</tr>
<tr>
<td>5.4</td>
<td>IIS virtual folder and port setup (third step)</td>
<td>62</td>
</tr>
<tr>
<td>5.5</td>
<td>Confirm installation (forth step)</td>
<td>63</td>
</tr>
<tr>
<td>5.6</td>
<td>Setup installing (fifth step)</td>
<td>64</td>
</tr>
<tr>
<td>5.7</td>
<td>Setup finished (sixth step)</td>
<td>65</td>
</tr>
<tr>
<td>6.1</td>
<td>Microsoft Visual Studio .NET 2003’s Solution Explorer (for the OLE solution)</td>
<td>74</td>
</tr>
<tr>
<td>6.2</td>
<td>Microsoft Visual Studio .NET 2003’s Debug Configuration Manager (for the OLE solution)</td>
<td>75</td>
</tr>
<tr>
<td>6.3</td>
<td>Microsoft Visual Studio .NET 2003’s Release Configuration Manager (for the OLE solution)</td>
<td>75</td>
</tr>
<tr>
<td>6.4</td>
<td>Microsoft Visual Studio .NET 2003’s database table creation/alteration (for the OLE solution)</td>
<td>76</td>
</tr>
<tr>
<td>6.5</td>
<td>Microsoft Visual Studio .NET 2003’s database relationship creation/alteration (for the OLE solution)</td>
<td>77</td>
</tr>
<tr>
<td>6.6</td>
<td>The User Interface configuration</td>
<td>78</td>
</tr>
<tr>
<td>6.7</td>
<td>The File System configuration</td>
<td>78</td>
</tr>
<tr>
<td>6.8</td>
<td>The Registry configuration</td>
<td>78</td>
</tr>
<tr>
<td>7.1</td>
<td>New database concept</td>
<td>81</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Software processes within the *IEEE 1074: Software Developing Life Cycle Processes* ........................................ 6

3.1 The attributes within the entity types *Users, AETQuestions*, and *AETAnswers* .................................................. 38

3.2 The attributes within the entity types *Classes, Roles*, and *ThreadSubscriptions* .................................................. 38

3.3 The attributes within the entity types *ClassUsers* and *OtherUsers* ................................................................. 39

3.4 The attribute domains for the attributes in Table 3.1 ................................................................. 40

3.5 The attribute domains for the attributes in Table 3.2 ................................................................. 40

3.6 The attribute domains for the attributes in Table 3.3 ................................................................. 41

3.7 The highlighted and created primary and composite keys from Table 3.1 ................................. 42

3.8 The highlighted and created primary and composite keys from Table 3.2 ................................. 42

3.9 The highlighted and created primary and composite keys from Table 3.3 ................................. 42

6.1 The Documentation, Requirements Elicitation and Analysis, and Design work structure ......................... 68
6.2 The Training, Implementation, Testing, and Deployment work structure   69
6.3 Risk types and the risks that may be encountered (Risk Identification)  70
6.4 Risks, their probability and effects (Risk Analysis)                   71
6.5 Risks and their strategies (Risk Strategy)                         72
6.6 Resource requirements                                               73
## List of Listings

2.1 Interface example in C# .............................................. 19

2.2 C/C++ example of how and where memory is allocated and deallocated 20

2.3 C# example of where memory is allocated and deallocated ............ 20

2.4 FOR XML EXPLICIT stored procedure ................................. 23

2.5 Output from the stored procedure in Listing 2.4 ...................... 23

H.1 Class.aspx ............................................................. 195

H.2 Class.aspx.cs .......................................................... 196

H.3 Configuration.aspx .................................................... 200

H.4 Configuration.aspx.cs ................................................ 201

H.5 Default.aspx ........................................................... 202

H.6 Default.aspx.cs .......................................................... 204

H.7 Default.css .............................................................. 204

H.8 Footer.ascx .............................................................. 214

H.9 Footer.ascx.cs .............................................................. 214
<table>
<thead>
<tr>
<th>Listing</th>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.10</td>
<td>Global.asax</td>
<td>214</td>
</tr>
<tr>
<td>H.11</td>
<td>Global.asax.cs</td>
<td>214</td>
</tr>
<tr>
<td>H.12</td>
<td>Header.ascx</td>
<td>218</td>
</tr>
<tr>
<td>H.13</td>
<td>Header.ascx.cs</td>
<td>219</td>
</tr>
<tr>
<td>H.14</td>
<td>Login.aspx</td>
<td>221</td>
</tr>
<tr>
<td>H.15</td>
<td>Login.aspx.cs</td>
<td>222</td>
</tr>
<tr>
<td>H.16</td>
<td>Management.aspx</td>
<td>222</td>
</tr>
<tr>
<td>H.17</td>
<td>Management.aspx.cs</td>
<td>223</td>
</tr>
<tr>
<td>H.18</td>
<td>Menu.ascx</td>
<td>224</td>
</tr>
<tr>
<td>H.19</td>
<td>Menu.ascx.cs</td>
<td>225</td>
</tr>
<tr>
<td>H.20</td>
<td>Submission.aspx</td>
<td>226</td>
</tr>
<tr>
<td>H.21</td>
<td>Submission.aspx.cs</td>
<td>227</td>
</tr>
<tr>
<td>H.22</td>
<td>Web.config</td>
<td>228</td>
</tr>
<tr>
<td>H.23</td>
<td>ClassControls/AETSubmissionList.ascx</td>
<td>228</td>
</tr>
<tr>
<td>H.24</td>
<td>ClassControls/AETSubmissionList.ascx.cs</td>
<td>228</td>
</tr>
<tr>
<td>H.25</td>
<td>ClassControls/ClassAnnouncements.ascx</td>
<td>229</td>
</tr>
<tr>
<td>H.26</td>
<td>ClassControls/ClassAnnouncements.ascx.cs</td>
<td>232</td>
</tr>
<tr>
<td>H.27</td>
<td>ClassControls/InformationAET.ascx</td>
<td>233</td>
</tr>
<tr>
<td>H.28</td>
<td>ClassControls/InformationAET.ascx.cs</td>
<td>234</td>
</tr>
<tr>
<td>H.29</td>
<td>ClassControls/MultipleChoiceAET.ascx</td>
<td>236</td>
</tr>
</tbody>
</table>
LIST OF LISTINGS

H.30 ClassControls/MultipleChoiceAET.ascx.cs .................................. 237
H.31 ClassControls/ShortResponseAET.ascx .................................... 242
H.32 ClassControls/ShortResponseAET.ascx.cs ................................ 243
H.33 ClassControls/ShortResponseAETAnswers.ascx ....................... 247
H.34 ClassControls/ShortResponseAETAnswers.ascx.cs ................... 247
H.35 ClassControls/ThreadPosts.ascx .......................................... 248
H.36 ClassControls/ThreadPosts.ascx.cs ....................................... 250
H.37 ClassControls/Threads.ascx .............................................. 252
H.38 ClassControls/Threads.ascx.cs ........................................... 253
H.39 ConfigurationControls/ApplicationConfiguration.ascx ............ 254
H.40 ConfigurationControls/ApplicationConfiguration.ascx.cs .......... 254
H.41 ConfigurationControls/ClassConfiguration.ascx .................... 255
H.42 ConfigurationControls/ClassConfiguration.ascx.cs .................. 255
H.43 ConfigurationControls/ClassEnrollmentConfiguration.ascx ...... 256
H.44 ConfigurationControls/ClassEnrollmentConfiguration.ascx.cs ..... 256
H.45 ConfigurationControls/InstitutionConfiguration.ascx .............. 256
H.46 ConfigurationControls/InstitutionConfiguration.ascx.cs ........... 257
H.47 ConfigurationControls/RoleAssignmentConfiguration.ascx ....... 257
H.48 ConfigurationControls/RoleAssignmentConfiguration.ascx.cs ....... 257
H.49 ConfigurationControls/UserConfiguration.ascx ..................... 258
<table>
<thead>
<tr>
<th></th>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.50</td>
<td>ConfigurationControls/UserConfiguration.ascx.cs</td>
<td>258</td>
</tr>
<tr>
<td>H.51</td>
<td>HomeControls/ErrorMessage.ascx</td>
<td>259</td>
</tr>
<tr>
<td>H.52</td>
<td>HomeControls/ErrorMessage.ascx.cs</td>
<td>261</td>
</tr>
<tr>
<td>H.53</td>
<td>HomeControls/ForumThreadSubscriptions.ascx</td>
<td>262</td>
</tr>
<tr>
<td>H.54</td>
<td>HomeControls/ForumThreadSubscriptions.ascx.cs</td>
<td>262</td>
</tr>
<tr>
<td>H.55</td>
<td>HomeControls/InstitutionAnnouncements.ascx</td>
<td>263</td>
</tr>
<tr>
<td>H.56</td>
<td>HomeControls/InstitutionAnnouncements.ascx.cs</td>
<td>266</td>
</tr>
<tr>
<td>H.57</td>
<td>HomeControls/QuickLinks.ascx</td>
<td>267</td>
</tr>
<tr>
<td>H.58</td>
<td>HomeControls/QuickLinks.ascx.cs</td>
<td>268</td>
</tr>
<tr>
<td>H.59</td>
<td>HomeControls/UpcomingDates.ascx</td>
<td>269</td>
</tr>
<tr>
<td>H.60</td>
<td>HomeControls/UpcomingDates.ascx.cs</td>
<td>271</td>
</tr>
<tr>
<td>H.61</td>
<td>LoginControls/Authentication.ascx</td>
<td>273</td>
</tr>
<tr>
<td>H.62</td>
<td>LoginControls/Authentication.ascx.cs</td>
<td>274</td>
</tr>
<tr>
<td>H.63</td>
<td>ManagementControls/ClassAnnouncementManagement.ascx</td>
<td>275</td>
</tr>
<tr>
<td>H.64</td>
<td>ManagementControls/ClassAnnouncementManagement.ascx.cs</td>
<td>276</td>
</tr>
<tr>
<td>H.65</td>
<td>ManagementControls/ClassManagement.ascx</td>
<td>276</td>
</tr>
<tr>
<td>H.66</td>
<td>ManagementControls/ClassManagement.ascx.cs</td>
<td>277</td>
</tr>
<tr>
<td>H.67</td>
<td>ManagementControls/ClassNoteManagement.ascx</td>
<td>277</td>
</tr>
<tr>
<td>H.68</td>
<td>ManagementControls/ClassNoteManagement.ascx.cs</td>
<td>277</td>
</tr>
<tr>
<td>H.69</td>
<td>ManagementControls/InformationAETManagement.ascx</td>
<td>278</td>
</tr>
<tr>
<td>H.70</td>
<td>ManagementControls/InformationAETManagement.ascx.cs</td>
<td>278</td>
</tr>
<tr>
<td>H.71</td>
<td>ManagementControls/InstitutionAnnouncementManagement.ascx</td>
<td>279</td>
</tr>
<tr>
<td>H.72</td>
<td>ManagementControls/InstitutionAnnouncementManagement.ascx.cs</td>
<td>279</td>
</tr>
<tr>
<td>H.73</td>
<td>ManagementControls/MultipleChoiceAETManagement.ascx</td>
<td>280</td>
</tr>
<tr>
<td>H.74</td>
<td>ManagementControls/MultipleChoiceAETManagement.ascx.cs</td>
<td>280</td>
</tr>
<tr>
<td>H.75</td>
<td>ManagementControls/ShortResponseAETManagement.ascx</td>
<td>281</td>
</tr>
<tr>
<td>H.76</td>
<td>ManagementControls/ShortResponseAETManagement.ascx.cs</td>
<td>281</td>
</tr>
<tr>
<td>H.77</td>
<td>ManagementControls/UserManagement.ascx</td>
<td>281</td>
</tr>
<tr>
<td>H.78</td>
<td>ManagementControls/UserManagement.ascx.cs</td>
<td>282</td>
</tr>
<tr>
<td>H.79</td>
<td>MenuControls/Assignments.ascx</td>
<td>282</td>
</tr>
<tr>
<td>H.80</td>
<td>MenuControls/Assignments.ascx.cs</td>
<td>283</td>
</tr>
<tr>
<td>H.81</td>
<td>MenuControls/Classes.ascx</td>
<td>284</td>
</tr>
<tr>
<td>H.82</td>
<td>MenuControls/Classes.ascx.cs</td>
<td>285</td>
</tr>
<tr>
<td>H.83</td>
<td>MenuControls/Exams.ascx</td>
<td>286</td>
</tr>
<tr>
<td>H.84</td>
<td>MenuControls/Exams.ascx.cs</td>
<td>286</td>
</tr>
<tr>
<td>H.85</td>
<td>MenuControls/Notes.ascx</td>
<td>288</td>
</tr>
<tr>
<td>H.86</td>
<td>MenuControls/Notes.ascx.cs</td>
<td>288</td>
</tr>
<tr>
<td>H.87</td>
<td>MenuControls/Tutorials.ascx</td>
<td>289</td>
</tr>
<tr>
<td>H.88</td>
<td>MenuControls/Tutorials.ascx.cs</td>
<td>290</td>
</tr>
<tr>
<td>H.89</td>
<td>SubmissionControls/Class.ascx</td>
<td>291</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>H.90</td>
<td>SubmissionControls/Class.ascx.cs</td>
<td>291</td>
</tr>
<tr>
<td>H.91</td>
<td>SubmissionControls/User.ascx</td>
<td>292</td>
</tr>
<tr>
<td>H.92</td>
<td>SubmissionControls/User.ascx.cs</td>
<td>292</td>
</tr>
<tr>
<td>H.93</td>
<td>Address.cs</td>
<td>293</td>
</tr>
<tr>
<td>H.94</td>
<td>AETAnswer.cs</td>
<td>294</td>
</tr>
<tr>
<td>H.95</td>
<td>AETQuestion.cs</td>
<td>309</td>
</tr>
<tr>
<td>H.96</td>
<td>Authentication.cs</td>
<td>330</td>
</tr>
<tr>
<td>H.97</td>
<td>Class.cs</td>
<td>332</td>
</tr>
<tr>
<td>H.98</td>
<td>Configuration.cs</td>
<td>338</td>
</tr>
<tr>
<td>H.99</td>
<td>DataBridge.cs</td>
<td>342</td>
</tr>
<tr>
<td>H.100</td>
<td>ErrorAndSecurity.cs</td>
<td>358</td>
</tr>
<tr>
<td>H.101</td>
<td>Limitations.cs</td>
<td>366</td>
</tr>
<tr>
<td>H.102</td>
<td>MD5Encryption.cs</td>
<td>367</td>
</tr>
<tr>
<td>H.103</td>
<td>Post.cs</td>
<td>368</td>
</tr>
<tr>
<td>H.104</td>
<td>Role.cs</td>
<td>382</td>
</tr>
<tr>
<td>H.105</td>
<td>Thread.cs</td>
<td>387</td>
</tr>
<tr>
<td>H.106</td>
<td>User.cs</td>
<td>392</td>
</tr>
<tr>
<td>H.107</td>
<td>oledb.sql</td>
<td>403</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The idea of the ‘Online Learning Environment’ using ASP.NET (OLE) was spawned from an interest in software engineering. Software engineering is quite a broad area, so the idea was to involve as many software engineering tasks as possible. These include software project management and enterprise-level application architectural design, implementation, testing, and deployment.

The initial ideas and research required a small amount of experience in application design and implementation. Throughout the writer’s own technical experience and University studies, experience within Microsoft Active Server Pages (ASP), C and C++, and databases (SQL) in design, implementation, testing, and deployment had already been achieved. For the purposes of this project however, the writer sought to expand his current level of experience in these fields, to new, modern, and exciting fields.

The main inspiration for this project came from using an already established enterprise-level Web application from the company WebCT, Inc. and their educational environment application, Vista.
1.1 Structure of the Dissertation

The dissertation begins with Chapter 2, introducing methodologies and techniques researched in order to gain an idea as to how development of the project would take place. Chapter 6 then outlines the activities involved in software project management. The design of the OLE project is then detailed in Chapter 3, while Chapter 4 displays how the OLE project was implemented. Chapter 5 details the deployment of the OLE project. The final chapter, Chapter 7, provides project conclusions and ideas for future projects to follow on from the OLE.

The appendices of this dissertation include (in order):

A. *Project Specification* - A contract between the University of Southern Queensland and the writer to ensure a pass grade (minimum) within the courses ENG4111 and ENG4112, Research Project 1 and 2.

B. *Requirements Analysis Document* - A document detailing the requirements of the OLE project.

C. *System Design Document* - A document detailing the system design of the OLE project.

D. *Object Design Document* - A document detailing the object design of the OLE project.

E. *Test Manual* - A document detailing the test procedures for the OLE project.

F. *Web Application Model* - A Unified Modeling Language model of the OLE Web application.

G. *Project Gantt Chart* - A chronological gantt chart that lists tasks and their completion times.

H. *Source Code* - The source code generated from the OLE project’s implementation.
Chapter 2

Methodology and Techniques

Throughout the OLE project, different methodologies and techniques into software engineering were required to be researched. The areas researched within this project consisted of software project management, programming languages, development environments and tools, and software project implementation methods and techniques.

**Application Programming Interface (API)** ‘The set of programming language constructs or statements that can be coded in an application program to invoke the specific functions and services provided by an underlying operating system or service program.’ (Microsoft Developer Network 2003, his/snagloss_1b4x.htm)

**Common Language Runtime (CLR)** ‘The engine at the core of managed code execution. The runtime supplies managed code with services such as cross-language integration, code access security, object lifetime management, and debugging and profiling support.’ (Microsoft Developer Network 2003, p. Netstart/html/cpgloc.htm)

**Database Management System (DBMS)** ‘A software system that enables users to define, create, maintain, and control access to the database.’ (Connolly & Begg 2002, p. 16)

**Dynamic Link Library (DLL)** A file containing a library of functions that can be dynamically linked to many applications. When called upon, it allocates itself to
2.1 Software Project Development

Software Project Development consists of processes that make up the development of a software application. Some of these processes are:

- Requirements Elicitation
- Requirements Analysis
- System Design
- Object Design
- Implementation
- Testing

Firstly, the Requirements Elicitation process focuses on defining the purpose of the software system, generating functional and non-functional requirements from an initial problem statement. A functional requirement is an ‘area of functionality the system must support’ (Bruegge & Dutoit 2000, p. 718), whereas a non-functional requirement is a ‘user-visible constraint on the system’ (Bruegge & Dutoit 2000, p. 724).

Secondly, the Requirements Analysis process uses the generated products from the Requirements Elicitation process that generates an analysis model that is composed of three other model:s:

- **Functional Model** - Use cases and scenarios.
- **Analysis Object Model** - Class and object diagrams.
2.2 Software Project Management

- Dynamic Model - Statechart and sequence diagrams.

Thirdly, the System Design process uses the generated products from the Requirements Analysis and Requirements Elicitation process to generate design goals and decompose the subsystems. Fourthly, the Object Design process uses the subsystem decompositions to generate a class diagram. Finally, the Implementation process uses the generated class diagram to produce working source code, which can then be tested via the Testing process.

2.2 Software Project Management

Software Project Management consists of activities and techniques to monitor and maintain a certain level of performance in the development of a software project. Some of the activities in software project management are:

- Scheduling the workload over a period of time;
- Dividing the workload over a range of human resources;
- Defining and manage the risks throughout the project; and
- Documenting the progress, technical specifications, and functional/non-functional requirements.

The life cycles for software projects vary from organisation-to-organisation, however, the activities involved are consistent throughout. The Institute of Electrical and Electronics Engineers (IEEE) have constructed a standard for the development life cycle within software projects known as the IEEE 1074: Standard for Developing Life Cycle Processes.

Bruegge & Dutoit (2000, pp. 460-468) describes the IEEE 1074: Standard for Developing Life Cycle Processes, from the IEEE (1997). Table 2.1 gives the process groups and processes within each group under the IEEE 1074 specification.
The first process group from Table 2.1, *Project Management*, is designed to monitor and control the project throughout the software life cycle. The next process, *Pre-development* researches the concept through different ways in order to understand and appreciate the project in more detail. The *Development* process follows by defining the software’s requirements, then designing and implementing the software. *Post-development* includes the installation of the software, while offering support and maintenance. The last processes under *Integral Processes* are performed throughout the duration of the project, which include software configuration management, documentation development, and training.
2.2 Software Project Management

2.2.1 Software Process Models

The implementation of a software application has a number of models to approach the process of software project management. Several software process models were detailed by Sommerville (2001, pp. 44-50). Specifically, two of these models are:

- *The waterfall model* - The fundamental activities within the development of a software project are represented as phases such as requirements specification, software design, implementation, and testing.

- *Evolutionary development* - Interleaves the fundamental activities of a software project. Abstract specifications then create an initial system to be refined with customer input to satisfy the requirements of the customer.

The main difference between these two models processes is the order in which each activity occurs. To clarify, using the *waterfall* process model (Figure 2.1) in an ideal situation only allows back-tracking to the previous activities when the last activity has been reached. Whereas, the *evolutionary development* process model (Figure 2.2) allows back-tracking between all activities.

![The Waterfall Process Model](image)

Figure 2.1: The *Waterfall* Process Model
2.2 Software Project Management

2.2.2 Software Project Documentation

Documentation of a software application is an important component within every software project. It relates to software project management by allowing project managers to record alterations throughout the project. Within an organisation, executives monitor and keep progress reports and information relating to all projects. Another reason for keeping documentation is so that if project staff leave the organisation, new project staff can research the documentation and understand the project.

The four documents that will be described in detail are:

- **Requirements Analysis Document (RAD)** - Describes system’s functional and non-functional requirements and serves as a contract between the developers and the client. The audience for the RAD includes the client, the users, project management, the system analysts, and the system designers (Bruegge & Dutoit 2004, pp. 151-153).

- **System Design Document (SDD)** - Describes the design by subsystem decomposition (with the use of UML class diagrams), hardware and software mapping (with the use of UML deployment diagrams), access control, control flow mechanisms, data management, and boundary conditions. The audience for the SDD includes system architects, project management, and the developers who implement each subsystem (Bruegge & Dutoit 2004, pp. 282-284).

- **Object Design Document (ODD)** - Describes object design and the guidelines
2.2 Software Project Management

followed for subsystem interfaces, decomposition of subsystems into packages and classes, and class interfaces. The audience for the ODD includes system architects, developers who implement the subsystem, and testers (Bruegge & Dutoit 2004, pp. 373-379).

- Test Manual - Describes the scope, approach, and records the differences between the expected output and the results of the test. The audience for the test manual includes developers who implement the system, testers, and system architects (Bruegge & Dutoit 2004).

- User Manual describes the instructions in using the software application. This is so that users can refer to it for information and instructions on the software application. The audience for the User Manual includes users that interact with the system, developers, and testers (Bruegge & Dutoit 2004).

From Section 2.2.1, it was mentioned that this project would follow the evolutionary development process model. This means that these four documents are not completed sequentially. That is, the SDD does not have to commence before the RAD, or the ODD before the SDD. To clarify, these documents will interleave between each other until completed.

Commonly, however, the document that commences first is the RAD, since it defines the functions and features that will exist within the software project.

Requirements Analysis Document

Figure 2.3 outlines the template used for the RAD. It has been modified from its original outline, specified by Bruegge & Dutoit (2004, p. 152), so that its section headings fit the OLE project.
Section 1 of the RAD involves an introduction into the system, providing information about its purpose and scope. Section 2 provides an analysis of a current system’s functions and features. Section 3 is the largest section of this document. It lists the functional and non-functional requirements that will exist for the proposed system. The non-functional requirements are made up of the system’s usability, reliability, performance, supportability, implementation, interface, and operation requirements. It
2.2 Software Project Management
details system models using UML diagrams, which describe the system with respect to
the functional and non-functional requirements.

**System Design Document**

Figure 2.4 outlines the template used for the SDD. It has been modified from its original
outline, specified by Bruegge & Dutoit (2004, p. 283), so that its section headings fit
the OLE project.

<table>
<thead>
<tr>
<th>System Design Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
</tr>
<tr>
<td>1.1 Purpose of the System</td>
</tr>
<tr>
<td>1.2 Design Goals</td>
</tr>
<tr>
<td>1.3 Definitions, Acronyms, and Abbreviations</td>
</tr>
<tr>
<td>1.5 Overview</td>
</tr>
<tr>
<td>2. Current Software Architecture</td>
</tr>
<tr>
<td>3. Proposed Software Architecture</td>
</tr>
<tr>
<td>3.1 Overview</td>
</tr>
<tr>
<td>3.2 Subsystem decomposition</td>
</tr>
<tr>
<td>3.3 Hardware/Software Mapping</td>
</tr>
<tr>
<td>3.4 Persistent Data Management</td>
</tr>
<tr>
<td>3.5 Access Control and Security</td>
</tr>
<tr>
<td>3.6 Global Software Control</td>
</tr>
<tr>
<td>3.7 Boundary Conditions</td>
</tr>
<tr>
<td>4. Subsystem Services</td>
</tr>
</tbody>
</table>

Figure 2.4: The outline of the System Design Document (SDD)

Section 1 of the SDD is similar to the RAD’s introduction. It provides an introduction
to the system’s design goals, in addition to describing purpose of the system. Section 2 briefly details the current system’s architecture. Section 3 details the proposed sys-
tem’s overall architecture. This consists of decomposing and detailing the subsystems,
mapping hardware and software, defining how persistent data is managed, security and access control features, how the software is controlled, and the conditions of start-up
and errors. Section 4 then defines the subsystems and the services they provide.
Once the first revision of this document started to take shape, the implementation of this project was commence.

**Object Design Document**

Figure 2.5 outlines the template used for the ODD. It has been modified from its original outline, specified by Bruegge & Dutoit (2004, p. 376), so that its section headings fit the OLE project.

<table>
<thead>
<tr>
<th>Object Design Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
</tr>
<tr>
<td>1.1 Object Design Trade-Offs</td>
</tr>
<tr>
<td>1.2 Interface Documentation Guidelines</td>
</tr>
<tr>
<td>1.3 Definitions, Acronyms, and Abbreviations</td>
</tr>
<tr>
<td>1.4 Overview</td>
</tr>
<tr>
<td>2. Packages</td>
</tr>
<tr>
<td>3. Class Interfaces</td>
</tr>
</tbody>
</table>

Figure 2.5: The outline of the Object Design Document (ODD)

Section 1 of the ODD introduces the reader to the object design trade-offs and documentation guidelines for the interface. Section 2 provides a detailed description of packages within the system. Section 3 details the class interfaces, which are used throughout the system.

This document flows through a number of revisions in an attempt to increase efficiency and decrease complexity of the class interfaces.

**Test Manual**

Figure 2.6 outlines the sections used for the Test Manual. It is a combination of the two documents, Test Plan and Test Case Specification, specified by Bruegge & Dutoit (2004, pp. 476-477).
Section 1 provides a brief introduction of how the testing will be carried out. Section 2 is split into unit and integrated testing. Unit testing tests the classes individually and integrated testing tests the user interfaces, which integrates the classes from the project.

### 2.2.3 Software Project Schedule Management

A software project’s schedule is vital to its development. Deadlines can be created so that developers stay on time. This project activity can only be set by individuals with experience in the respective software field. For this project, seven components were fashioned:

- **Documentation** - Lists the documents that are required to be completed and in which order.

- **Requirements Elicitation and Analysis** - Lists the process required to produce the requirements and analyse them.

- **Design** - Lists the process required to produce the design of the system.

- **Training** - Lists the languages required to be learnt before implementation.

- **Implementation** - Lists the steps involved in implementing the system.

- **Testing** - Lists the process required to test two other components. They are:
  - **Unit Testing** - The back-end components of the system.
Integration Testing - All of the components with the system, working together.

• Deployment - Lists the process required to deploy the system.

2.2.4 Software Project Risk Management

All projects, software or other, have unexpected issues that may arise throughout its development. Risk Management is a common safeguard that ensures that these issues are minimised or averted. Sommerville (2001, pp. 84-91) describes some of the risks involved when developing a software project. He divides risk management into three sections:

• Risk Identification - Identifies and lists the risks that may be encountered within the design and implementation.

• Risk Analysis - Lists and analyses the risks that may affect or jeopardise the project; along with the probability of affect upon, and the effect it may impose on the project.

• Risk Strategy - Lists the risks from the risk analysis section and strategies that may be carried out in order to avoid such a danger.

2.3 Programming Languages

This section introduces the programming models and languages that were researched in order to implement a modern software application. Firstly, Microsoft .NET and Microsoft .NET Framework will be detailed, while then describing ASP.NET, C# .NET, ADO.NET, XML, SQL, HTML, and CSS. Their acronyms will also be defined in each respective sub-section.

The primary languages will be described in greater depth, introducing their main features and components.
2.3 Programming Languages

Microsoft .NET

The Microsoft .NET platform is used for building, running, and experiencing the next generation of distributed applications. It extends over clients, servers, and services. Microsoft Corporation (2001a) outlines Microsoft .NET and that it consists of:

- **A unified programming model** - Enables developers to build applications and XML Web services;

- **Client software** - This helps developers deliver a deep and compelling user experience across a family of devices;

- **Servers** - Integrates, executes, operates, and manages applications and XML Web services;

- **XML Web services** - Helps developers create a simple and integrated user experience; and

- **Tools** - Are used to develop Windows and Web applications, and XML Web services.

Microsoft .NET Framework 1.1

The Microsoft .NET Framework is the programming model of the Microsoft .NET platform. It allows for building, deploying, and running applications and XML Web services. It also manages much of the inner workings of .NET, enabling developers to concentrate on writing business logic code for their software applications (Microsoft Corporation 2001a). The Microsoft .NET Framework consists of two main components:

- **Common Language Runtime (CLR)**

- **Class Libraries**

Figure 2.7 depicts the Microsoft .NET Framework and how its components exist with each other (Microsoft Corporation 2001a).
The Common Language Runtime (CLR) is the core engine responsible for services such as language integration, and memory, process, and thread management. It also has the role of managing development time features such as: strong type naming, cross language exception handling, and dynamic binding. This reduces the amount of code to be written by developers in order to turn business logic into reusable components (Microsoft Corporation 2001b).

The Class Libraries provide what the name suggests, a library of classes for the use of developers. Microsoft Corporation (2001b) lists some of these as:

- **Base classes** - Provide standard functionality for developers such as, string manipulation, input/output, network communications, thread management, text management, and many others;
- **Data classes** - Support data management and include SQL classes for manipulating data stores through a standard SQL interface;
- **XML classes** - Enable XML data manipulation, searching, and translations;
- **Web Forms classes** - Include the classes that enable developers to rapidly create Web graphical user interface (GUI) applications; and
- **Windows Forms classes** - Support developers in creating GUI for Windows. This provides a common, consistent development interface across all languages supported by the Microsoft .NET Framework.
2.3 Programming Languages

2.3.1 Microsoft Active Server Pages .NET 1.1

Microsoft Active Server Pages .NET (ASP.NET) is more than the next version of Microsoft Active Server Pages 3.0 (ASP); it is a Web development platform that provides the services that are required for developers to design and build enterprise-class Web applications (Microsoft Corporation 2004a).

The following sub-sections will briefly describe some of the main components within ASP.NET.

Web Forms

Microsoft Developer Network (2003, p. Netstart/html/cpglow.htm) defines a Web Form as an ‘ASP.NET page framework, which consists of programmable Web pages ... that contain reusable server controls’. To clarify, a Web form is a Web page that is compiled dynamically when it is requested by a client. Once a client requests the page, the ASP.NET Web server executes the methods defined by the programmer, with its result sent back to the client.

Web User Controls

Microsoft Developer Network (2003, p. Netstart/html/cpglou.htm) defines a Web User Control as:

‘A server control that is authored declaratively using the same syntax as an ASP.NET page and is saved as a text file with an .ascx extension. User controls allow page functionality to be partitioned and reused. Upon first request, the page framework parses a user control into a class that derives from System.Web.UI.UserControl and compiles that class into an assembly, which it reuses on subsequent requests.’

To clarify, a Web user control is identical to a Web form, however, they are designed to be included within Web forms. For example, most Web pages use a menu to navigate
2.3 Programming Languages

throughout the site. So that the menu does not have to be repeated for each page, a
Web user control (menu.ascx, for example) is included at a specific location within all
ASP.NET pages.

Code-Behind

*Code-Behind* can be defined in two forms, files and classes. A *code-behind file* is ‘a
code file containing the page class that implements the program logic of a Web Forms
or ASP.NET mobile Web Forms application’ (Microsoft Developer Network 2003, p.
Netstart/html/cpgloc.htm). A *code-behind class* is ‘a class that is accessed by an .aspx
file, but resides in a separate file (such as a .dll or .cs file). For example, you can write a
code-behind class that creates an ASP.NET custom server control, contains code that is
called from an .aspx file, but does not reside within the .aspx file’ (Microsoft Developer

This defines a very important advantage for programmers since the logic of an ASP.NET
page can exist outside of the page’s presentation code. Linking pages to classes also
allows programmers to develop powerful methods and properties that interact with the
page in a more logical manner.

2.3.2 Microsoft C# .NET 1.1

*Microsoft C# .NET* is the most common programming/scripting language for ASP.NET
and other methods. It is a modern, object-oriented, and type-safe language and en-
ables programmers to build a wide range of different applications for the new Microsoft
.NET platform. C#, pronounced "see-sharp", is designed to enable rapid development
to the C++ programmer without sacrificing the power and control that have been the
trademarks of C and C++ (Microsoft Corporation 2004e).

In 2001, the International Standards Office (ISO) received Microsoft C# .NET to be
standardised. In April of 2003, it was ratified as ISO/IEC 23270 (Microsoft Corporation
2004d).
In the following sub-sections, a brief description will be given to some of the main concepts and components within C# .NET.

**Value-Types versus Reference-Types**

In order to explain value and reference-types correctly, their definitions must be described. A *value-type* is a ‘data type that is represented by the type’s actual value’ (Microsoft Developer Network 2003, Netstart/html/cpglov.htm). To clarify, changing a value-type variable changes the actual value of the variable. Some value-typed data-types are `int`, `char`, enumerations, and structures.

A *reference-type* is a ‘data type that is represented by a reference (similar to a pointer) to the type’s actual value’ (Microsoft Developer Network 2003, Netstart/html/cpglor.htm). To clarify, a reference-type pointed to a place in memory where the actual value was held. This allows multiple reference-type variables to point to the same location, each changing the same actual value. All classes are reference-types.

**Interfaces**

An interface provides a framework to which a class must conform. That is, it supplies a definition of properties and/or methods which must be implemented by any derived class. Listing 2.1 shows an example of an interface.

```
Listing 2.1: Interface example in C#

public interface IManage
{
    bool isReadOnly
    {
        get;
        set;
    }
    void Create();
    void Delete();
    void Update();
}
```
Classes

A class within C# is made up of methods (virtual and abstract), attributes, properties, and static members. These class members are identical to C++, with minor changes. One of these changes is that in C++, a class can only inherit one other class. In C#, his ability has been replaced by enabling a class to inherit an infinite amount of interfaces.

Garbage Collection

Garbage collection is not a new concept within programming languages. It is ‘the process of transitive tracing through all pointers to actively used objects in order to locate all objects that can be referenced, and then arranging to reuse any heap memory that was not found during this trace’ (Microsoft Developer Network 2003, p. Netstart/html/cpglog.htm).

C and C++ do not have garbage collection. When memory was allocated to a variable, it must be deallocated. Listing 2.2 demonstrates an example of this.

Listing 2.2: C/C++ example of how and where memory is allocated and deallocated

```c
1 int main ( )
2 {
3     int mem[] = new int[100] // Allocated memory of 100 ints
4     // mem = null; // Would not deallocate the memory (no garbage collection)
5     delete mem; // Allocated memory now deallocated
6 }
```

C#, however, has this feature, and the allocated memory is deallocated once it is no longer referenced by a variable. Listing 2.3 demonstrates an example of this.

Listing 2.3: C# example of how and where memory is allocated and deallocated

```c
1 static void Main(string[] args)
```
2.3 Programming Languages

```c
{  
    int[] mem = new int[100];  // Allocated memory of 100 ints
    mem = null;  // Allocated memory now deallocated (garbage collection)
}
```

2.3.3 Microsoft Active Data Objects .NET 1.1

Microsoft Active Data Objects .NET, also known as ADO.NET, is defined by Microsoft Developer Network (2003, p. Netstart/html/cpglog.htm) to be:

‘The suite of data access technologies included in the .NET Framework class libraries that provide access to relational data and XML. ADO.NET consists of classes that make up the DataSet (such as tables, rows, columns, relations, and so on), .NET Framework data providers, and custom type definitions (such as SqlTypes for SQL Server).’

To clarify, ADO.NET consists of data connectivity features via the .NET Framework in the form of database and XML. The .NET Framework namespace that provides these class libraries is `System.Data`.

2.3.4 Extensible Markup Language

Extensible Markup Language (XML) is a simple and very flexible text format derived from the language, Standard Generalized Markup Language (SGML). It is playing an increasingly successful and important role in the exchange of a wide variety of data on the Web and elsewhere.

Microsoft Developer Network (2003, p. Netstart/html/cpgloe.htm) defines XML as providing ‘a uniform method for describing and exchanging structured data that is independent of applications or vendors.’ XML specification also provides more complex tools to maintain and manipulate XML.
2.3 Programming Languages

Extensible Markup Language Schema Definitions

*Extensible Markup Language Schema Definitions* (XSD) are a tool to structure and maintain that structure (through validation) in XML data files. This is done by defining data-types to the different nodes’ attributes and elements. Also, it defines the structure of the node by having a set of rules that determine how many elements and attributes exist under the root node.

Extensible Stylesheet Language Transformations

*Extensible Stylesheet Language Transformations* (XSLT) is another tool for XML files that can manipulate the way in which the data is displayed to the user. It allows the XML data to be unchanged so that other XSLT files can manipulate the data dynamically.

2.3.5 Structured Query Language

*Structured Query Language* (SQL) is the standard language for database operations and commands. Simple programming syntaxes allow tables, rows, and columns to be added to a database easily and efficiently. SQL also allows fast and efficient listings of database entries to be viewed by users.

Stored Procedures (Microsoft SQL Server specific)

Microsoft Developer Network (2003, architec/8_ar_da_0nxv.htm) defines a *stored procedure* as a ‘precompiled collection of Transact-SQL statements stored under a name and processed as a unit’. To clarify, it stores a precompiled query within the database as the database changes. This increases server efficiency by stopping the requirement to process and compile query requests.
XML Support

The support given by Microsoft SQL Server 2000 for XML is provided by transforming database data in the form of an XML document. This XML document can then by read and transformed by the application that requests the data. Listing 2.4 demonstrates a stored procedure that implements a \texttt{FOR XML EXPLICIT} mode. Listing 2.5 details an example XML document that would be compiled from Listing 2.4.

```sql
LISTING 2.4: \texttt{FOR XML EXPLICIT} stored procedure


cREATE PROCEDURE dbo.FindClass
  (@classID CHAR(10))
AS
  SELECT
    1 AS Tag,
    NULL AS Parent,
    NULL AS [Classes!1],
    NULL AS [Class!2!classID],
    NULL AS [Class!2!Name!element],
    NULL AS [Class!2!denyOtherUsers]
  UNION ALL
  SELECT
    2 AS Tag,
    1 AS Parent,
    NULL,
    Classes.classID,
    Classes.className,
    Classes.denyOtherUsers
  FROM Classes
  WHERE classID = @classID
  FOR XML EXPLICIT
```

LISTING 2.5: Output from the stored procedure in Listing 2.4

```xml
<Classes>
  <Class classID="ENG4111" denyOtherUsers="1">
    <Name>Research Project 1</Name>
  </Class>
</Classes>
```
2.3.6 Hyper-Text Markup Language

HyperText Markup Language (HTML) is made up of tags such as `<h1>` and `</h1>` to structure text into headings, paragraphs, images, lists as well as hypertext links so that data and information can be displayed in an appropriate and eye catching method. HTML is another language that is derived from the language SGML.

2.3.7 Cascade Style Sheeting

Cascading Style Sheets (CSS) is a simple programming mechanism used for adding styles to HTML Web documents. These include colours, fonts, spacing, table sizing, as well as a number of other style attributes. It was designed to speed up Web page development by specifying a style class in place of typing code repeatedly.

2.4 Development Environments and Tools

Every software project requires an environment under which development occurs. Tools are also important in assisting in the development of the project. The following subsections will briefly describe the environments and tools utilized within this project.

2.4.1 Microsoft Visual Studio .NET 2003

Microsoft Visual Studio .NET 2003 delivers the developer productivity required to deliver a wide range of professional software applications. Its Integrated Development Environment (IDE) provides a consistent interface for all languages, including Microsoft Visual Basic .NET, Microsoft Visual C++ .NET, Microsoft Visual C# .NET, and Microsoft Visual J# .NET. Developers can take advantage of shared visual designers
2.4 Development Environments and Tools

to design and build rich Windows-based applications and dynamic Web applications that render in any browser (Microsoft Corporation 2004f).

2.4.2 Microsoft ASP.NET Web Matrix

Microsoft ASP.NET Web Matrix allows developers to write ASP.NET code productively and efficiently. Designed and developed by some of the team that created ASP.NET, it has been completely written using the Microsoft .NET Framework and Microsoft C# .NET (Microsoft Corporation 2004b). Although it is not as powerful and diverse as Microsoft Visual Studio .NET 2003, it offers subtle advantages.

The Matrix Class Browser is one of these advantages as it lists all the classes (along with their properties, methods, and events) within the Microsoft .NET Framework.

2.4.3 Microsoft Windows Server 2003

Microsoft Windows Server 2003 is a multipurpose operating system capable of handling a diverse set of server roles. Some of these server roles include the ability to be a Web server with Web application services and a mail server. Microsoft Windows Server 2003 is the first operating system to completely integrate the Microsoft .NET Framework on installation (Microsoft Corporation 2002b).

2.4.4 Microsoft Internet Information Service 6.0

Internet Information Services 6.0 (IIS) is a powerful Web server which is included with all Microsoft Windows Server 2003 editions. It provides a highly reliable, manageable, scalable, and secure Web application infrastructure and environment. IIS allows developers to quickly and easily deploy Web sites as well as providing a high-performance platform for applications built using Microsoft ASP.NET and the Microsoft .NET Framework (Microsoft Corporation 2003d).
2.4 Development Environments and Tools

2.4.5 Microsoft SQL Server 2000 and 2005

*Microsoft SQL Server 2000* is an industry leader in database development and deployment. Its speed and efficiency in adding, deleting, altering, and searching data has made its mark in the *Database Management System* (DBMS) industry (Microsoft Corporation 2002c).

The *Microsoft Desktop Engine 2000* (MSDE) is a data engine built and based on core Microsoft SQL Server 2000 technology for database and application developers. It is a reliable storage engine and query processor for desktop extensions of enterprise applications. The common technology base shared between Microsoft SQL Server 2000 and MSDE allows developers to build applications that can scale seamlessly from portable computers to multiprocessor clusters. An example is the Microsoft SQL Server 2000 Enterprise Edition database server (Microsoft Corporation 2003c).

Microsoft has recently released a technical preview beta version of their upcoming *Microsoft SQL Server 2005*. This technical preview has been released as an *Express Edition* which is the equivalent to MSDE, described above, with its core engine their primary difference. The SQL Server 2005 engine offers (Microsoft Corporation 2003b):

- **.NET Framework Hosting** - Developers will be able to create database objects using languages such as Microsoft Visual C# .NET.
- **XML Technologies** - Native support for storage and query of XML documents.
- **Transact-SQL Enhancements** - Including error handling and recursive query capabilities.

2.4.6 Microsoft Developer Network

Throughout the history of Microsoft, a very important and vital tool has been offered (free of charge) to all public developers. The *Microsoft Developer Network* (MSDN) offers a substantial information database for Microsoft products and *Application Programming Interfaces* (APIs), including Windows, MS-DOS, DirectX, SQL Server, and
2.5 Software Project Implementations

recently, the .NET Framework. This information is offered in the form of white-papers, case studies, source code examples, and class interface descriptions.

Microsoft makes this library available on disc and on the Internet (Microsoft Developer Network 2004). With this information, programmers can develop their software so that it works within the Windows platform.

As an extension to MSDN, Microsoft offers discussion newsgroups where Microsoft employees, Microsoft’s Most Valuable Professionals (MVPs), and volunteers offer their advice and help in the situations where developers encounter difficulties or problems.

2.5 Software Project Implementations

Throughout the software industry, there are three implementation types:

- Executable Applications
- Web Site
- Web Applications

Executable Applications

An Executable Application is an executable file that can be a versatile, dynamically formed computer program that allows for full interaction with data. These programs are subject to acute synchronisation errors when accessing and communicating with databases. Each executable application is platform dependent, meaning that it requires a specific operating system in order to operate correctly. In addition, each computer must have an executable file available to access and manipulate data. An Executable Application can be identified by the file extension .exe.
2.5 Software Project Implementations

Web Sites

A Web Site is a neat and orderly set out page that statically displays data and allows a limited interaction with data. These pages have a restricted range of data manipulation techniques. A Web site can be used via the Internet or an Intranet network by using a third party Web browser and can be identified by the file extensions `.htm` and `.html`.

2.5.1 Web Applications

A Web Application is a combination of a Web site and an executable application. This means that Web application can be a neat and orderly laid out page that dynamically displays and manipulates data. It can be used via the Internet or an Intranet network by the use of a third party Web browser. A Web application can be identified by the file extension `.asp`, `.aspx`, `.jsp`, `.cgi`, and `.php`.

Web application’s are more complex than they appear. When implementing, they can be divided into a tier application architecture, which can range from a one-tier to an $n^{th}$-tier application. The most common are two-tier and three-tier application architectures. Figure 2.8 details these architectures.
The business rules component shown above (Figure 2.8) manipulates data from the data source so that other components can use the data. Common to all tier architectures are the different layers each possess. The four layers are:

- **Presentation Layer** - This layer presents data to the user via a user interface.
- **Workflow Layer** - The business rules on the client-side. That is, it controls the user’s input and other processes on the client.
- **Business Logic Layer** - The business rules on the server-side. That is, it controls the manipulation and flow of data on the server.
- **Data Access Layer** - Allows the business rules access to the data source.

**One-Tier Architecture**

The one-tier architecture contains all layers inside the application itself. In a large-scale environment, this type of application would be difficult to maintain. This type of architecture would be acceptable for small systems such as a small retail operation.
2.5 Software Project Implementations

Two-Tier Architecture

The two-tier architecture splits the application by creating the Data Access Layer (second tier). This allows the programmer to implement an application with multi-user access more easily than with a one-tier system. This type of architecture is suitable for small to medium systems that allows multiple users to access the system at the same time.

Three-Tier Architecture

The three-tier architecture splits the application again by creating the Business Logic Layer (second tier) alongside the Data Access Tier (third tier). This layer not only contains the code to determine the data’s purpose, but also code that determines how and when to manipulate the data.

This architecture was chosen as the back-bone of this project, because it allows:

- Users to interact with the application
- The server to manipulate the data to suit the database
- The database to store and serve data
- Users to access data simultaneously

N-Tier Architecture

The n-tier architecture is very similar to the three-tier architecture. The only difference is, in its decision to split its layers into a greater number, the work load is decreased in different processes. One example is the splitting of the Data Access Layer up into an SQL Server database and an XML Web Service.
Chapter 3

Software Design

This chapter describes the design behind the system and database architectures of the OLE project. It also identifies the Microsoft .NET Framework components and

Section 3.1 explains the thought behind the system architecture, while Section 3.2 explains the database architecture. Finally, Section 3.3 describes the different namespaces and members of the Microsoft .NET Framework.

Database Management System (DBMS) ‘A software system that enables users to define, create, maintain, and control access to the database.’ (Connolly & Begg 2002, p. 16)

3.1 System Architecture

In Chapter 2, specifically Section 2.5.1, the different application architectures were discussed. The OLE utilized the three-tier architecture, shown in Figure 3.1, which can also be found and described in the System Design Document (Appendix C).
3.1 System Architecture

Presentation Tier

This tier (subsystem) consists of two layers. They are the Presentation and Workflow layers. The Presentation layer consists of code that presents the data to the user. The languages used under this layer were HTML and CSS.

The Workflow layer consists of code that creates and manipulates data to fit within the Presentation layer. The language used was combination of C# .NET and ASP.NET.

Business Logic Tier

This tier involves a layer called the Business Logic layer. This layer consists of classes that are used within the OLE. The language used here was C# .NET.
3.1 System Architecture

Data Access Tier

This tier consists of the Data Access layer. It provides classes that can realise the storage, retrieval, and query of persistent objects within the Data Source. The languages used here were ADO.NET and SQL.

3.1.1 System-to-Data Relationship

Since the OLE was also a research project, a hybrid method was taken towards the relationship between the system and data. Two methods were used to manage data utilized, specifically XML files and an SQL Server database.

In order to create this hybrid method, a sealed class was designed to offer properties and methods relating to both SQL and XML data management. The name given to this class reflected its purpose; to offer a bridge between the system and data. The name, DataBridge was created.

Since SQL is very broad in its method calls, its connection class was offered as a property, so that its calls can be manually invoked. XML was used quite differently. It held two properties that gave the class and institution directories. These directories sorted the XML files. The class also offered many methods that returned next files’ paths and existing files’ paths.

3.1.2 Data Storage Techniques

Two types of data storage techniques were used within the OLE project. These two techniques are the most used methods within the development industry. They are:

- SQL databases; and
- XML files.
In order for these two techniques to exist within the project, a programming model that utilized these two techniques was required.

Section 3.2 describes the SQL database’s design, but not its connection to the XML file design. It will be noted here that within Section 3.4 within the SDD (Appendix C), the design and structure of the XML file and SQL database is documented. The SQL database is used for efficiently recovering data, since the database only holds superficial and important information (i.e. small amounts of information or information that will constantly be accessed). The XML files hold all of the information so that both techniques do not have to be called in order to get retrieve the requested information.

An example:

An assignment will have an identifier (ID), a due date, and a question to accompany it (simplified example). The SQL database will hold the:

- **ID** - A unique number to create a link between programming calls.
- **Due Date** - A small value that will be constantly called upon.

The XML file will hold all the values so:

- **ID** - As seen above.
- **Due Date** - Holds and the SQL database does not have to be called in order to retrieve this value.
- **Question** - Is a large field that can be any size.

There are two main reasons why this programming model was designed. They are:

- To learn both data storage techniques; and
- As a way to lessen the complication of the SQL database since they can only hold up to 8000 characters within a **VARCHAR** field.
3.2 Database Architecture

Before a database can be completed, it must pass through a number of design steps. Connolly & Begg (2002, pp. 420-421) describes the database design methodology, which was used in order to generate a database for the OLE. This methodology is divided into three main sections, each containing steps:

- **Conceptual Database Design** - ‘The process of constructing a model of the information used in an enterprise, independent of all physical considerations.’ (Connolly & Begg 2002, p. 419)
  - *Step 1* - Build a local conceptual data model.

- **Logical Database Design** - ‘The process of constructing a model of the information used in an enterprise based on a specific data model, but independent of a particular DBMS and other physical considerations.’ (Connolly & Begg 2002, p. 419)
  - *Step 2* - Build and validate a local logical data model.
  - *Step 3* - Build and validate a global logical data model.

- **Physical Database Design** - ‘The process of producing a description of the implementation of the database on secondary storage...’ (Connolly & Begg 2002, p. 419)
  - *Step 4* - Translate the global logical data model.

The steps shown above also split into sub-steps. Figure 3.2 display these steps in further detail, however, only steps 1 and 2 will be described.
3.2 Database Architecture

<table>
<thead>
<tr>
<th>Step 1 - Build a local conceptual data model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1.1 - Identify entity types</td>
</tr>
<tr>
<td>Step 1.2 - Identify relationship types</td>
</tr>
<tr>
<td>Step 1.3 - Identify and associate attributes with entity or relationship types</td>
</tr>
<tr>
<td>Step 1.4 - Determine attribute domains</td>
</tr>
<tr>
<td>Step 1.5 - Determine candidate and primary key attributes</td>
</tr>
<tr>
<td>Step 2 - Build and validate a local logical data model</td>
</tr>
<tr>
<td>Step 2.1 - Derive relations for local logical data model</td>
</tr>
<tr>
<td>Step 2.2 - Validate relations using normalization</td>
</tr>
<tr>
<td>Step 2.3 - Define integrity constraints</td>
</tr>
</tbody>
</table>

Figure 3.2: The steps in the Database Design Methodology (Steps 1 to 2)

This section will follow through the described steps, informing how the OLEs database was generated.

**Step 1.1 - Identify entity types**

Connolly & Begg (2002, p. 331) defines an *entity type* as, ‘a group of objects with the same properties, which are identified by the enterprise as having an independent existence’. Using this definition, the OLE has the following entity types:

- **Users** - An individual that contributes to the system.
- **Classes** - A course for users to enrol in and contribute.
- **Roles** - Three roles that define a user’s type of contributions.
- **AETQuestions** - An Assignment, Exam, or Tutorial question within a class.
- **AETAnswers** - An answer to an AETQuestion
- **ThreadSubscriptions** - A subscription to a forum thread, defined by the user.
3.2 Database Architecture

Step 1.2 - Identify relationship types

Connolly & Begg (2002, p. 334) defines a relationship type as, ‘a set of meaningful associations among entity types’. With this definition, the OLE has the following relationship types:

Users (1..1) Creates ⊳ (0..*) AETQuestions

A User can create zero-to-many AETQuestion(s).

An AETQuestion can have one User.

Users (0..*) Maintains ⊳ (0..*) Classes

A User can maintain zero-to-many Class(es).

A Class can have zero-to-many User(s).

Users (0..*) Supervises ⊳ (0..*) Users

A User can supervise zero-to-many User(s).

A User can be supervised by zero-to-many User(s).

Users (1..1) Owns ⊳ (0..*) AETAnswers

A User can own zero-to-many AETAnswer(s).

An AETAnswer can have one User.

Users (0..*) Has ⊳ (0..*) ThreadSubscriptions

A User can have zero-to-many ThreadSubscription(s).

A ThreadSubscription can have zero-to-many User(s).

Classes (0..*) Has ⊳ (0..*) Users

A Class can have zero-to-many User(s).

A User can have zero-to-many Class(es).

Classes (1..1) Has ⊳ (0..*) AETQuestions

A Class can have zero-to-many AETQuestion(s).

An AETQuestion can have one Class.

Roles (0..3) Assigns ⊳ (0..*) Users

A Role can assign zero-to-many User(s).

A User can have zero-to-many Role(s).
### 3.2 Database Architecture

**Step 1.3 - Identify and associate attributes with entity or relationship types**

Connolly & Begg (2002, p. 338) defines an *attribute* as, ‘a property of an entity or a relationship type’. With this definition, Table 3.1, 3.2, and 3.3 will detail the attributes that were used within each entity.

Table 3.1: The attributes within the entity types *Users, AETQuestions, and AETAnswers*

<table>
<thead>
<tr>
<th>Users</th>
<th>AETQuestions</th>
<th>AETAnswers</th>
</tr>
</thead>
<tbody>
<tr>
<td>userID</td>
<td>aetQID</td>
<td>aetQID</td>
</tr>
<tr>
<td>firstName</td>
<td>aetQType</td>
<td>aetQType</td>
</tr>
<tr>
<td>middleName</td>
<td>aetQTitle</td>
<td>aetQType</td>
</tr>
<tr>
<td>lastName</td>
<td>aetQDescription</td>
<td>aetQID</td>
</tr>
<tr>
<td>password</td>
<td>aetQQuestionType</td>
<td>classID</td>
</tr>
<tr>
<td>dob</td>
<td>aetQDueDateTime</td>
<td>aetQDueDateTime</td>
</tr>
<tr>
<td>postalStreet</td>
<td>aetQHaltSubmissionDateTime</td>
<td>aetASubmittedDate</td>
</tr>
<tr>
<td>postalSuburb</td>
<td>aetQViewQuestionDateTime</td>
<td>aetAMarked</td>
</tr>
<tr>
<td>postalState</td>
<td>aetQViewMarkedAnswersDateTime</td>
<td></td>
</tr>
<tr>
<td>postalPostcode</td>
<td>phoneNumber</td>
<td>aetAMarked</td>
</tr>
<tr>
<td>postalCountry</td>
<td>mobileNumber</td>
<td>aetQAmountOfQuestions</td>
</tr>
<tr>
<td>phoneNumber</td>
<td>emailAddress</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: The attributes within the entity types *Classes, Roles, and ThreadSubscriptions*

<table>
<thead>
<tr>
<th>Classes</th>
<th>Roles</th>
<th>ThreadSubscriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>classID</td>
<td>rolesID</td>
<td>classID</td>
</tr>
<tr>
<td>className</td>
<td></td>
<td>threadID</td>
</tr>
<tr>
<td>denyOtherUsers</td>
<td></td>
<td>classID</td>
</tr>
</tbody>
</table>
3.2 Database Architecture

Table 3.3: The attributes within the entity types *ClassUsers* and *OtherUsers*

<table>
<thead>
<tr>
<th>ClassUsers</th>
<th>OtherUsers</th>
</tr>
</thead>
<tbody>
<tr>
<td>classID</td>
<td>userID</td>
</tr>
<tr>
<td>userID</td>
<td>roleID</td>
</tr>
<tr>
<td>roleID</td>
<td></td>
</tr>
</tbody>
</table>

Step 1.4 - Determine attribute domains

Connolly & Begg (2002, p. 338) defines an *attribute domain* as, ‘the set of allowable values for one or more attributes’. These values are defined by SQL data types:

- **BIGINT (BI)** - An integer with a set character length of 8.
- **BIT (B)** - Equivalent to a boolean value with a set character length of 1.
- **CHAR (C)** - A variable character length that stores values of a specific length.
- **DATETIME (DT)** - A date and time value with a set character length of 8.
- **INT (I)** - An integer with a set character length of 4.
- **VARCHAR (VC)** - A variable character length that stores values of a variable length.

Tables 3.4, 3.5, and 3.6 lists the values beside each attribute within [ ] parentheses. For example, a **VARCHAR** of character length 50, for the attribute *firstName*, would be [VC50].
### 3.2 Database Architecture

Table 3.4: The attribute domains for the attributes in Table 3.1

<table>
<thead>
<tr>
<th>Users</th>
<th>AETQuestions</th>
<th>AETAnswers</th>
</tr>
</thead>
<tbody>
<tr>
<td>userID [I]</td>
<td>aetQID [BI]</td>
<td>aetQID [BI]</td>
</tr>
<tr>
<td>firstName [VC50]</td>
<td>classID [C7]</td>
<td>classID [C7]</td>
</tr>
<tr>
<td>middleName [VC50]</td>
<td>aetQType [VC20]</td>
<td>aetQType [VC20]</td>
</tr>
<tr>
<td>lastName [VC50]</td>
<td>aetQTitle [VC100]</td>
<td>userID [I]</td>
</tr>
<tr>
<td>password [C47]</td>
<td>aetQDescription [VC30]</td>
<td>aetASubmittedDateTime [DT]</td>
</tr>
<tr>
<td>dob [DT]</td>
<td>aetQQuestionType [VC20]</td>
<td>aetAMarked [B]</td>
</tr>
<tr>
<td>postalStreet [VC50]</td>
<td>aetQViewQuestionDateTime [DT]</td>
<td></td>
</tr>
<tr>
<td>postalSuburb [VC50]</td>
<td>aetQDueDateTime [DT]</td>
<td></td>
</tr>
<tr>
<td>postalState [VC50]</td>
<td>aetQHaltSubmissionDateTime [DT]</td>
<td></td>
</tr>
<tr>
<td>postalPostcode [VC10]</td>
<td>aetQViewMarkedAnswersDateTime [DT]</td>
<td></td>
</tr>
<tr>
<td>postalCountry [VC50]</td>
<td>aetQAmountOfQuestions [I]</td>
<td></td>
</tr>
<tr>
<td>phoneNumber [VC15]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mobileNumber [VC15]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>emailAddress [VC50]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.5: The attribute domains for the attributes in Table 3.2

<table>
<thead>
<tr>
<th>Classes</th>
<th>Roles</th>
<th>ThreadSubscriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>classID [C7]</td>
<td>rolesID [VC50]</td>
<td>classID [C7]</td>
</tr>
<tr>
<td>className [VC50]</td>
<td></td>
<td>userID [I]</td>
</tr>
<tr>
<td>denyOtherUsers [B]</td>
<td></td>
<td>threadID [I]</td>
</tr>
</tbody>
</table>
3.2 Database Architecture

Table 3.6: The attribute domains for the attributes in Table 3.3

<table>
<thead>
<tr>
<th>ClassUsers</th>
<th>OtherUsers</th>
</tr>
</thead>
<tbody>
<tr>
<td>classID [C7]</td>
<td>userID [I]</td>
</tr>
<tr>
<td>userID [I]</td>
<td>roleID [VC50]</td>
</tr>
<tr>
<td>roleID [VC50]</td>
<td></td>
</tr>
</tbody>
</table>

Step 1.5 - Determine candidate and primary key attributes

Within a database entity, there are three types of key attributes. They are:

- **Candidate Key** - ‘The minimal set of attributes that uniquely identifies each occurrence of an entity type.’ (Connolly & Begg 2002, p. 340)

- **Primary Key** - ‘The candidate key that is selected to uniquely identify each occurrence of an entity type.’ (Connolly & Begg 2002, p. 341)

- **Composite Key** - ‘A candidate key that consists of two or more attributes.’ (Connolly & Begg 2002, p. 341)

With these definitions, Tables 3.7, 3.8, and 3.9 will highlight the primary and composite keys from the set attributes detailed in Tables 3.1, 3.2, and 3.3. Primary keys are defined by having a \{PK\} notation beside an attribute. A composite key is defined by having more than one attribute with the \{PK\} notation.
### 3.2 Database Architecture

#### Table 3.7: The highlighted and created primary and composite keys from Table 3.1

<table>
<thead>
<tr>
<th>Users</th>
<th>AETQuestions</th>
<th>AETAnswers</th>
</tr>
</thead>
<tbody>
<tr>
<td>userID {PK}</td>
<td>aetQID {PK}</td>
<td>aetQID {PK}</td>
</tr>
<tr>
<td>firstName</td>
<td>classID {PK}</td>
<td>aetQType {PK}</td>
</tr>
<tr>
<td>middleName</td>
<td>aetQType {PK}</td>
<td>aetQType {PK}</td>
</tr>
<tr>
<td>lastName</td>
<td>aetQDescription</td>
<td>userID {PK}</td>
</tr>
<tr>
<td>password</td>
<td>aetQQuestionType</td>
<td>aetQDescription</td>
</tr>
<tr>
<td>dob</td>
<td>aetQDueDateTime</td>
<td>aetQAmountOfQuestions</td>
</tr>
<tr>
<td>postalStreet</td>
<td>aetQViewQuestionDateTime</td>
<td></td>
</tr>
<tr>
<td>postalSuburb</td>
<td>aetQDueDateTime</td>
<td></td>
</tr>
<tr>
<td>postalState</td>
<td>aetQHaltSubmissionDateTime</td>
<td></td>
</tr>
<tr>
<td>postalPostcode</td>
<td>aetQViewMarkedAnswersDateTime</td>
<td></td>
</tr>
<tr>
<td>postalCountry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phoneNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mobileNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>emailAddress</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Table 3.8: The highlighted and created primary and composite keys from Table 3.2

<table>
<thead>
<tr>
<th>Classes</th>
<th>Roles</th>
<th>ThreadSubscriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>classID {PK}</td>
<td>rolesID {PK}</td>
<td>classID {PK}</td>
</tr>
<tr>
<td>className</td>
<td></td>
<td>userID {PK}</td>
</tr>
<tr>
<td>denyOtherUsers</td>
<td></td>
<td>threadID {PK}</td>
</tr>
</tbody>
</table>

#### Table 3.9: The highlighted and created primary and composite keys from Table 3.3

<table>
<thead>
<tr>
<th>ClassUsers</th>
<th>OtherUsers</th>
</tr>
</thead>
<tbody>
<tr>
<td>classID {PK}</td>
<td>userID {PK}</td>
</tr>
<tr>
<td>userID {PK}</td>
<td>roleID {PK}</td>
</tr>
<tr>
<td>roleID {PK}</td>
<td></td>
</tr>
</tbody>
</table>
3.2 Database Architecture

Step 2.1 - Derive relations for local logical data model

Within a database, there are two types of entities. They are:

- **Strong Entity Types** - ‘An entity type that is not existence-dependent on some other entity type.’ (Connolly & Begg 2002, p. 342)

- **Weak Entity Types** - ‘An entity type that is existence-dependent on some other entity type.’ (Connolly & Begg 2002, p. 343)

From these definitions, the strong and weak entity types within the database will be defined. Once defined, the relationships between entities will be detailed.

**Strong Entity Types:**

**Users** (userID, ...)
- **Primary Key** userID

**Classes** (classID, ...)
- **Primary Key** classID

**Roles** (roleID)
- **Primary Key** roleID

**Weak Entity Types:**

**AETAnswers** (aetQID, classID, aetQType, userID, ...)
- **Alternate Key** aetQType
- **Foreign Key** aetQID
- **Foreign Key** classID
- **Foreign Key** userID

**AETQuestions** (aetQID, classID, aetQType, ...)
- **Primary Key** aetQID
- **Alternate Key** aetQType
- **Foreign Key** classID

**ClassUsers** (classID, userID, roleID)
- **Foreign Key** classID
3.2 Database Architecture

Foreign Key userID

Foreign Key roleID

OtherUsers (userID, roleID)

Foreign Key userID

Foreign Key roleID

ThreadSubscriptions (threadID, classID, userID)

Primary Key threadID

Foreign Key classID

Foreign Key userID

Relationships:

AETAnswers (aetQID, classID, aetQType, userID, ...)

Foreign Key aetQID references AETQuestions(aetQID)

Foreign Key classID references Classes(classID)

Foreign Key userID references Users(userID)

AETQuestions (aetQID, classID, aetQType, ...)

Foreign Key classID references Classes(classID)

ClassUsers (classID, userID, roleID)

Foreign Key classID references Classes(classID)

Foreign Key userID references Users(userID)

Foreign Key roleID references Roles(roleID)

OtherUsers (userID, roleID)

Foreign Key userID references Users(userID)

Foreign Key roleID references Roles(roleID)

ThreadSubscriptions (threadID, classID, userID)

Foreign Key classID references Classes(classID)

Foreign Key userID references Users(userID)
Step 2.2 - Validate relations using normalization

Connolly & Begg (2002, p. 376) defines a normalization as, ‘a technique for producing a set of relations with desirable properties given the data requirements of an enterprise’. With this definition, the entities within the OLE are already in its maximum normal form (with the exception of $AETAnswers(aetQType)$ and $AETQuestions(aetQType)$).

Step 2.3 - Define integrity constraints

This step will define the referential integrity between entities within the OLE. That is, how records are updated/deleted when a referenced record is updated/deleted.

**Relationships:**

- **AETAnswers** $(aetQID, classID, aetQType, userID, ...)$
  
  **Foreign Key** $aetQID$ references $AETQuestions(aetQID)$
  
  ON UPDATE CASCADE, ON DELETE CASCADE

- **AETQuestions** $(aetQID, classID, aetQType, ...)$
  
  **Foreign Key** $classID$ references $Classes(classID)$
  
  ON UPDATE CASCADE, ON DELETE CASCADE

- **ClassUsers** $(classID, userID, roleID)$
  
  **Foreign Key** $classID$ references $Classes(classID)$
  
  ON UPDATE CASCADE, ON DELETE CASCADE

- **OtherUsers** $(userID, roleID)$
  
  **Foreign Key** $userID$ references $Users(userID)$
3.3 Design using the Microsoft .NET Framework

This section describes the main .NET Framework components (classes, enumerations, and interfaces) that were used in the OLEs implementation. Microsoft Developer Network (2003) describes the main namespaces that were used in the OLE:

- **Microsoft.Win32** - ‘The Microsoft.Win32 namespace provides two types of classes: those that handle events raised by the operating system and those that manipulate the system registry.’ (Microsoft Developer Network 2003, p. cpref/html/frlrfMicrosoftWin32.htm)

- **System** - ‘The System namespace contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.’ (Microsoft Developer Network 2003, p. cpref/html/frlrfSystem.htm)

- **System.Collections** - ‘The System.Collections namespace contains interfaces and classes that define various collections of objects, such as lists, queues, bit arrays, hashtables and dictionaries.’ (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemCollections.htm)

- **System.Data** - ‘The System.Data namespace consists mostly of the classes that constitute the ADO.NET architecture. The ADO.NET architecture enables you
3.3 Design using the Microsoft .NET Framework

to build components that efficiently manage data from multiple data sources.'
(Microsoft Developer Network 2003, p. cpref/html/frlrfSystemData.htm)


- **System.IO** - ‘The System.IO namespace contains types that allow reading and writing to files and data streams, and types that provide basic file and directory support.’ (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemIO.htm)

- **System.Security.Cryptography** - ‘The System.Security.Cryptography namespace provides cryptographic services, including secure encoding and decoding of data, as well as many other operations, such as hashing, random number generation, and message authentication.’ (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemSecurityCryptography.htm)


- **System.Text** - ‘The System.Text namespace contains classes representing ASCII, Unicode, UTF-7, and UTF-8 character encodings; abstract base classes for converting blocks of characters to and from blocks of bytes; and a helper class that manipulates and formats String objects without creating intermediate instances of String.’ (Microsoft Developer Network 2003, )

- **System.Xml** - ‘The System.Xml namespace provides standards-based support for processing XML.’ (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemXml.htm)

- **System.Xml.Serialization** - ‘The System.Xml.Serialization namespace contains classes that are used to serialize objects into XML format documents or streams.’ (Microsoft Developer Network 2003, p. cpref/html/frlrfSystemXmlSerialization.htm)

The following classes, enumerations, and interfaces were used:
3.3 Design using the Microsoft .NET Framework

- **Microsoft.Win32.RegistryKey class** - Opens access to the Win32 registry and encapsulates any or new registry keys, allowing alterations, deletions, and creations.

- **System.BitConverter class** - Converts base data types to an array of bytes and also converts an array of bytes to base data types.

- **System.Collections.ArrayList class** - Offers methods and properties that manipulate object arrays. That is, the inserting, removing, and relocating of objects within the list.

- **System.Data.CommandType enumeration** - Lists the types of commands that can be executed by an System.Data.SqlClient.SqlCommand class.

- **System.Data.SqlDbType enumeration** - Lists the data types that exist within an SQL Server database (e.g. CHAR, VARCHAR, INT, etc.).

- **System.Data.SqlClient.SqlConnection class** - Creates a connection to an existing SQL Server database server when supplied a correct connection string.


- **System.Data.SqlClient.SqlDataAdapter class** - The ability to read data which is returned from an System.Data.SqlClient.SqlCommand command execution.

- **System.IO.Directory class** - Creates, deletes, and renames a directory within the local filesystem. It can also return an array of files that exist within the directory.

- **System.IO.File class** - Creates, deletes, and renames a file within the local filesystem. It can also provide detailed information about the file.

- **System.Security.Cryptography.MD5CryptoServiceProvider class** - Converts a string to an MD5 encrypted byte[].

- **System.Security.Principal.IPrincipal interface** - Provides methods and properties that define a .NET Framework user. The interface is used in many ASP.NET authentication processes.
• **System.Text.ASCIIEncoding class** - Converts an ASCII character set to other character encodings.

• **System.Xml.XmlDocument class** - The ability to load an XML file and manipulate its contents.

• **System.Xml.XmlTextReader class** - The ability to read through an XML file.

• **System.Xml.XmlTextWriter class** - The ability to write an XML file.

• **System.Xml.Serialization.XmlSerializer class** - The ability to convert attributed class properties into an XML document, using the `Serialize()` method. This process can also be reversed by using the `Deserialize()` method.

• **System.Xml.Serialization.XmlArrayAttribute class** - Defines a property as an XML node with items (`System.Xml.Serialization.XmlArrayItem`).

• **System.Xml.Serialization.XmlArrayItem class** - Defines an array property’s items as XML nodes within the `System.Xml.Serialization.XmlArrayAttribute` node.

• **System.Xml.Serialization.XmlTextAttribute class** - Defines a property to exist as text within an XML element.

• **System.Xml.Serialization.XmlRootAttribute class** - Defines a class to be the root XML node of an XML document.

• **System.Xml.Serialization.XmlElementAttribute class** - Defines a property to be an XML element under the XML root node.

• **System.Xml.SerializationXmlAttribute class** - Defines a property to be an attribute of an XML element.
Chapter 4

Software Implementation

The implementation of the OLE project involved the use of all programming languages listed under Section 2.3. However, the two most significant programming languages were ASP.NET and XML.

This chapter will start by explaining some of the problematic issues revolving around the project’s implementation, in Section 4.1. Section 4.2 will then explain some of the decisions behind the implementation using ASP.NET, while Section 4.3 will explain the decisions made using XML.

**Windows Registry** ‘The registry acts as a central repository of information for the operating system and the applications on a computer. The registry is organized in a hierarchical format, based on a logical ordering of the elements stored within it...’ (Microsoft Developer Network 2003, cpref/html/frlrfmicrosoftwin32registrykeyclasstopic.htm)

**Win32** Microsoft Windows in 32-bit executable code.
4.1 Implementation Issues

This section contains some implementation issues that could not be resolved, researched (due to lack of time), or implemented due to the lack of support of features within the software.

4.1.1 XML in Microsoft SQL Server 2000

In Section 2.3.5, Microsoft SQL Server’s support for XML was discussed. SQL Server 2000 however, does not support an XML document formed by the SQL data-type, BINARY (formed by a stored procedure using FOR XML EXPLICIT mode). SQL Server 2005 provides native support for XML, allowing this problem to be bypassed.

Originally, a user’s password was hashed into a C# byte[] data-type so that it could be stored in an SQL BINARY field. Instead, the C# byte[] data-type was converted to a string of hexadecimal values and then stored in a an SQL CHAR field. This was decided since SQL Server 2000’s support for XML was not as required and SQL Server 2005 had not been completely developed by Microsoft.

4.1.2 Registry and File System Permissions

During the implementation process of the setup application for the OLE project, a few issues were encountered during post-installation testing of the Web application. It was noticed that if the data files were created under a directory without full permissions given to Everyone (a default NTFS user, defining every user under the operating system), files were unable to be created, altered, or removed. Read-only access was the only permission given. This was also consistent with the values within the Windows registry.
4.2 Implementation using ASP.NET

4.1.3 Native XML Architecture

The choice in only implementing a small portion of XML’s features and capabilities was based on the time required to research against the time required for all the other activities within the project. However, foundation knowledge for all of XML’s features and capabilities was researched.

4.2 Implementation using ASP.NET

This section describes the primary impacts on the OLE project’s implementation while using ASP.NET. Some of these implementation impacts were discovered while implementing ASP.NET authentication, and creating Web forms and user controls.

4.2.1 Authentication

Authentication under ASP.NET works differently to most other programming languages. A number of procedures must have been completed in order for ASP.NET authentication to be enabled. J. D. Meier & Vasireddy (2002a) defines these procedures in its How-To article. The main steps as provided in this article are:

- **Step 1** - Configure the Web application for Forms Authentication.
- **Step 2** - Generate an Authentication Ticket for the authenticated users.
- **Step 3** - Create a class that implements the IPrincipal interface.
- **Step 4** - Create the CustomPrincipal object.

The obvious reason for Microsoft to publish an article such as this one is to ensure that ASP.NET is practiced the way it was designed to be. It also allows ASP.NET developers to move from one ASP.NET application to another without having to learn a the previous developers’ implementation.
Step 1 - Configure the Web application for Forms Authentication

This step involves the configuration of the Web.config source code to enable Forms Authentication. In essence, the only change required is the setting of the Uniform Resource Locator (URL) of the user login page, changing the `<authentication>` element’s `mode` attribute to `Forms`, and defining an XML element under the `<authentication>` node called `<deny>`, with its `users` attribute set as `?`. This step’s implementation can be found within Listing H.22, between lines 6 and 11.

Step 2 - Generate an Authentication Ticket for the authenticated users

This step involves the addition of a `bool` valued variable called `isAuthenticated`. A method is called to check for a user’s valid password, and its result is returned to `isAuthenticated` (successful as `true` and unsuccessful as `false`). On the success of a user’s authentication, a `System.Web.Security.FormsAuthenticationTicket` is created with the user’s details, encrypted, and stored as a Web cookie on the user’s computer. The user is then redirected to the page the requested before being redirected to the login page.

This step’s implementation can be found within Listing H.62, between lines 39 and 83.

Step 3 - Create a class that implements the IPrincipal interface

This step involves the integration of the `IPrincipal` interface with the OLE’s `User` class. The integration consists of a method, `bool IsInRole(string)`, and a property, `System.Security.Principal.IIdentity Identity`. The `IsInRole()` method is used by ASP.NET to identify the user’s assigned roles (i.e. Student, Instructor, and/or Administrator). The `Identity` property is made up of the decrypted cookie, created by the previous step.

The roles that the user is assigned to can be a problem. For example, a user could have a role Student, but a student of which class? This problem is overcome by invoking a
4.3 Implementation using XML

method that confirms their role status for a particular class when a class function is accessed.

This step’s implementation can be found within Listing H.106, between lines 21 and 58.

Step 4 - Create the CustomPrincipal object

Creating a IPrincipal object involves the implementation of the User object, created from the FormsAuthenticationTicket in Step 2, within the application authentication event handler. The event handler is initiated by decrypting the cookie kept on the user’s computer. If this cookie is not found, the authentication process is cancelled and the user must be asked for their authentication details once again. However, if the cookie is found and has not expired, the user will then be redirected to the page they first requested.

This step’s implementation can be found within Listing H.11, between lines 86 and 134.

4.2.2 Web Forms and User Controls

Within the OLE project, a Web form is a subsystem, and a Web user control was a component of its respective subsystem. For example, the menu is a component that exists within every subsystem. So the menu was implemented as a Web user control.

This created a powerful implementation, allowing the Web user controls to be reused, rather than having to be recoded on each Web form.

4.3 Implementation using XML

This section details some examples of the XML data structures that were used within the OLE project in order to store data. It also explains how these data structures were assembled and disassembled using the .NET Framework.
4.4 The Incomplete Web Application

4.3.1 XML Data Structures and Database Compatibility

The data structures designed in the OLE project were based on the tables inside the SQL Server database’s tables. These tables then formed the classes inside the Business Logic tier of the application. The reason for this was to ensure a compatibility between the database and Web application.

This compatibility was achieved by the .NET Framework’s serialisation and deserialisation class, which is discussed in the following sub-section.

4.3.2 Serialization and Deserialization

Within the Microsoft’s .NET Framework, an XML based class known as XmlSerializer is found under the System.Xml.Serialization namespace. This class offers the ability to ‘serialise’ a class and its members into an XML document that can then be ‘deserialised’ into a new instance of that same class.

This method was used to form the XML data structures easily and effectively. It allowed the SQL stored procedures that were converted into XML documents to be easily ‘deserialised’ into a new instance of its respective class.

4.4 The Incomplete Web Application

This section provides several screenshots of the OLE Web application within Microsoft Internet Explorer 6.0. Since the OLE Web application was not completely implemented, only a few screenshots of the application are shown.

The first screenshot, Figure 4.1, shows the Login page for users to access the Web application. Figure 4.2 depicts the Home page for user ‘Jonathan Trinder’ (userID: 26). Finally, Figure 4.3 shows the Class page, with ‘Assignment 1’ for class, ‘Research Project 1’ (courseID: ENG4111).
4.4 The Incomplete Web Application

**Figure 4.1:** Login page

**Figure 4.2:** Home page for user ‘Jonathan Trinder’ (userID: 26)
Figure 4.3: ‘Assignment 1’ for ‘Research Project 1’ (courseID: ENG4111) within the Class page
Chapter 5

Software Deployment

This chapter describes the step-by-step procedures required to deploy the ‘Online Learning Environment’ in a Microsoft Windows Server 2003 environment. It will first display a prerequisite option that is required for the application to run successfully in Microsoft Internet Information Services 6.0 (Section 5.1). The chapter will then explain the installation process of the OLE within Microsoft Windows Server 2003 (Section 5.2).

**Virtual Folder** A folder that is treated by Microsoft Windows as a normal folder, but does not correspond to a physical directory.

**Server/Machine Name** A unique name that identifies a computer to the network. A computer can also be referred to as having the name, `localhost`.

### 5.1 Configuration of Microsoft Windows Server 2003

Within *Windows Server 2003*, ASP.NET support is disabled throughout the operating system (specifically IIS). The OLE requires ASP.NET and therefore ASP.NET support must to be enabled.

In order to enable ASP.NET, navigation to Windows’ *Control Panel* occurred, then *Add/Remove Programs* was opened. A button inside this window called the *Windows
5.2 Installation - The Setup Application

Component Wizard was clicked and then the Application Server was selected and the Details button was clicked. Figure 5.1 shows this window and the option to check ‘enabled’.

Figure 5.1: The Application Server window from the Windows Component Wizard, under Windows Server 2003

5.2 Installation - The Setup Application

This section describes the steps involved in the installation the OLE. The installation fails when the presence of Microsoft Internet Information Services 5.0+ (ISS) and Microsoft .NET Framework 1.0+ is not found. This is by design because the OLE uses both components.

The files required for the setup application to execute, are:

- Setup.exe - Required to execute the setup application.
- OLEWebSetup.msi - Holds the data and files required for installation. This can be executed on some Windows operating systems without the use of Setup.exe. However, using Setup.exe to execute the installation is recommended.
5.2 Installation - The Setup Application

- **Setup.ini** - Holds meta-data for the **Setup.exe** file that redirects the user to the Microsoft .NET Framework’s download page in the case that the user does not have it installed. It also describes which **.msi** file holds the installation data.

**Step 1 - Welcome Screen**

Figure 5.2 details the first interactive screen for the user. It is usually shown when all pre-requisites are met. To proceed, the user clicks on *Next*. Otherwise, the user clicks *Cancel*.

![Welcome screen](image)

Figure 5.2: Welcome screen (first step)

**Step 2 - Configuration**

Figure 5.3 details the next screen the user encounters. It contains a textbox that asks for the server name of their SQL Server application. By default, the textbox contains **localhost**, which is the name of the machine in which they executed the setup application.
Step 3 - Installation Address

Figure 5.4 details the screen following the previous step. It asks for which virtual folder should the files be copied to. Also, it asks what port the application should set the virtual folder to use.
Figure 5.4: IIS virtual folder and port setup (third step)

Step 4 - Installation Confirmation

Figure 5.5 details the screen before installation commences. This screen appears in the occurrence of a mistake by a user, providing the ability to use Back so that the settings can be altered.
Step 5 - Installation Progress

Figure 5.6 details the installation’s progress by the use of a progress bar. It continues once the bar has been completely filled.
5.2 Installation - The Setup Application

Figure 5.6: Setup installing (fifth step)

Step 6 - Installation Complete

Figure 5.7 details the installation’s status. If the installation does not encounter a problem or error, this screen will appear.
Step 7 - Database Installation

Once the setup application has successfully installed the OLE, the user must execute a .sql file within their Microsoft SQL Server 2000+ database, installing the database required for the OLE to work correctly. This file is located in the virtual directory, specified in Step 3.
Chapter 6

Software Project Management

The software project management activities presented in this chapter make up the ‘nuts and bolts’ that construct the management of the OLE project. It allows the progress of its development to be followed, in addition to mapping out the project development’s structure.

The first section of this chapter, Section 6.1, describes how documents were maintained throughout the project’s development. Section 6.2 details the project’s schedule, while Section 6.3 lists the possible risks that may have posed unwanted issues towards the project. Section 6.4 identifies the resources required in order for the project to be implemented. Finally, Section 6.5 details the project’s key development environment, Microsoft Visual Studio .NET 2003, and how it managed the implementation of the OLE project’s programming source code.

6.1 Documentation

This project generated documents (outlined in Section 2.2.2) as draft revisions. A small change in the document structure implies a small change in the revision title. For example, the revision title of Draft 1.1 increases to Draft 1.2. If a large change in the document occurs, the revision title increases from Draft 1.2 to Draft 2.1. Once a document has been completed, the revision title is removed.
6.2 Schedule Management

This system was introduced to keep a record and back-up of the direction these documents were moving. If questions arose to why a feature was taken out from later document revisions, but existed in earlier revisions, the earlier revisions would be consulted to investigate why it was removed.

6.2 Schedule Management

Tables 6.1 and 6.2 shows the project’s work structure and an estimation of completion times. The headings include:

- **Step** - Provides the step identification number.
- **Description** - Provides the name of the task.
- **Estimate** - Provides the estimated time to complete the task.
6.2 Schedule Management

Table 6.1: The Documentation, Requirements Elicitation and Analysis, and Design work structure

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Project Specification</td>
<td>5 hours</td>
</tr>
<tr>
<td>1.2</td>
<td>Project Appreciation</td>
<td>50 hours</td>
</tr>
<tr>
<td>1.3</td>
<td>Requirements Analysis Document (RAD)</td>
<td>30 hours</td>
</tr>
<tr>
<td>1.4</td>
<td>System Design Document (SDD)</td>
<td>9 days</td>
</tr>
<tr>
<td>1.5</td>
<td>Object Design Document (SDD)</td>
<td>9 days</td>
</tr>
<tr>
<td>1.6</td>
<td>Project Dissertation</td>
<td>5 weeks</td>
</tr>
<tr>
<td>1.7</td>
<td>Test Manual</td>
<td>8 days</td>
</tr>
<tr>
<td>1.8</td>
<td>User Manual</td>
<td>10 hours</td>
</tr>
<tr>
<td>2</td>
<td>Requirements Elicitation and Analysis</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>Needs analysis</td>
<td>3 days</td>
</tr>
<tr>
<td>2.2</td>
<td>Secure required resources</td>
<td>1 day</td>
</tr>
<tr>
<td>2.3</td>
<td>Obtain supervisor approval</td>
<td>2 hours</td>
</tr>
<tr>
<td>3</td>
<td>Design</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Review RAD</td>
<td>1 day</td>
</tr>
<tr>
<td>3.2</td>
<td>Design Presentation Tier</td>
<td>3 days</td>
</tr>
<tr>
<td>3.3</td>
<td>Design Business Logic Tier</td>
<td>3 days</td>
</tr>
<tr>
<td>3.4</td>
<td>Design Data Access Tier</td>
<td>3 days</td>
</tr>
</tbody>
</table>
Table 6.2: The Training, Implementation, Testing, and Deployment work structure

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Training</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>C# .NET</td>
<td>6 days</td>
</tr>
<tr>
<td>4.2</td>
<td>ASP.NET</td>
<td>6 days</td>
</tr>
<tr>
<td>4.3</td>
<td>SQL</td>
<td>3 days</td>
</tr>
<tr>
<td>4.4</td>
<td>XML</td>
<td>6 days</td>
</tr>
<tr>
<td>5</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Review RAD, SDD, and ODD</td>
<td>5 days</td>
</tr>
<tr>
<td>5.2</td>
<td>Develop code</td>
<td>30 days</td>
</tr>
<tr>
<td>5.3</td>
<td>Implementation testing (Debugging)</td>
<td>10 days</td>
</tr>
<tr>
<td>6</td>
<td>Testing</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>Develop unit test plans</td>
<td>2 days</td>
</tr>
<tr>
<td>6.2</td>
<td>Develop integration test plans</td>
<td>2 days</td>
</tr>
<tr>
<td>6.3</td>
<td>Unit Testing</td>
<td></td>
</tr>
<tr>
<td>6.3.1</td>
<td>Test component modules</td>
<td>2 days</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Modify code</td>
<td>1 day</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Re-test modified code</td>
<td>1 day</td>
</tr>
<tr>
<td>6.4</td>
<td>Integration Testing</td>
<td></td>
</tr>
<tr>
<td>6.4.1</td>
<td>Test module integration</td>
<td>2 days</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Modify code</td>
<td>1 day</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Re-test modified code</td>
<td>1 day</td>
</tr>
<tr>
<td>7</td>
<td>Deployment</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>Deploy software</td>
<td>2 days</td>
</tr>
</tbody>
</table>

Since the programming and development experience of the writer was limited to different languages and applications, the time estimation was approximated as best as possible.

6.3 Risk Management

Briefly discussed in Section 2.2.4, risk management is split into three section:
6.3 Risk Management

- Risk Identification
- Risk Analysis
- Risk Strategy

Risk Identification

Table 6.3 identifies and lists the risks that may be encountered within the design and development of this project (Sommerville 2001, pp. 87-88). They are organised into a Risk Type column outlined in the table below (Table 6.3).

Table 6.3: Risk types and the risks that may be encountered (Risk Identification)

<table>
<thead>
<tr>
<th>Risk Type</th>
<th>Risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>Computer hardware failure</td>
</tr>
<tr>
<td></td>
<td>Computer security compromised</td>
</tr>
<tr>
<td></td>
<td>Data loss</td>
</tr>
<tr>
<td>People</td>
<td>Illness</td>
</tr>
<tr>
<td>Organisational</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Tools</td>
<td>Software unavailability</td>
</tr>
<tr>
<td></td>
<td>Software evaluation expiration</td>
</tr>
<tr>
<td>Requirements</td>
<td>Changes to requirements (major design rework)</td>
</tr>
<tr>
<td>Estimation</td>
<td>Time required for development (underestimate)</td>
</tr>
<tr>
<td></td>
<td>Rate of bug fixing (underestimate)</td>
</tr>
<tr>
<td></td>
<td>Size of the software (underestimate)</td>
</tr>
</tbody>
</table>

Risk Analysis

Table 6.4 lists and analyses the risks that may affect or jeopardise the project; along with the probability of affect upon, and the effect it may impose on the project (Sommerville 2001, pp. 88-89).

Each risk has a probability rating as either:
6.3 Risk Management

- Very low (<10%)
- Low (10-25%)
- Medium (25-50%)
- High (50-75%)
- Very high (>75%)

The effects of each risk are assessed as either:

- Catastrophic
- Serious
- Tolerable
- Insignificant

Table 6.4: Risks, their probability and effects (Risk Analysis)

<table>
<thead>
<tr>
<th>Risk</th>
<th>Probability</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer hardware failure</td>
<td>Medium</td>
<td>Catastrophic</td>
</tr>
<tr>
<td>Computer security compromised</td>
<td>Medium</td>
<td>Serious</td>
</tr>
<tr>
<td>Data loss</td>
<td>Medium</td>
<td>Catastrophic</td>
</tr>
<tr>
<td>Illness</td>
<td>High</td>
<td>Insignificant</td>
</tr>
<tr>
<td>Software unavailability</td>
<td>Low</td>
<td>Catastrophic</td>
</tr>
<tr>
<td>Software evaluation expiration</td>
<td>Very High</td>
<td>Tolerable</td>
</tr>
<tr>
<td>Changes to requirements</td>
<td>Very Low</td>
<td>Serious</td>
</tr>
<tr>
<td>Time required for development (underestimate)</td>
<td>Medium</td>
<td>Serious</td>
</tr>
<tr>
<td>Rate of bug fixing (underestimate)</td>
<td>Medium</td>
<td>Tolerable</td>
</tr>
<tr>
<td>Size of the software (underestimate)</td>
<td>High</td>
<td>Catastrophic</td>
</tr>
</tbody>
</table>

Risk Strategy

Table 6.5 lists the risks from Table 6.4 and strategies that may be carried out in order to avoid such a danger. These strategies need to be followed in order for work momentum
6.4 Resource Requirements

to be maintained.

Table 6.5: Risks and their strategies (Risk Strategy)

<table>
<thead>
<tr>
<th>Risk</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer hardware failure</td>
<td>Archive constant backups of data using compact disc writing and/or secondary hard drive.</td>
</tr>
<tr>
<td>Computer security compromised</td>
<td>Deploy a software firewall to prevent any unauthorised access into computer system.</td>
</tr>
<tr>
<td>Data loss</td>
<td>Archive constant printings of documentation.</td>
</tr>
<tr>
<td>Illness</td>
<td>Leave an opening at the end of the project due date to allow for such an event.</td>
</tr>
<tr>
<td>Software unavailability</td>
<td>Locate and deploy a different software source.</td>
</tr>
<tr>
<td>Software evaluation expiration</td>
<td>Backup data and format computer system to allow the software to be used again (this cancels the expiration date).</td>
</tr>
<tr>
<td>Changes to requirements (design rework)</td>
<td>Allow for an opening at the end of the project due date to allow for such an event.</td>
</tr>
<tr>
<td>Time required for development (underestimate)</td>
<td>Develop and document the system with the ability to add support for certain, non-crucial, components.</td>
</tr>
<tr>
<td>Rate of bug fixing (underestimate)</td>
<td>Develop and document the system with the ability to add support for certain, non-crucial, components.</td>
</tr>
<tr>
<td>Size of the software (underestimate)</td>
<td>Develop and document the system with the ability to add support for certain, non-crucial, components.</td>
</tr>
</tbody>
</table>

6.4 Resource Requirements

Table 6.6 lists the resources required to develop this project. The total budget came to $159, and was not exceeded at any point within the project’s development.
6.5 Implementation Management using Microsoft Visual Studio .NET

Table 6.6: Resource requirements

<table>
<thead>
<tr>
<th>Need</th>
<th>Resources</th>
<th>Status</th>
<th>Cost</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Workstation</td>
<td>1GHz+, 256Mb+ RAM</td>
<td>Acquired</td>
<td>N/A</td>
<td>Own workstation</td>
</tr>
<tr>
<td>Database Server</td>
<td>1GHz+, 256Mb+ RAM</td>
<td>Acquired</td>
<td>N/A</td>
<td>Same as Development Workstation</td>
</tr>
<tr>
<td>Server Operating System</td>
<td>Microsoft Windows Server 2003</td>
<td>Acquired</td>
<td>Free</td>
<td>120-day limit</td>
</tr>
<tr>
<td>Database Server</td>
<td>Microsoft SQL Server 2000 (MSDE)</td>
<td>Acquired</td>
<td>Free</td>
<td></td>
</tr>
<tr>
<td>Development Environment</td>
<td>Microsoft Visual Studio .NET 2003 Academic</td>
<td>Acquired</td>
<td>$159</td>
<td></td>
</tr>
</tbody>
</table>

6.5 Implementation Management using Microsoft Visual Studio .NET

In Section 2.4.1, Microsoft Visual Studio .NET 2003 (VS.NET) was briefly described. In this section, the ways in which VS.NET is used will be displayed.

VS.NET files are managed by a solution, which contains a number of projects. The projects are usually the different components that exist within a software project. Figure 6.1 displays the Solution Explorer that exists within VS.NET for the OLE solution.
Within the OLE solution, four projects exist:

- **Database Source Code** - Maintains a connection to the development database.
- **OLEBusinessLogic** - The main class library.
- **OLEWebApplication** - Manages the Web pages and Web components.
- **OLETestingConsoleApplication** - Used to test the main class library (OLEBusinessLogic).
- **OLEWebSetup** - The installation application.

VS.NET’s Configuration Manager allows the developer to define the compilation/build process of the solution’s projects so that project dependencies can be maintained. Figures 6.2 and 6.3 show the two configurations that exist within the OLE solution:

- **Debug** (Figure 6.2) - Optimized compilation/build for testing and debugging.
- **Release** (Figure 6.3) - Optimized compilation/build for speed in a release environment.
6.5 Implementation Management using Microsoft Visual Studio .NET

The following two sub-sections below will describe the projects that effectively managed the database and setup application.

6.5.1 Database Source Code Project

The database for SQL Server was created without the use of programming code in VS.NET (excluding stored procedures). This allowed more time to be given to the projects that required programming code.
6.5 Implementation Management using Microsoft Visual Studio .NET

Figure 6.4 details the design of a table within VS.NET. It allows the definition of primary keys, data types, and null values to the table’s fields.

Figure 6.4: Microsoft Visual Studio .NET 2003’s database table creation/alteration (for the OLE solution)

Figure 6.5 details the referential integrity (relationships) between the tables in the database. A one-to-infinity relationship is shown by a key symbol (one) and an infinite symbol (infinity).
6.5 Implementation Management using Microsoft Visual Studio .NET

6.5.2 OLEWebSetup Project

A setup application was created using VS.NET via a project named OLEWebSetup. Below in Figures 6.6, 6.7, and 6.8, the configuration of the setup application is shown. By default, a Web setup project automatically detects whether Microsoft Internet Information Services (IIS) is installed.

Figure 6.6 details the screens shown as the user proceeds through the setup process. All screens are included by default, except the Textboxes (A) screen. It has been included so that the user can provide a server name for where the SQL Server database will exist.
6.5 Implementation Management using Microsoft Visual Studio .NET

Figure 6.6: The *User Interface* configuration

Figure 6.7 details where the OLE’s files are to be installed during the setup. The *Web Application Folder* is the virtual folder created by the installation, within IIS. VS.NET solutions allow its *projects* to interact with each other, which is shown by the OLEWebApplication’s *Content Files* and *Primary Output* existing within the setup application’s file system.

Figure 6.7: The *File System* configuration

Figure 6.8 shows the registry values that will be created when the installation of the OLE occurs. It allows the OLE to retrieve information after installation.

Figure 6.8: The *Registry* configuration
Chapter 7

Conclusions

Throughout this project, a large amount of research has been performed, allowing a large portion of the ‘Online Learning Environment’ to be completed, as well as allowing the acquisition of experience in:

- **Development of an enterprise-level software application** - Designing the architecture, and implementation and deployment the software application.

- **Small-scale project management** - Documenting the requirements and design of the software application, and scheduling the tasks and managing the risks of the project.

- **Research and incorporation of different programming models and languages** - Including the .NET Framework, ASP.NET, C# .NET, SQL, XML, HTML, and CSS.

While the result of the ‘Online Learning Environment’ was not as successful as anticipated, the journey of learning new and exciting skills within the field of software engineering was invaluable.
7.1 Further Work

Benefits of this project would not occur solely from continuation of the current progress. Rather, it would benefit the projectee to re-design the entire system, learning from the ideas in this project and improving on them.

The following sub-sections will provide a more detailed scope for a future project, derived from the ‘Online Learning Environment’. It will propose a larger system perspective, not confined to students and instructors. It will also propose a system architecture with focus on the database and data retrieval.

XML Web Services

An XML Web Service acts as a bridge between applications and data via the Hyper-Text Transfer Protocol (HTTP). Under the .NET Framework, an application or Web application can access an XML Web Service via a network. This technology is used from the Internet and has the ability to provide excellent grounds to commence further research and development.

New Data Storage and Retrieval Concept

The OLE utilized two techniques in order to store data (Section 3.1.2). As mentioned before, that programming model was designed so that the two techniques could be learnt. However, the following programming model provides an easier alternative to implementation of a future project as well as providing greater efficiency.

Since an SQL database field can only hold 8000 characters using the SQL data-type, VARCHAR. An improvement is to create multiple entries for a field that exceed the character limitation. Figure 7.1 details this new concept.
The existence of XML within this new architecture will no have to be thrown away either. Unlike Microsoft SQL Server 2000, Microsoft SQL Server 2005 has native XML support. Meaning that the new programming model can use XML in order to retrieve data from the new database concept.

7.2 The Future Project

The idea for this future project comes from initial project plans. Lack of knowledge however, in Web development forced the complexity of this project to be reduced to the project described in this dissertation.

The idea for the project involves the process of merging different institution departments, such as retail stores, financial and human resource management, and, learning environment (this project). A small example of this project’s potential is described in a use case (below).

**Bookshop Use Case**

A student accesses their class Web site (learning environment site) at the start of the semester, and then clicks on a menu item called ‘Text Books’. This would displays a page that lists the books required for the course, and allows the student to add them to the Bookshop’s (retail store) shopping cart. The Bookshop’s printer can then print this order and allow employees to package the books and send them to the student.
Due to the scale of the project, an option for future work is to take a collaborative approach among students.
References


Coleridge, R. (2002), An introduction to the duwamish books sample application,
REFERENCES


REFERENCES


REFERENCES


REFERENCES


Appendix A

Project Specification
PROJECT SPECIFICATION

FOR: Timothy Ross Vriesema

TOPIC: 'Online Learning Environment' using ASP.NET

SUPERVISORS: Dr. Hong Zhou & Dr. John Leis

PROJECT AIM: This project aims to design and develop a generalised learning environment application for educational institutions that endeavours to use modern programming languages, application environments, and software project management techniques (and methodologies). The software will include features such as:

- Assignment, tutorial, and exam management; and
- Discussion forums.

PROGRAMME: Issue F, 26 August 2004

1. Research of the Microsoft .NET Framework and development languages required for the software project’s development. These languages include C# .NET, ASP.NET, XML, and SQL.

2. Research the different processes for managing software projects and select an appropriate means to do so.

3. Investigate the project requirements and generate a Requirements Analysis Document (RAD).

4. Design the software project’s overall and internal (object) architecture and generate a System Design Document (SDD) and an Object Design Document (ODD).

If time permits:

5. Develop the software application and alter the SDD and ODD documents at the discretion of the development process.

6. Produce a testing procedure and then perform the tests while generating a Test Manual.

7. Generate a User Manual for information on how to operate the software application.

AGREED:

_________________ (Student Signature) _________________ (Supervisor Signature)
___ / ___ / ___ (Date) ___ / ___ / ___ (Date)

_________________ (Supervisor Signature)
___ / ___ / ___ (Date)
Appendix B

Requirements Analysis Document
‘Online Learning Environment’
using ASP.NET

Requirements Analysis Document
Thursday, 14 October 2004

Name
Timothy Vriesema
tripix@msn.com

Student Number
0011122446

University Program
Bachelor of Engineering
(Software Engineering)
AND
Bachelor of Business
(Management and Leadership)
# TABLE OF CONTENTS

## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 PURPOSE OF THE SYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>1.2 SCOPE OF THE SYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS</td>
<td>1</td>
</tr>
<tr>
<td>1.4 OVERVIEW</td>
<td>1</td>
</tr>
<tr>
<td>CURRENT SYSTEM</td>
<td>3</td>
</tr>
<tr>
<td>2.1 OVERVIEW</td>
<td>3</td>
</tr>
<tr>
<td>2.2 FUNCTIONS AND FEATURES</td>
<td>3</td>
</tr>
<tr>
<td>2.2.1 Multi-Level Administrative Roles</td>
<td>3</td>
</tr>
<tr>
<td>2.2.2 Announcements</td>
<td>3</td>
</tr>
<tr>
<td>2.2.3 Assessments</td>
<td>3</td>
</tr>
<tr>
<td>2.2.4 Notes</td>
<td>4</td>
</tr>
<tr>
<td>2.2.5 Discussion Forum</td>
<td>4</td>
</tr>
<tr>
<td>2.2.6 Internal Mail</td>
<td>4</td>
</tr>
<tr>
<td>2.2.7 File Manager</td>
<td>4</td>
</tr>
<tr>
<td>PROPOSED SYSTEM</td>
<td>5</td>
</tr>
<tr>
<td>3.1 OVERVIEW</td>
<td>5</td>
</tr>
<tr>
<td>3.2 FUNCTIONAL REQUIREMENTS</td>
<td>5</td>
</tr>
<tr>
<td>3.2.1 Components Shared by All Roles</td>
<td>5</td>
</tr>
<tr>
<td>3.2.1.1 Login Authentication</td>
<td>5</td>
</tr>
<tr>
<td>3.2.1.2 Quick Links</td>
<td>5</td>
</tr>
<tr>
<td>3.2.1.3 Institution Announcements</td>
<td>5</td>
</tr>
<tr>
<td>3.2.1.4 Personal Information Configuration</td>
<td>5</td>
</tr>
<tr>
<td>3.2.2 Components for the Administrator Role</td>
<td>6</td>
</tr>
<tr>
<td>3.2.2.1 Recent Errors</td>
<td>6</td>
</tr>
<tr>
<td>3.2.2.2 Database Configuration</td>
<td>6</td>
</tr>
<tr>
<td>3.2.2.3 E-Mail Configuration</td>
<td>6</td>
</tr>
<tr>
<td>3.2.2.4 Institution Configuration</td>
<td>6</td>
</tr>
<tr>
<td>3.2.2.5 Terminology Configuration</td>
<td>6</td>
</tr>
<tr>
<td>3.2.2.6 Web Application Startup and Shutdown</td>
<td>6</td>
</tr>
<tr>
<td>3.2.2.7 Error Logging</td>
<td>7</td>
</tr>
<tr>
<td>3.2.2.8 Login Logging</td>
<td>7</td>
</tr>
<tr>
<td>3.2.2.9 User Management</td>
<td>7</td>
</tr>
<tr>
<td>3.2.2.10 Institution Announcements Management</td>
<td>7</td>
</tr>
<tr>
<td>3.2.3 Components Shared by Instructor and Student Roles</td>
<td>7</td>
</tr>
<tr>
<td>3.2.3.1 Assignments, Exams, and Tutorials</td>
<td>7</td>
</tr>
<tr>
<td>3.2.3.2 Assignment, Exam and Tutorial Solutions</td>
<td>7</td>
</tr>
<tr>
<td>3.2.3.3 Upcoming Important Dates</td>
<td>7</td>
</tr>
<tr>
<td>3.2.3.4 Class Announcements</td>
<td>8</td>
</tr>
<tr>
<td>3.2.3.5 Notes</td>
<td>8</td>
</tr>
<tr>
<td>3.2.3.6 Forums</td>
<td>8</td>
</tr>
<tr>
<td>3.2.4 Components for the Instructor Role</td>
<td>8</td>
</tr>
<tr>
<td>3.2.4.1 Assignment, Exam, and Tutorial Management</td>
<td>8</td>
</tr>
<tr>
<td>3.2.4.2 Class Authorisation</td>
<td>9</td>
</tr>
<tr>
<td>3.2.4.3 Class Announcements Management</td>
<td>9</td>
</tr>
<tr>
<td>3.2.4.4 Notes Management</td>
<td>9</td>
</tr>
<tr>
<td>3.2.4.5 Assignment, Exam and Tutorial Answer Viewing and Downloading</td>
<td>10</td>
</tr>
<tr>
<td>3.2.4.6 Assignment, Exam and Tutorial Submission List</td>
<td>10</td>
</tr>
<tr>
<td>3.2.5 Components for the Student Role</td>
<td>10</td>
</tr>
<tr>
<td>3.2.5.1 Forum Thread Subscriptions</td>
<td>10</td>
</tr>
<tr>
<td>3.2.5.2 Assignment, Exam and Tutorial Submission</td>
<td>10</td>
</tr>
<tr>
<td>3.3 NON-FUNCTIONAL REQUIREMENTS</td>
<td>10</td>
</tr>
</tbody>
</table>

'Online Learning Environment' using ASP.NET

Copyright © 2004 by Timothy Vriesema

Requirements Analysis Document
3.3.1 Usability.............................................................................................................. 10
3.3.2 Reliability.......................................................................................................... 11
3.3.3 Performance .................................................................................................... 11
3.3.4 Supportability .................................................................................................. 11
3.3.5 Implementation ............................................................................................... 11
3.3.6 Interface .......................................................................................................... 11
3.3.7 Operation ......................................................................................................... 11
3.4 SYSTEM MODELS................................................................................................... 11
3.4.1 Scenarios .......................................................................................................... 12
3.4.2 Use Case Models.............................................................................................. 16
  3.4.2.1 Use Case Diagram – Class Subsystem ................................................................. 17
  3.4.2.2 Use Case Diagram – Configuration Subsystem ..................................................... 18
  3.4.2.3 Use Case Diagram – Submission and Management Subsystems ...................... 19
  3.4.2.4 Use Case Diagrams – Login and Home Subsystems ......................................... 20
3.4.3 Object Model................................................................................................... 21
  3.4.3.1 Class Diagram .................................................................................................. 21
  3.4.3.2 Class – User ...................................................................................................... 21
  3.4.3.3 Abstract Class – AETAnswer .............................................................................. 22
  3.4.3.4 Inherited Class – AETInformationAnswer : AETAnswer .................................. 22
  3.4.3.5 Inherited Class – AETMultipleChoiceAnswer : AETAnswer .............................. 22
  3.4.3.6 Inherited Class – AETShortResponseAnswer : AETAnswer ............................. 22
  3.4.3.7 Abstract Class – AETQuestion ............................................................................ 22
  3.4.3.8 Inherited Class – AETInformationQuestion : AETQuestion ............................... 23
  3.4.3.9 Inherited Class – AETMultipleChoiceQuestion : AETQuestion .......................... 23
  3.4.3.10 Inherited Class – AETShortResponseQuestion : AETQuestion .......................... 23
  3.4.3.11 Class – Role .................................................................................................. 23
  3.4.3.12 Class – Thread ............................................................................................... 23
  3.4.3.13 Class – Post .................................................................................................... 24
  3.4.3.14 Abstract Class – ClassAnnouncement : Post ..................................................... 24
  3.4.3.15 Inherited Class – ClassThreadPost : Post ......................................................... 24
  3.4.3.16 Inherited Class – ClassNote : Post .................................................................. 24
  3.4.3.17 Inherited Class – InstitutionalAnnouncement : Post ........................................ 25
  3.4.3.18 Inherited Class – TemplateAnnouncement : Post ........................................... 25
3.4.4 Dynamic Models ............................................................................................. 25
  3.4.4.1 Sequence: Assignment, Exam, and Tutorial Posting and Answering ............. 25
  3.4.4.2 Sequence: Assignment, Exam, and Tutorial Marking ...................................... 26
3.4.5 User Interface ................................................................................................. 26

4 GLOSSARY................................................................................................................... 28
1 INTRODUCTION

The following subsections will provide an introduction into the document’s purpose, the scope of the proposed system, and an overview of the rest of this document.

1.1 Purpose of the System

The system that this document is generated from is the ‘Online Learning Environment’ using ASP.NET (OLE). It aims to develop a system for educational institutions and their needs. This primarily includes class management by instructors, for enrolled students.

1.2 Scope of the System

The OLE will be utilized by students of an educational institution to:

- View and submit class assignments, tutorials, and/or exams online;
- Edit personal details (i.e. residential and email addresses);
- View announcements from both instructors and the educational institution; and
- Discussing events, questions, and statements via a dynamic class forum.

Instructors will also have distinct advantages from this system. These include:

- Assignment, tutorial, and exam management (i.e. creating, altering, and deleting questions);
- Assignment, tutorial, and exam marking from online submissions from students;
- Edit personal details; and
- Interacting with students via the dynamic class forum.

1.3 Definitions, Acronyms, and Abbreviations

Before reading the Requirements Analysis Document, some abbreviations are presented below that will be used.

- **AET**: Assignment, Exam, and Tutorial
- **ADO.NET**: Microsoft Active Data Objects .NET
- **ASP.NET**: Microsoft Active Server Pages .NET
- **IIS**: Microsoft Internet Information Service 6.0
- **OLE**: ‘Online Learning Environment’ using ASP.NET

1.4 Overview

This document gives a detailed description of the system’s perspective, functions, user characteristics, constraints, and
assumptions and dependencies. Following this general description, it documents the functional and non-functional requirements within the system. This involves a detailed description of what features are expected.

Footnotes have been used throughout this document to help refer the relationships between each component and feature.
2 CURRENT SYSTEM

Within the University of Southern Queensland, a system already exists with features similar to the OLE. This system is called Vista, designed and developed by WebCT.

2.1 Overview

WebCT Vista offers a number of features that will far exceed the expectations of the OLE. This is due to budget and time constraints.

Vista has features that allow students and instructors to meet in a neutral environment.

2.2 Functions and Features

This section will point out some functions and features not provided by the OLE.

2.2.1 Multi-Level Administrative Roles

Vista has many different Administrator roles that allow different people access to certain areas of the system. Some of these roles are:
- System Administrator
- Backup Administrator
- Database Administrator

2.2.2 Announcements

Announcements allow Section Instructors to provide Students with information about the class.

2.2.3 Assessments

This feature within Vista allows Section Instructors to manage assessment items within their respective classes. These assessment items consist of questions such as:
- Multiple choice
- Fill-in blanks
- Match-up definitions
- Complete paragraphs

Section Instructors define the correct answers to these questions before they become available to Students so that the system can automatically mark Students’ answers.
2.2.4 Notes
The notes feature that exists within Vista allows Section Instructors to manage notes about the class so that Students can view the Instructors’ view and perspective on certain class modules.

2.2.5 Discussion Forum
Vista provides a discussion forum so that Section Instructors and Students can communicate about the class. Vista allows Section Instructors to delete messages that violate class rules (e.g. the answer to an assessment item). Students do not have this ability.

2.2.6 Internal Mail
Each class section has an internal mail account where e-mail is passed between Section Instructors and/or other Students. It cannot be used to communicate with outside e-mail addresses.

2.2.7 File Manager
Vista has a built-in file manager for instructors to add files in order for students to download and view.
3 PROPOSED SYSTEM

This section provides preliminary information and design for the OLE.

3.1 Overview

The OLE system’s preliminary information and design has been described below showing its functional and non-functional requirements, and its system models. Some system models include:

- Scenarios
- Use Case models
- Object models
- Dynamic models
- User Interfaces

3.2 Functional Requirements

The following sub-sections detail the different functional requirements for each user role. These roles are:

- Administrators
- Instructors
- Students

3.2.1 Components Shared by All Roles

This section details the components that will be accessed by all users and roles.

3.2.1.1 Login Authentication

All users will be able to access the Web application by the use of their issued username and password.

3.2.1.2 Quick Links

All users will have access to a list of Web links, linking to other parts of the Web application that will be used on a regular basis. The Web links for each user will be automatically chosen for them.

3.2.1.3 Institution Announcements

This allows all users to be kept up-to-date through displays of news, events, and developments that are happening within the institution.

3.2.1.4 Personal Information Configuration

All users are required to supply personal information in order for their username to exist within the Web application’s database. Users will be able to alter this information simply and quickly. Such information that users will be able to alter are:

- Residential address
- E-Mail address
• Postal address

3.2.2 Components for the Administrator Role
This section details the components that will be accessed by the Administrators within the Web application.

3.2.2.1 Recent Errors
This allows Administrators to be shown recent errors so that they can be aware of recent problems within the Web application.

3.2.2.2 Database Configuration
This page allows Administrators to set the server name or IP address that holds a Microsoft SQL Server database for the Web application to access. The will also be able to set the Web application to create the database on the database server.

3.2.2.3 E-Mail Configuration
Administrators will be able to set the e-mail server POP3 address so that e-mails can be sent.

3.2.2.4 Institution Configuration
This page will allow Administrators to personalise the Web application by specifying educational institution information that may show up throughout it. Some information that will be changeable is:
• Institution name
• Institution abbreviation
• Institution logo image

3.2.2.5 Terminology Configuration
Many educational institutions have different terminology. For example, an instructor could be a teacher, examiner, or tutor. This functional requirement allows the entire Web application to be adapted to individual institutions by dynamically altering the terminology set by Administrators. The terms that will be varied include:
• Institution
• Class
• Instructor
• Student
• Assignment
• Exam
• Tutorial

Administrators will be the only users that will view the generalised terms. All other roles will view the terminology set by Administrators.

3.2.2.6 Web Application Startup and Shutdown
This allows the Administrators to start and stop the Web application. A reason for stopping the Web application may be for database upgrades, configuration changes, or error repairs.
3.2.2.7 Error Logging
This log will show errors that have occurred within the Web application. It will display the page the error occurred on, the date and time the error occurred and the error exception or message.

3.2.2.8 Login Logging
This log will display successful and unsuccessful logins to the Web application. It will display the IP address that attempted the login, the date and time the attempt was made, along with the username that was being used.

3.2.2.9 User Management
This allows Administrators to create, delete, and alter all user information. When the Administrators create a user, that user must be assigned to a certain role. Students will then be assigned to the classes they are attending and Instructors assigned to the classes they control.

Administrators can also delete users that no longer need to exist within the database and alter a user’s role association and information.

3.2.2.10 Institution Announcements Management
This gives the Administrators an ability to manage the institution announcements1 and management features such as posting, editing, and deleting.

3.2.3 Components Shared by Instructor and Student Roles
This section details the components that will be accessed by all users within the roles Instructors and Students.

3.2.3.1 Assignments, Exams, and Tutorials
This allows Instructors and Students to view assignment, exam, and tutorial questions for their class.

3.2.3.2 Assignment, Exam and Tutorial Solutions
Instructors and Students will be able to download and view solutions to an assignment, exam, or tutorial2 question. The Instructors will upload these solutions3.

3.2.3.3 Upcoming Important Dates
This allows Instructors and Students to view a list of due dates for certain assignments, exams, and tutorials. Each entry in the list will be a link to the question.

1 See Institution Announcements, section 3.2.1.3
2 See Assignments, Exams, and Tutorials, section 3.2.3.1
3 See Assignment, Exam, and Tutorial Management, section 3.2.4.1
3.2.3.4 **Class Announcements**

This allows *Instructors* and *Students* to view news, events, and developments related to a certain class.

3.2.3.5 **Notes**

*Instructors* and *Students* will be able to download notes. The *Instructors* will upload these notes\(^4\).

3.2.3.6 **Forums**

*Instructors* and *Students* will be able to post, read, and reply to messages on certain threads within the forum. This allows all users to communicate in a neutral environment, viewed by all.

Each post to the forum cannot be edited or deleted. However, *Instructors* will be able to delete any post as it may violate collusion rules within the educational institution.

3.2.4 **Components for the Instructor Role**

This section details the components that will be accessed by the *Instructors* within the Web application.

3.2.4.1 **Assignment, Exam, and Tutorial Management**

All assignments, exams, and tutorials\(^2\) can be added, deleted, and altered. Each question will have an ‘Updated’ date at the bottom that displays the date it was last modified. *Instructors* will also be able to manually sort these questions so that they are displayed in an order they are satisfied with. Assignments, exams, and tutorials\(^2\) will also have no upper or lower limit, allowing the amount of questions to be completely flexible.

Different assignment, exam, and tutorial\(^2\) types may also exist within the Web application. The different types are:

- **Question** – Created by *Instructors*, giving *Students* a question for them to submit online or by another method;
- **Multiple Choice** – Consisting of a number questions, each displaying a series of answers which may be chosen; and
- **Short Answer** – Consisting of a number of questions and space provided under each for *Students* to input an answer.

The *Instructors* will also be able to input links and images to support the questions.

The **Multiple Choice** type varies from the other types of assignment, exam, and tutorials\(^2\). *Instructors* input the correct answers into the Web application for each of the questions; thereby allowing the Web application to assign a mark automatically after the task has been submitted.

---

\(^4\) See *Notes Management*, section 3.2.4.4
Each type will also have a mode of submission. They are:

- **None** – *Instructors* will give their own description of how to submit (i.e. hand in at a certain office); and

- **File Submission** – *Students* will upload a file via the Web application.

The *Multiple Choice* and *Short Answer* types, however, have no submission mode. These are automatically submitted. When the *File Submission* mode is used, *Instructors* will set the allowed file types so that *Students* do not submit the wrong file type.

Each assignment, exam, and tutorial\(^2\) will require a due date to be set. Alongside this, *Instructors* may set limits on the questions. These limits include:

- **Time Allow** – *Instructors* can set a date to which the question may be viewed. The question is automatically configured to be displayed to users after this date;

- **Time Between** – *Instructors* can set two date and times for the question to be accessed between and as above, the question is automatically configured to be displayed to users between these dates; and

- **Time Extend** – *Instructors* can enable or disable submissions after the due date.

*Instructors* can upload the assignment, exam, and tutorial solutions\(^5\) via the Web application as a file that may be downloaded by *Students*. *Students* can also view it as typed text. These solutions can have the option to set the *Time Allow* limit (detailed above) for the Web application to make them available for all users. The solution can be made available immediately instead of using a limit.

**3.2.4.2 Class Authorisation**

This allows *Instructors* to choose if other *Students*, not enrolled in the particular class, can visit the *Instructors’* pages. This is to prevent assignment, exam, and tutorial solutions\(^5\) becoming readily available to the public.

**3.2.4.3 Class Announcements Management**

This gives the *Instructors* an ability to manage the class announcements\(^6\). This includes posting, editing, and deleting.

**3.2.4.4 Notes Management**

This will allow *Instructors* to upload files to allow other users the ability to download and view them\(^7\). This gives the *Instructors* the ability to add, delete, and update files.

---

\(^5\) See *Assignment, Exam and Tutorial Solutions*, section 3.2.3.2

\(^6\) See *Class Announcements*, section 3.2.3.4

\(^7\) See *Notes*, section 3.2.3.5
3.2.4.5 Assignment, Exam and Tutorial Answer Viewing and Downloading

Instructors will be able to view the Short Answer type submissions within the class site and assign a mark to Students. The Multiple Choice type will not be able to be viewed because the Web application assigns a mark automatically. Instructors can download files submitted by Students and manually enter a mark via the Web application.

3.2.4.6 Assignment, Exam and Tutorial Submission List

Instructors will be able to view a list containing names of Students. This list will also contain a statement (true or false) as to whether a submission was made by a Students user and if the submission was before the due date.

3.2.5 Components for the Student Role

This section details the components that will be accessed by the Students within the Web application.

3.2.5.1 Forum Thread Subscriptions

This allows Students to subscribe to certain threads within forums so that when a message is posted, extensive searching for its location does not have to occur. Each subscription will be a link to a thread and will show the number of new postings.

3.2.5.2 Assignment, Exam and Tutorial Submission

This allows Students to submit an assignment, exam, or tutorial via the Web application. If File Submission mode has been chosen, Students will be required to supply a file with their answer on it. An error will be given if Students attempts to submit the incorrect file type.

3.3 Non-Functional Requirements

This section describes the aspects of the Web application that are not directly related to its functional behaviour. These aspects are:

- Usability
- Reliability
- Performance
- Supportability
- Implementation
- Interface
- Operation
- Packaging

3.3.1 Usability

The users of this Web application will range from all departments within an educational system and therefore suggests that most users will not be completely computer literate. This will then require the Web application to be logical, coherent, and easy-to-use.

8 See Forums, section 3.2.3.6
The system will also need to keep the standards that most popular user interfaces implement to allow users to feel comfortable while using and interacting with it.

A User Manual will be included so that Administrators, Instructors, and Students can follow instructions on how to use all features and components.

3.3.2 Reliability
This system requires a security feature such as a username and password model so that unauthorised users cannot access it.

3.3.3 Performance
This system requires an unlimited amount of concurrent users to operate it so that it can be accessed any time, every time.

3.3.4 Supportability
This section defines the supportability and maintainability of the Web application by users. It details which users are required to maintain the system.

3.3.5 Implementation
This system requires all users to utilize a Web browser. Since Web browsers are deployed on almost all operating system platforms, the Web application can be accessed anywhere.

The Web application requires a Web server that is ASP.NET compatible in order to execute and operate correctly.

3.3.6 Interface
Data within the system should be kept inside a database. This data would include information about the users of the system and their personal details.

3.3.7 Operation
This system will require users that configure many of the system’s settings. Some of these settings would affect the:

- Database
- Institution variables

3.4 System Models
This section gives scenarios and models based on the OLE software system. Only the key functions have been given due to the amount of functions the OLE provides.

9 See Database Configuration, section 3.2.2.2
10 See Institution Configuration, section 3.2.2.4
3.4.1 Scenarios

This section details the primary features within the OLE as scenarios. The scenarios consist of a name, relevant actors, and the flow of events to take place.

<table>
<thead>
<tr>
<th>Scenario Name</th>
<th>Participating Actors</th>
<th>Flow of Events</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Login Authentication</strong></td>
<td>julie:Administrator</td>
<td>1. Julie visits the OLE Web application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. The OLE prompts Julie to enter her username and password to gain access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Julie then types her username and password (where the password is hidden on the screen so others do not see it).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. The OLE then gives Julie access to the Web application.</td>
</tr>
<tr>
<td><strong>Manage Institution Announcements</strong></td>
<td>julie:Administrator</td>
<td>1. Julie visits and logs into the OLE Web application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Then, Julie accesses the Institution Announcements management.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Julie then adds/deletes/alters an announcement.</td>
</tr>
<tr>
<td><strong>View Institution Announcement</strong></td>
<td>tim:Student</td>
<td>1. Tim visits and logs into the OLE Web application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Then, Tim accesses and views the Institution Announcements.</td>
</tr>
<tr>
<td><strong>Alter Personal Information</strong></td>
<td>tim:Student</td>
<td>1. Tim visits and logs into the OLE Web application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Then, Tim accesses and alters his personal information.</td>
</tr>
<tr>
<td><strong>Configure Database</strong></td>
<td>julie:Administrator</td>
<td>1. Julie visits and logs into the OLE Web application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Then, Julie accesses and alters the database’s server name.</td>
</tr>
<tr>
<td><strong>Configure E-Mail</strong></td>
<td>julie:Administrator</td>
<td></td>
</tr>
<tr>
<td>Scenario Name</td>
<td>Participating Actors</td>
<td>Flow of Events</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Manage Roles</td>
<td>julie:Administrator</td>
<td>1. Julie visits and logs into the OLE Web application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Then, Julie accesses and alters the e-mail server name.</td>
</tr>
<tr>
<td>Configure Institution</td>
<td>julie:Administrator</td>
<td>1. Julie visits and logs into the OLE Web application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Then, Julie accesses and alters the institution name and logo.</td>
</tr>
<tr>
<td>Configure Terminology</td>
<td>julie:Administrator</td>
<td>1. Julie visits and logs into the OLE Web application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Then, Julie accesses and alters the terminology to be used within the OLE.</td>
</tr>
<tr>
<td>Web Application Running Status</td>
<td>julie:Administrator</td>
<td>1. Julie visits and logs into the OLE Web application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Then, Julie accesses and alters the OLE Web application’s running state to ‘stop’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This will stop all users from using the server-side scripts.</td>
</tr>
<tr>
<td>Error Logging</td>
<td>julie:Administrator,</td>
<td>1. Tim visits and logs into the OLE Web application and receives an error when he tries to</td>
</tr>
<tr>
<td></td>
<td>tim:Student</td>
<td>access a page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. The server then records the time, date, the username that Tim had used, and the error given.</td>
</tr>
</tbody>
</table>

'Online Learning Environment' using ASP.NET
Copyright © 2004 by Timothy Vriesema
Requirements Analysis Document
### Scenario Name: Login Logging

**Participating Actors:** julie:Administrator

**Flow of Events:**
1. Tim visits and logs into the OLE Web application.
2. The server then records the time, date, and username that Tim had used.
3. Julie then accesses the OLE and views the log.

---

### Scenario Name: Manage Assignment, Exam, and Tutorial

**Participating Actors:** john:Instructor

**Flow of Events:**
1. John visits and logs into the OLE Web application.
2. Then, John adds an assignment to his class. He types the question and posts it immediately.
3. John then uploads a file with the questions solution.

---

### Scenario Name: View Assignment, Exam, and Tutorial

**Participating Actors:** tim:Student

**Flow of Events:**
1. Tim visits and logs into the OLE Web application.
2. Then, Tim visits and views the assignment question page.

---

### Scenario Name: View Assignment, Exam, and Tutorial Solution

**Participating Actors:** tim:Student

**Flow of Events:**
1. Tim visits and logs into the OLE Web application.
2. Then, Tim visits and views the assignment question page. He then clicks and downloads the question’s solution.

---

### Scenario Name: Submit Assignment, Exam, and Tutorial Answer

**Participating Actors:** tim:Student

**Flow of Events:**
1. Tim visits and logs into the OLE Web application.
2. Then, Tim visits and views the assignment question page.
3. Tim then uploads his assignment file via the page.
<table>
<thead>
<tr>
<th>Scenario Name</th>
<th>View Assignment, Exam, and Tutorial Submission List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>john:Instructor</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. John visits and logs into the OLE Web application.  
2. Then, John adds an assignment to his class. He types the question and posts it immediately.  
3. John then uploads a file with the questions solution. |

<table>
<thead>
<tr>
<th>Scenario Name</th>
<th>View Assignment, Exam, and Tutorial Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>john:Instructor</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. John visits and logs into the OLE Web application.  
2. Then, John adds an assignment to his class. He types the question and posts it immediately.  
3. John then uploads a file with the questions solution. |

<table>
<thead>
<tr>
<th>Scenario Name</th>
<th>Manage Class Announcement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>john:Instructor</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. John visits and logs into the OLE Web application.  
2. Then, John adds an announcement to the class page.  
3. He then alters the announcement by adding some more information. |

<table>
<thead>
<tr>
<th>Scenario Name</th>
<th>View Class Announcement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>tim:Student</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. Tim visits and logs into the OLE Web application.  
2. Then, Tim visits the class page and views the announcements. |

<table>
<thead>
<tr>
<th>Scenario Name</th>
<th>Manage Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>john:Instructor</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. John visits and logs into the OLE Web application.  
2. Then, John adds a note to the class page.  
3. He then alters the note by adding some more information. |

<table>
<thead>
<tr>
<th>Scenario Name</th>
<th>View Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>tim:Student</td>
</tr>
</tbody>
</table>
Flow of Events 1. Tim visits and logs into the OLE Web application.
2. Then, Tim visits the class page and views the notes.

Scenario Name View Forum Thread
Participating Actors john:Instructor
          tim:Student
Flow of Events 1. Both, John and Tim visit and log into the OLE Web application.
2. Then, Tim and John visit the class page and view the forum.
3. John posts on the forum.
4. Tim, then replies to John’s post.

Scenario Name Class Authorisation
Participating Actors john:Instructor
Flow of Events 1. John visits and logs into the OLE Web application.
2. Then, John visits and alters the class authorisation configuration.

### 3.4.2 Use Case Models

This section generates use case diagrams from scenarios above. It involves certain actors that interact with the OLE and its components. These actors are:

- **Administrators** – The actors that administrate and manage the software system.
- **Students** – The actors that use and interact with the software system, guided by an Instructor.
- **Instructors** – The actors that use and interact with the software system, guiding more than or one Student.
3.4.2.1 Use Case Diagram – Class Subsystem

![Use Case Diagram](image)

**Figure 3.1:** The use case diagram of the Class subsystem (UML)

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>ViewClassAnnouncements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>tim:Student</td>
</tr>
<tr>
<td>Flow of Events</td>
<td>1. Tim visits the class announcements page and reads the announcement Jon has entered.</td>
</tr>
<tr>
<td>Entry Conditions</td>
<td>• The Student is logged into the OLE.</td>
</tr>
<tr>
<td>Exit Conditions</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>ViewAET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>tim:Student</td>
</tr>
<tr>
<td>Flow of Events</td>
<td>1. Tim visits the Assignments page and clicks on the first assignment. 2. The OLE then displays the questions that are associated with that assignment.</td>
</tr>
<tr>
<td>Entry Conditions</td>
<td>• The Student is logged into the OLE.</td>
</tr>
<tr>
<td>Exit Conditions</td>
<td>• Receive an acknowledgement of success or fail.</td>
</tr>
</tbody>
</table>
### Use Case Name
SubmitAETAnswer

### Participating Actors
**tim:** Student

### Flow of Events
1. Tim visits the Assignments page and clicks on the first assignment.
2. The OLE then displays the questions that are associated with that assignment.
3. He then fills out the questions and clicks ‘Submit’.

### Entry Conditions
- **The Student** is logged into the OLE.

### Exit Conditions
- Receive an acknowledgement of success or fail.

---

3.4.2.2 Use Case Diagram – Configuration Subsystem

![Use Case Diagram](image_url)

**Figure 3.2:** The use case model of the Configuration subsystem (UML)

### Use Case Name
ConfigureUserInformation

### Participating Actors
**tim:** Student

### Flow of Events
1. Tim visits the Configuration page and configures his user information.
2. He changes the current home address to his new home address.

### Entry Conditions
- **The Student** is logged into the OLE.

### Exit Conditions
- Receive an acknowledgement of success or fail.
3.4.2.3 Use Case Diagram – Submission and Management Subsystems

Figure 3.3: The use case model of the Submission and Management subsystems (UML)

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>ManageInstitutionAnnouncements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>john:Administrator</td>
</tr>
<tr>
<td>Flow of Events</td>
<td>1. Jon adds two announcements by entering information and file attachments.</td>
</tr>
<tr>
<td></td>
<td>2. He notices a problem with one and decides to delete it completely.</td>
</tr>
<tr>
<td></td>
<td>3. He then notices a minor spelling error in the other announcement so he alters its information.</td>
</tr>
<tr>
<td>Entry Conditions</td>
<td>The Administrator is logged into the OLE.</td>
</tr>
<tr>
<td>Exit Conditions</td>
<td>Receive an acknowledgement of success or fail.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>ManageClassAnnouncements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>john:Instructor</td>
</tr>
<tr>
<td>Flow of Events</td>
<td>1. Jon adds two announcements by entering information and file attachments.</td>
</tr>
<tr>
<td></td>
<td>2. He notices a problem with one and decides to delete it completely.</td>
</tr>
</tbody>
</table>

'Online Learning Environment' using ASP.NET
Copyright © 2004 by Timothy Vriesema
Requirements Analysis Document
decides to delete it completely.
3. He then notices a minor spelling error in
the other announcement so he alters its
information.

<table>
<thead>
<tr>
<th>Entry Conditions</th>
<th>Exit Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The Instructor is logged into the OLE.</td>
<td>• Receive an acknowledgement of success or fail.</td>
</tr>
</tbody>
</table>

### 3.4.2.4 Use Case Diagrams – Login and Home Subsystems

**Figure 3.4:** The use case model of the Login and Home subsystems (UML)

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Login</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating Actors</td>
<td>tim:AllRoles</td>
</tr>
<tr>
<td>Flow of Events</td>
<td>1. Tim visits the OLE and views the Login page.</td>
</tr>
<tr>
<td></td>
<td>2. He types his user ID and password.</td>
</tr>
<tr>
<td></td>
<td>3. The OLE issues an error describing his username or password is incorrect.</td>
</tr>
</tbody>
</table>
### 3.4.3 Object Model

This section will detail the classes that will be utilized within the ‘Online Learning Environment’ Web application. Section 3.4.3.1 details the class model diagram. The sections following the class model diagram describe, in detail, each class and its attributes and methods.

#### 3.4.3.1 Class Diagram

Figure 3.5: Class Diagram for the OLE Web application (UML)

#### 3.4.3.2 Class – User

This class contains properties and methods for users.

Properties (not all listed):
- `FirstName` – The user’s first name.
- `LastName` – The user’s surname.
- `ResidentAddress` – The user’s residential address.
- `PostalAddress` – The user’s postal address.
- `EmailAddress` – The user’s e-mail address.
- `PhoneNumber` – The user’s phone number.
• Password – The user’s password within the OLE.
• UserID – The user’s unique identifier.

3.4.3.3 Abstract Class – AETAnswer
This abstract class provides base methods and properties for all assignment, exam, and tutorial answers.

Properties:
• AETQID – The AET’s unique identifier.
• AETType – The type of AET.
• ClassID – The associated class ID.
• Marked – True if the answer has been marked and false if not.
• SubmittedDateTime – The date and time the user submitted their answer.
• UserID – The associated user ID.

3.4.3.4 Inherited Class – AETInformationAnswer : AETAnswer
This inherited class defines properties and methods for Information AET answers.

Properties:
• AETAFilename – The filename that contains the user’s answer.

3.4.3.5 Inherited Class – AETMultipleChoiceAnswer : AETAnswer
This inherited class defines properties and methods for Multiple Choice AET answers.

Properties:
• Answers – Structures that include the user’s answers.

3.4.3.6 Inherited Class – AETShortResponseAnswer : AETAnswer
This inherited class defines properties and methods for Short Response AET answers.

Properties:
• Answers – Structures that include the user’s answers.

3.4.3.7 Abstract Class – AETQuestion
This abstract class provides base methods and properties for all assignment, exam, and tutorial questions.

Properties:
• AETQID – The AET’s unique identifier.
• AETType – The type of AET.
• AmountOfQuestions – The amount of questions within the AET.
• ClassID
- Description – The description of the AET.
- DueDateTime – The date and time the AET is due.
- HaltSubmissionDateTime – The date and time to stop allowing answers from students.
- QuestionType – Assignment, Exam, or Tutorial?
- Title – The title of the question
- ViewMarkedAnswersDateTime – The date and time students can view their marked answers.
- ViewQuestionDateTime – The date and time students can view the question

3.4.3.8 Inherited Class – AETInformationQuestion : AETQuestion
This inherited class defines properties and methods for Information AET questions.

Properties:
- AETSFilename – The filename of the solutions.
- Questions – The questions associated with the AET.

3.4.3.9 Inherited Class – AETMultipleChoiceQuestion : AETQuestion
This inherited class defines properties and methods for Multiple Choice AET questions.

Properties:
- Questions – The questions associated with the AET.

3.4.3.10 Inherited Class – AETShortResponseQuestion : AETQuestion
This inherited class defines properties and methods for Short Response AET questions.

Properties:
- Answers – The correct answers entered by an instructor.
- Questions – The questions associated with the AET.

3.4.3.11 Class – Class
This class defines properties and methods for classes.

Properties:
- ClassID – The class’ unique string identifier.
- ClassName – The class’ name.
- DenyOtherUsers – If true, non-enrolled users cannot view the class and its functions.

3.4.3.12 Class – Role
This class defines properties and methods for roles.

Properties:
• RoleID – The role’s name (unique identifier).

3.4.3.13 Class – Thread

This class defines properties and methods for class threads.

Properties:
• ClassID – The associated class ID.
• CreatedDateTime – The date and time created.
• CreatorName – The creator’s name.
• ThreadID – The thread’s unique identifier.
• Title – The thread’s title.

3.4.3.14 Abstract Class – Post

This abstract class provides base methods and properties for all posts.

Properties:
• AttachmentFilenames – The file attachments associated with the post.
• CreatedDateTime – The date and time created.
• CreatorName – The creator’s name.
• Information – The information associated with the post.
• PostID – The post’s unique identifier.
• UpdatedDateTime – The date and time altered/updated.

3.4.3.15 Inherited Class – ClassAnnouncement : Post

This inherited class defines properties and methods for class announcements.

Properties:
• Title – The announcement’s title.
• ClassID – The associated class ID.

3.4.3.16 Inherited Class – ClassThreadPost : Post

This inherited class defines properties and methods for class thread posts.

Properties:
• ClassID – The associated class ID.
• ThreadID – The associated thread ID.
• Withhold – If true, it won’t be displayed to students.

3.4.3.17 Inherited Class – ClassNote : Post

This inherited class defines properties and methods for class notes.

Properties:
• ClassID – The associated class ID.
• Description – The description of the note.
• Title – The note’s title.

3.4.3.18 Inherited Class – InstitutionalAnnouncement : Post
This inherited class defines properties and methods for institutional announcements.

Properties:
• Title – The announcement’s title.

3.4.4 Dynamic Models
This section details dynamic models generated from the functional requirements.

3.4.4.1 Sequence: Assignment, Exam, and Tutorial Posting and Answering
Figure 3.6 shows the sequence in which an Instructor posts and assignment, exam, or tutorial and then how a Student interacts receives the question and posts an answer.

Figure 3.6: Sequence Diagram for Posting and Answering AETs (UML)

---

11 See Functional Requirements, section 3.2
3.4.4.2 Sequence: Assignment, Exam, and Tutorial Marking

Figure 3.7 shows the sequence in which an Instructor receives an AET submission list, receives the answer file, and then posts a mark for the answer. The Student then visits and receives the mark assigned to that particular answer.

![Sequence Diagram for Marking AETs (UML)](image)

3.4.5 User Interface

Figure 3.8 shows the Web page layout for the login page when a user first visits the OLE Web application. Once passed the login procedure, Figure 3.9 shows the Web page layout for the rest of the OLE Web application. These pages will consist of hyperlinks, pictures, and input forms.

Figure 3.10 shows the components that will be used within the Web page forms in order to input data. They consist of:

- **Text box** – This allows users to type text that does not require more than one line.
- **Button** – This allows users to execute an operation.
- **Combo box** – This allows users to select a choice from within a box.
- **Multi-line Text box** – This allows users to type text that requires one or more lines.

---

12 See, Figure 3.10: Components for Web Forms
• **Check box** – This allows users to select an option.
• **Radio button** – This allows users to choose one option from several options, similar to the **Comb box**, but all options are shown.
• **Tooltips** – This allows users to view information about a certain object as their mouse pointer pauses above it.

![Figure 3.8: Web layout for the OLE Web application's login page](image)

![Figure 3.9: Web layout of the OLE Web application](image)

![Figure 3.10: Components for Web Forms](image)
## 4 GLOSSARY

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Administrator</strong></td>
<td>A system user that manages the software systems.</td>
</tr>
<tr>
<td><strong>Button</strong></td>
<td>This allows users to execute an operation.</td>
</tr>
<tr>
<td><strong>Check box</strong></td>
<td>This allows users to select an option.</td>
</tr>
<tr>
<td><strong>Class</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Combo box</strong></td>
<td>This allows users to select a choice from within a box.</td>
</tr>
<tr>
<td><strong>Instructor</strong></td>
<td>An individual that operates a class within the educational institution.</td>
</tr>
<tr>
<td><strong>Multi-line text box</strong></td>
<td>This allows users to type text that requires one or more lines.</td>
</tr>
<tr>
<td><strong>Radio button</strong></td>
<td>This allows users to choose one option from several options, similar to the Comb box, but all options are shown.</td>
</tr>
<tr>
<td><strong>Role</strong></td>
<td>A name describing what a group of people do.</td>
</tr>
<tr>
<td><strong>Student</strong></td>
<td>An individual that populates a class and is lead by an Instructor.</td>
</tr>
<tr>
<td><strong>Text box</strong></td>
<td>This allows users to type text that does not require more than one line.</td>
</tr>
<tr>
<td><strong>Tooltips</strong></td>
<td>This allows users to view information about a certain object as their mouse pointer pauses above it.</td>
</tr>
<tr>
<td><strong>Web application</strong></td>
<td>An application that is used via the Internet or internal Intranet.</td>
</tr>
</tbody>
</table>
Appendix C

System Design Document
‘Online Learning Environment’
using ASP.NET

SYSTEM DESIGN DOCUMENT
Sunday, 24 October 2004

NAME
Timothy Vriesema
tripix@msn.com

STUDENT NUMBER
0011122446

UNIVERSITY PROGRAM
Bachelor of Engineering
(Software Engineering)
AND
Bachelor of Business
(Management and Leadership)
# Table of Contents

## Table of Contents

INTRODUCTION ........................................................................................................... 1  
1.1 PURPOSE OF THE SYSTEM ................................................................................. 1  
1.2 DESIGN GOALS..................................................................................................... 1  
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS ................................................. 1  
1.4 OVERVIEW ............................................................................................................ 1  
2 PROPOSED SOFTWARE ARCHITECTURE .................................................................. 2  
2.1 OVERVIEW ............................................................................................................ 2  
2.1.1 Presentation Tier ............................................................................................... 2  
2.1.2 Business Logic Tier .......................................................................................... 3  
2.1.3 Data Access Tier .............................................................................................. 3  
2.2 SUBSYSTEM DECOMPOSITION ........................................................................... 3  
2.3 HARDWARE/SOFTWARE MAPPING ...................................................................... 3  
2.4 PERSISTENT DATA MANAGEMENT .................................................................... 3  
2.4.1 Database .......................................................................................................... 4  
2.4.2 XML File Storage ............................................................................................. 5  
2.5 ACCESS CONTROL AND SECURITY ................................................................... 5  
2.5.1 ASP.NET Forms-Based Authentication ............................................................. 5  
2.5.2 Encryption ....................................................................................................... 6  
2.6 GLOBAL SOFTWARE CONTROL ....................................................................... 6  
2.7 BOUNDARY CONDITIONS .................................................................................. 6  
2.7.1 Start-up Behaviours ......................................................................................... 7  
2.7.2 Error Behaviours ............................................................................................. 7  
3 SUBSYSTEM SERVICES ............................................................................................ 8  
3.1 LOGIN SUBSYSTEM .............................................................................................. 8  
3.2 HOME SUBSYSTEM ............................................................................................. 8  
3.2.1 Institution Announcements ............................................................................. 8  
3.2.2 Quick Links ...................................................................................................... 8  
3.2.3 Upcoming Dates ............................................................................................... 8  
3.2.4 Application and Security Errors ...................................................................... 8  
3.2.5 Thread Subscriptions ...................................................................................... 8  
3.3 CLASS SUBSYSTEM ............................................................................................. 8  
3.3.1 Assignments, Exams, and Tutorials ................................................................. 9  
3.3.1.1 Submission List ............................................................................................ 9  
3.3.2 Class Announcements ..................................................................................... 9  
3.3.3 Notes ................................................................................................................ 9  
3.3.4 Forum .............................................................................................................. 9  
3.4 SUBMISSION SUBSYSTEM .................................................................................. 9  
3.5 MANAGEMENT SUBSYSTEM ............................................................................... 9  
3.5.1 Assignment, Exam, and Tutorial Management .............................................. 9  
3.5.2 Class Announcement Management ................................................................. 9  
3.5.3 Institution Announcement Management ....................................................... 10  
3.5.4 User Management ............................................................................................ 10  
3.5.5 Class Management .......................................................................................... 10  
3.6 CONFIGURATION SUBSYSTEM ........................................................................ 10  
3.6.1 Application Configuration ............................................................................... 10  
3.6.2 Institution Configuration ............................................................................... 10  
3.6.3 Class Configuration ....................................................................................... 10  
3.6.4 User Configuration ......................................................................................... 10  
3.6.5 Role Assignment Configuration ................................................................... 10  
3.6.6 Class Enrolment Configuration .................................................................. 10

'The Online Learning Environment' using ASP.NET  
Copyright © 2004 by Timothy Vriesema  
System Design Document
1 INTRODUCTION

The following sub-sections will provide an introduction to the purpose of the system and design goals towards the OLE project.

1.1 Purpose of the System

The system that this document is generated from is the ‘Online Learning Environment’ using ASP.NET (OLE). It aims to develop a system for educational institutions and their needs. This primarily includes class management by instructors, for enrolled students.

1.2 Design Goals

Throughout the design process, design goals were set in order to give a deeper purpose to the process. The OLE project is not designed to be profitable or to be sold to organisations; it is designed as a learning experience for its developers.

1.3 Definitions, Acronyms, and Abbreviations

Before reading the System Design Document, some abbreviations are presented below that will be used.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AET</td>
<td>Assignment, Exam, and Tutorial</td>
</tr>
<tr>
<td>ADO.NET</td>
<td>Microsoft Active Data Objects .NET</td>
</tr>
<tr>
<td>ASP.NET</td>
<td>Microsoft Active Server Pages .NET</td>
</tr>
<tr>
<td>IIS</td>
<td>Microsoft Internet Information Service 6.0</td>
</tr>
<tr>
<td>OLE</td>
<td>‘Online Learning Environment’ using ASP.NET</td>
</tr>
</tbody>
</table>

1.4 Overview

This document provides detail on the system design of the OLE project. It details the proposed software architecture for the project and then details the services to be provided, sorted by subsystem.
# 2 PROPOSED SOFTWARE ARCHITECTURE

This section details the overall architecture to the OLE. It gives an overview of the system, along with its subsystem decomposition, hardware and software mappings, persistent data management, access control and security, global software control, and boundary conditions.

## 2.1 Overview

The OLE has been split into a three-tier architecture. These tiers are (Figure 2.1):

- Presentation Tier
- Business Logic Tier
- Data Access Tier

![Figure 2.1: The three tiers of the ‘Online Learning Environment’](image)

### 2.1.1 Presentation Tier

This tier (subsystem) consists of two layers. They are:

- Presentation Layer
- Workflow Layer

The *Presentation Layer* consists of code that presents the data to the user. The languages used will be HTML and CSS.

The *Workflow Layer* consists of code that creates and manipulates data to fit within the Presentation Layer. The language used here is a mixture of C# .NET and ASP.NET.
2.1.2 Business Logic Tier
This tier involves a layer called the Business Logic Layer. This layer consists of classes that are used within the OLE. The language used here is C# .NET.

2.1.3 Data Access Tier
This tier consists of the Data Access Layer. It provides classes that can realise the storage, retrieval, and query of persistent objects within the Data Source. The languages used here is ADO.NET and SQL.

2.2 Subsystem Decomposition
The Presentation Tier (section 2.1.1) is the only tier that has been split into subsystems. These subsystems are:

- Login – Provides an interface for users to enter username and password information in order to access the OLE.
- Home – Provides an interface for users to see an overview of the OLE.
- Submission – Provides an interface for Administrators to add users and classes to the OLE.
- Class – Provides an interface for users to view class related materials.
- Management – Provides an interface for users to manage institution and/or class materials.
- Configuration – Provides an interface for users to configure application, institution, user, and/or class materials.

These subsystems and their services are described in section 3.

2.3 Hardware/Software Mapping
The OLE is split between two primary systems, the client, and server. The client users access the OLE via a Web browser and the server holds all data and code to manipulate the client’s responses.

2.4 Persistent Data Management
The persistent data within the OLE is stored in either or both of two ways. They are:

- Database; and/or
- XML files.

This design was chosen to allow for research in both persistent data storage methods. If the OLE were to be re-programmed, the database would be used because of its quick methods of accessing data.

Below, these two methods will be described as to how they are implemented within the OLE.
2.4.1 Database

There are three core tables within the database. They are Classes, Roles, and Users. This means that all other tables use the primary keys from within those three tables to make up their own primary keys.

From Figure 2.2, there exist eight tables. They are (with descriptions):

- **AETAnswers** – Stores the submitted date and its marked status for an AET answer.
- **AETQuestions** – Stores information about an AET question (i.e. title, description, due date, etc.).
- **Classes** – Stores information about the classes that exist.
- **ClassUsers** – Stores which users belong to which classes.
- **OtherUsers** – Stores which users are Administrators.
- **Roles** – Stores the roles that exist.
- **ThreadSubscriptions** – Stores a user’s subscription to certain threads.
- **Users** – Stores information about existing users.
2.4.2 XML File Storage

The OLE creates and manipulates many XML files. These files have to be kept in a directory on the system. This directory is unavailable to all remote users (i.e. Web clients). Figure 2.3 displays an example of the OLE directory’s structure.

![XML File Storage Diagram](image)

Figure 2.3: An example of the OLE data directory's structure

From the above example, two classes exist with IDs ENG4111, and ENG4112. The folder structure of ENG4112 is identical to ENG4111 so it has been omitted from the figure.

It can be seen that ENG4111 has six subfolders, Announcements, Assignments, Exams, Form, Notes, and Tutorials. Within each of these directories, a list of XML files exists. These XML files have incremented filenames from 1 to 99999999 (padded with 0). For example, the fifth XML file within the directory would be ‘00000005.xml’. The 125th file would be ‘00000125.xml’.

Each XML file is accompanied with a directory containing the same number as the filename. For example, ‘00000001.xml’ would have a directory with the name ‘00000001’.

Note that the above is just an example of the OLEs directory structure. Some subfolders’ method differs from one to another.

2.5 Access Control and Security

The following sections will describe the authentication methods used within the OLE.

2.5.1 ASP.NET Forms-Based Authentication

ASP.NET provides an authentication method through web page forms. This is known as Forms-Based Authentication.
The Forms-Based Authentication within ASP.NET can be implemented in conjunction with the Microsoft .NET interface, System.Security.Principal.IPrincipal. This interface is inherited by user classes so it is passed around the application.

2.5.2 Encryption
The Microsoft .NET Framework provides many different cryptography classes that can be utilised. The method that will be used for encrypting passwords is MD5.

2.6 Global Software Control
As described in section 2.1, three tiers exist within the OLE. Therefore, there must be a way for all tiers to synchronise. This is done by two methods. They are:
- Dynamic Link Libraries (DLL); and
- SQL Stored Procedures.

A SQL Stored Procedure is code that pre-caches database entries so that the data is ready for retrieval. These procedures are kept in the Data Access Tier and referenced by the Business Logic Tier via two ADO.NET classes, System.Data.SqlClient.SqlConnection and System.Data.SqlClient.SqlCommand.

The Business Logic Tier (Business Logic layer) is compiled into a DLL and is referenced by the Presentation Tier.

The Presentation Tier is a little more complex than the other two tiers. It has two layers, Presentation Layer and Workflow Layer. The Workflow Layer is a compiled into a DLL for the Presentation Layer to access.

The Presentation Layer is dynamically compiled by a Web server (ASP.NET compatible). It compiles the Presentation Layer by automatically referencing DLL files located within the application’s \bin directory.

2.7 Boundary Conditions
The OLE is a Web application that not only uses a Web server; it uses the Windows registry to access configuration values. These registry values are set upon the installation of the OLE. These registry values are held under the windows registry key, HKEY_LOCAL_MACHINE\SOFTWARE\Online Learning Environment.

The registry values consist of:
• DataDirectory – The directory in which all OLE data files will be kept (default value, C:\Program Files\Online Learning Environment).
• SqlServerName – The computer name of where the Microsoft SQL Server database exists (default name, localhost).

2.7.1 Start-up Behaviours
For the OLE to start, the Web server requires a client to access the OLE’s address. This then starts by retrieving registry values (described above) and detecting if the data directory exists. If the directory does not exist, the OLE assumes it is starting for the first time and creates the necessary directories.

If the OLE is executing for the first time, a username and password is required to logon and create new users and classes. This username is, Administrator. The user, Administrator, has the password, OnlineLearningEnvironment (case-sensitive).

2.7.2 Error Behaviours
All errors within the OLE are thrown by exceptions. These exceptions are captured and displayed in an easy to read manner to users other than Administrators. The full exception message is stored as an Application Error and shown to all Administrators when they visit the application.
3 SUBSYSTEM SERVICES

This section details the services that exist within each subsystem. Many of these services can be found in more detail under the Requirements Analysis Document.

3.1 Login Subsystem

The Login subsystem contains only one service called the Authentication service. It allows users to enter a username and password to allow access to the OLE.

3.2 Home Subsystem

The Home subsystem is where users are directed to by default after authentication by the Login subsystem. It is designed to give an overview of the other subsystems’ services. These overview services are described in the sections below.

3.2.1 Institution Announcements

All users can view these announcements. They give information on events throughout the institution.

3.2.2 Quick Links

This allows the users within the Instructors and/or Students roles to have a list of links to their enrolled classes.

3.2.3 Upcoming Dates

Users within the roles Instructors and/or Students can view their upcoming AET due dates. This allows the users to plan and schedule their semester effectively.

3.2.4 Application and Security Errors

This service is only available to users within the Administrators role. It displays security and application errors.

3.2.5 Thread Subscriptions

Users within the Students role can add a thread to a list (subscribing to it) so that they can be notified of new responses quickly and without re-navigating to the thread again.

3.3 Class Subsystem

The Class subsystem consists of services that assist a class within the institution. These services allow users to interact with class activities such as assessment items and discussion forums.
3.3.1 Assignments, Exams, and Tutorials
The AET service allows students to visit and take the assessment. Instructors can then calculate the completed assessments, issuing students their performance score.

3.3.1.1 Submission List
This service allows users within the Instructors role to view whether or not students have submitted their AET and their submission dates and times.

3.3.2 Class Announcements
The class announcements service is a periodical update on events that are current within the class. All users within a class can view these announcements.

3.3.3 Notes
This service is quite similar to the class announcements service. However, notes are information on difficult questions, frequently asked questions, or class changes.

3.3.4 Forum
The forum is where instructors and students visit and create threads based on certain categories. Within these threads, any user can place a question and await an answer from another user.

3.4 Submission Subsystem
The Submission subsystem is designed to allow access to users within the Administrators role. It consists of two services:
- Class submission; and
- User submission.

These allow Administrators to add new classes and users to the OLE.

3.5 Management Subsystem
This subsystem deals with the management side of the application. This can include institution management along with class management.

3.5.1 Assignment, Exam, and Tutorial Management
This service allows instructors to manage AETs by creating, altering, and deleting them.

3.5.2 Class Announcement Management
Instructors of a class can manage these announcements by creating, altering, and deleting them.
3.5.3 **Institution Announcement Management**
Administrators can manage these announcements by creating, altering, and deleting them.

3.5.4 **User Management**
Administrators can manage all users within the institution by creating, altering, and deleting them.

3.5.5 **Class Management**
Administrators can manage all classes within the institution by creating, altering, and deleting them.

3.6 **Configuration Subsystem**
This subsystem contains services that configure different parts of the institution and application.

3.6.1 **Application Configuration**
This service allows administrators to alter application dependent variables such as the SQL Server database.

3.6.2 **Institution Configuration**
The institution configuration service allows administrators to alter institutional terminology such as assignment, class, and tutorial.

3.6.3 **Class Configuration**
This allows instructors to alter class configurations such as whether or not to allow non-enrolled users to access particular areas.

3.6.4 **User Configuration**
The user configuration service allows all users to alter their personal information.

3.6.5 **Role Assignment Configuration**
This allows administrators to assign different roles to different users.

3.6.6 **Class Enrolment Configuration**
Administrators can enrol users to any class under the Instructors or Students roles.
Appendix D

Object Design Document
‘Online Learning Environment’
using ASP.NET

OBJECT DESIGN DOCUMENT
Monday, 25 October 2004

NAME
Timothy Vriesema
tripix@msn.com

STUDENT NUMBER
0011122446

UNIVERSITY PROGRAM
Bachelor of Engineering
(Software Engineering)
AND
Bachelor of Business
(Management and Leadership)
# TABLE OF CONTENTS

## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
</tr>
<tr>
<td>1.1</td>
<td>OBJECT DESIGN TRADE-OFFS</td>
</tr>
<tr>
<td>1.2</td>
<td>INTERFACE DOCUMENTATION GUIDELINES</td>
</tr>
<tr>
<td>1.3</td>
<td>DEFINITIONS, ACRONYMS, AND ABBREVIATIONS</td>
</tr>
<tr>
<td>1.4</td>
<td>OVERVIEW</td>
</tr>
<tr>
<td>2</td>
<td>PACKAGES</td>
</tr>
<tr>
<td>2.1</td>
<td>LOGIN SUBSYSTEM</td>
</tr>
<tr>
<td>2.2</td>
<td>HOME SUBSYSTEM</td>
</tr>
<tr>
<td>2.3</td>
<td>CLASS SUBSYSTEM</td>
</tr>
<tr>
<td>2.4</td>
<td>SUBMISSION SUBSYSTEM</td>
</tr>
<tr>
<td>2.5</td>
<td>MANAGEMENT SUBSYSTEM</td>
</tr>
<tr>
<td>2.6</td>
<td>CONFIGURATION SUBSYSTEM</td>
</tr>
<tr>
<td>2.7</td>
<td>MENU PACKAGE</td>
</tr>
<tr>
<td>3</td>
<td>CLASS INTERFACES</td>
</tr>
<tr>
<td>3.1</td>
<td>ONLINELEARNINGENVIRONMENT</td>
</tr>
<tr>
<td>3.1.1</td>
<td>AddressLimitations</td>
</tr>
<tr>
<td>3.1.2</td>
<td>AddressStructure</td>
</tr>
<tr>
<td>3.1.3</td>
<td>AETAnswer</td>
</tr>
<tr>
<td>3.1.4</td>
<td>AETQuestion</td>
</tr>
<tr>
<td>3.1.5</td>
<td>AETInformationAnswer</td>
</tr>
<tr>
<td>3.1.6</td>
<td>AETInformationQuestion</td>
</tr>
<tr>
<td>3.1.7</td>
<td>AETLimitations</td>
</tr>
<tr>
<td>3.1.8</td>
<td>AETMultipleChoiceAnswer</td>
</tr>
<tr>
<td>3.1.9</td>
<td>AETMultipleChoiceQuestion</td>
</tr>
<tr>
<td>3.1.10</td>
<td>AETShortResponseAnswer</td>
</tr>
<tr>
<td>3.1.11</td>
<td>AETShortResponseQuestion</td>
</tr>
<tr>
<td>3.1.12</td>
<td>AETType</td>
</tr>
<tr>
<td>3.1.13</td>
<td>ApplicationError</td>
</tr>
<tr>
<td>3.1.14</td>
<td>Authentication</td>
</tr>
<tr>
<td>3.1.15</td>
<td>Class</td>
</tr>
<tr>
<td>3.1.16</td>
<td>ClassAnnouncement</td>
</tr>
<tr>
<td>3.1.17</td>
<td>ClassLimitations</td>
</tr>
<tr>
<td>3.1.18</td>
<td>ClassNote</td>
</tr>
<tr>
<td>3.1.19</td>
<td>ClassThreadPost</td>
</tr>
<tr>
<td>3.1.20</td>
<td>Configuration</td>
</tr>
<tr>
<td>3.1.21</td>
<td>DataBridge</td>
</tr>
<tr>
<td>3.1.22</td>
<td>FileLimitations</td>
</tr>
<tr>
<td>3.1.23</td>
<td>InformationQuestion</td>
</tr>
<tr>
<td>3.1.24</td>
<td>InstitutionAnnouncement</td>
</tr>
<tr>
<td>3.1.25</td>
<td>MD5Encryption</td>
</tr>
<tr>
<td>3.1.26</td>
<td>MultipleChoiceAnswer</td>
</tr>
<tr>
<td>3.1.27</td>
<td>MultipleChoiceChoice</td>
</tr>
<tr>
<td>3.1.28</td>
<td>MultipleChoiceQuestion</td>
</tr>
<tr>
<td>3.1.29</td>
<td>Post</td>
</tr>
<tr>
<td>3.1.30</td>
<td>QuestionType</td>
</tr>
<tr>
<td>3.1.31</td>
<td>Role</td>
</tr>
<tr>
<td>3.1.32</td>
<td>RoleID</td>
</tr>
<tr>
<td>3.1.33</td>
<td>RoleLimitations</td>
</tr>
<tr>
<td>3.1.34</td>
<td>SecurityError</td>
</tr>
<tr>
<td>3.1.35</td>
<td>ShortAETAnswer</td>
</tr>
</tbody>
</table>
3.1.36 ShortAETQuestion
3.1.37 ShortResponseAnswer
3.1.38 ShortResponseQuestion
3.1.39 Thread
3.1.40 User
3.1.41 UserLimitations
1 INTRODUCTION

The following sub-sections will provide an introduction to the object design trade-offs and interface documentation guidelines towards the OLE project.

1.1 Object Design Trade-Offs

While designing the OLE project, particular attention was given to the persistent storage component. Since the project was undertaken as a learning activity, efficiency was sacrificed by using two storage techniques rather than one.

1.2 Interface Documentation Guidelines

Throughout the implementation of the OLE project, different naming conventions carry different components within the source code.

- C# Namespaces – OnlineLearningEnvironment
- C# Structure – OnlineLearningEnvironment
- C# Enumeration – OnlineLearningEnvironment
- C# Classes – OnlineLearningEnvironment
- C# Properties – OnlineLearningEnvironment
- C# Attribute – onlineLearningEnvironment
- XML Element – OnlineLearningEnvironment
- XML Attribute – onlineLearningEnvironment
- SQL Table – OnlineLearningEnvironment
- SQL Attribute – onlineLearningEnvironment

1.3 Definitions, Acronyms, and Abbreviations

Before reading the Test Manual, some abbreviations are presented below that will be used.

AET Assignment, Exam, and Tutorial
ADO.NET Microsoft Active Data Objects .NET
ASP.NET Microsoft Active Server Pages .NET
IIS Microsoft Internet Information Service 6.0
OLE ‘Online Learning Environment’ using ASP.NET

1.4 Overview

This document provides detail on the object design of the OLE project. Firstly it details the packages inside each subsystem and secondly it describes the class interfaces.
2 Packages

This section details the packages found inside the ‘OLE’. However, since the application is split into three tiers, only the Presentation tier contains packages.

In the following sub-sections, each subsystem will be split into their respective packages. Note that since the Presentation tier is, in essence, an ASP.NET Web application, each subsystem is an .aspx file. Packages are split into .ascx files that are included within the subsystem’s .aspx file.

2.1 Login Subsystem

The Login subsystem is contained in the file Login.aspx. The following packages are located within this subsystem:

- **Authentication** – Authenticates a user to access the OLE.
  - Authentication.ascx

2.2 Home Subsystem

The Home subsystem is contained in the file Default.aspx. The following packages are located within this subsystem:

- **Error Messages** – Displays recent error messages (Administrators only).
  - ErrorMessages.ascx
- **Forum Thread Subscriptions** – Displays a user’s subscriptions to threads.
  - ForumThreadSubscriptions.ascx
- **Institution Announcements** – Displays announcements for the institution.
  - InstitutionAnnouncements.ascx
- **Quick Links** – Displays the classes a user is enrolled in.
  - QuickLinks.ascx
- **Upcoming Dates** – Displays upcoming AET dates.
  - UpcomingDates.ascx

2.3 Class Subsystem

The Class subsystem is contained in the file Class.aspx. The following packages are located within this subsystem:

- **AET Submission List** – Displays the users who have submitted for a specific AET.
  - AETSubmissionList.ascx
- **Class Announcements** – Displays the announcements for a class.
  - ClassAnnouncements.ascx
- **Information AET Question** – Display an Information AET question for a specific class.
2.4 Submission Subsystem

The Submission subsystem is contained in the file Submission.aspx. The following packages are located within this subsystem:

- **User Submission** – Submit a new user.
  - User.ascx

- **Class Submission** – Submit a new class.
  - Class.ascx

2.5 Management Subsystem

The Management subsystem is contained in the file Management.aspx. The following packages are located within this subsystem:

- **Class Announcement Management** – Manages the class announcements.
  - ClassAnnouncementManagement.ascx

- **Class Management** – Manages the classes.
  - ClassManagement.ascx

- **Class Note Management** – Manages the class notes.
  - ClassNoteManagement.ascx

- **Information AET Management** – Manages the Information AETs for a class.
  - InformationAETManagement.ascx

- **Institution Announcement Management** – Manages the institution announcements.
  - InstitutionAnnouncementManagement.ascx

- **Multiple-Choice AET Management** – Manages the Multiple-Choice AETs for a class.
  - MultipleChoiceAETManagement.ascx

- **Short-Response AET Management** – Manages the Short-Response AETs for a class.
  - ShortResponseAETManagement.ascx

- **User Management** – Manages the users.
  - UserManagement.ascx
2.6 Configuration Subsystem

The Configuration subsystem is contained in the file Configuration.aspx. The following packages are located within this subsystem:

- **Application Configuration** – Configure the Web application.
  - ApplicationConfiguration.ascx
- **Class Configuration** – Configure a class.
  - ClassConfiguration.ascx
- **Class Enrollment Configuration** – Configure class user enrolments.
  - ClassEnrollmentConfiguration.ascx
- **Institution Configuration** – Configure the institution.
  - InstitutionConfiguration.ascx
- **Role Assignment Configuration** – Configure a user’s roles.
  - RoleAssignmentConfiguration.ascx
- **User Configuration** – Configure a user’s personal details.
  - UserConfiguration.ascx

2.7 Menu Package

The Menu package is contained in the file Menu.ascx. The following packages are located within this package:

- **Assignments** – List a user’s assignments for a specific class.
  - Assignments.ascx
- **Classes** – List a user’s classes.
  - Classes.ascx
- **Exams** – List a user’s exams for a specific class.
  - Exams.ascx
- **Notes** – List a user’s notes for a specific class.
  - Notes.ascx
- **Tutorials** – List a user’s notes for a specific class.
  - Tutorials.ascx
3 CLASS INTERFACES

This section details all classes that exist within the 'OLE', along with descriptions on constructors, destructors, attributes, properties, and methods.

3.1 OnlineLearningEnvironment

The OnlineLearningEnvironment namespace contains classes from within the Business Logic Tier. They are concerned with describing and storage of data information.

3.1.1 AddressLimitations

<table>
<thead>
<tr>
<th>Filename</th>
<th>Limitations.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>Property limitations for structure, AddressStructure.</td>
</tr>
</tbody>
</table>

Attributes

const int Street

The maximum string length of the combined street number, name, and type

const int Suburb

The maximum string length of the suburb name

const int State

The maximum string length of the state name

const int Postcode

The maximum string length of the postcode number

const int Country

The maximum string length of the country name

3.1.2 AddressStructure

<table>
<thead>
<tr>
<th>Filename</th>
<th>Address.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>The structure of a postal/residential Address.</td>
</tr>
</tbody>
</table>

Attributes

string Street

Street number and name.

string Suburb

Suburb/City.

string State

Suburb/City.

string Postcode

Postcode of the country.

string Country

Country.
3.1.3 AETAnswer

<table>
<thead>
<tr>
<th>Filename</th>
<th>AETAnswer.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Abstract Class</td>
</tr>
<tr>
<td>Description</td>
<td>An abstract class, which provides shared properties and methods for AET answers</td>
</tr>
</tbody>
</table>

**Constructors**

- AETAnswer()
  - Used for XmlSerializer purposes
- AETAnswer(DataBridge)
  - Abstract constructor

**Properties**

- int UserID
  - The UserID associated with the AETAnswer
- int AETQID
  - The ID of the AETQuestion associated with the AETAnswer
- AETType AETType
  - The AETType of the associated AETQuestion
- string ClassID
  - The ClassID associated class
- DateTime SubmittedDateTime
  - The date that the answer was submitted
- bool Marked
  - Whether or not the AETAnswer has been completely marked
- DataBridge DataBridge
  - The DataBridge that connects to the data

**Methods**

- Create()
  - Uses current instance’s properties and stores the data
- Alter()
  - Uses current instance’s properties and re-stores the data
- Delete()
  - Uses the current instance’s properties and deletes the stored data

3.1.4 AETQuestion

<table>
<thead>
<tr>
<th>Filename</th>
<th>AETQuestion.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Abstract Class</td>
</tr>
<tr>
<td>Description</td>
<td>An abstract class which provides shared properties and methods for AETQuestions.</td>
</tr>
</tbody>
</table>

**Constructors**

- AETQuestion(QuestionType)
  - Abstract constructor.
- AETQuestion(AETType, QuestionType, DataBridge)
  - Abstract constructor
Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AETQID</td>
<td>The unique ID.</td>
</tr>
<tr>
<td>AETType</td>
<td>The associated type.</td>
</tr>
<tr>
<td>ClassID</td>
<td>The associated class ID.</td>
</tr>
<tr>
<td>QuestionType</td>
<td>The associated question type.</td>
</tr>
<tr>
<td>Title</td>
<td>The associated title.</td>
</tr>
<tr>
<td>Description</td>
<td>The associated description (usually 'Assignment', 'Exam', or 'Tutorial').</td>
</tr>
<tr>
<td>DueDateTime</td>
<td>The date &amp; time the answers are due.</td>
</tr>
<tr>
<td>HaltSubmissionsDateTime</td>
<td>The date &amp; time no more answers are accepted.</td>
</tr>
<tr>
<td>ViewMarkedAnswersDateTime</td>
<td>The date &amp; time students are able to see their marked answers.</td>
</tr>
<tr>
<td>ViewQuestionDateTime</td>
<td>The date &amp; time students can view the question.</td>
</tr>
<tr>
<td>DataBridge</td>
<td>The DataBridge that connects to the data.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create()</td>
<td>Uses current instance's properties and stores the data.</td>
</tr>
<tr>
<td>Alter()</td>
<td>Uses current instance's properties and re-stores the data.</td>
</tr>
<tr>
<td>Delete()</td>
<td>Uses the current instance's properties and deletes the stored data.</td>
</tr>
</tbody>
</table>

3.1.5 AETInformationAnswer

<table>
<thead>
<tr>
<th>Filename</th>
<th>AETAnswer.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td>Description</td>
<td>An inherited class of AETAnswer for Information answers</td>
</tr>
<tr>
<td>Inherits</td>
<td>AETAnswer</td>
</tr>
</tbody>
</table>

Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AETInformationAnswer()</td>
<td>Used for XmlSerializer purposes</td>
</tr>
<tr>
<td>AETInformationAnswer(DataBridge)</td>
<td>Main constructor</td>
</tr>
</tbody>
</table>
Properties

---

**int AETAFilename**

*The filename of where the answer exists*

---

**Static Methods**

---

FindAETInformationAnswer()

*Finds an AETInformationAnswer*

---

### 3.1.6 AETInformationQuestion

**Filename**

AETQuestion.cs

**Type**

C# Sealed Class

**Description**

A sealed class that contains properties and methods of an Information Question.

**Inherits**

AETQuestion

---

### Constructors

---

AETInformationQuestion()

*Used for XmlSerializer purposes*

AETInformationQuestion(AETType, DataBridge)

*Main constructor*

---

### Properties

---

InformationQuestion[] Questions

*An array of information questions.*

string AETSFilename

*The filename of the provided solutions file.*

---

### Static Methods

---

FindAETInformationQuestion()

*Find a specific AETInformationQuestion.*

---

### 3.1.7 AETLimitations

**Filename**

Limitations.cs

**Type**

C# Structure

**Description**

Property limitations for classes, AETQuestion and AETAnswer.

---

### Attributes

---

const int Title

*The maximum string length of the title.*

const int Description

*The maximum string length of the description.*

const int AETType

*The maximum string length of the AETType.*

const int AETQuestionType

*The maximum string length of the AETQuestionType.*
3.1.8 **AETMultipleChoiceAnswer**

<table>
<thead>
<tr>
<th><strong>Filename</strong></th>
<th>AETAnswer.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>An inherited class of AETAnswer for Multiple Choice answers</td>
</tr>
<tr>
<td><strong>Inherits</strong></td>
<td>AETAnswer</td>
</tr>
</tbody>
</table>

**Constructors**

- AETMultipleChoiceAnswer()
  *Used for XmlSerializer purposes*
- AETMultipleChoiceAnswer(DataBridge)
  *Main constructor*

**Attributes**

- MultipleChoiceAnswer[] answers
  *An array of answers*

**Static Methods**

- FindAETMultipleChoiceAnswer()
  *Finds an AETMultipleChoiceAnswer*

3.1.9 **AETMultipleChoiceQuestion**

<table>
<thead>
<tr>
<th><strong>Filename</strong></th>
<th>AETQuestion.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>A sealed class that contains properties and methods of a Multiple Choice Question.</td>
</tr>
<tr>
<td><strong>Inherits</strong></td>
<td>AETQuestion</td>
</tr>
</tbody>
</table>

**Constructors**

- AETMultipleChoiceQuestion()
  *Used for XmlSerializer purposes*
- AETMultipleChoiceQuestion(AETType, DataBridge)
  *Main constructor*

**Properties**

- MultipleChoiceQuestion[] Questions
  *An array of multiple choice questions.*

**Static Methods**

- FindAETMultipleChoiceQuestion()
  *Find a specific AETMultipleChoiceQuestion.*

3.1.10 **AETShortResponseAnswer**

<table>
<thead>
<tr>
<th><strong>Filename</strong></th>
<th>AETAnswer.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>An inherited class of AETAnswer for Short Response answers</td>
</tr>
<tr>
<td><strong>Inherits</strong></td>
<td>AETAnswer</td>
</tr>
</tbody>
</table>
Constructors

AETShortResponseAnswer()
   Used for XmlSerializer purposes

AETShortResponseAnswer(DataBridge)
   Main constructor

Attributes

ShortResponseAnswer[] answers
   An array of answers

Static Methods

FindAETShortResponseAnswer()
   Finds an AETShortResponseAnswer

3.1.11 AETShortResponseQuestion

Filename  AETQuestion.cs
Type      C# Sealed Class
Description A sealed class that contains properties and methods of a Short Response Question.
Inherits  AETQuestion

Constructors

AETShortResponseQuestion()
   Used for XmlSerializer purposes

AETShortResponseQuestion(AETType, DataBridge)
   Main constructor

Properties

ShortResponseQuestion[] Questions
   An array of short response questions.

ShortResponseAnswer[] Answers
   An array of short response question's answers (entered by Instructors).

Static Methods

FindAETShortResponseQuestion()
   Find a specific AETShortResponseQuestion.

3.1.12 AETType

Filename  AETQuestion.cs
Type      C# Enumeration
Description The type of the AETQuestion.

Enumerates

Assignment
   An assignment.

Exam
   An exam.
3.1.13 **ApplicationError**

<table>
<thead>
<tr>
<th>Filename</th>
<th>ErrorAndSecurity.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>A class that holds information about application errors.</td>
</tr>
</tbody>
</table>

**Constructors**

- **ApplicationError()**
  - Used for XmlSerializer purposes.
- **ApplicationError(DataBridge)**
  - Main constructor.

**Static Methods**

- **FindApplicationError()**
  - Find a specific application error.
- **FindApplicationErrors()**
  - Find all application errors.

**Methods**

- **Create()**
  - Creates an instance of the error as an xml file.
- **Delete()**
  - Delete the specified error ID.

**Properties**

- **int ErrorID**
  - The ID of the error.
- **string Information**
  - The ID of the associated user.
- **DateTime CreatedDateTime**
  - The date and time the error was created.
- **DataBridge DataBridge**
  - The DataBridge that connects to the data.

3.1.14 **Authentication**

<table>
<thead>
<tr>
<th>Filename</th>
<th>Authentication.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>A sealed class that contains static methods to check authentication.</td>
</tr>
</tbody>
</table>

**Constructors**

- **Authentication()**
  - Disabled constructor.
- **Authentication(DataBridge)**
Main constructor

Properties

DataBridge DataBridge
The DataBridge that connects to the data.

Static Methods

CheckAuthentication()
Check username and password.

3.1.15 Class

<table>
<thead>
<tr>
<th>Filename</th>
<th>Class.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td>Description</td>
<td>A sealed class that contains methods and properties for classes.</td>
</tr>
</tbody>
</table>

Constructors

Class()
Used for XmlSerializer purposes.

Class(string, DataBridge)
Main Constructor.

Properties

string ClassID
The unique ID.

string ClassName
The associated class name.

bool DenyOtherUsers
Whether or not non-enrolled users can visit the class’ section.

Methods

Create()
Uses current instance’s properties and stores the data.

Alter()
Uses current instance’s properties and re-stores the data.

Delete()
Uses the current instance’s properties and deletes the stored data.

EnrolUser()
Enrol a user to the class.

DeEnrolUser()
DeEnrol a user to the class.

Static Methods

IsInClass()
Find all roles a user is enrolled in.

FindClass()
Find a class.

FindClasses()
Find the classes associated with a user.

### 3.1.16 ClassAnnouncement

<table>
<thead>
<tr>
<th>Filename</th>
<th>Post.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td>Description</td>
<td>A sealed class that contains properties and methods of a class announcement.</td>
</tr>
</tbody>
</table>

#### Constructors

- **ClassAnnouncement()**
  
  *Used for XmlSerializer purposes.*

- **ClassAnnouncement(string, DataBridge)**
  
  *Main Constructor.*

#### Properties

- **string ClassID**
  
  *The associated class ID.*

- **string Title**
  
  *The associated title.*

#### Static Methods

- **FindClassAnnouncement()**
  
  *Find a specific class announcement.*

- **FindClassAnnouncements()**
  
  *Finds all class announcements for a specific class.*

### 3.1.17 ClassLimitations

<table>
<thead>
<tr>
<th>Filename</th>
<th>Limitations.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>Property limitations for class, Class</td>
</tr>
</tbody>
</table>

#### Attributes

- **const int ClassID**
  
  *The maximum string length of the class ID*

- **const int ClassName**
  
  *The maximum string length of the class name*

### 3.1.18 ClassNote

<table>
<thead>
<tr>
<th>Filename</th>
<th>Post.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td>Description</td>
<td>A sealed class that contains properties and methods of a class note.</td>
</tr>
</tbody>
</table>

#### Constructors

- **ClassNote()**
  
  *Used for XmlSerializer purposes.*

- **ClassNote(string, DataBridge)**
  
  *Main Constructor.*
### Properties

<table>
<thead>
<tr>
<th>string ClassID</th>
<th>The associated class ID.</th>
</tr>
</thead>
<tbody>
<tr>
<td>string Title</td>
<td>The associated title.</td>
</tr>
<tr>
<td>string Description</td>
<td>The associated description.</td>
</tr>
</tbody>
</table>

### Static Methods

<table>
<thead>
<tr>
<th>FindClassNote()</th>
<th>Find a specific class note.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindClassNotes()</td>
<td>Finds all class notes for a specific class.</td>
</tr>
</tbody>
</table>

### 3.1.19 ClassThreadPost

<table>
<thead>
<tr>
<th>Filename</th>
<th>Post.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td>Description</td>
<td>A sealed class that contains properties and methods of a class thread post.</td>
</tr>
</tbody>
</table>

#### Constructors

<table>
<thead>
<tr>
<th>ClassThreadPost()</th>
<th>Used for XmlSerializer purposes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassThreadPost(string, int, DataBridge)</td>
<td>Main Constructor.</td>
</tr>
</tbody>
</table>

#### Properties

<table>
<thead>
<tr>
<th>string ClassID</th>
<th>The associated class ID.</th>
</tr>
</thead>
<tbody>
<tr>
<td>int ThreadID</td>
<td>The associated thread ID.</td>
</tr>
<tr>
<td>bool Withhold</td>
<td>Whether or not to show this thread post.</td>
</tr>
</tbody>
</table>

#### Static Methods

<table>
<thead>
<tr>
<th>FindClassThreadPost()</th>
<th>Find a specific class thread post.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindClassThreadPost()</td>
<td>Finds all class thread posts for a specific class.</td>
</tr>
</tbody>
</table>

### 3.1.20 Configuration

<table>
<thead>
<tr>
<th>Filename</th>
<th>Configuration.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td>Description</td>
<td>A sealed class that contains methods and properties for configuration.</td>
</tr>
</tbody>
</table>

#### Constructors

<table>
<thead>
<tr>
<th>Configuration()</th>
<th></th>
</tr>
</thead>
</table>
Properties

string InstitutionName
   The institution name.
string InstitutionNameAbbreviation
   The institution name's abbreviation.
string InstitutionLogoImage
   The institution's logo image for the web site header.
string Institution
   Institution terminology.
string Class
   Class terminology.
string Assignment
   Assignment terminology.
string Exam
   Exam terminology.
string Tutorial
   Tutorial terminology.
string Note
   Note terminology.

DataBridge DataBridge
   The DataBridge that connects to the data.

Methods

Create()
   Uses current instance's properties and stores the data.
Alter()
   Uses current instance's properties and re-stores the data.

Static Methods

FindConfiguration()
   Find configuration.

3.1.21 DataBridge

Filename  DataBridge.cs
Type      C# Sealed Class
Description  The class that provides properties and methods to connect to Xml files and an SQL Server database.

Constructors

DataBridge(SqlConnection, string, string, bool)
   Main Constructor.
DataBridge()
   Disabled Constructor.

Properties
SqlConnection SqlConnection
    The connection to an SQL Server database.

string ClassesDirectory
    The directory that holds all class critical XML files.

string InstitutionDirectory
    The directory that holds all institution critical XML files.

**Methods**

GetClassAnnouncementFilePath()
    Get class announcement's file path.

GetClassAnnouncementsDirectory()
    Get the class' announcements directory.

GetClassAssignmentAnswerFilePath()
    Get a 'new' class assignment answer's file path.

GetClassAssignmentAnswersDirectory()
    Get the class' assignment answer directory.

GetClassAssignmentFilePath()
    Get class assignment's file path.

GetClassAssignmentsDirectory()
    Get the class' assignments directory.

GetClassComponentDirectory()
    Gets a class component's directory.

GetClassDirectory()
    Get a class' files directory.

GetClassExamAnswerFilePath()
    Get a 'new' class exam answer's file path.

GetClassExamAnswersDirectory()
    Get the class' exam answer directory.

GetClassExamFilePath()
    Get class exam's file path.

GetClassExamsDirectory()
    Get the class' exams directory.

GetClassNoteFilePath()
    Get class note's file path.

GetClassNotesDirectory()
    Get the class' notes directory.

GetClassSubComponentDirectory()
    Gets a class component's sub-directory.

GetClassThreadFilePath()
    Get class thread's file path.

GetClassThreadPostFilePath()
    Get class thread post's file path.

GetClassThreadPostsDirectory()
    Get the class thread post's directory.

GetClassThreadsDirectory()
    Get the class' threads directory.

GetClassTutorialAnswerFilePath()
    Get a 'new' class tutorial answer's file path.

GetClassTutorialAnswersDirectory()
    Get the class' tutorial answer directory.

GetClassTutorialFilePath()
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetClassTutorialsDirectory()</td>
<td>Get the class' tutorials directory.</td>
</tr>
<tr>
<td>GetConfigurationFilePath()</td>
<td>Get configuration file path.</td>
</tr>
<tr>
<td>GetFilePath()</td>
<td>Retrieves a file path with a certain name for a file.</td>
</tr>
<tr>
<td>GetInstitutionAnnouncementFilePath()</td>
<td>Gets a file path within the institution announcement directory.</td>
</tr>
<tr>
<td>GetInstitutionAnnouncementsDirectory()</td>
<td>Gets the institution's announcement directory.</td>
</tr>
<tr>
<td>GetInstitutionApplicationErrorFilePath()</td>
<td>Gets a file path within the institution application errors directory.</td>
</tr>
<tr>
<td>GetInstitutionApplicationErrorsDirectory()</td>
<td>Gets the institution's application errors directory.</td>
</tr>
<tr>
<td>GetInstitutionComponentDirectory()</td>
<td>Gets an institution component's directory.</td>
</tr>
<tr>
<td>GetInstitutionSecurityErrorFilePath()</td>
<td>Gets a file path within the institution security errors directory.</td>
</tr>
<tr>
<td>GetInstitutionSecurityErrorsDirectory()</td>
<td>Gets the institution's security errors directory.</td>
</tr>
<tr>
<td>GetNextClassAnnouncementFilePath()</td>
<td>Get next file path to exist within the institution security errors directory.</td>
</tr>
<tr>
<td>GetNextClassAssignmentFilePath()</td>
<td>Get a 'new' class assignment's file path.</td>
</tr>
<tr>
<td>GetNextClassExamFilePath()</td>
<td>Get a 'new' class exam's file path.</td>
</tr>
<tr>
<td>GetNextClassNoteFilePath()</td>
<td>Get a 'new' class note's file path.</td>
</tr>
<tr>
<td>GetNextClassThreadFilePath()</td>
<td>Get a 'new' class thread's file path.</td>
</tr>
<tr>
<td>GetNextClassThreadPostFilePath()</td>
<td>Get a 'new' class thread post's file path.</td>
</tr>
<tr>
<td>GetNextClassTutorialFilePath()</td>
<td>Get a 'new' class tutorial's file path.</td>
</tr>
<tr>
<td>GetNextFilePath()</td>
<td>Retrieves a 'new' file path with a numeric number above the last in the directory (+1).</td>
</tr>
<tr>
<td>GetNextInstitutionAnnouncementFilePath()</td>
<td>Get next file path to exist within the institution announcement directory.</td>
</tr>
<tr>
<td>GetNextInstitutionApplicationErrorFilePath()</td>
<td>Get next file path to exist within the application errors directory.</td>
</tr>
<tr>
<td>GetNextInstitutionSecurityErrorFilePath()</td>
<td>Get next file path to exist within the institution security errors directory.</td>
</tr>
</tbody>
</table>
GetRootDirectory()

    Gets the root data file directory (the directory before the class and institution directory).

3.1.22 FileLimitations

<table>
<thead>
<tr>
<th>Filename</th>
<th>Limitations.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>Limitations for filenames used within the OLE.</td>
</tr>
</tbody>
</table>

Attributes

const int FileNameLength
    The character length of a filename
const char PadCharacter
    The character used to pad the filename

3.1.23 InformationQuestion

<table>
<thead>
<tr>
<th>Filename</th>
<th>AETQuestion.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>Properties that belong to an InformationQuestion.</td>
</tr>
</tbody>
</table>

Attributes

int questionID
    The question's unique ID.

string title
    The question's title.

string question
    The question's question.

3.1.24 InstitutionAnnouncement

<table>
<thead>
<tr>
<th>Filename</th>
<th>Post.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td>Description</td>
<td>A sealed class that contains properties and methods of an institution announcement.</td>
</tr>
</tbody>
</table>

Constructors

InstitutionAnnouncement()
    Used for XmlSerializer purposes.

InstitutionAnnouncement(DataBridge)
    Main Constructor.

Properties

string Title
    The associated title.

Static Methods

FindInstitutionAnnouncement()
    Find a specific institution announcement.

FindInstitutionAnnouncement()
Finds all institution announcements.

3.1.25 MD5Encryption

<table>
<thead>
<tr>
<th>Filename</th>
<th>MD5Encryption.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Class</td>
</tr>
<tr>
<td>Description</td>
<td>This class is used to encrypt strings of data to an MD5 hash string.</td>
</tr>
</tbody>
</table>

**Static Methods**

- **EncryptString()**
  - Encrypts a string to a MD5 hash in the form of a byte array.
- **EncryptStringToByteString()**
  - Encrypts a string into a readable MD5 hash string.

3.1.26 MultipleChoiceAnswer

<table>
<thead>
<tr>
<th>Filename</th>
<th>AETQuestion.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>Properties that belong to a MultipleChoiceQuestion.</td>
</tr>
</tbody>
</table>

**Attributes**

- **int questionID**
  - The QuestionID of the AETQuestion.
- **int choiceID**
  - The answer choiceID from the AETQuestion's choices.
- **bool answeredCorrectly**
  - Assigned true by the application equal to AETQuestion.Questions.correctChoice.

3.1.27 MultipleChoiceChoice

<table>
<thead>
<tr>
<th>Filename</th>
<th>AETQuestion.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>Properties that belong to a MultipleChoiceQuestion's choice.</td>
</tr>
</tbody>
</table>

**Attributes**

- **int choiceID**
  - The choice's unique ID.
- **string choiceText**
  - The choice's text.

3.1.28 MultipleChoiceQuestion

<table>
<thead>
<tr>
<th>Filename</th>
<th>AETQuestion.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>Properties that belong to an MultipleChoiceQuestion.</td>
</tr>
</tbody>
</table>

**Attributes**

- **int questionID**
  - The question's unique ID.
string title
    The question's title.
string question
    The question's question.
bool correctChoice
    True if the answer is correct, false if otherwise.
MultipleChoiceChoice[] choices
    An array of choices to accompany the question.

3.1.29 Post

<table>
<thead>
<tr>
<th>Filename</th>
<th>Post.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Abstract Class</td>
</tr>
<tr>
<td>Description</td>
<td>An abstract class which provides shared properties and methods for posts.</td>
</tr>
</tbody>
</table>

Constructors

Post()
    Used for XmlSerializer purposes.
Post(string, DataBridge)
    Abstract Constructor.

Properties

int PostID
    The unique ID.
string CreatorName
    The name of the creator.
DateTime CreatedDateTime
    The date & time created.
DateTime UpdatedDateTime
    The date & time updated.
string Information
    The associated information.
string[] AttachmentFilenames
    An array of attachment filenames.
DataBridge DataBridge
    The DataBridge that connects to the data.

Methods

Create()
    Uses current instance's properties and stores the data.
Alter()
    Uses current instance's properties and re-stores the data.
Delete()
    Uses the current instance's properties and deletes the stored data.

3.1.30 QuestionType

<table>
<thead>
<tr>
<th>Filename</th>
<th>AETQuestion.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Enumeration</td>
</tr>
</tbody>
</table>
### 3.1.31 Role

**Filename**  Role.cs  
**Type**  C# Sealed Class  
**Description**  A class which holds properties and methods that define a role.

#### Constructors

- **Role()**  
  Used for XmlSerializer purposes.
- **Role(int, DataBridge)**  
  Main constructor.

#### Static Methods

- **FindRoles()**  
  Find all roles associated with a user's ID.
- **FindRolesIDs()**  
  Find all roles associated with a user's ID.

#### Methods

- **AssignUser()**  
  Assign a specific user to a role.
- **AssignUserToClass()**  
  Assign a specific user to a role within a specific class.
- **UnassignUser()**  
  Unassign a specific user to a role.
- **UnassignUserToClass()**  
  Unassign a specific user from a role within a specific class.

#### Properties

- **string RoleID**  
  A string ID that defines a role.
- **DataBridge DataBridge**  
  The DataBridge that connects to the data.

### 3.1.32 RoleID

**Filename**  Role.cs  
**Type**  C# Enumeration  
**Description**  An enumeration which holds which roles exist within
the OLE.

**Enumerates**

**Administrators**
*The highest authenticated users within the OLE.*

**Instructors**
*Users that controls Students within the OLE.*

**Students**
*The lowest authenticated users within the OLE.*

### 3.1.33 RoleLimitations

<table>
<thead>
<tr>
<th>Filename</th>
<th>Limitations.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>C# Structure</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Property limitations for class, Role.</td>
</tr>
</tbody>
</table>

**Attributes**

const int RoleID
*The maximum string length of a role ID.*

### 3.1.34 SecurityError

<table>
<thead>
<tr>
<th>Filename</th>
<th>ErrorAndSecurity.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>A class that holds information about security errors.</td>
</tr>
</tbody>
</table>

**Constructors**

SecurityError()
*Used for XmlSerializer purposes.*

SecurityError(DataBridge)
*Main constructor.*

**Static Methods**

FindSecurityError()
*Find a specific security error.*

FindSecurityErrors()
*Find all security errors.*

**Methods**

Create()
*Creates an instance of the error as an xml file.*

Delete()
*Delete the specified error ID.*

**Properties**

int SecurityID
*The ID of the error.*

int UserID
*The ID of the associated user.*

string IPAddress
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreatedDateTime</td>
<td>The date and time the error was created.</td>
</tr>
<tr>
<td>DataBridge</td>
<td>The DataBridge that connects to the data.</td>
</tr>
</tbody>
</table>

### 3.1.35 ShortAETAnswer

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>AETAnswer.cs</td>
</tr>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>A structure similar to AETAnswer, but without the answer properties.</td>
</tr>
</tbody>
</table>

#### Static Methods

- **FindShortAETAnswer()**
  - Finds an AETInformationAnswer
- **FindShortAETAnswers()**
  - Finds an AETInformationAnswer

#### Properties

- **int UserID**
  - The UserID associated with the AETAnswer
- **int AETQID**
  - The ID of the AETQuestion associated with the AETAnswer
- **AETType**
  - The AETType of the associated AETQuestion
- **string ClassID**
  - The ClassID associated class
- **DateTime SubmittedDateTime**
  - The date that the answer was submitted
- **bool Marked**
  - Whether or not the AETAnswer has been completely marked
- **DataBridge**
  - The DataBridge that connects to the data

### 3.1.36 ShortAETQuestion

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>AETQuestion.cs</td>
</tr>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>A structure similar to AETQuestion, but without the question properties.</td>
</tr>
</tbody>
</table>

#### Static Methods

- **FindShortAETQuestion()**
  - Finds a specific ShortAETQuestion.
- **FindShortAETQuestions()**
  - Finds the properties of AETQuestions for a particular Class ID (without the questions).

#### Properties

- **int AETQID**
The unique ID.

AETType AETType
The associated type.

string AmountOfQuestions
The ClassID associated class

string ClassID
The associated class ID.

QuestionType QuestionType
The associated question type.

string Title
The associated title.

string Description
The associated description (usually 'Assignment', 'Exam', or 'Tutorial').

DateTime DueDateTime
The date & time the answers are due.

DateTime HaltSubmissionsDateTime
The date & time no more answers are accepted.

DateTime ViewMarkedAnswersDateTime
The date & time students are able to see their marked answers.

DateTime ViewQuestionDateTime
The date & time students can view the question.

DataBridge DataBridge
The DataBridge that connects to the data.

3.1.37 ShortResponseAnswer

Filename AETQuestion.cs
Type C# Structure
Description The structure of a short response answer.

Attributes

int questionID
The QuestionID of the AETQuestion.

string answer
A written answer to the question.

bool answeredCorrectly
An Instructor will set true if answer is correct and false if incorrect.

3.1.38 ShortResponseQuestion

Filename AETQuestion.cs
Type C# Structure
Description Properties that belong to a ShortResponseQuestion.

Attributes

int questionID
The question's unique ID.

string title
3.1.39 Thread

<table>
<thead>
<tr>
<th>Filename</th>
<th>Thread.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td>Description</td>
<td>A sealed class that contains methods and properties for classes.</td>
</tr>
</tbody>
</table>

**Constructors**

Thread()

Used for XmlSerializer purposes.

Thread(DataBridge)

Main Constructor.

**Properties**

<table>
<thead>
<tr>
<th>int</th>
<th>ThreadID</th>
</tr>
</thead>
<tbody>
<tr>
<td>The ID associated to the thread.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>ClassID</th>
</tr>
</thead>
<tbody>
<tr>
<td>The ID of the class associated to the thread.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>The title of the thread.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>CreatorName</th>
</tr>
</thead>
<tbody>
<tr>
<td>The thread creator's full name.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DateTime</th>
<th>CreatedDateTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>The date and time the thread was created.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DataBridge</th>
<th>DataBridge</th>
</tr>
</thead>
<tbody>
<tr>
<td>The DataBridge that connects to the data.</td>
<td></td>
</tr>
</tbody>
</table>

**Methods**

Create()

Uses current instance's properties and stores the data.

Alter()

Uses current instance's properties and re-stores the data.

Delete()

Uses the current instance's properties and deletes the stored data.

CreateThreadSubscription()

Creates a subscription to a thread for a user.

DeleteThreadSubscription()

Deletes a subscription to a thread for a user.

**Static Methods**

FindThread()

Find a specific thread that belongs within a specific class.

FindAllThreads()

Finds all threads that belong to a specific class ID.
### 3.1.40 User

<table>
<thead>
<tr>
<th>Filename</th>
<th>User.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Sealed Class</td>
</tr>
<tr>
<td>Description</td>
<td>A sealed class that contains methods and properties for classes.</td>
</tr>
<tr>
<td>Implemented</td>
<td>IPrincipal</td>
</tr>
</tbody>
</table>

#### Constructors

- **User()**
  - Used for XmlSerializer purposes.
- **User(DataBridge)**
  - Main Constructor.
- **User(IIdentity, DataBridge)**
  - Main Constructor.
- **User(int, IIdentity, DataBridge)**
  - Main Constructor.

#### Properties

- **int UserID**
  - The user's numeric ID. Also used as username.
- **string Password**
  - The user's password in MD5 encryption form.
- **SetEncryptedPassword**
  - Used to set and convert the user's password from a string to an MD5 encryption string.
- **string[] RoleIDs**
  - The user's roles.
- **string FirstName**
  - The user's first name.
- **string MiddleName**
  - The user's middle name.
- **string LastName**
  - The user's last name.
- **string FullName**
  - Gets the user's full name from the first, middle, and last name properties.
- **DateTime DOB**
  - The user's date of birth.
- **AddressStructure PostalAddress**
  - The user's postal address.
- **AddressStructure ResidentialAddress**
  - The user's residential address.
- **string PhoneNumber**
  - The user's home phone number.
- **string MobileNumber**
  - The user's mobile phone number.
- **string EmailAddress**
  - The user's email address.
- **DataBridge DataBridge**
  - The DataBridge that connects to the data.
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create()</td>
<td>Uses current instance's properties and stores the data.</td>
</tr>
<tr>
<td>Alter()</td>
<td>Uses current instance's properties and re-stores the data.</td>
</tr>
<tr>
<td>Delete()</td>
<td>Uses the current instance's properties and deletes the stored data.</td>
</tr>
</tbody>
</table>

### Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindUser()</td>
<td>Find a specific user.</td>
</tr>
<tr>
<td>FindUsers()</td>
<td>Find all users within a specific class.</td>
</tr>
</tbody>
</table>

### 3.1.41 UserLimitations

<table>
<thead>
<tr>
<th>Filename</th>
<th>Limitations.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>C# Structure</td>
</tr>
<tr>
<td>Description</td>
<td>Property limitations for class, User</td>
</tr>
</tbody>
</table>

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const int Password</td>
<td>The maximum string length of the password</td>
</tr>
<tr>
<td>const int FirstName</td>
<td>The maximum string length of the first name</td>
</tr>
<tr>
<td>const int MiddleName</td>
<td>The maximum string length of the middle name</td>
</tr>
<tr>
<td>const int LastName</td>
<td>The maximum string length of the last name</td>
</tr>
<tr>
<td>const int PhoneNumber</td>
<td>The maximum string length of the phone number</td>
</tr>
<tr>
<td>const int EmailAddress</td>
<td>The maximum string length of the email address</td>
</tr>
</tbody>
</table>
Appendix E

Test Manual
‘Online Learning Environment’
using ASP.NET

TEST MANUAL
Saturday, 16 October 2004

NAME
Timothy Vriesema
tripix@msn.com

STUDENT NUMBER
0011122446

UNIVERSITY PROGRAM
Bachelor of Engineering  
(Software Engineering)  
AND  
Bachelor of Business  
(Management and Leadership)
# Table of Contents

**TABLE OF CONTENTS**

**TABLE OF CONTENTS**

1. **Introduction** .................................................................................................................. 1
2. **Testing** ............................................................................................................................ 2
  2.1 **Unit Testing** ............................................................................................................... 5
    2.1.1 User ...................................................................................................................... 5
    2.1.2 Class .................................................................................................................... 7
    2.1.3 MD5Encryption .................................................................................................. 10
    2.1.4 Authentication .................................................................................................... 11
    2.1.5 Configuration ...................................................................................................... 11
    2.1.6 Role .................................................................................................................... 13
    2.1.7 DataBridge ......................................................................................................... 14
    2.1.8 ApplicationError .................................................................................................. 14
    2.1.9 SecurityError ...................................................................................................... 14
    2.1.10 Thread ................................................................................................................ 14
    2.1.11 AETInformationAnswer : AETAnswer : AETQuestion .................................. 14
    2.1.12 AETMultipleChoiceAnswer : AETAnswer : AETQuestion .................................. 14
    2.1.13 AETShortResponseAnswer : AETAnswer : AETQuestion .................................. 15
    2.1.14 AETInformationQuestion : AETQuestion ....................................................... 15
    2.1.15 AETMultipleChoiceQuestion : AETQuestion .................................................. 15
    2.1.16 AETShortResponseQuestion : AETQuestion .................................................... 15
    2.1.17 InstitutionAnnouncement : Post ........................................................................ 15
    2.1.18 ClassAnnouncement : Post ................................................................................ 15
    2.1.19 ClassNote : Post ................................................................................................ 15
    2.1.20 ClassThreadPost : Post ..................................................................................... 15
  2.2 **Integration Testing** .................................................................................................. 15
    2.2.1 Login Site ........................................................................................................... 15
    2.2.2 Default Site ......................................................................................................... 15
    2.2.3 Class Site ............................................................................................................ 15
    2.2.4 Configuration Site ............................................................................................... 15
    2.2.5 Management Site ............................................................................................... 15
    2.2.6 Submission Site ................................................................................................. 16

**Appendix A** ...................................................................................................................... 17

**Testing Source Code** ...................................................................................................... 17
1 INTRODUCTION

This document has been generated in order to perfect the quality of the release source code. It consists of testing that attempts to ‘break and repair’ faulty source code.

The document starts with testing the different units within the application, while re-coding any faulty portions of the code. Once all units have been tested, the units are integrated into the presentation environment, where faulty presentation or logic code is re-coded to reflect the release result of the application.
# 2 Testing

This section will test the different components within the application. All tests assume that the data from previous tests exists. For example, the first test is of the User class. When the Class class is tested, it will assume that the test data created from the User class exists.

## Test Data

The test data to be used for these tests are as follows. This data should also exist once all tests are complete.

<table>
<thead>
<tr>
<th>Type</th>
<th>Data</th>
</tr>
</thead>
</table>
| **Class 1**          | ClassID: ENG4111  
ClassID: ENG4111  
ClassName: Research Project 1  
DenyOtherUsers: true |
| **Class 2**          | ClassID: ENG4112  
ClassName: Research Project 2  
DenyOtherUsers: false |
| **Class Announcement 1** | ClassID: ENG4111  
Information: “This is the first class announcement for ENG4111.”  
FileAttachments: None  
CreatorName: Timothy Ross Vriesema  
CreatedDateTime: Now  
UpdatedDateTime: Now |
| **Class Announcement 2** | ClassID: ENG4111  
Information: “This is the second announcement for ENG4111.”  
FileAttachments: None  
CreatorName: Timothy Ross Vriesema  
CreatedDateTime: Now  
UpdatedDateTime: Now |
| **Assignment 1**     | ClassID: ENG4111  
AETType: Assignment  
QuestionType: Information  
AmountOfQuestions: 2  
Title: Colours of the Earth  
Description: Assignment 1  
DueDateTime: Now + 1month  
HaltSubmissionsDateTime: Now + 1month + 1 day  
ViewMarkedAnswersDateTime: Now + 2months  
ViewQuestionDateTime: Now |
| **Exam 1**           | ClassID: ENG4111  
AETType: Exam  
QuestionType: Multiple Choice  
AmountOfQuestions: 2 |
Title: Colours of Animals  
Description: Exam 1  
DueDateTime: Now + 1 month  
HaltSubmissionsDateTime: Now + 1 month + 1 day  
ViewMarkedAnswersDateTime: Now + 2 months  
ViewQuestionDateTime: Now

Exam 2  
ClassID: ENG4112  
AETType: Exam  
QuestionType: Multiple Choice  
AmountOfQuestions: 2  
Title: Definitions  
Description: Exam 1  
DueDateTime: Now + 1 month  
HaltSubmissionsDateTime: Now + 1 month + 1 day  
ViewMarkedAnswersDateTime: Now + 2 months  
ViewQuestionDateTime: Now

Tutorial 1  
ClassID: ENG4111  
AETType: Tutorial  
QuestionType: Short Response  
AmountOfQuestions: 2  
Title: Colours of Materials  
Description: Tutorial 1  
DueDateTime: Now + 1 month  
HaltSubmissionsDateTime: Now + 1 month + 1 day  
ViewMarkedAnswersDateTime: Now + 2 months  
ViewQuestionDateTime: Now

User 1  
FirstName: Timothy  
MiddleName: Ross  
LastName: Vriesema  
EmailAddress: tim@home.com  
DOB: 20/05/1983  
Password: tim  
PhoneNumber: 1234567890  
MobileNumber: 1234567890  
Street: 1 Main Street  
Suburb: ASuburb  
State: SomeState  
Postcode: 1000  
Country: Australia

User 2  
FirstName: Jonathan  
LastName: Trinder  
EmailAddress: jon@home.com  
DOB: 18/05/1983  
Password: jon  
PhoneNumber: 0987654321  
MobileNumber: 0987654321  
Street: 50 Main Street  
Suburb: ASuburb  
State: SomeState  
Postcode: 1000
Extensive testing of this tier has already been accomplished throughout the Business Logic tier. The SQL Server stored procedures were used within each class. This is mapped out in the Table 2.1 below.

<table>
<thead>
<tr>
<th>Stored Procedure</th>
<th>Associated Business Logic Tier Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlterAETAnswer</td>
<td>AETAnswer.Alter()</td>
</tr>
<tr>
<td>AlterAETQuestion</td>
<td>AETQuestion.Alter()</td>
</tr>
<tr>
<td>AlterClass</td>
<td>Class.Alter()</td>
</tr>
<tr>
<td>AlterUser</td>
<td>User.Alter()</td>
</tr>
<tr>
<td>AssignUser</td>
<td>Role.AssignUser()</td>
</tr>
<tr>
<td>AssignUserToClass</td>
<td>Role.AssignUserToClass()</td>
</tr>
<tr>
<td></td>
<td>Class.EnrolUser()</td>
</tr>
<tr>
<td>Authentication</td>
<td>Authentication.CheckAuthentication()</td>
</tr>
<tr>
<td>CreateAETAnswer</td>
<td>AETAnswer.Create()</td>
</tr>
<tr>
<td>CreateAETQuestion</td>
<td>AETQuestion.Create()</td>
</tr>
<tr>
<td>CreateClass</td>
<td>Class.Create()</td>
</tr>
<tr>
<td>CreateThreadSub</td>
<td>Thread.Create()</td>
</tr>
<tr>
<td>CreateUser</td>
<td>User.Create()</td>
</tr>
<tr>
<td>UnassignUser</td>
<td>Role.UnassignUser()</td>
</tr>
<tr>
<td>UnassignUserToClass</td>
<td>Role.UnassignUserToClass()</td>
</tr>
<tr>
<td>DeleteAETAnswer</td>
<td>AETAnswer.Delete()</td>
</tr>
<tr>
<td>DeleteAETQuestion</td>
<td>AETQuestion.Delete()</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>Class.Delete()</td>
</tr>
<tr>
<td>DeleteThreadSub</td>
<td>Thread.Delete()</td>
</tr>
<tr>
<td>DeleteUser</td>
<td>User.Delete()</td>
</tr>
<tr>
<td>FindClass</td>
<td>Class.FindClass()</td>
</tr>
<tr>
<td>FindClasses</td>
<td>Class.FindClasses()</td>
</tr>
<tr>
<td>FindRolesIDs</td>
<td>Role.FindRolesIDs()</td>
</tr>
<tr>
<td>FindShortAETAnswer</td>
<td>ShortAETAnswer.FindShortAETAnswer()</td>
</tr>
<tr>
<td>FindShortAETAnswers</td>
<td>ShortAETAnswer.FindShortAETAnswers()</td>
</tr>
<tr>
<td>FindShortAETQuestion</td>
<td>ShortAETQuestion.FindShortAETQuestion()</td>
</tr>
<tr>
<td>FindShortAETQuestions</td>
<td>ShortAETQuestion.FindShortAETQuestions()</td>
</tr>
<tr>
<td>FindUser</td>
<td>User.FindUser()</td>
</tr>
<tr>
<td>FindUsers</td>
<td>User.FindUsers()</td>
</tr>
<tr>
<td>IsInClass</td>
<td>Class.IsInClass()</td>
</tr>
</tbody>
</table>
2.1 Unit Testing

This section is designed to test each individual class as a unit, rather than an integrated system. This is to 'iron' out the majority of the programming faults that may exist before integration testing.

If unit testing is not completed prior to the integration testing, faults may be more difficult to locate.

Each sub-section is split into three sections. They are:
- Specification – Provides criteria that must succeed in order for the test to be successful.
- Description – Provides information on how the test is carried out.
- Analysis – Provides an analysis of the test’s output, from both the console application and persistent data.

2.1.1 User

This class holds a user’s information. Within this test, two users will be added to the database, while one will be altered once it has been entered.

**Specification**

In order for the test on this class to pass, there should be:
1. No errors when using methods:
   a. Create();
   b. Alter();
   c. Delete();
   d. FindUser();
2. User 2’s mobile number should change from 1 to 0987654321.
3. User 2’s duplicate should be deleted.
4. An exception should be thrown when User 2’s duplicate cannot be found.
5. User1 and User 2 must have different (but incremental) user IDs.

**Description**

The data used for this test is User 1 and User 2.

Below is the source code that replaces the //**** Test code is entered in here ****// comment, listed under Appendix A.

```csharp
User user1 = new User(db);
user1.FirstName = "Timothy";
user1.MiddleName = "Ross";
user1.LastName = "Vriesema";
user1.MobileNumber = "1234567890";
user1.PhoneNumber = "1234567890";
user1.EmailAddress = "tim@home.com";
user1.DOB = new DateTime(1983, 5, 20);
```

user1.SetEncryptedPassword = "tim";
AddressStructure a1 = new AddressStructure();
a1.Street = "1 Main Street";
a1.Suburb = "ASuburb";
a1.State = "SomeState";
a1.Postcode = "1000";
a1.Country = "Australia";
user1.PostalAddress = user1.ResidentAddress = a1;
int userID1 = user1.Create(); // Add user

User user2 = new User(db);
user2.FirstName = "Jonathan";
user2.MiddleName = "";
user2.LastName = "Trinder";
user2.MobileNumber = "1"; // should be 0987654321
user2.PhoneNumber = "0987654321";
user2.EmailAddress = "jon@home.com";
user2.DOB = new DateTime(1983, 5, 18);
user2.SetEncryptedPassword = "jon";
AddressStructure a2 = new AddressStructure();
a2.Street = "1 Main Street";
a2.Suburb = "ASuburb";
a2.State = "SomeState";
a2.Postcode = "1000";
a2.Country = "Australia";
user2.PostalAddress = user2.ResidentAddress = a2;
int userID2 = user2.Create(); // Add user

int userID3 = user2.Create(); // Create a duplicate
User duplicateUser = User.FindUser(userID3, db);
duplicateUser.Delete(); // Delete duplicate
try
{
    User checkUser = User.FindUser(userID3, db);
}
catch // Exception thrown, userID not found
{
    Console.WriteLine("User not found!\n");
}

User alterUser = User.FindUser(userID2, db);
alterUser.MobileNumber = "0987654321";
alterUser.Alter(); // Alter the user's data

User[] foundUsers = new User[2];
foundUsers[0] = User.FindUser(userID1, db);
foundUsers[1] = User.FindUser(userID2, db);
foreach(User u in foundUsers)
{
    Console.WriteLine(u.UserID + " : " + u.FullName + ";" + u.MobileNumber);
}

**Analysis**

Below is the screen output from the following test.

User not found!

25 : Timothy Ross Vriesema;1234567890
26 : Jonathan Trinder;0987654321
It shows that User 1 was given the user ID of 25 and User 2 was given the ID of 26 (different but incremental, specification 5). User 2’s mobile number was altered (specification 2).

When the duplicate user was deleted (specification 3), a message was displayed when it could not be found (specification 4).

No errors were found while invoking methods listed under specification 1a through to 1d.

<table>
<thead>
<tr>
<th>Table 2.2: Data within the database table User after the test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SQL Server Data – User table</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>Timothy Ross Vriesema</td>
</tr>
<tr>
<td>1234567890</td>
</tr>
<tr>
<td>26</td>
</tr>
<tr>
<td>Jonathan Trinder</td>
</tr>
<tr>
<td>0987654321</td>
</tr>
</tbody>
</table>

### 2.1.2 Class

This class holds details of a certain institution class. Within this test, two classes will be added to the database, while a user is enrolled in both classes.

**Specification**

In order for the test on this class to pass, there should be:

1. **No errors when using methods:**
   a. Create();
   b. Alter();
   c. Delete();
   d. FindClass();
   e. FindClasses();
   f. EnrolUser();
   g. DeEnrolUser();
2. Class 2’s name should change from “Research 2” to “Research Project 2”.
3. Duplication of an existing class ID should be denied and an exception should be thrown.
4. An exception should be thrown when a class ID cannot be found.
5. Enrolment of User 1 and User 2 in the two classes has no problems.
Description
The data used for this test is Class 1 and Class 2.

Below is the source code that replaces the //**** Test code is entered in here ****// comment, listed under Appendix A.

```csharp
Class class1 = new Class("ENG4111", db);
class1.ClassName = "Research Project 1";
class1.DenyOtherUsers = true;
string classID1 = class1.Create(); // Create class

class1.EnrolUser(25, RoleID.Instructors); // Enrol instructor
class1.DeEnrolUser(25, RoleID.Instructors); // DeEnrol instructor

class1.EnrolUser(26, RoleID.Students); // Enrol student

Class class2 = new Class("ENG4112", db);
class2.ClassName = "Research 2"; // Left out "Project"
class2.DenyOtherUsers = false;
string classID2 = class2.Create(); // Create class

try
{
    class2.Create(); // Duplicate class
}
catch (Exception e) // Thrown exception, class id found
{
    Console.WriteLine(e.Message);
}

class2.EnrolUser(26, RoleID.Students); // Enrol student

Class alterClass = Class.FindClass(classID2, db);
alterClass.ClassName = "Research Project 2";
alterClass.Alter(); // Alter class name

try
{
    Class foundClass = Class.FindClass("FAKE", db);
}
catch // Thrown exception when class not found
{
    Console.WriteLine("Class not found!");
}

// Entered and altered classes
Console.WriteLine("Entered and altered classes:");
Class[] foundClasses = new Class[2];
foundClasses[0] = Class.FindClass(classID1, db);
foundClasses[1] = Class.FindClass(classID2, db);
foreach (Class c in foundClasses)
{
    Console.WriteLine(c.ClassID + " : " + c.ClassName);
}

// User 26's classes
Console.WriteLine("User 26's classes:");
Class[] usersClasses = Class.FindClasses(26, db);
foreach (Class c in usersClasses)
{
    Console.WriteLine(c.ClassID + " : " + c.ClassName);
}
```

Analysis

Below is the screen output from the following test.

ClassID exists!

Class not found!

Entered and altered classes:
ENG4111 : Research Project 1
ENG4112 : Research Project 2

User 26's classes:
ENG4111 : Research Project 1
ENG4112 : Research Project 2

The two classes (1 & 2) were added while Class 2's name was altered from “Research 2” to “Research Project 2” (specification 2). When an attempt to duplicate Class 2 was made, an exception was thrown (specification 3).

When a fake class ID was asked to be found, an exception was thrown, satisfying specification 4.

User 1 was enrolled as an Instructor to Class 1 and User 2 was enrolled as a Student under both Class 1 and Class 2 (specification 5).

No errors were found while invoking methods listed under specification 1a through to 1g.

Table 2.3: Data within the database table Classes after the test

<table>
<thead>
<tr>
<th>SQL Server Data – Classes table</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG4111  Research Project 1  1</td>
</tr>
<tr>
<td>ENG4112  Research Project 2  0</td>
</tr>
</tbody>
</table>

Table 2.4: Data within the database table ClassUsers after the test

<table>
<thead>
<tr>
<th>SQL Server Data – ClassUsers table</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG4111   26  Students</td>
</tr>
<tr>
<td>ENG4112   26  Students</td>
</tr>
</tbody>
</table>

Table 2.5: Directories found within the \Classes directory after the test

<table>
<thead>
<tr>
<th>OLE Data Directory – \Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ENG4111</td>
</tr>
<tr>
<td>\ENG4111\Announcements</td>
</tr>
<tr>
<td>\ENG4111\Assignments</td>
</tr>
<tr>
<td>\ENG4111\Exams</td>
</tr>
<tr>
<td>\ENG4111\Forum</td>
</tr>
<tr>
<td>\ENG4111\Notes</td>
</tr>
<tr>
<td>\ENG4111\Tutorials</td>
</tr>
<tr>
<td>ENG4112</td>
</tr>
<tr>
<td>\ENG4112\Announcements</td>
</tr>
</tbody>
</table>
2.1.3 MD5Encryption

This class holds a static method that allows a password to be converted to an encrypted string. Within this test, three strings (two identical and one different) will be compared against each other.

**Specification**

In order for the test on this class to pass, there should be:

1. No errors when using methods:
   a. `EncryptStringToByteString();`

2. Three encrypted strings that will be tested against each other. They are:
   a. `Password`
   b. `Password`
   c. `password`

3. Case-sensitivity.

**Description**

Below is the source code that replaces the `//**** Test code is entered in here ****//` comment, listed under Appendix A.

```csharp
string[] e = new string[3];
e[0] = MD5Encryption.EncryptStringToByteString("Password");
e[1] = MD5Encryption.EncryptStringToByteString("Password");
e[2] = MD5Encryption.EncryptStringToByteString("password");
foreach (string s in e)
{
    Console.WriteLine(s);
}
Console.WriteLine("e[0] vs e[1] : " +
(e[0].CompareTo(e[1]) == 0 ? "Equal" : "Not equal");
Console.WriteLine("e[0] vs e[2] : " +
(e[0].CompareTo(e[2]) == 0 ? "Equal" : "Not equal");
```

**Analysis**

Below is the screen output from the following test.

```
5F-4D-CC-3B-5A-A7-65-D6-1D-83-27-DE-B8-82-CF-99
e[0] vs e[1] : Equal
e[0] vs e[2] : Not equal
```

The encrypted strings were identical for the first and second, but not for the third (specification 2). This concludes that there is case-sensitivity (specification 3).
No errors were found while invoking methods listed under specification 1a.

### 2.1.4 Authentication

**Specification**

In order for the test on this class to pass, there should be:

1. No errors when using methods:
   a. CheckAuthentication();
2. The password for User 1 matches the password that exists in the database.

**Description**

The data used for this test is User 1.

Below is the source code that replaces the `//**** Test code is entered in here ****//` comment, listed under Appendix A.

```csharp
// Correct authentication
bool a = Authentication.CheckAuthentication(25, "tim", db);
// Incorrect authentication
bool b = Authentication.CheckAuthentication(25, "somefakepassword", db);

Console.WriteLine("\"tim\" : " + (a ? "Correct" : "Incorrect"));
Console.WriteLine("\"somefakepassword\" : " + (b ? "Correct" : "Incorrect"));
```

**Analysis**

Below is the screen output from the following test.

"tim" : Correct
"somefakepassword" : Incorrect

The password, “tim”, is the same as the password shown under User 1’s data (specification 2).

No errors were found while invoking methods listed under specification 1a.

### 2.1.5 Configuration

This class stores information about the application and institution’s configuration, such as terminology values and SQL Server names. Within this test, a configuration file will be created and its institution name will be altered.
**Specification**

In order for the test on this class to pass, there should be:

1. No errors when using methods:
   - a. Alter();
   - b. Create();
   - c. FindConfiguration();

2. When created without any properties set, no error occurs and default values are created.

3. The XML file containing the data is created under the Data Directory.

**Description**

Below is the source code that replaces the //**** Test code is entered in here ****// comment, listed under Appendix A.

```csharp
Configuration config1 = new Configuration(db);
Console.WriteLine("Default: " + config1.InstitutionName); 
config1.Create(); // Create data

// Retrieve data
Configuration config2 = Configuration.FindConfiguration(db);
config2.InstitutionName = "New Institution Name"; 
config2.Alter();

// Re-retrieve data
config2 = Configuration.FindConfiguration(db);
Console.WriteLine("New: " + config2.InstitutionName);
```

**Analysis**

Below is the screen output from the following test.

Default: Online Learning Environment
New: New Institution Name

When the file was created, default values were entered without an error (specification 2). This file was also created under the data directory (specification 3).

No errors were found while invoking methods listed under specification 1a through to 1c.

Below displays the new data within the XML file.

```xml
<?xml version="1.0" encoding="utf-8"?>
    <InstitutionName>New Institution Name</InstitutionName>
    <InstitutionNameAbbreviation>OLE</InstitutionNameAbbreviation>
    <InstitutionLogoImage>Images\header.jpg</InstitutionLogoImage>
</Configuration>
```
2.1.6 Role

This class holds the name of a role. Within this test, users will be added to different roles and then have the roles revoked.

Specification

In order for the test on this class to pass, there should be:

1. No errors when using methods:
   a. AssignUserToClass();
   b. UnassignUserToClass();
   c. AssignUser();
   d. UnassignUser();
   e. FindRoles();
   f. FindRoleIDs();

2. User 1 should be an Instructor of Class 1 and an Administrator.
3. User 1’s roles should be retrieved.
4. The roles of User 1 should be taken away.

Description

The data used for this test is User 1 and Class 1.

Below is the source code that replaces the //**** Test code is entered in here ****// comment, listed under Appendix A.

```csharp
Role role1 = new Role(RoleID.Administrators, db); 
role1.AssignUser(25); // Add user to role

Role role2 = new Role(RoleID.Instructors, db); 
role2.AssignUserToClass(25, "ENG4111");

// Get RoleIDs
Console.WriteLine("25's RoleIDs:");
RoleID[] roleIDs1 = Role.FindRolesIDs(25, db);
foreach (RoleID r in roleIDs1) 
{ 
    Console.WriteLine(r.ToString());
}

// Get Roles
Console.WriteLine("25's Roles:");
Role[] roles1 = Role.FindRoles(25, db);
foreach (Role r in roles1) 
{ 
    Console.WriteLine(r.RoleID.ToString());
}

role1.UnassignUser(25);
```
User 1 was put in the role of Instructor for Class 1 and an Administrator (specification 2). The roles User 1 was in were retrieved and displayed to the screen (specification 3) and then the two roles were taken away (specification 4).

No errors were found while invoking methods listed under specification 1a through to 1f.

Table 2.6: Data within the database table ClassUsers after the test

<table>
<thead>
<tr>
<th>SQL Server Data – ClassUsers table</th>
</tr>
</thead>
</table>

Table 2.7: Data within the database table OtherUsers after the test

<table>
<thead>
<tr>
<th>SQL Server Data – OtherUsers table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Administrators</td>
</tr>
</tbody>
</table>

2.1.7 **DataBridge**
Untested.

2.1.8 **ApplicationError**
Untested.

2.1.9 **SecurityError**
Untested.

2.1.10 **Thread**
Untested.

2.1.11 **AETInformationAnswer : AETAnswer**
Untested.

2.1.12 **AETMultipleChoiceAnswer : AETAnswer**
Untested.
2.1.13 AETShortResponseAnswer : AETAnswer
Untested.

2.1.14 AETInformationQuestion : AETQuestion
Untested.

2.1.15 AETMultipleChoiceQuestion : AETQuestion
Untested.

2.1.16 AETShortResponseQuestion : AETQuestion
Untested.

2.1.17 InstitutionAnnouncement : Post
Untested.

2.1.18 ClassAnnouncement : Post
Untested.

2.1.19 ClassNote : Post
Untested.

2.1.20 ClassThreadPost : Post
Untested.

2.2 Integration Testing
This section will integrate the classes into a working environment in order to locate presentation and programming faults.

Each sub-section will be presented in its release code structure in order to perfect the end result.

2.2.1 Login Site
Untested.

2.2.2 Default Site
Untested.

2.2.3 Class Site
Untested.

2.2.4 Configuration Site
Untested.

2.2.5 Management Site
Untested.
2.2.6 Submission Site

Untested.
using OnlineLearningEnvironment;
using System;

namespace OLETestingConsoleApplication
{
    class OLETest
    {
        static void Main(string[] args)
        {
            // Standard Connection string to the SQL Server database
            string conn = "server=localhost;" +
                "Trusted_Connection=true;database=oledb";
            // The shared DataBridge
            // Folders already exist
            DataBridge db = new DataBridge(
                new System.Data.SqlClient.SqlConnection(conn),
                @"C:\Online Learning Environment\Classes",
                @"C:\Online Learning Environment\Institution",
                true
            );

            try
            {
                //**** Test code is entered in here ****/
            }
            catch (Exception e)
            {
                // All error data is written to screen
                Console.WriteLine(e.ToString());
            }
        }
    }
}
Appendix F

Web Application Model
Appendix G

Project Gannt Chart
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Documentation</td>
<td>96 days</td>
<td>Mon 1/03/04</td>
</tr>
<tr>
<td>2</td>
<td>Project Specification</td>
<td>5 hrs</td>
<td>Mon 1/03/04</td>
</tr>
<tr>
<td>3</td>
<td>Project Appreciation</td>
<td>50 hrs</td>
<td>Mon 1/03/04</td>
</tr>
<tr>
<td>4</td>
<td>Requirements Analysis Document (RAD)</td>
<td>30 hrs</td>
<td>Wed 10/03/04</td>
</tr>
<tr>
<td>5</td>
<td>System Design Document (SDD)</td>
<td>9 days</td>
<td>Mon 22/03/04</td>
</tr>
<tr>
<td>6</td>
<td>Object Design Document (SDD)</td>
<td>9 days</td>
<td>Fri 28/05/04</td>
</tr>
<tr>
<td>7</td>
<td>Project Dissertation</td>
<td>5 wks</td>
<td>Mon 6/09/04</td>
</tr>
<tr>
<td>8</td>
<td>Test Manual</td>
<td>8 days</td>
<td>Fri 27/08/04</td>
</tr>
<tr>
<td>9</td>
<td>User Manual</td>
<td>10 hrs</td>
<td>Wed 15/09/04</td>
</tr>
<tr>
<td>10</td>
<td>Documentation Complete</td>
<td>0 days</td>
<td>Fri 8/10/04</td>
</tr>
<tr>
<td>11</td>
<td>Requirements Elicitation and Analysis</td>
<td>8 days</td>
<td>Mon 1/03/04</td>
</tr>
<tr>
<td>12</td>
<td>Needs analysis</td>
<td>3 days</td>
<td>Mon 1/03/04</td>
</tr>
<tr>
<td>13</td>
<td>Secure required resources</td>
<td>1 day</td>
<td>Mon 8/03/04</td>
</tr>
<tr>
<td>14</td>
<td>Obtain supervisor approval</td>
<td>2 hrs</td>
<td>Fri 19/03/04</td>
</tr>
<tr>
<td>15</td>
<td>Requirements Analysis Complete</td>
<td>0 days</td>
<td>Fri 19/03/04</td>
</tr>
<tr>
<td>16</td>
<td>Design</td>
<td>10 days</td>
<td>Fri 19/03/04</td>
</tr>
<tr>
<td>17</td>
<td>Review RAD</td>
<td>1 day</td>
<td>Fri 19/03/04</td>
</tr>
<tr>
<td>18</td>
<td>Design Presentation Tier</td>
<td>3 days</td>
<td>Mon 22/03/04</td>
</tr>
<tr>
<td>19</td>
<td>Design Business Logic Tier</td>
<td>3 days</td>
<td>Mon 29/03/04</td>
</tr>
<tr>
<td>20</td>
<td>Design Data Access Tier</td>
<td>3 days</td>
<td>Mon 5/04/04</td>
</tr>
<tr>
<td>21</td>
<td>Design Complete</td>
<td>0 days</td>
<td>Mon 12/04/04</td>
</tr>
<tr>
<td>22</td>
<td>Training</td>
<td>15 days</td>
<td>Mon 12/04/04</td>
</tr>
<tr>
<td>23</td>
<td>C# .NET</td>
<td>6 days</td>
<td>Mon 12/04/04</td>
</tr>
<tr>
<td>24</td>
<td>ASP.NET</td>
<td>6 days</td>
<td>Mon 19/04/04</td>
</tr>
<tr>
<td>25</td>
<td>SQL</td>
<td>3 days</td>
<td>Fri 30/04/04</td>
</tr>
<tr>
<td>26</td>
<td>XML</td>
<td>6 days</td>
<td>Mon 3/05/04</td>
</tr>
<tr>
<td>27</td>
<td>Training Complete</td>
<td>0 days</td>
<td>Mon 17/05/04</td>
</tr>
<tr>
<td>28</td>
<td>Implementation</td>
<td>40 days</td>
<td>Mon 17/05/04</td>
</tr>
<tr>
<td>29</td>
<td>Review RAD, SDD, and ODD</td>
<td>5 days</td>
<td>Mon 17/05/04</td>
</tr>
<tr>
<td>30</td>
<td>Implement code</td>
<td>30 days</td>
<td>Fri 28/05/04</td>
</tr>
<tr>
<td>31</td>
<td>Implementation testing (Debugging)</td>
<td>10 days</td>
<td>Mon 26/07/04</td>
</tr>
<tr>
<td>32</td>
<td>Implementation Complete</td>
<td>0 days</td>
<td>Wed 18/08/04</td>
</tr>
<tr>
<td>33</td>
<td>Testing</td>
<td>12 days</td>
<td>Wed 18/08/04</td>
</tr>
<tr>
<td>34</td>
<td>Develop unit test plans</td>
<td>2 days</td>
<td>Wed 18/08/04</td>
</tr>
<tr>
<td>35</td>
<td>Develop integration test plans</td>
<td>2 days</td>
<td>Mon 23/08/04</td>
</tr>
<tr>
<td>36</td>
<td>Unit Testing</td>
<td>4 days</td>
<td>Fri 27/08/04</td>
</tr>
<tr>
<td>37</td>
<td>Test component modules</td>
<td>2 days</td>
<td>Fri 27/08/04</td>
</tr>
<tr>
<td>38</td>
<td>Modify code</td>
<td>1 day</td>
<td>Wed 1/09/04</td>
</tr>
<tr>
<td>39</td>
<td>Re-test modified code</td>
<td>1 day</td>
<td>Fri 3/09/04</td>
</tr>
<tr>
<td>40</td>
<td>Unit Testing Complete</td>
<td>0 days</td>
<td>Mon 6/09/04</td>
</tr>
<tr>
<td>41</td>
<td>Integration Testing</td>
<td>4 days</td>
<td>Mon 6/09/04</td>
</tr>
<tr>
<td>42</td>
<td>Test module integration</td>
<td>2 days</td>
<td>Mon 6/09/04</td>
</tr>
<tr>
<td>43</td>
<td>Modify code</td>
<td>1 day</td>
<td>Fri 10/09/04</td>
</tr>
<tr>
<td>44</td>
<td>Re-test modified code</td>
<td>1 day</td>
<td>Mon 13/09/04</td>
</tr>
<tr>
<td>45</td>
<td>Integration Testing Complete</td>
<td>0 days</td>
<td>Wed 15/09/04</td>
</tr>
<tr>
<td>46</td>
<td>Testing Complete</td>
<td>0 days</td>
<td>Wed 15/09/04</td>
</tr>
<tr>
<td>47</td>
<td>Deployment</td>
<td>2 days</td>
<td>Wed 15/09/04</td>
</tr>
<tr>
<td>48</td>
<td>Deploy software</td>
<td>2 days</td>
<td>Wed 15/09/04</td>
</tr>
<tr>
<td>49</td>
<td>Deployment Complete</td>
<td>0 days</td>
<td>Mon 20/09/04</td>
</tr>
<tr>
<td>50</td>
<td>Project Complete</td>
<td>0 days</td>
<td>Fri 8/10/04</td>
</tr>
</tbody>
</table>

![Gantt chart](chart.png)
<table>
<thead>
<tr>
<th>ID</th>
<th>14 Jun '04</th>
<th>21 Jun '04</th>
<th>28 Jun '04</th>
<th>5 Jul '04</th>
<th>12 Jul '04</th>
<th>19 Jul '04</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>F</td>
<td>S</td>
<td>M</td>
<td>T</td>
<td>W</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix H

Source Code

H.1 OLEWebApplication Project

Listing H.1: Class.aspx

```xml
%@ Register TagPrefix="OLE" TagName="Threads" Src="ClassControls/Threads.ascx"
%@ Register TagPrefix="OLE" TagName="ThreadPosts" Src="ClassControls/ThreadPosts.ascx"
%@ Register TagPrefix="OLE" TagName="ClassAnnouncements" Src="ClassControls/ClassAnnouncements.ascx"
%@ Register TagPrefix="OLE" TagName="InformationAET" Src="ClassControls/InformationAET.ascx"
%@ Register TagPrefix="OLE" TagName="MultipleChoiceAET" Src="ClassControls/MultipleChoiceAET.ascx"
%@ Register TagPrefix="OLE" TagName="ShortResponseAET" Src="ClassControls/ShortResponseAET.ascx"
%@ Register TagPrefix="OLE" TagName="Footer" Src="Footer.ascx"
%@ Register TagPrefix="OLE" TagName="Menu" Src="Menu.ascx"
%@ Register TagPrefix="OLE" TagName="Header" Src="Header.ascx"

<%@ Page language="c#" Codebehind="Class.aspx.cs" AutoEventWireup="false" Inherits="OnlineLearningEnvironment.Interface.ClassPage" codePage="28591" debug="False" %>

<!DOCTYPE HTML PUBLIC "+//W3C//DTD HTML 4.0 Transitional//EN">

<HEAD>

<TITLE>'Online Learning Environment'</TITLE>

<META http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<META name="GENERATOR" Content="Microsoft Visual Studio.NET 7.1"/>

<META name="CODE_LANGUAGE" Content="C#"/>

<META name="vs_defaultClientScript" content="JavaScript"/>

<META name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">

<LINK href="Default.css" type="text/css" rel="stylesheet">

</HEAD>

<BODY MS_POSITIONING="FlowLayout">
```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

Listing H.2: Class.aspx.cs
namespace OnlineLearningEnvironment.Interface
{
    public class ClassPage : System.Web.UI.Page
    {
        protected OnlineLearningEnvironment.Interface.HeaderControl OLEHeader;
        protected OnlineLearningEnvironment.Interface.MenuControl OLEMenu;
        protected OnlineLearningEnvironment.Interface.ClassControls.
            ClassAnnouncements OLEClassAnnouncements;
        protected OnlineLearningEnvironment.Interface.ClassControls.
            Threads OLETreads;
        protected OnlineLearningEnvironment.Interface.ClassControls.
            ThreadPosts OLETThreadPosts;
        protected OnlineLearningEnvironment.Interface.ClassControls.
            InformationAET OLEInformationAET;
        protected OnlineLearningEnvironment.Interface.ClassControls.
            MultipleChoiceAET OLEMultipleChoiceAET;
        protected OnlineLearningEnvironment.Interface.ClassControls.
            ShortResponseAET OLEShortResponseAET;

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private int _threadID = 0;
        public int ThreadID
        {
            get { return this._threadID; }
            set { this._threadID = value; }
        }

        private int _aetQID = 0;
        public int AETQID
        {
            get { return this._aetQID; }
            set { this._aetQID = value; }
        }

        private AETType _aetType = AETType.NotSet;
        public AETType AETType
        {
            get { return this._aetType; }
            set { this._aetType = value; }
        }

        private int _questionID = 1;
        public int QuestionID
        {
            get { return this._questionID; }
            set { this._questionID = value; }
        }
    }
}
private void Page_Load(object sender, System.EventArgs e)
{
    this.OLEHeader.PageName = (Application["Configuration"]
as Configuration).Class;
    this.OLEMenu.PageName = (Application["Configuration"] as
Configuration).Class;
    if (Request.QueryString["classID"] != null)
    {
        this.ClassID = Request.QueryString["classID"]; //
    }
    if (Request.QueryString["threadID"] != null)
    {
        try
        {
            this.ThreadID = Int32.Parse(Request.
            QueryString["threadID"]);
        }
        catch {}
    }
    if (Request.QueryString["aetQID"] != null)
    {
        try
        {
            this.AETQID = Int32.Parse(Request.
            QueryString["aetQID"]);
        }
        catch {}
    }
    if (Request.QueryString["questionID"] != null)
    {
        try
        {
            this.QuestionID = Int32.Parse(Request.
            QueryString["questionID"]);
        }
        catch {}
    }
    if (Request.QueryString["aetType"] != null)
    {
        switch (Request.QueryString["aetType"])
        {
            case "Assignment":
                this.AETType = AETType.Assignment;
                break;
            case "Exam":
                this.AETType = AETType.Exam;
                break;
            case "Tutorial":
                this.AETType = AETType.Tutorial;
                break;
        }
    }
    this.OLEHeader.SubPageName = this.ClassID;
this.OLEMenu.ClassID = this.ClassID;

switch (Request.QueryString["view"]) {
  case "Announcements":
    this.OLEClassAnnouncements.Enabled = true;
    this.OLEClassAnnouncements.ClassID = this.ClassID;
    break;
  case "Threads":
    this.OLEThreads.Enabled = true;
    this.OLEThreads.ClassID = this.ClassID;
    break;
  case "ThreadPosts":
    this.OLEThreadPosts.Enabled = true;
    this.OLEThreadPosts.ClassID = this.ClassID;
    this.OLEThreadPosts.ThreadID = this.ThreadID;
    break;
  case "InformationAET":
    this.OLEInformationAET.Enabled = true;
    this.OLEInformationAET.ClassID = this.ClassID;
    this.OLEInformationAET.AETQID = this.AETQID;
    this.OLEInformationAET.AETType = this.AETType;
    this.OLEInformationAET.QuestionID = this.QuestionID;
    break;
  case "MultipleChoiceAET":
    this.OLEMultipleChoiceAET.Enabled = true;
    this.OLEMultipleChoiceAET.ClassID = this.ClassID;
    this.OLEMultipleChoiceAET.AETQID = this.AETQID;
    this.OLEMultipleChoiceAET.AETType = this.AETType;
    this.OLEMultipleChoiceAET.QuestionID = this.QuestionID;
    break;
  case "ShortResponseAET":
    this.OLEShortResponseAET.Enabled = true;
    this.OLEShortResponseAET.ClassID = this.ClassID;
    this.OLEShortResponseAET.AETQID = this.AETQID;
    this.OLEShortResponseAET.AETType = this.AETType;
    this.OLEShortResponseAET.QuestionID = this.QuestionID;
    break;
  case null:
    default:
    this.OLEClassAnnouncements.Enabled = true;
    this.OLEClassAnnouncements.ClassID = this.ClassID;
Listing H.3: Configuration.aspx

```csharp
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
}
```

---

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>'Online Learning Environment'</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft Visual Studio .NET 7.1">
<meta name="CODE_LANGUAGE" content="C#">
<meta name="vs_default_client_script" content="JavaScript">
<meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
<link href="Default.css" type="text/css" rel="stylesheet">
</head>
<body MS_POSITIONING="FlowLayout">
<form method="post" runat="server" ID="Form1">
<table class="PageFramework" cellspacing="0" cellpadding="0">
<tr class="PageFramework_Header" colspan="2">
<td class="PageFramework_Header" colspan="2">
<OLE:Header id="OLEHeader" PageName="Configuration" runat="server"></OLE:Header>
</td>
</tr>
</table>
</form>
</body>
</html>
```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace OnlineLearningEnvironment.Interface
{
    /// <summary>
    /// Not implemented
    /// </summary>
    public class ConfigurationPage : System.Web.UI.Page
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
            #region Web Form Designer generated code
            override protected void OnInit(EventArgs e)
            {
                InitializeComponent();
                base.OnInit(e);
            }

            private void InitializeComponent()
            {
                this.Load += new System.EventHandler(this.Page_Load);
            }
            #endregion
        }
    }
}
Listing H.5: Default.aspx

```xml
<%@ Register TagPrefix="OLE" TagName="Footer" Src="Footer.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Header" Src="Header.ascx" %>
<%@ Register TagPrefix="OLE" TagName="ErrorMessages" Src="HomeControls/ErrorMessages.ascx" %>
<%@ Register TagPrefix="OLE" TagName="ForumThreadSubscriptions" Src="HomeControls/ForumThreadSubscriptions.ascx" %>
<%@ Register TagPrefix="OLE" TagName="InstitutionAnnouncements" Src="HomeControls/InstitutionAnnouncements.ascx" %>
<%@ Register TagPrefix="OLE" TagName="QuickLinks" Src="HomeControls/QuickLinks.ascx" %>
<%@ Register TagPrefix="OLE" TagName="UpcomingDates" Src="HomeControls/UpcomingDates.ascx" %>
<%@ Page language="c#" Codebehind="Default.aspx.cs" AutoEventWireup="false" Inherits="OnlineLearningEnvironment.Interface.HomePage" codePage="28591" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
<title>'Online Learning Environment'</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
<meta name="CODE_LANGUAGE" Content="C#">
<meta name="vs_defaultClientScript" content="JavaScript">
<meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
<link href="Default.css" type="text/css" rel="stylesheet">
</head>
<body MS_POSITIONING="FlowLayout">
<form method="post" runat="server">
<table class="PageFramework" cellspacing="0" cellpadding="0">
<tr>
<td class="PageFramework_Header">
<OLE:Header id="OLEHeader" PageName="Home" runat="server"></OLE:Header></td>
</tr>
<tr>
<td class="PageFramework.Body" valign="top">
<table width="100%" cellspacing="0" cellpadding="0">
<tr>
<td width="80%" valign="top">
<table cellspacing="0" cellpadding="0" width="100%">
<tr>
<td align="center">OLE: InstitutionAnnouncements id="OLEInstitutionAnnouncements" Runat="server" /></td>
</tr>
</table>
</td>
</tr>
</table>
</td>
</tr>
</table>
</form>
</body>
</html>
```
Listing H.6: Default.aspx.cs

```csharp
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace OnlineLearningEnvironment.Interface
{
    public class HomePage : System.Web.UI.Page
    {

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (Context.User.IsInRole(RoleID.Administrators.ToString()))
                this.OLEErrorMessages.Enabled = true;
            else
                this.OLEErrorMessages.Enabled = false;
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.7: Default.css

```css
P {
    font-size: 10pt;
    vertical-align: baseline;
    color: black;
    line-height: normal;
    font-style: normal;
```
H.1 OLEWebApplication Project

```plaintext
font-family: Sans-Serif;
letter-spacing: normal;
text-align: left;
}

P. Copyright
{
font-size: 8pt;
vertical-align: baseline;
color: black;
line-height: normal;
font-style: normal;
font-family: Sans-Serif;
letter-spacing: normal;
text-align: center;
}

H1
{
font-size: 18pt;
vertical-align: baseline;
color: black;
line-height: normal;
font-style: italic;
font-family: Sans-Serif;
letter-spacing: normal;
text-align: left;
font-variant: small-caps;
}

H2
{
font-size: 16pt;
vertical-align: baseline;
color: black;
line-height: normal;
font-style: italic;
font-family: Sans-Serif;
letter-spacing: normal;
text-align: left;
font-variant: normal;
}

H3
{
font-size: 14pt;
vertical-align: baseline;
color: black;
line-height: normal;
font-style: italic;
font-family: Sans-Serif;
letter-spacing: normal;
text-align: left;
font-variant: normal;
}
```
BODY
{
    margin: 0px;
    background-color: #6699CC;
}

A
{
    color: #990000;
    text-decoration: none;
}

A:active
{
    color: #990000;
    text-decoration: none;
}

A:hover
{
    color: #990000;
    text-decoration: underline;
}

A:link
{
    color: #990000;
    text-decoration: none;
}

TABLE.PageFramework
{
    padding: 0px;
    margin: 0px;
    border: none 0;
    width: 100%;
    height: 100%;
}

TD.PageFramework.Header
{
    padding: 0px;
    margin: 0px;
    border: none 0;
    height: 100px;
}

TD.PageFramework.Menu
{
    padding: 0px;
    margin: 0px;
    border: none 0;
    text-align: left;
    vertical-align: top;
}

TD.PageFramework.Body
{
    height: 100%;
    width: 100%;
    padding: 0px;
margin: 0px;
border: none 0;
vertical-align: top;
}

TD. PageFramework_LoginBody
{
height: 100%;
padding: 0px;
margin: 0px;
border: none 0;
}

TD. PageFramework_Copyright
{
padding: 0px;
margin: 0px;
border: none 0;
text-align: center;
}

A. HeaderFramework_links
{
color: #ffffff;
text-decoration: none;
}

A: active.HeaderFramework_links
{
color: #ffffff;
text-decoration: none;
}

A: hover.HeaderFramework_links
{
color: #ffffff;
text-decoration: underline;
}

A: link.HeaderFramework_links
{
color: #ffffff;
text-decoration: none;
}

TABLE. HeaderFramework
{
padding: 0px;
margin: 0px;
border: none 0;
width: 100%;
background-color: #0066cc;
}

TD. HeaderFramework_TopLeft
{
height: 100px;
}

TD. HeaderFramework_TopRightLeft
{
width: 25px;
height: 25px;
background-position: center 50%;
background-image: url(Images/topleft.jpg);
}

TD. HeaderFramework_TopRightMiddle
{
  background-position: center 50%;
  font-size: 10pt;
  background-image: url(Images/topright.jpg);
  color: #FFFFFF;
  background-repeat: repeat-x;
  font-family: Arial;
  height: 25px;
  text-align: right;
}

TD. HeaderFramework_TopRightRight
{
  width: 5px;
  background-position: center 50%;
  background-image: url(Images/topright.jpg);
  background-repeat: repeat-x;
}

TD. HeaderFramework_TopRightBottom
{
  font-size: 10pt;
  font-family: Arial;
  color: #FFFFFF;
  text-align: right;
}

TD. HeaderFramework_Bottom
{
  background-position: center 50%;
  font-size: 12pt;
  background-image: url(Images/border.jpg);
  background-repeat: repeat-x;
  font-family: Arial;
  height: 35px;
  text-align: center;
  font-variant: small-caps;
}

TABLE. LoginFramework
{
  background-color: #FFFFFF;
  width: 250px;
  padding: 0px;
  margin: 0px;
  border: none 0;
}

TD. LoginFramework_Username
{
  font-size: 12pt;
  font-family: Arial;
  text-align: left;
}

TD. LoginFramework_Password
{  
  font-size: 12pt;
  font-family: Arial;
  text-align: left;
}

TD. LoginFramework_RememberMe

{  
  font-size: 12pt;
  font-family: Arial;
  text-align: right;
}

TD. LoginFramework_Validators

{  
  font-size: 10pt;
  font-family: Arial;
  text-align: center;
}

TABLE. PostFramework

{  
  font-size: 12pt;
  font-family: Arial;
  width: 98%;
  border: none 0;
  border-bottom: #0066cc 3px solid;
}

TH. PostFramework_Head

{  
  font-size: 13pt;
  color: #DDDDDD;
  border-bottom: #6699cc 3px solid;
  font-variant: small-caps;
  border-top: 1px solid #6699cc;
  vertical-align: middle;
  height: 30px;
  background-position: center 50%;
  background-image: url(Images/header2.jpg);
  background-repeat: repeat-x;
}

TD. PostFramework_PostData

{  
  vertical-align: top;
  width: 20%;
  text-align: center;
  background-color: #DDDDDD;
  border-bottom: solid 2px #AAAAAA;
}

TD. PostFramework_PostInformation

{  
  vertical-align: top;
  width: 80%;
}
H.1 OLEWebApplication Project

```
background-color: #EEEEEE;
border-bottom: solid 2px #AAAAAA;
}
TD. PostFramework. AlternatePostInformation
{
  vertical-align: top;
  width: 80%;
  background-color: #e6e6e6;
  border-bottom: solid 2px #AAAAAA;
}
DIV. PostFramework. Name
{
  font-size: 11pt;
  font-weight: bold;
  font-style: italic;
}
DIV. PostFramework. DateDescription
{
  font-size: 9pt;
  font-weight: bold;
}
DIV. PostFramework. Date
{
  font-size: 9pt;
}
DIV. PostFramework. Files
{
  font-size: 9pt;
  font-weight: bold;
}
DIV. PostFramework. Filenames
{
  font-size: 9pt;
}
TABLE. ErrorFramework
{
  font-size: 12pt;
  font-family: Arial;
  width: 98%;
  border: none 0;
  border-bottom: #0066cc 3px solid;
}
TH. ErrorFramework. Header
{
  font-size: 13pt;
  color: #DDDDDD;
  border-bottom: #6699cc 3px solid;
  font-variant: small-caps;
  border-top: 1px solid #6699cc;
  vertical-align: middle;
  height: 30px;
  background-position: center 50%;
  background-image: url(Images/header2.jpg);
  background-repeat: repeat-x;
```
H.1 OLEWebApplication Project

```plaintext
} 
TH. ErrorFramework_Columns 
{
     text-align: center;
     background-color: #AAAAA;
     border-bottom: solid 3px #888888;
 }
TD. ErrorFramework_Date 
{
     width: 20%;
     text-align: center;
     background-color: #DDDDDD;
     border-bottom: solid 2px #AAAAAA;
 }
TD. ErrorFramework_Data 
{
     width: 40%;
     vertical-align: top;
     text-align: center;
     background-color: #EEEEEE;
     border-bottom: solid 2px #AAAAAA;
 }
TD. ErrorFramework_Information 
{
     width: 80%;
     vertical-align: top;
     text-align: left;
     background-color: #EEEEEE;
     border-bottom: solid 2px #AAAAAA;
 }
TD. ErrorFramework_AltData 
{
     width: 40%;
     vertical-align: top;
     text-align: center;
     background-color: #e6e6e6;
     border-bottom: solid 2px #AAAAAA;
 }
TD. ErrorFramework_AltInformation 
{
     width: 80%;
     vertical-align: top;
     text-align: left;
     background-color: #e6e6e6;
     border-bottom: solid 2px #AAAAAA;
 }
TABLE. QLFramework 
{
     font-size: 12pt;
     font-family: Arial;
     width: 98%;
     border: none 0;
     border-bottom: #0066cc 3px solid;
 }
```
H.1 OLEWebApplication Project

TH. QLFramework_Header
{
  font-size: 13pt;
  color: #DDDDDD;
  border-bottom: 3px solid #6699cc;
  font-variant: small-caps;
  border-top: 1px solid #6699cc;
  vertical-align: middle;
  height: 30px;
  background-position: center 50%;
  background-image: url(Images/header2.jpg);
  background-repeat: repeat-x;
}

TD. QLFramework_Class
{
  vertical-align: top;
  text-align: center;
  background-color: #EEEEEE;
  border-bottom: solid 2px #AAAAAA;
}

TD. QLFramework_AltClass
{
  vertical-align: top;
  text-align: center;
  background-color: #e6e6e6;
  border-bottom: solid 2px #AAAAAA;
}

TABLE. AEFramework
{
  font-size: 12pt;
  font-family: Arial;
  width: 98%;
  border: none 0;
  border-bottom: 3px solid #0066cc;
}

TH. AEFramework_Header
{
  font-size: 13pt;
  color: #DDDDDD;
  border-bottom: 3px solid #6699cc;
  font-variant: small-caps;
  border-top: 1px solid #6699cc;
  vertical-align: middle;
  height: 30px;
  background-position: center 50%;
  background-image: url(Images/header2.jpg);
  background-repeat: repeat-x;
}

TD. AEFramework_Date
{
  font-size: 10pt;
  text-align: center;
  vertical-align: middle;
  background-color: #DDDDDD;
```
H.1 OLEWebApplication Project

.1 TD. AEFramework_Description

{ width: 100%;
  vertical-align: top;
  text-align: center;
  background-color: #EEEEEE;
  border-bottom: solid 2px #AAAAAA;
}

.1 TD. AEFramework_AltDescription

{ width: 100%;
  vertical-align: top;
  text-align: center;
  background-color: #e6e6e6;
  border-bottom: solid 2px #AAAAAA;
}

.1 DIV. AEFramework.ClassID

{ font-size: 9pt;
}

.1 TABLE. AEFramework

{ font-size: 12pt;
  font-family: Arial;
  width: 98%;
  border: none 0;
  background-color: #EEEEEE;
  border-top: #0066cc 3px solid;
  border-bottom: #0066cc 3px solid;
}

.1 DIV. AEFramework.Title

{ padding: 10px 0px 0px 0px;
  font-size: 16pt;
  font-variant: small-caps;
}

.1 DIV. AEFramework_Description

{ font-size: 8pt;
}

.1 TD. AEFramework_Data

{ font-size: 8pt;
  text-align: center;
}

.1 DIV. AEFramework_QuestionTitle

{ font-weight: bold;
  font-size: 12pt;
  font-style: italic;
}

.1 DIV. AEFramework_Question
```
Listing H.8: Footer.ascx

```<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Footer.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.FooterControl" TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
<font color="#000000" size="1" face="Arial">Copyright 2004 by Timothy Vriesema</font>
```

Listing H.9: Footer.ascx.cs

```c#
namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class FooterControl : System.Web.UI.UserControl
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.10: Global.asax

```c#
<%@ Application Codebehind="Global.asax.cs" Inherits="OnlineLearningEnvironment.Interface.Global" %>
```

Listing H.11: Global.asax.cs

```c#
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
```
using System.Data.SqlClient;
using System.IO;
using System.Security.Principal;
using System.Web;
using System.Web.SessionState;
using System.Xml;

namespace OnlineLearningEnvironment.Interface
{
    {
        private System.ComponentModel.IContainer components = null;

        public Global()
        {
            InitializeComponent();
        }

        protected void Application_Start(Object sender, EventArgs e)
        {
            Microsoft.Win32.RegistryKey rkOLE = Microsoft.Win32.
                Registry.LocalMachine.OpenSubKey( @"SOFTWARE\Online\Learning\Environment" );

            Application["ClassesDirectory"] = (string)rkOLE.GetValue ("DataDirectory") + "\\" + "Classes";
            Application["InstitutionDirectory"] = (string)rkOLE.
                GetValue("DataDirectory") + "\\" + "Institution";
            Application["SqlConnectionString"] = "server=" + rkOLE.
                GetValue("SqlServerName") + ";Trusted_Connection=true ;database=oledb;" ;

            DataBridge db = new DataBridge ( new SqlConnection(Application["SqlConnectionString"]
                as string ),
                Application["ClassesDirectory"] as string ,
                Application["InstitutionDirectory"] as string ,
                false );

            if (!Directory.Exists((string)rkOLE.GetValue("DataDirectory")))
            {
                // Create Class directory
                Directory.CreateDirectory(Application["ClassesDirectory"] as string);

                // Create Institution directories
                Directory.CreateDirectory(Application["InstitutionDirectory"] as string);
                Directory.CreateDirectory((Application["InstitutionDirectory"] as string) + "\\" + "

Announcements")
Directory.CreateDirectory((Application["InstitutionDirectory"] as string) + "\" + "ApplicationErrors");

Configuration config = new Configuration(db);
config.Create();

// Configuration for the Application
Application["Configuration"] = OnlineLearningEnvironment.Configuration.FindConfiguration(db);
}

protected void Session_Start(Object sender, EventArgs e)
{
    Session["DataBridge"] = new DataBridge(
        new SqlConnection(Application["SqlConnectionString"] as string),
        Application["ClassesDirectory"] as string,
        Application["InstitutionDirectory"] as string,
        true
    );

    Session["SessionVariables"] = new ArrayList();
    (Session["SessionVariables"] as ArrayList).Add(Session["DataBridge"]);
}

protected void Application_BeginRequest(Object sender, EventArgs e)
{
}

protected void Application_EndRequest(Object sender, EventArgs e)
{
}

protected void Application_AuthenticateRequest(Object sender, EventArgs e)
{
    string cookieName = FormsAuthentication.FormsCookieName;
    HttpCookie httpCookie = Context.Request.Cookies[cookieName];

    if (httpCookie == null)
    {
    }
```

    return;
    }
FormsAuthenticationTicket aTicket = null;
try
{
    aTicket = FormsAuthentication.Decrypt(httpCookie.Value);
}
catch
{
    return;
}

if (aTicket == null)
{
    return;
}

FormsIdentity formsIdentity = new FormsIdentity(aTicket);

User user = new User(
    Int32.Parse(formsIdentity.Name),
    formsIdentity,
    new DataBridge(
        new SqlConnection(Application["SqlConnectionString"] as string),
        Application["ClassesDirectory"] as string,
        Application["InstitutionDirectory"] as string,
        true)
);
Context.User = user;

OnlineLearningEnvironment.User tempUser =
    OnlineLearningEnvironment.User.FindUser(
        Int32.Parse(Context.User.Identity.Name),
        (Context.User as User).DataBridge
    );
(Context.User as User).FirstName = tempUser.FirstName;
(Context.User as User).MiddleName = tempUser.MiddleName;
(Context.User as User).LastName = tempUser.LastName;
(Context.User as User).RoleIDs = Role.FindRolesIDs(
    Int32.Parse(Context.User.Identity.Name),
    (Context.User as User).DataBridge
);

protected void Application_Error(Object sender, EventArgs e)
{
}

protected void Session_End(Object sender, EventArgs e)
{

```
protected void Application_End(Object sender, EventArgs e)
{
}

#region Web Form Designer generated code
private void InitializeComponent()
{
    this.components = new System.ComponentModel.Container();
}
#endregion

Listing H.12:  Header.ascx
<TABLE class="HeaderFramework" cellPadding="0" cellSpacing="0" width="100%">
<TR>
    <TD class="HeaderFramework_TopLeft"><asp:Image id="imageLogo" runat="server" Height="100px" Width="420px" /></TD>
    <TD align="right" valign="top">
        <asp:Table cellSpacing="0" cellPadding="0" id="tableLinks" Runat="server">
            <asp:TableRow Runat="server">
                <asp:TableCell CssClass="HeaderFramework_TopRightLeft"></asp:TableCell>
                <asp:TableCell CssClass="HeaderFramework_TopRightMiddle">
                    <asp:HyperLink id="hyperHome" CssClass="HeaderFramework_Links" NavigateUrl="Default.aspx" runat="server">Home</asp:HyperLink>
                    <font>|</font>
                    <asp:HyperLink id="hyperManagement" CssClass="HeaderFramework_Links" NavigateUrl="Management.aspx" runat="server">Management</asp:HyperLink>
                    <font>|</font>
                    <asp:HyperLink id="hyperConfiguration" CssClass="HeaderFramework_Links" NavigateUrl="Configuration.aspx" runat="server">Configuration</asp:HyperLink>
                    <font>|</font>
                    <asp:HyperLink id="hyperClasses" CssClass="HeaderFramework_Links" NavigateUrl="Class.aspx" runat="server">Classes</asp:HyperLink>
                </asp:TableCell>
            </asp:TableRow>
        </asp:Table>
    </TD>
</TR>
</TABLE>
namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Collections;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class HeaderControl : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Label labelPageName;
        protected System.Web.UI.WebControls.Label labelWelcome;
        protected System.Web.UI.WebControls.HyperLink hyperHome;
        protected System.Web.UI.WebControls.HyperLink hyperConfiguration;
        protected System.Web.UI.WebControls.HyperLink hyperClasses;
        protected System.Web.UI.WebControls.LinkButton buttonLogout;
    }
}
protected System.Web.UI.WebControls.Image imageLogo;
protected System.Web.UI.WebControls.HyperLink
    hyperSubmission;
protected System.Web.UI.WebControls.Table tableLinks;

private string _pageTitle;
public string PageTitle
{
    get { return this._pageTitle; }
    set { this._pageTitle = value; }
}

private string _subPageTitle = null;
public string SubPageTitle
{
    get { return this._subPageTitle; }
    set { this._subPageTitle = value; }
}

private void Page_Load(object sender, System.EventArgs e)
{
    this.labelPageName.Text = this.PageTitle;
    if (this.SubPageTitle != null)
        this.labelPageName.Text += " − " + this.SubPageTitle;

    this.imageLogo.ImageUrl = (Application["Configuration"]
        as Configuration).InstitutionLogoImage;

    if (this.PageTitle == "Login")
        this.tableLinks.Visible = false;
    else
        this.labelWelcome.Text = "Welcome" + "<i>" + (Context.User as User).FullName + "</i>" + "(" + (Context.User as User).UserID + ")";
}

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion

private void buttonLogout_Click(object sender, System.
    EventArgs e)
{
    // Kill session variables
    if (Session["SessionVariables"] != null)
```csharp
    object [] os = (object []) (Session ["SessionVariables"] as ArrayList).ToArray (typeof (object));
    for (int i = 0; i < os.Length; i++)
    {
        os [i] = null;
    }
    FormsAuthentication . SignOut ();
    Response . Redirect ("Login . aspx");
```

Listing H.14: Login.aspx
Listing H.15: Login.aspx.cs

```csharp
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace OnlineLearningEnvironment.Interface
{
    public class LoginPage : System.Web.UI.Page
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.16: Management.aspx

```html
<%@ Register TagPrefix="OLE" TagName="Footer" Src="Footer.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Menu" Src="Menu.ascx" %>
<%@ Register TagPrefix="OLE" TagName="Header" Src="Header.ascx" %>
<%@ Page language="c#" Codebehind="Management.aspx.cs"
AutoEventWireup="false" Inherits="OnlineLearningEnvironment.Interface.ManagementPage" codePage="28591"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
</head>
```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace OnlineLearningEnvironment.Interface
{
    /// <summary>
    /// Not implemented
    /// </summary>
}
{
private void Page_Load(object sender, System.EventArgs e)
{
}
#endregion
override protected void OnInit(EventArgs e)
{
 InitializeComponent();
 base.OnInit(e);
}
private void InitializeComponent()
{
 this.Load += new System.EventHandler(this.Page_Load);
}
#endregion
}

Listing H.18: Menu.ascx

%@ Register TagPrefix="OLE" TagName=" Tutorials" Src="MenuControls/Tutorials.ascx" %>
%@ Register TagPrefix="OLE" TagName=" Notes" Src="MenuControls/Notes.ascx" %>
%@ Register TagPrefix="OLE" TagName="Exams" Src="MenuControls/Exams.ascx" %>
%@ Register TagPrefix="OLE" TagName="Assignments" Src="MenuControls/Assignments.ascx" %>
%@ Register TagPrefix="OLE" TagName="Classes" Src="MenuControls/Classes.ascx" %>
<asp:Table Height="1px" Width="150px" CellPadding="0" CellSpacing="0"
 Runat="server">
</asp:Table>
<asp:Table ID="tableClass" Visible="False" Runat="server">
<asp:TableRow>
<asp:TableCell><asp:HyperLink ID="hyperAnnouncements" Runat="server">Announcements</asp:HyperLink></asp:TableCell>
</asp:TableRow>
<asp:TableRow>
<asp:TableCell><asp:HyperLink ID="hyperForum" Runat="server">Forum</asp:HyperLink></asp:TableCell>
</asp:TableRow>
</asp:Table>
<OLE:Notes id="OLENotes" Visible="False" runat="server"></OLE:Notes>
<OLE:Assignments id="OLEAssignments" Visible="False" runat="server"></OLE:Assignments>
<OLE:Exams id="OLEExams" Visible="False" runat="server"></OLE:Exams>
namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class MenuControl : System.Web.UI.UserControl
    {
            Notes OLENotes;
            Assignments OLEAssignments;
            Exams OLEExams;
            Tutorials OLETutorials;
            Classes OLEClasses;

        protected System.Web.UI.WebControls.Table tableClass;
        protected System.Web.UI.WebControls.HyperLink
            hyperAnnouncements;
        protected System.Web.UI.WebControls.HyperLink hyperForum;

        private string _pageName;
        public string PageName
        {
            get { return this._pageName; }
            set { this._pageName = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.PageName == (Application["Configuration"] as
                Configuration).Class)
            {
                if (this.ClassID != null)
                {
                    this.hyperAnnouncements.NavigateUrl = "Class.
```csharp
Listing H.20: Submission.aspx

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace OnlineLearningEnvironment.Interface
{
    /// <summary>
    /// Not implemented
    /// </summary>
    public class SubmissionPage : System.Web.UI.Page
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }
    }
}
#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion

Listing H.22: Web.config

```xml
<configuration>
  <system.web>
    <compilation defaultLanguage="c#" debug="true" />
    <customErrors mode="RemoteOnly" />
    <authentication mode="Forms">
      <forms name="OLEAuthentication" loginUrl="Login.aspx"
        protection="All" path="/" />
    </authentication>
    <authorization>
      <deny users="?" />
    </authorization>
    <trace enabled="false" requestLimit="10" pageOutput="false"
      traceMode="SortByTime" localOnly="true" />
    <sessionState
      mode="InProc"
      stateConnectionString="tcpip=127.0.0.1:42424"
      sqlConnectionString="data source=127.0.0.1;Trusted_Connection=yes"
      cookieless="false"
      timeout="20"
    />
    <globalization
      requestEncoding="utf-8"
      responseEncoding="utf-8"
    />
  </system.web>
</configuration>
```

Listing H.23: ClassControls/AETSubmissionList.ascx

```csharp
```

Listing H.24: ClassControls/AETSubmissionList.ascx.cs
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class AETSubmissionList : System.Web.UI.UserControl
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
            
        #endregion Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
    }

    Listing H.25: ClassControls/ClassAnnouncements.ascx

<%@ Control Language=”c#” AutoEventWireup=”false” Codebehind=”ClassAnnouncements.ascx.cs” Inherits=”OnlineLearningEnvironment.Interface.ClassControls.ClassAnnouncements” TargetSchema=”http://schemas.microsoft.com/intellisense/ie5”%>
<asp:Repeater id=”repeater” runat=”server”>
<HeaderTemplate>
<table class=”PostFramework” cellspacing=”0” cellpadding=”0”>
<tr height=”20px”>
<td></td>
</tr>
<tr>
<th class=”PostFramework_Header” colspan=”2”>
Announcements</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td class=”PostFramework_PostData”>
<div class=”PostFramework_Name”>
(OnlineLearningEnvironment.ClassAnnouncement)
</div>
</td>
</tr>
</ItemTemplate>
</Repeater>
Container.DataItem).CreatorName%></div>
</div>
<div class="PostFramework_DateDescription">Created</div>
<div class="PostFramework_Date">%(OnlineLearningEnvironment.ClassAnnouncement) Container.DataItem).CreatedDateTime.ToString()%</div>
</div>
</div>
</div>
</div>
</div>
<div class="PostFramework_DateDescription">Updated</div>
<div class="PostFramework_Date">%(OnlineLearningEnvironment.ClassAnnouncement) Container.DataItem).UpdatedDateTime.ToString()%</div>
</div>
</div>
</div>
</div>
<td class="PostFramework_PostInformation">
</div>
<asp:Repeater ID="fileAttachmentsRepeater1" DataSource="%(OnlineLearningEnvironment.ClassAnnouncement) Container.DataItem).AttachmentFilenames%" Runat="server">
<HeaderTemplate>
<blockquote>
<table width="100%">
<tr>
<td colspan="2">
<div class="PostFramework_Files">File Attachments:</div>
</td>
</tr>
<tr>
<td width="15px"></td>
<td>
<div class="PostFramework_Filenames">
</td>
</tr>
</table>
</blockquote>
</HeaderTemplate>
<ItemTemplate>%(Container.DataItem%)</ItemTemplate>
<SeparatorTemplate>&nbsp;</SeparatorTemplate>
<FooterTemplate></FooterTemplate>
</td>
</td>
</asp:Repeater>
</td>
</div>
</div>
</div>
</div>
<AlternatingItemTemplate>
  <tr>
    <td class="PostFramework_PostData">
      <div class="PostFramework_Name"><%#((OnlineLearningEnvironment.ClassAnnouncement)Container.DataItem).CreatorName%></div>
      <div class="PostFramework_DateDescription">Created</div>
      <div class="PostFramework_Date"><%#((OnlineLearningEnvironment.ClassAnnouncement)Container.DataItem).CreatedDateTime.ToString()%></div>
      <div class="PostFramework_DateDescription">Updated</div>
      <div class="PostFramework_Date"><%#((OnlineLearningEnvironment.ClassAnnouncement)Container.DataItem).UpdatedDateTime.ToString()%></div>
    </td>
    <td class="PostFramework_AlternatePostInformation">
      <div class="PostFramework_Files">
        File Attachments:
      </div>
    </td>
  </tr>
</AlternatingItemTemplate>
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Collections;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassAnnouncements : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }
        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                if (this.ClassID != null)
                {
                    ClassAnnouncement[] announcements =
                        ClassAnnouncement.FindClassAnnouncements(this.
                        ClassID, Session["DataBridge"] as DataBridge);
                    if (announcements != null)
                    {
                    }
                }
Array.Reverse(announcements);

this.repeater.DataSource = announcements;
this.repeater.DataBind();
}
}
else
    this.Visible = false;
}

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endif

Listing H.27: ClassControls/InformationAET.ascx

<!@ Control Language="c#" AutoEventWireup="false" Codebehind="InformationAET.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.ClassControls.InformationAET" TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
<table cellspacing="0" cellpadding="0"><tr height="20px"><td><td><tr>
<asp:Table ID="" Runat="server"/>
<table cellspacing="0" cellpadding="0" class="IAETFramework">
<tr>
    <th>
        <div class="IAETFramework_Title"><asp:Label ID="" labelTitle" Runat="server"></asp:Label></div>
        <div class="IAETFramework_Description"><asp:Label ID="" labelDescription" Runat="server"></asp:Label></div>
    </th>
</tr>
<tr class="IAETFramework_Data">
    <div>Due Date: <asp:Label ID="" labelDueDate" Runat="server" /></div>
</tr>
<tr height="15px"></tr>
<asp:Repeater ID="" repeaterQuestions" Runat="server">
    <ItemTemplate>
Listing H.28: ClassControls/InformationAET.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class InformationAET : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Label labelTitle;
        protected System.Web.UI.WebControls.Label labelDescription;
        protected System.Web.UI.WebControls.Label labelQuestion;
        protected System.Web.UI.WebControls.Label labelDueDate;
        protected System.Web.UI.WebControls.Repeater repeaterQuestions;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }
    }
}
```
private int _aetQID = 0;
public int AETQID
{
    get { return this._aetQID; }
    set { this._aetQID = value; }
}

private AETType _aetType = AETType.NotSet;
public AETType AETType
{
    get { return this._aetType; }
    set { this._aetType = value; }
}

private int _questionID = 1;
public int QuestionID
{
    get { return this._questionID; }
    set { this._questionID = value; }
}

private AETInformationQuestion currentAETIQ = null;

private void Page_Load(object sender, System.EventArgs e)
{
    if (this.Enabled)
    {
        this.Visible = true;

        if (this.AETQID != 0 && this.AETType != AETType.NotSet && this.ClassID != null)
        {
            if (Request.QueryString["questionID"] != null)
            {
                try
                {
                    this.QuestionID = Int32.Parse(Request.QueryString["questionID"]);
                }
                catch {}
            }

            string uniqueQID = this.ClassID + "|" + this.AETQID.ToString() + "|" + this.AETType.ToString() + "|" + "Question";
            if (Session[uniqueQID] == null)
            {
                Session[uniqueQID] = this.currentAETIQ = AETInformationQuestion.FindAETInformationQuestion(this.ClassID, this.AETQID, this.AETType, Session["DataBridge"] as DataBridge) as AETInformationQuestion;
            }
        }
    }
}
```csharp
this.currentAETIQ.DataBridge = Session["DataBridge"] as DataBridge;

else
    this.currentAETIQ = Session[uniqueQID] as AETInformationQuestion;

this.labelTitle.Text = this.currentAETIQ.Title;
this.labelDescription.Text = this.currentAETIQ.Description;
this.labelDueDate.Text = this.currentAETIQ.DueDateTime.Date.ToLongDateString();

this.repeaterQuestions.DataSource = this.currentAETIQ.Questions;
this.repeaterQuestions.DataBind();
}
else
    this.Visible = false;

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion

Listing H.29: ClassControls/MultipleChoiceAET.ascx
```
Listing H.30: ClassControls/MultipleChoiceAET.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
}
```
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

public class MultipleChoiceAET : System.Web.UI.UserControl
{
    protected System.Web.UI.WebControls.Repeater repeaterQuestions1;
    protected System.Web.UI.WebControls.Repeater repeaterQuestions2;
    protected System.Web.UI.WebControls.Button buttonAnswer;
    protected System.Web.UI.WebControls.Label labelTitle;
    protected System.Web.UI.WebControls.Label labelDescription;
    protected System.Web.UI.WebControls.Label labelQuestion;

    private bool _enabled = false;
    public bool Enabled
    {
        get { return this._enabled; }
        set { this._enabled = value; }
    }

    private string _classID = null;
    public string ClassID
    {
        get { return this._classID; }
        set { this._classID = value; }
    }

    private int _aetQID = 0;
    public int AETQID
    {
        get { return this._aetQID; }
        set { this._aetQID = value; }
    }

    private AETType _aetType = AETType.NotSet;
    public AETType AETType
    {
        get { return this._aetType; }
        set { this._aetType = value; }
    }

    private int _questionID = 1;
    public int QuestionID
    {
        get { return this._questionID; }
        set { this._questionID = value; }
    }

    private AETMultipleChoiceQuestion currentAETMCQ = null;
    private AETMultipleChoiceAnswer currentAETMCA = null;

    private void Page_Load(object sender, System.EventArgs e)
if (this.Enabled)
{
    this.Visible = true;
    if (!this.IsPostBack)
    {
        if (this.AETQID != 0 && this.AETType != AETType.NotSet && this.ClassID != null)
        {
            try
            {
                this.QuestionID = Int32.Parse(Request.QueryString["questionID"]);
            }
            catch {}
```csharp
92 {
    Session[uniqueAID] = this;
    currentAETMCA = new AETMultipleChoiceAnswer(Session["DataBridge"] as DataBridge);
    this.currentAETMCA.AETQID = this.AETQID;
    this.currentAETMCA.AETType = this.AETType;
    this.currentAETMCA.ClassID = this.ClassID;
    this.currentAETMCA.UserID = Int32.Parse(Context.Identity.Name);
    this.currentAETMCA.answers = new MultipleChoiceAnswer[this.currentAETMCQ.AmountOfQuestions];
    this.currentAETMCA.Marked = false;
    this.currentAETMCA.SubmittedDateTime = DateTime.Now;
    this.currentAETMCA.Create();
} else {
    this.currentAETMCA.DataBridge = Session["DataBridge"] as DataBridge;
}
} else {
    this.currentAETMCA = Session[uniqueAID] as AETMultipleChoiceAnswer;
}

if (this.currentAETMCQ != null) {
    int[] questionIDs = new int[this.currentAETMCQ.AmountOfQuestions];
    for (int i = 0; i < questionIDs.Length; i++)
        questionIDs[i] = i + 1;

    this.repeaterQuestions1.DataSource = this.repeaterQuestions2.DataSource = questionIDs;
    this.repeaterQuestions1.DataBind();
    this.repeaterQuestions2.DataBind();
    this.labelTitle.Text = this.currentAETMCQ.Title;
    this.labelDescription.Text = this.currentAETMCQ.Description;
    this.labelQuestion.Text = this.currentAETMCQ.Questions[this.
```
foreach (MultipleChoiceChoice mcc in this.currentAETMCQ.Questions[this.QuestionID - 1].choices)
{
    this.rblChoices.Items.Add(new ListItem(mcc.choiceText, mcc.choiceID.ToString()));
}

int answer;
if (answer == this.currentAETMCA.answers[this.QuestionID - 1].choiceID != 0)
    this.rblChoices.Items[answer - 1].Selected = true;
else
    this.Visible = false;

private void ButtonAnswer_Click(object sender, System.EventArgs e)
{
    string uniqueQID = this.ClassID + "|" + this.AETQID.ToString() + "|" + "Question";
    string uniqueAID = this.ClassID + "|" + this.AETQID.ToString() + "|" + "Answer";
    if (Session[uniqueAID] != null)
    {
        this.currentAETMCQ = Session[uniqueQID] as AETMultipleChoiceQuestion;
        this.currentAETMCA = Session[uniqueAID] as AETMultipleChoiceAnswer;
        this.currentAETMCA.answers[this.QuestionID - 1].questionID = this.QuestionID;
        this.currentAETMCA.answers[this.QuestionID - 1].choiceID = Int32.Parse(this.rblChoices.SelectedValue);
        bool answeredCorrectly = (this.currentAETMCA.answers[this.QuestionID - 1].choiceID == this.currentAETMCQ.Questions[this.QuestionID - 1].correctChoice);
        this.currentAETMCA.answers[this.QuestionID - 1].answeredCorrectly = answeredCorrectly;
        this.currentAETMCA.SubmittedDateTime = DateTime.Now;
        this.currentAETMCA.Alter();
    }
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
    this.buttonAnswer.Click += new System.EventHandler(this.ButtonAnswer_Click);
}

Listing H.31: ClassControls/ShortResponseAET.ascx

```xml
%@ Control Language="c#" AutoEventWireup="false" Codebehind="ShortResponseAET.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.ClassControls.ShortResponseAET" TargetSchema="http://schemas.microsoft.com/intellisense/ie5%>
<table cellspacing="0" cellpadding="0">
    <tr>
        <td><asp:Label ID="labelTitle" Runat="server"></asp:Label></td>
    </tr>
    <tr>
        <td><asp:Label ID="labelDescription" Runat="server"></asp:Label></td>
    </tr>
    <tr>
        <td><asp:Repeater ID="repeaterQuestions1" Runat="server">
            <ItemTemplate>
                <a href="Class.aspx?view=ShortResponseAET&
                    classID=<%#Request.QueryString["classID"]%>&
                    aetQID=<%#Request.QueryString["aetQID"]%>&
                    aetType=<%#Request.QueryString["aetType"]%>&
                    questionID=<%#Container.DataItem%>&
                    classID=<%#Container.DataItem%"></a>
            </ItemTemplate>
            <SeparatorTemplate>
                &nbsp;
            </SeparatorTemplate>
        </asp:Repeater>
    </tr>
    <tr>
        <td><asp:Label ID="labelQuestion" Runat="server"></td>
    </tr>
```
Listing H.32: ClassControls/ShortResponseAET.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class ShortResponseAET : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeaterQuestions1;
        protected System.Web.UI.WebControls.Repeater repeaterQuestions2;
        protected System.Web.UI.WebControls.Label labelTitle;
        protected System.Web.UI.WebControls.Label labelDescription;
        protected System.Web.UI.WebControls.Label labelQuestion;
        protected System.Web.UI.WebControls.TextBox textboxAnswer;
        protected System.Web.UI.WebControls.LinkButton buttonAnswer;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }
    }
}
```
private string _classID = null;
public string ClassID
{
    get { return this._classID; }
    set { this._classID = value; }
}

private int _aetQID = 0;
public int AETQID
{
    get { return this._aetQID; }
    set { this._aetQID = value; }
}

private AETType _aetType = AETType.NotSet;
public AETType AETType
{
    get { return this._aetType; }
    set { this._aetType = value; }
}

private int _questionID = 1;
public int QuestionID
{
    get { return this._questionID; }
    set { this._questionID = value; }
}

private AETShortResponseQuestion currentAETSRQ = null;
protected System.Web.UI.WebControls.RequiredFieldValidator RequiredFieldValidator1;
private AETShortResponseAnswer currentAETSRA = null;

private void Page_Load(object sender, System.EventArgs e)
{
    if (this.Enabled)
    {
        this.Visible = true;
        if (!this.IsPostBack)
        {
            if (this.AETQID != 0 && this.AETType != AETType.NotSet && this.ClassID != null)
            {
                if (Request.QueryString["questionID"] != null)
                {
                    try
                    {
                        this.QuestionID = Int32.Parse(Request.QueryString["questionID"]);
                    }
                    catch {}
string uniqueQID = this.ClassID + "|" + this.AETQID.ToString() + "|" + this.AETType.ToString() + "|" + "Question";
if (Session[uniqueQID] == null)
{
    Session[uniqueQID] = this.currentAETSRQ = AETShortResponseQuestion == AETShortResponseQuestion(this.ClassID, this.AETQID, this.AETType, Session["DataBridge"] as DataBridge) as AETShortResponseQuestion;
    this.currentAETSRQ.DataBridge = Session["DataBridge"] as DataBridge;
}
else
{
    this.currentAETSRQ = Session[uniqueQID] as AETShortResponseQuestion;
}

string uniqueAID = this.ClassID + "|" + this.AETQID.ToString() + "|" + this.AETType.ToString() + "|" + "Answer";
if (Session[uniqueAID] == null)
{
    Session[uniqueAID] = this.currentAETSRA = AETShortResponseAnswer = AETShortResponseAnswer(this.AETQID, this.ClassID, Int32.Parse(Context.User.Identity.Name), this.AETType, Session["DataBridge"] as DataBridge) as AETShortResponseAnswer;
    if (Session[uniqueAID] == null)
    {
        this.currentAETSRA.AETQID = this.AETQID;
        this.currentAETSRA.AETType = this.AETType;
        this.currentAETSRA.ClassID = this.ClassID;
        this.currentAETSRA.UserID = Int32.Parse(Context.User.Identity.Name);
        this.currentAETSRA.answers = new ShortResponseAnswer[this.currentAETSRQ.AmountOfQuestions];
        this.currentAETSRA.Marked = false;
        this.currentAETSRA.SubmittedDateTime
    = DateTime.Now;
    this.currentAETSRA.Create();
}
else
{
    this.currentAETSRA.DataBridge = 
    Session["DataBridge"] as
    DataBridge;
}
else
{
    this.currentAETSRA = Session[uniqueAID] 
    as AETShortResponseAnswer;
}

if (this.currentAETSRQ != null)
{
    int[] questionIDs = new int[this.
    currentAETSRQ.AmountOfQuestions];
    for (int i = 0; i < questionIDs.Length; i++)
    questionIDs[i] = i + 1;
    this.repeaterQuestions1.DataSource = 
    this.repeaterQuestions2.DataSource = questionIDs;
    this.repeaterQuestions1.DataBind();
    this.repeaterQuestions2.DataBind();
    this.labelTitle.Text = this.
    currentAETSRQ.Title;
    this.labelDescription.Text = this.
    currentAETSRQ.Description;
    this.labelQuestion.Text = this.
    currentAETSRQ.Questions[this.
    QuestionID - 1].question;
    if (this.currentAETSRA.answers[this.
    QuestionID - 1].answer != null)
    this.textboxAnswer.Text = this.
    currentAETSRA.answers[this.
    QuestionID - 1].answer;
}
else
this.Visible = false;
}

private void ButtonAnswer_Click(object sender, System.
    EventArgs e)
{
    string uniqueAID = this.ClassID + "|" + this.AETQID.
Listing H.33: ClassControls/ShortResponseAETAnswers.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
}
```

Listing H.34: ClassControls/ShortResponseAETAnswers.ascx

```csharp
```
public class ShortResponseAETAnswers : System.Web.UI.UserControl
{
    private void Page_Load(object sender, System.EventArgs e)
    {
    
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
        InitializeComponent();
        base.OnInit(e);
    }

    private void InitializeComponent()
    {
        this.Load += new System.EventHandler(this.Page_Load);
    }
    #endregion
}

Listing H.35: ClassControls/ThreadPosts.ascx

<table class="PostFramework" cellspacing="0" cellpadding="0">
<tr height="20px">
<td>
</td>
</tr>
<th class="PostFramework_Header" colspan="2"><asp:Label ID="labelThreadTitle" Runat="server"></asp:Label></th>
</tr>
<asp:Repeater ID="repeater" runat="server">
<ItemTemplate>
<tr>
<td class="PostFramework_PostData">
<div class="PostFramework_Name">%(OnlineLearningEnvironment.ClassThreadPost.Container.DataItem.CreatorName)</div>
<div class="PostFramework_DateDescription">Created</div>
<div class="PostFramework_Date">%(OnlineLearningEnvironment.ClassThreadPost.Container.DataItem.CreatedDateTime.ToString())</div>
<div class="PostFramework_DateDescription">Updated</div>
<div class="PostFramework_Date">%(OnlineLearningEnvironment.ClassThreadPost.Container.DataItem.UpdatedDateTime.ToString())</div>
</td>
</tr>
</ItemTemplate>
</asp:Repeater>
</table>
<div><span>&lt;%#((OnlineLearningEnvironment . ClassThreadPost ) Container . DataItem ) . Information%&gt;</span></div>

<asp:Repeater ID="fileAttachmentsRepeater1" DataSource="&lt;%#((OnlineLearningEnvironment . ClassThreadPost ) Container . DataItem ) . AttachmentFilenames%&gt;" Runat="server">

<HeaderTemplate>

<blockquote>
<table width="100%">
<tr>
<td colspan="2">
<div class="PostFramework_Files">
File Attachments:
</div>
</td>
</tr>
<tr>
<td width="15px"></td>
<td>
<div class="PostFramework_Filenames">
</div>
</td>
</tr>
</table>
</FooterTemplate>

<ItemTemplate>
&lt;%#Container . DataItem%&gt;
</ItemTemplate>

<SeparatorTemplate>
&nbsp;
</SeparatorTemplate>

<FooterTemplate>
</FooterTemplate>

&lt;/aspx:Repeater&gt;
</ItemTemplate>

<AlternatingItemTemplate>
<tr>
<td class="PostFramework_PostData">

<div class="PostFramework_Name">&lt;%#((OnlineLearningEnvironment . ClassThreadPost ) Container . DataItem ) . CreatorName%&gt;</div>

<div class="PostFramework_DateDescription">Created</div>&nbsp;

<div class="PostFramework_Date">&lt;%#((OnlineLearningEnvironment . ClassThreadPost ) Container . DataItem ) . CreatedDateTime . ToString ()%&gt;</div>

</td>
</tr>
</AlternatingItemTemplate>
Listing H.36: ClassControls/ThreadPosts.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ClassControls {

    // Updated
    <div class="PostFramework_Date"><%#((OnlineLearningEnvironment.Container.DataItem).UpdatedDateTime.ToString())%></div>

    <td class="PostFramework_AltPostInfo">
        <div><%#((OnlineLearningEnvironment.ClassThreadPost)Container.DataItem).Information%></div>
        <asp:Repeater ID="fileAttachmentsRepeater2" DataSource="<%#((OnlineLearningEnvironment.ClassThreadPost)Container.DataItem).AttachmentFilenames%>" Runat="server">
            <HeaderTemplate>
                <table width="100%">
                    <tr>
                        <td colspan="2">
                            <div class="PostFramework_Files">
                                File Attachments:
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td width="15px"></td>
                        <td>
                            <div class="PostFramework_Filenames">
                                %Container.DataItem%
                            </div>
                        </td>
                    </tr>
                    <SeparatorTemplate>
                        &nbsp;
                    </SeparatorTemplate>
                </table>
            </HeaderTemplate>
            <ItemTemplate>
                <%#Container.DataItem%>
            </ItemTemplate>
            <SeparatorTemplate>
                &nbsp;
            </SeparatorTemplate>
            <FooterTemplate>
                <div></div>
            </FooterTemplate>
        </asp:Repeater>
    </td>
</table>
```

```
Listing H.36: ClassControls/ThreadPosts.ascx.cs
```
using System;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

/// <summary>
/// Not implemented
/// </summary>
public class ThreadPosts : System.Web.UI.UserControl
{
    protected System.Web.UI.WebControls.Repeater repeater;
    protected System.Web.UI.WebControls.Label labelThreadTitle;

    private bool _enabled = false;
    public bool Enabled
    {
        get { return this._enabled; }
        set { this._enabled = value; }
    }

    private int _threadID = 0;
    public int ThreadID
    {
        get { return this._threadID; }
        set { this._threadID = value; }
    }

    private string _classID = null;
    public string ClassID
    {
        get { return this._classID; }
        set { this._classID = value; }
    }

    private void Page_Load(object sender, System.EventArgs e)
    {
        if (this.Enabled)
        {
            this.Visible = true;
            if (this.ThreadID != 0)
            {
                Thread thread = Thread.FindThread(this.ClassID, this.ThreadID, Session["DataBridge"] as DataBridge);
                if (thread != null)
                    this.labelThreadTitle.Text = thread.Title;

                ClassThreadPost[] classThreadPosts = ClassThreadPost.FindClassThreadPosts(this.ClassID, this.ThreadID, Session["DataBridge"] as DataBridge);
                if (classThreadPosts != null)
                {
                    // Code to handle classThreadPosts
                }
            }
        }
    }
}
Listing H.37: ClassControls/Threads.ascx

```xml
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table width="100%">
        </HeaderTemplate>
    <ItemTemplate>
        <tr>
        </tr>
    </ItemTemplate>
    <SeparatorTemplate>
        <tr height="5px" bgcolor="#ffffff"></tr>
    </SeparatorTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```
namespace OnlineLearningEnvironment.Interface.ClassControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class Threads : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                if (this.ClassID != null)
                {
                    Thread[] threads = Thread.FindAllThreads(this.ClassID, Session["DataBridge"] as DataBridge);
                    if (threads != null)
                    {
                        this.repeater.DataSource = threads;
                        this.repeater.DataBind();
                    }
                }
                else
                { this.Visible = false; }
            }
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
{ initializeComponent();
    base.OnInit(e);
}

private void initializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}
#endregion

Listing H.39: ConfigurationControls/ApplicationConfiguration.ascx

namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ApplicationConfiguration : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }
        #endregion
    }
}

Listing H.40: ConfigurationControls/ApplicationConfiguration.ascx.cs
```csharp
Listing H.41: ConfigurationControls/ClassConfiguration.ascx

```csharp
namespace OnlineLearningEnvironment.Interface.ConfigurationControls
{

    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassConfiguration : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion

        }
```
Listing H.43: ConfigurationControls/ClassEnrollmentConfiguration.ascx

```xml
```

Listing H.44: ConfigurationControls/ClassEnrollmentConfiguration.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ConfigurationControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassEnrollmentConfiguration : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.45: ConfigurationControls/InstitutionConfiguration.ascx

```xml
```
namespace OnlineLearningEnvironment.Interface
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class InstitutionConfiguration : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}

namespace OnlineLearningEnvironment.Interface.ConfigurationControls
{
    using System;
    using System.Data;
}
using System.Drawing;
using System.Web;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

/// <summary>
/// Not implemented
/// </summary>
public class RoleAssignmentConfiguration : System.Web.UI.UserControl
{
    private bool _enabled = false;
    public bool Enabled
    {
        get { return this._enabled; } 
        set { this._enabled = value; }
    }

    private void Page_Load(object sender, System.EventArgs e)
    {
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
        InitializeComponent();
        base.OnInit(e);
    }

    private void InitializeComponent()
    {
        this.Load += new System.EventHandler(this.Page_Load);
    }

    #endregion
}

Listing H.49: ConfigurationControls/UserConfiguration.ascx


Listing H.50: ConfigurationControls/UserConfiguration.ascx.cs

namespace OnlineLearningEnvironment.Interface.ConfigurationControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
/// Not implemented
/// </summary>

class UserConfiguration : System.Web.UI.UserControl
{
    private bool _enabled = false;
    public bool Enabled
    {
        get { return this._enabled; }
        set { this._enabled = value; }
    }
}

private void Page_Load(object sender, System.EventArgs e)
{
}

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion

Listing H.51: HomeControls/ErrorMessage.ascx

<asp:Repeater id="repeaterSecurityErrors" runat="server">
    <HeaderTemplate>
        <table class="ErrorFramework" cellspacing="0" cellpadding="0">
            <tr height="20px">
                <td><th class="ErrorFramework Header" colspan="3">Security Errors</th></tr>
                <tr>
                    <th class="ErrorFramework_Column">Date/Time</th>
                    <th class="ErrorFramework_Column">UserID</th>
                    <th class="ErrorFramework_Column">IP Address</th>
                </tr>
            </tr>
        </table>
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
        </tr>
    </ItemTemplate>
</asp:Repeater>
#### H.1 OLEWebApplication Project

```xml
<asp:Repeater id="repeaterApplicationErrors" runat="server">
  <HeaderTemplate>
    <table class="ErrorFramework" cellspacing="0" cellpadding="0">
      <tr height="20px">
        <td class="ErrorFramework_Date">
          ©%
          ([OnlineLearningEnvironment.ApplicationError.Container.DataItem].CreatedDateTime.ToString())
        </td>
        <td class="ErrorFramework_Information">
          ©%
        </td>
      </tr>
    </table>
  </HeaderTemplate>
  <ItemTemplate>
    <tr>
      <td class="ErrorFramework_Date">
        ©%
        ([OnlineLearningEnvironment.ApplicationError.Container.DataItem].CreatedDateTime.ToString())
      </td>
      <td class="ErrorFramework_Information">
        ©%
      </td>
    </tr>
  </ItemTemplate>
  <AlternatingItemTemplate>
    <tr>
      <td class="ErrorFramework_Date">
        ©%
        ([OnlineLearningEnvironment.ApplicationError.Container.DataItem].CreatedDateTime.ToString())
      </td>
      <td class="ErrorFramework_Information">
        ©%
      </td>
    </tr>
  </AlternatingItemTemplate>
</asp:Repeater>
```
Listing H.52: HomeControls/ErrorMessage.ascx.cs

```
namespace OnlineLearningEnvironment.Interface.HomeControls {

    using System;
    using System.Collections;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class ErrorMessages : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeaterApplicationErrors;

        private bool _enabled = true;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                SecurityError[] securityErrors = SecurityError.FindSecurityErrors(Session["DataBridge"] as DataBridge);
                if (securityErrors != null)
                {
                    ArrayList tempArray = new ArrayList(securityErrors);
                    if (tempArray.Count > 5)
                        tempArray.RemoveRange(5, tempArray.Count - 5);
                    securityErrors = (SecurityError[])tempArray;
                }
            }
        }
    }
}
```
ToArray(typeof(SecurityError));
    Array.Reverse(securityErrors);
    this.repeaterSecurityErrors.DataSource = securityErrors;
    this.repeaterSecurityErrors.DataBind();
}

ApplicationError[] applicationErrors =
    ApplicationError.FindApplicationErrors(Session["DataBridge"] as DataBridge);
if (applicationErrors != null)
{
    ArrayList tempArray = new ArrayList(
        applicationErrors);
    if (tempArray.Count > 5)
        tempArray.RemoveAtRange(5, tempArray.Count - 5);
    applicationErrors = (ApplicationError[])
        tempArray.ToArray(typeof(ApplicationError));
    Array.Reverse(applicationErrors);
    this.repeaterApplicationErrors.DataSource = applicationErrors;
    this.repeaterApplicationErrors.DataBind();
}
else
    this.Visible = false;

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}
private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion

Listing H.53: HomeControls/ForumThreadSubscriptions.ascx

namespace OnlineLearningEnvironment.Interface.HomeControls
{
    ///
    ///</summary>

    Listing H.54: HomeControls/ForumThreadSubscriptions.ascx.cs
```csharp
using System;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

/// <summary>
/// Not implemented
/// </summary>
public class ForumThreadSubscriptions : System.Web.UI.UserControl
{
    private bool _enabled = true;
    public bool Enabled
    {
        get { return this._enabled; }
        set { this._enabled = value; }
    }

    private void Page_Load(object sender, System.EventArgs e)
    {
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
        InitializeComponent();
        base.OnInit(e);
    }

    private void InitializeComponent()
    {
        this.Load += new System.EventHandler(this.Page_Load);
    }
    #endregion
}
```

Listing H.55: HomeControls/InstitutionAnnouncements.ascx

```xml
<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table class="PostFramework" cellspacing="0" cellpadding="0">
            <tr height="20px">
                <td></td>
            </tr>
            <tr>
                <th class="PostFramework_Header" colspan="2">
```

Announcements</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td class="PostFramework_PostData">
<div class="PostFramework_Name"><%#((OnlineLearningEnvironment.InstitutionAnnouncement)Container.DataItem).CreatorName%></div>
<div class="PostFramework_DateDescription">Created</div>
</td>
</tr>
</ItemTemplate>
</td>
</tr>
</td>
</ItemTemplate>
</table>
</blockquote>
</td>
</tr>
<tr>
<td width="15px"></td>
<td width="15px"></td>
</tr>
</td>
</ItemTemplate>
</table>
</ItemTemplate>
</HeaderTemplate>

File Attachments:
</tr></table></blockquote></FooterTemplate></asp:Repeater></td></tr></ItemTemplate></AlternatingItemTemplate></tr><td class="PostFramework PostData">
  <div class="PostFramework Name">
    <div><%#((OnlineLearningEnvironment . InstitutionAnnouncement . Container . DataItem) . CreatorName%></div>
  </div>
  <div class="PostFramework DateDescription">
    <div class="PostFramework Date">Created</div>
  </div>
  <div class="PostFramework DateDescription">
    <div class="PostFramework Date">Updated</div>
  </div>
</td>
<td class="PostFramework AlternatePostInformation">
  <div><%#((OnlineLearningEnvironment . InstitutionAnnouncement . Container . DataItem) . Information%></div>
    Runat="server">
    HeaderTemplate
    <blockquote>
      <table width="100%">
        <tr>
          <td colspan="2">
            <div class="PostFramework Files">File Attachments:</div>
          </td>
        </tr>
        <tr>
          <td width="15px"></td>
          <td><div class="PostFramework Filenames">%#Container . DataItem%<%#Container . DataItem%</div></td>
        </tr>
        <tr>
          </td>
        </tr>
        <tr>
          <td><div class="PostFramework Filenames">%#Container . DataItem%<%#Container . DataItem%</div></td>
        </tr>
        <tr>
          </td>
        </tr>
      </table>
    </td>
    <td>
      <div class="PostFramework Files">File Attachments:</div>
    </td>
  </asp:Repeater>
</td>
namespace OnlineLearningEnvironment.Interface.HomeControls
{
    using System;
    using System.Collections;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class InstitutionAnnouncements : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = true;

        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                InstitutionAnnouncement[] announcements =
                    InstitutionAnnouncement.FindInstitutionAnnouncements(Session["DataBridge"] as DataBridge);
                if (announcements != null)
                {
                    Array.Reverse(announcements);
                    this.repeater.DataSource = announcements;
                }
            }
        }
    }
}
private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    InitializeComponent();
    base.OnInit(e);
}
#endregion

Listing H.57:  HomeControls/QuickLinks.ascx

<asp:Repeater id="repeater" runat="server">
    <HeaderTemplate>
        <table class="QLFramework" cellspacing="0" cellpadding="0">
            <tr height="20px"><td></td><td class="QLFramework Header">Quick Links</td></tr>
        </table>
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td class="QLFramework Class"><a href="Class.aspx?classID=" classID="((OnlineLearningEnvironment.Class)Container.DataItem).ClassID%"
                OnlineLearningEnvironment.Class(Container.DataItem).ClassID%>>(OnlineLearningEnvironment.Class(Container.DataItem).ClassName%)/a"></td>
        </tr>
    </ItemTemplate>
    <AlternatingItemTemplate>
        <tr>
                OnlineLearningEnvironment.Class(Container.DataItem).ClassID%>>(OnlineLearningEnvironment.Class(Container.DataItem).ClassName%)/a"></td>
        </tr>
    </AlternatingItemTemplate>
namespace OnlineLearningEnvironment.Interface.HomeControls
{
    using System;
    using System.Collections;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class QuickLinks : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = true;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                Class[] classes = Class.FindClasses(Int32.Parse(Context.User.Identity.Name), Session["DataBridge"] as DataBridge);
                if (classes != null)
                {
                    this.repeater.DataSource = classes;
                    this.repeater.DataBind();
                }
            }
            else
            {
                this.Visible = false;
            }

            #region Web Form Designer generated code
            override protected void OnInit(EventArgs e)
            {
                InitializeComponent();
                base.OnInit(e);
            }

            private void InitializeComponent()
            {
            }
        }
    }
}
```csharp
{ this.Load += new System.EventHandler(this.Page_Load);

#endregion

Listing H.59: HomeControls/UpcomingDates.ascx
```
ShortAETQuestion) Container . DataItem ) . ClassID
&netQID=<%#((OnlineLearningEnvironment .
ShortAETQuestion) Container . DataItem ) . AETQID
&netType=<%#((OnlineLearningEnvironment .
ShortAETQuestion) Container . DataItem ) . AETType
>Description%</a></div>
</td>
</tr>
</AlternatingItemTemplate>
</FooterTemplate>
</table>
<asp:Repeater id="repeaterExams" runat="server">
<HeaderTemplate>
<table class="AEFramework" cellspacing="0" cellpadding="0">
<tr height="20px"><td><br /></td></tr>
<tr>
<th class="AEFramework_Header" colspan="2">Exams</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td class="AEFramework_Date"><%#((
OnlineLearningEnvironment . ShortAETQuestion )
Container . DataItem ) . DueDateTime></td>
<td class="AEFramework_Description">
<a href="Class.aspx?view=<%#((
OnlineLearningEnvironment . ShortAETQuestion )
Container . DataItem ) . QuestionType%&netClassID
&netQID=<%#((OnlineLearningEnvironment .
ShortAETQuestion) Container . DataItem ) . ClassID
&netQID=<%#((OnlineLearningEnvironment .
ShortAETQuestion) Container . DataItem ) . AETQID
&netType=<%#((OnlineLearningEnvironment .
ShortAETQuestion) Container . DataItem ) . AETType
>Description%</a></td>
</div>
</ItemTemplate>
</AlternatingItemTemplate>
<tr><td class="AEFramework_Date"><%#((
OnlineLearningEnvironment . ShortAETQuestion )
Container . DataItem ) . ClassID"></td>
Listing H.60: HomeControls/UpcomingDates.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.HomeControls
{
    using System;
    using System.Collections;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class UpcomingDates : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeaterAssignments;
        protected System.Web.UI.WebControls.Repeater repeaterExams;

        private bool _enabled = true;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                Class[] classes = Class.FindClasses(Int32.Parse(
```
if (classes != null)
{
    ArrayList assignments = new ArrayList();
    bool isAssignments = false;
    ArrayList exams = new ArrayList();
    bool isExams = false;
    foreach (Class c in classes)
    {
        try
        {
            assignments.AddRange(ShortAETQuestion.FindShortAETQuestions(c.ClassID, AETType.Assignment, Session["DataBridge"] as DataBridge));
            isAssignments = true;
        }
        catch
        {
            if (!isAssignments)
                isAssignments = false;
        }
        try
        {
            exams.AddRange(ShortAETQuestion.FindShortAETQuestions(c.ClassID, AETType.Exam, Session["DataBridge"] as DataBridge));
            isExams = true;
        }
        catch
        {
            if (!isExams)
                isExams = false;
        }
    }
    if (isAssignments)
    {
        this.repeaterAssignments.DataSource = assignments;
        this.repeaterAssignments.DataBind();
    }
    if (isExams)
    {
        this.repeaterExams.DataSource = exams;
        this.repeaterExams.DataBind();
    }
}
else
    this.Visible = false;
override protected void OnInit(EventArgs e) {
    InitializeComponent();
    base.OnInit(e);
}

private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion

Listing H.61: LoginControls/Authentication.ascx

```csharp
<table class="LoginFramework">
    <tr>
        <td class="LoginFramework_Username">Username:</td>
    </tr>
    <tr>
        <td align="right"><asp: textbox id="textBoxUsername" tabIndex="1" runat="server" MaxLength="255" Width="200px"></asp: textbox></td>
    </tr>
    <tr>
        <td class="LoginFramework_Password">Password:</td>
    </tr>
    <tr>
        <td align="right"><asp: textbox id="textBoxPassword" TextMode="Password" tabIndex="2" runat="server" MaxLength="255" Width="200px"></asp: textbox></td>
    </tr>
    <tr>
        <td class="LoginFramework_RememberMe"><asp: CheckBox id="checkBoxRememberMe" tabIndex="3" runat="server" Text="Remember me" /></td>
    </tr>
    <tr>
        <td class="LoginFramework_Login"><asp: button id="loginButton" tabIndex="4" runat="server" Text="Login"></asp: button></td>
    </tr>
    <tr>
        <td class="LoginFramework_Validators"><asp: requiredfieldvalidator id="validatorUsernameExists" runat="server" id="labelNotFound" Visible="false" runat="server" ForeColor="Red">Invalid username or password!</asp: requiredfieldvalidator></td>
    </tr>
    <tr>
        <td class="LoginFramework_Validators"><asp: Label id="labelNotFound" Visible="false" runat="server" ForeColor="Red">Invalid username or password!</asp: Label></td>
    </tr>
```
Listing H.62: Login Controls/Authentication.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.LoginControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Authentication : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Button loginButton;
        protected System.Web.UI.WebControls.TextBox textBoxUsername;
        protected System.Web.UI.WebControls.RequiredFieldValidator validatorUsernameExists;
        protected System.Web.UI.WebControls.RequiredFieldValidator validatorPasswordExists;
        protected System.Web.UI.WebControls.CheckBox checkBoxRememberMe;
        protected System.Web.UI.WebControls.Label labelNotFound;
        protected System.Web.UI.WebControls.TextBox textBoxPassword;

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.loginButton.Click += new System.EventHandler(this.loginButton_Click);
            this.Load += new System.EventHandler(this.Page_Load);
        }
    }
}
```
private void loginButton_Click(object sender, System.EventArgs e)
{
    bool isAuthenticated;

    try
    {
        isAuthenticated = OnlineLearningEnvironment.Authentication.CheckAuthentication(
            Int32.Parse(this.textBoxUsername.Text),
            this.textBoxPassword.Text,
            Session["DataBridge"] as DataBridge);
    }
    catch (System.FormatException)
    {
        // Not authenticated
        isAuthenticated = false;
    }

    // Authenticated?
    if (isAuthenticated)
    {
        FormsAuthenticationTicket aTicket = new
            FormsAuthenticationTicket(1,
            this.textBoxUsername.Text,
            DateTime.Now,
            DateTime.Now.AddMonths(1),
            this.checkBoxRememberMe.Checked,
            null);

        string encryptedATicket = FormsAuthentication.Encrypt(aTicket);

        HttpCookie httpCookie = new HttpCookie(
            FormsAuthentication.FormsCookieName,
            encryptedATicket);
        Response.Cookies.Add(httpCookie);

        FormsAuthentication.RedirectFromLoginPage(
            this.textBoxUsername.Text,
            this.checkBoxRememberMe.Checked);
    }
    else
    {
        // Display not found message
        this.labelNotFound.Visible = true;
    }
}
Listing H.63: ManagementControls/ClassAnnouncementManagement.ascx

```csharp
```

Listing H.64: ManagementControls/ClassAnnouncementManagement.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassAnnouncementManagement : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.65: ManagementControls/ClassManagement.ascx

```csharp
```
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassManagement : System.Web.UI.UserControl
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}

namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ClassNoteManagement : System.Web.UI.UserControl
    {
    }
}
Listing H.69: ManagementControls/InformationAETManagement.ascx

```csharp

Listing H.70: ManagementControls/InformationAETManagement.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    public class InformationAETManagement : System.Web.UI.UserControl
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }
        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```
Listing H.71: ManagementControls/InstitutionAnnouncementManagement.ascx

```xml
```

Listing H.72: ManagementControls/InstitutionAnnouncementManagement.ascx.cs

```csharp
namespace OnlineLearningEnvironment.UserInterface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class InstitutionAnnouncementManagement : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
    }
}
```
Listing H.73: ManagementControls/MultipleChoiceAETManagement.ascx

```csharp
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="MultipleChoiceAETManagement.ascx.cs" Inherits="OnlineLearningEnvironment.Interface.ManagementControls.MultipleChoiceAETManagement" TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>

Listing H.74: ManagementControls/MultipleChoiceAETManagement.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ManagementControls {
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class MultipleChoiceAETManagement : System.Web.UI.UserControl {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e) {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```
Listing H.75: ManagementControls/ShortResponseAETManagement.ascx

```csharp
```

Listing H.76: ManagementControls/ShortResponseAETManagement.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class ShortResponseAETManagement : System.Web.UI.UserControl
    {
        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.77: ManagementControls/UserManagement.ascx

```csharp
```
Listing H.78: ManagementControls/UserManagement.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.ManagementControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    /// Not implemented
    /// </summary>
    public class UserManagement : System.Web.UI.UserControl
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }

        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Listing H.79: MenuControls/Assignments.ascx

```xml
<asp:Repeater id="repeater" runat="server">
<HeaderTemplate>
<table>
<tr><td><i>%#(Application["Configuration"] as OnlineLearningEnvironment.Configuration).Assignment%s</i></td></tr>
</HeaderTemplate>
<ItemTemplate><tr>
```

```xml
```
namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Assignments : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                if (this.ClassID != null)
                {
                    try
                    ...
```csharp
{  
    ShortAETQuestion[] sAETQuestions =  
    ShortAETQuestion.FindShortAETQuestions(  
        this.ClassID, AETType.Assignment, Session  
        ["DataBridge"] as DataBridge);  
    this.repeater.DataSource = sAETQuestions;  
    this.repeater.DataBind();  
}  
}  
catch {}  
}  
else  
    this.Visible = false;  
}  
#endif  
override protected void OnInit(EventArgs e)  
{  
    InitializeComponent();  
    base.OnInit(e);  
}  
private void InitializeComponent()  
{  
    this.Load += new System.EventHandler(this.Page_Load);  
}  
#endif  
Listing H.81: MenuControls/Classes.ascx
```

```xml
<brp:Repeater id="repeater" runat="server">
  <HeaderTemplate>
    <table>
      <tr>
        <td><b><i>%#(Application["Configuration"] as OnlineLearningEnvironment.Configuration).Class%s</i></b></td>
      </tr>
    </HeaderTemplate>
  <ItemTemplate>
    <tr>
```
namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Classes : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (this.Enabled)
            {
                this.Visible = true;
                try
                {
                    Class[] classes = Class.FindClasses(Int32.Parse(Context.User.Identity.Name), Session["DataBridge"] as DataBridge);
                    this.repeater.DataSource = classes;
                    this.repeater.DataBind();
                }
                catch {} 
            }
            else
            { 
                this.Visible = false;
            }
        }
    }
}
private void InitializeComponent()
{
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion

Listing H.83: MenuControls/Exams.ascx

<asp:Repeater id=”repeater” runat=”server”>
    <HeaderTemplate>
        <table>
            <tr>
                <td><b><i>%#(Application[”Configuration”] as OnlineLearningEnvironment.Configuration).Exam%</i><b></td>
            </tr>
        </table>
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
        </tr>
    </ItemTemplate>
    <SeparatorTemplate>
        <tr>
            <td height=”5px” bgcolor=”#ffffff”></td>
        </tr>
    </SeparatorTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>

Listing H.84: MenuControls/Exams.ascx.cs

namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

public class Exams : System.Web.UI.UserControl
{
    protected System.Web.UI.WebControls.Repeater repeater;

    private bool _enabled = false;
    public bool Enabled
    {
        get { return this._enabled; }
        set { this._enabled = value; }
    }

    private string _classID = null;
    public string ClassID
    {
        get { return this._classID; }
        set { this._classID = value; }
    }

    private void Page_Load(object sender, System.EventArgs e)
    {
        if (this.Enabled)
        {
            this.Visible = true;
            if (this.ClassID != null)
            {
                try
                {
                    ShortAETQuestion[] sAETQuestions =
                        ShortAETQuestion.FindShortAETQuestions(
                            this.ClassID, AETType.Exam, Session["DataBridge"] as DataBridge);
                    this.repeater.DataSource = sAETQuestions;
                    this.repeater.DataBind();
                }
                catch {}
            }
            else
            {
                this.Visible = false;
            }
        }
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }
        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
Listing H.85: MenuControls/Notes.ascx

```csharp
<asp:Repeater id="repeater" runat="server">
  <HeaderTemplate>
    <table>
      <tr>
        <td><b><i>%#(Application["Configuration"] as OnlineLearningEnvironment.Configuration).Note%</i><b></td>
      </tr>
    </table>
  </HeaderTemplate>
  <ItemTemplate>
    <tr>
      <td width="80%" class="aspx?view=Notes&AET">
    </tr>
  </ItemTemplate>
  <SeparatorTemplate>
    <tr>
      <td height="5px" bgcolor="#ffffff"></td>
    </tr>
  </SeparatorTemplate>
  <FooterTemplate>
    </table>
  </FooterTemplate>
</asp:Repeater>
```

Listing H.86: MenuControls/Notes.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Notes : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;
        private bool _enabled = false;
        public bool Enabled
```
private string _classID = null;
public string ClassID
{
    get { return this._classID; }
    set { this._classID = value; }
}

private void Page_Load(object sender, System.EventArgs e)
{
    if (this.Enabled)
    {
        this.Visible = true;
        if (this.ClassID != null)
        {
            ClassNote[] classNotes = ClassNote.
                FindClassNotes(this.ClassID, Session["DataBridge"] as DataBridge);
            if (classNotes != null)
            {
                this.repeater.DataSource = classNotes;
                this.repeater.DataBind();
            }
        }
    }
    else
    {
        this.Visible = false;
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
        InitializeComponent();
        base.OnInit(e);
    }
    
    private void InitializeComponent()
    {
        this.Load += new System.EventHandler(this.Page_Load);
    }
    
    #endregion
Listing H.88:  MenuControls/Tutorials.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Interface.MenuControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    public class Tutorials : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Repeater repeater;

        private bool _enabled = false;
        public bool Enabled
        {
            get { return this._enabled; }
            set { this._enabled = value; }
        }

        private string _classID = null;
```
```csharp
    public string ClassID
    {
        get { return this._classID; }
        set { this._classID = value; }
    }

    private void Page_Load(object sender, System.EventArgs e)
    {
        if (this.Enabled)
        {
            this.Visible = true;
            if (this.ClassID != null)
            {
                try
                {
                    ShortAETQuestion[] sAETQuestions =
                        ShortAETQuestion.FindShortAETQuestions(
                            this.ClassID, AETType.Tutorial, Session["DataBridge"] as DataBridge);
                    this.repeater.DataSource = sAETQuestions;
                    this.repeater.DataBind();
                }
                catch {}
            }
            else
            {
                this.Visible = false;
            }
        }
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            InitializeComponent();
            base.OnInit(e);
        }
        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
```

Listing H.89: SubmissionControls/Class.ascx

```csharp
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Class.ascx.cs" Inherits="OnlineLearningEnvironment.Iface.SubmissionControls.Class" TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>

Listing H.90: SubmissionControls/Class.ascx.cs

```csharp
namespace OnlineLearningEnvironment.Iface.SubmissionControls
{
    using System;
```
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls;

/// <summary>
/// Not implemented
/// </summary>
public class Class : System.Web.UI.UserControl
{
    private bool _enabled = false;
    public bool Enabled
    {
        get { return this._enabled; }
        set { this._enabled = value; }
    }

    private void Page_Load(object sender, System.EventArgs e)
    {
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
        InitializeComponent();
        base.OnInit(e);
    }

    private void InitializeComponent()
    {
        this.Load += new System.EventHandler(this.Page_Load);
    }
    #endregion
}

Listing H.91: SubmissionControls/User.ascx

namespace OnlineLearningEnvironment.Interface.SubmissionControls
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.WebControls;

    /// <summary>
    //
H.2 OLEBusinessLogic Project

Listing H.93: Address.cs

```csharp
using System;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// The structure of a postal/residential Address.
    public struct AddressStructure
    {
        #region Properties
        /// Street number and name.
        private string _street;
        [XmlElement("Street")]  // Add XML attribute on property
        public string Street
        {
            get { return _street; }
        }
    }
}
```
using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;

/// <summary>
/// Suburb/City.
/// </summary>
private string _suburb;
[XmlElement("Suburb")]
public string Suburb
{
    get { return _suburb; }
    set { this._suburb = value; }
}

/// <summary>
/// State of the country.
/// </summary>
private string _state;
[XmlElement("State")]
public string State
{
    get { return _state; }
    set { this._state = value; }
}

/// <summary>
/// Postcode of the country.
/// </summary>
private string _postcode;
[XmlElement("Postcode")]
public string Postcode
{
    get { return _postcode; }
    set { this._postcode = value; }
}

/// <summary>
/// Country.
/// </summary>
private string _country;
[XmlElement("Country")]
public string Country
{
    get { return _country; }
    set { this._country = value; }
}
#endregion

Listing H.94: AETAnswer.cs
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    #region Structures
    /// <summary>
    /// The structure of a short response answer.
    /// </summary>
    public struct ShortResponseAnswer
    {
        /// <summary>
        /// The QuestionID of the AETQuestion.
        /// </summary>
        [XmlAttribute("questionID")]
        public int questionID;
        /// <summary>
        /// A written answer to the question.
        /// </summary>
        [XmlText()]
        public string answer;
        /// <summary>
        /// An Instructor will set true if answer is correct and false if incorrect.
        /// </summary>
        [XmlAttribute("answeredCorrectly")]
        public bool answeredCorrectly;
    }

    #endregion

    /// <summary>
    /// The structure of a multiple choice answer.
    /// </summary>
    public struct MultipleChoiceAnswer
    {
        /// <summary>
        /// The QuestionID of the AETQuestion.
        /// </summary>
        [XmlAttribute("questionID")]
        public int questionID;
        /// <summary>
        /// The answer choiceID from the AETQuestion’s choices.
        /// </summary>
        [XmlAttribute("choiceID")]
        public int choiceID;
        /// <summary>
        /// Assigned true by the application equal to AETQuestion. Questions.correctChoice.
        /// </summary>
        [XmlAttribute("answeredCorrectly")]
        public bool answeredCorrectly;
    }
}
/// <summary>
/// A structure similar to AETAnswer, but without the answer properties.
/// </summary>
public struct ShortAETAnswer
{
    #region Properties
    /// <summary>
    /// The UserID associated with the AETAnswer.
    /// </summary>
    private int _userID;
    [XmlAttribute("userID")]
    public int UserID
    {
        get { return this._userID; }
        set { this._userID = value; }
    }

    /// <summary>
    /// The ID of the AETQuestion associated with the AETAnswer.
    /// </summary>
    private int _aetQID;
    [XmlAttribute("aetQID")]
    public int AETQID
    {
        get { return this._aetQID; }
        set { this._aetQID = value; }
    }

    /// <summary>
    /// The AETType of the associated AETQuestion.
    /// </summary>
    private AETType _aetType;
    [XmlAttribute("aetType")]
    public AETType AETType
    {
        get { return this._aetType; }
        set { this._aetType = value; }
    }

    /// <summary>
    /// The ClassID associated class.
    /// </summary>
    private string _classID;
    [XmlAttribute("classID")]
    public string ClassID
    {
        get { return this._classID; }
        set { this._classID = value; }
    }

    /// <summary>
    /// The date that the answer was submitted.
    /// </summary>
    private DateTime _answerSubmitDate;
    [XmlAttribute("answerSubmitDate")]
    public DateTime AnswerSubmitDate
    {
        get { return this._answerSubmitDate; }
        set { this._answerSubmitDate = value; }
    }
}
private DateTime _submittedDateTime;
[XmlAttribute("submittedDateTime")]
public DateTime SubmittedDateTime
{
    get { return this._submittedDateTime; }
    set { this._submittedDateTime = value; }
}

private bool _marked;
[XmlAttribute("marked")]
public bool Marked
{
    get { return this._marked; }
    set { this._marked = value; }
}

#region Static Data Retrieving Methods
/// <summary>
/// Finds the properties of an AETAnswer (without the answers).
/// </summary>
/// <param name="aetQID">The associated AETQuestion ID.</param>
/// <param name="classID">The associated Class ID.</param>
/// <param name="userID">The owner of the AETAnswer.</param>
/// <param name="aetType">The type of the AETAnswer (Assignment, Exam, or Tutorial).</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <exception cref="System.Exception">Thrown when an AETAnswer isn’t found.</exception>
/// <returns>Returns the found structure ShortAETAnswer.</returns>
public static ShortAETAnswer FindShortAETAnswer(int aetQID, string classID, int userID, AETType aetType, DataBridge dataBridge)
{
    SqlCommand sqlCommand = new SqlCommand("FindShortAETAnswer", dataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@aetQID", SqlDbType.Int);
    sqlCommand.Parameters["@aetQID"].Value = aetQID;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = classID;
    sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
    sqlCommand.Parameters["@userID"].Value = userID;
    sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar, AETLimitations.AETType);
    sqlCommand.Parameters["@aetQType"].Value = aetType;
dataBridge SqlConnection.Open();
// Read the sql server's xml data
XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.ExecuteScalar();

// Retrieve the information and deserialize it
ShortAETAnswer shortAETAnswer = new ShortAETAnswer();
shortAETAnswer.AETQID = 0;
XmlSerializer xmlSerializer = new XmlSerializer(typeof(ShortAETAnswer));
while (xmlTextReader.Read())
if (xmlSerializer.CanDeserialize(xmlTextReader))
    shortAETAnswer = (ShortAETAnswer)xmlSerializer.Deserialize(xmlTextReader);

xmlTextReader.Close();
dataBridge SqlConnection.Close();

if (shortAETAnswer.AETQID == 0)
    throw new Exception("ShortAETAnswer not found!");

return shortAETAnswer;

/// <summary>
/// Finds the properties of AETAnswers for a particular Class ID (without the answers).
/// </summary>
/// <param name="aetQID">The associated AETQuestion ID.</param>
/// <param name="classID">The associated Class ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <exception cref="System.Exception">Thrown when AETAnswers aren't found.</exception>
/// <returns>Returns an array of ShortAETAnswers that were found.</returns>
public static ShortAETAnswer[] FindShortAETAnswers(int aetQID, string classID, AETType aetType, DataBridge dataBridge)
{
    SqlCommand sqlCommand = new SqlCommand("FindShortAETAnswers", dataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@aetQID", SqlDbType.Int); sqlCommand.Parameters["@aetQID"].Value = aetQID;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = classID;
    sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar, AETLimitations.AETType);
    sqlCommand.Parameters["@aetQType"].Value = aetType;
    dataBridge.SqlConnection.Open();

// Read the sql server's xml data
XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.ExecuteXmlReader();

// Retrieve the information and deserialize it
ArrayList shortAETAnswers = new ArrayList();
XmlSerializer xmlSerializer = new XmlSerializer(typeof(ShortAETAnswer));
while (xmlTextReader.Read())
if (xmlSerializer.CanDeserialize(xmlTextReader))
    shortAETAnswers.Add(xmlSerializer.Deserialize(xmlTextReader));

xmlTextReader.Close();
dataBridge.SqlConnection.Close();

if (shortAETAnswers.Count == 0)
    throw new Exception("ShortAETAnswers not found!");

return (ShortAETAnswer[])shortAETAnswers.ToArray(typeof(ShortAETAnswer));

#endregion

/// <summary>
/// An abstract class which provides shared properties and methods for AETAnswers.
/// </summary>
public abstract class AETAnswer
{
    #region Properties
    /// <summary>
    /// The UserID associated with the AETAnswer.
    /// </summary>
    private int _userID;
    [XmlAttribute("userID")]
    public int UserID
    {
        get { return this._userID; }
        set { this._userID = value; }
    }

    /// <summary>
    /// The ID of the AETQuestion associated with the AETAnswer.
    /// </summary>
    private int _aetQID;
    [XmlAttribute("aetQID")]
    public int AETQID
    {
        get { return this._aetQID; }
        set { this._aetQID = value; }
    }
    /// <summary>
private AETType _aetType;
[XmlAttribute("aetType")]
public AETType AETType
{
    get { return this._aetType; }
    set { this._aetType = value; }
}

private string _classID;
[XmlAttribute("classID")]
public string ClassID
{
    get { return this._classID; }
    set { this._classID = value; }
}

private DateTime _submittedDateTime;
[XmlAttribute("submittedDateTime")]
public DateTime SubmittedDateTime
{
    get { return this._submittedDateTime; }
    set { this._submittedDateTime = value; }
}

private bool _marked;
[XmlAttribute("marked")]
public bool Marked
{
    get { return this._marked; }
    set { this._marked = value; }
}

#endregion

#region Constructors
public AETAnswer() {}

#endregion

/// The AETType of the associated AETQuestion.
/// </summary>
private AETType _aetType;
[XmlAttribute("aetType")]
public AETType AETType
{
    get { return this._aetType; }
    set { this._aetType = value; }
}

/// The ClassID associated class.
/// </summary>
private string _classID;
[XmlAttribute("classID")]
public string ClassID
{
    get { return this._classID; }
    set { this._classID = value; }
}

/// The date that the answer was submitted.
/// </summary>
private DateTime _submittedDateTime;
[XmlAttribute("submittedDateTime")]
public DateTime SubmittedDateTime
{
    get { return this._submittedDateTime; }
    set { this._submittedDateTime = value; }
}

/// Whether or not the AETAnswer has been completely marked.
/// </summary>
private bool _marked;
[XmlAttribute("marked")]
public bool Marked
{
    get { return this._marked; }
    set { this._marked = value; }
}

#region Constructors
public AETAnswer() {}

/// Used for XmlSerializer purposes.
/// </summary>
public AETAnswer() {}

/// Abstract constructor.
/// </summary>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
```csharp
public AETAnswer(DataBridge dataBridge)
{
    this.DataBridge = dataBridge;
}
#endregion
#region Data Management Properties
/// <summary>
/// The DataBridge that connects to the data.
/// </summary>
private DataBridge _dataBridge = null;
[XmlIgnore()]
public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}
#endregion
#region Data Management Methods
/// <summary>
/// Uses current instance's properties and stores the data.
/// </summary>
public void Create()
{
    string aetAFilePath = null;
    string paddedUserID = null;

    // Get the filepath
    switch (this.AETType)
    {
    case AETType.Assignment:
        aetAFilePath = this.DataBridge.
            GetClassAssignmentAnswerFilePath(this.ClassID,
                this.AETQID, this.UserID, out paddedUserID, false);
        break;
    case AETType.Exam:
        aetAFilePath = this.DataBridge.
            GetClassExamAnswerFilePath(this.ClassID, this.
                AETQID, this.UserID, out paddedUserID, false);
        break;
    case AETType.Tutorial:
        aetAFilePath = this.DataBridge.
            GetClassTutorialAnswerFilePath(this.ClassID, this.
                AETQID, this.UserID, out paddedUserID, false);
        break;
    }

    // Write information to sql server
    SqlCommand sqlCommand = new SqlCommand("CreateAETAnswer",
        this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
```
sqlCommand.Parameters.AddWithValue("@aetQID", SqlDbType.BigInt);
sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
sqlCommand.Parameters.AddWithValue("@classID", SqlDbType.VarChar, ClassLimitations.ClassID);
sqlCommand.Parameters["@classID"].Value = this.ClassID;
sqlCommand.Parameters.AddWithValue("@aetQType", SqlDbType.VarChar, AETLimitations.AETType);
sqlCommand.Parameters["@aetQType"].Value = this.AETType.ToString();
sqlCommand.Parameters.AddWithValue("@userID", SqlDbType.BigInt);
sqlCommand.Parameters["@userID"].Value = this.UserID;
sqlCommand.Parameters.AddWithValue("@aetASubmittedDate", SqlDbType.DateTime);
sqlCommand.Parameters["@aetASubmittedDate"].Value = this.SubmittedDateTime;
sqlCommand.Parameters.AddWithValue("@aetAMarked", SqlDbType.Bit);
sqlCommand.Parameters["@aetAMarked"].Value = this.Marked;
this.DataBridge.SqlConnection.Open();
sqlCommand.ExecuteNonQuery();
this.DataBridge.SqlConnection.Close();

/// Write information to xml file
XmlSerializer xmlSerializer = new XmlSerializer(this.GetType());
XmlTextWriter xmlTextWriter;
xmlSerializer.Serialize(xmlTextWriter = new XmlTextWriter(aetAFilePath, Encoding.UTF8), this);
xmlTextWriter.Close();

/// <summary>
/// Uses current instance’s properties and re-stores the data.
/// </summary>
public void Alter()
{
    string aetAFilePath = null;
    string paddedUserID = null;

    // Get the filepath
    switch (this.AETType)
    {
    case AETType.Assignment:
        aetAddress = this.DataBridge.GetClassAssignmentAnswerFilePath(this.ClassID,
                                                            this.AETQID, this.UserID,
                                                            out paddedUserID,
                                                            true);
        break;
    case AETType.Exam:
        aetAddress = this.DataBridge.GetClassExamAnswerFilePath(this.ClassID,
                                                                  this.AETQID, this.UserID,
                                                                  out paddedUserID, true);
        break;
    }
case AETType.Tutorial:
    string aetAFilePath = this.DataBridge.GetClassTutorialAnswerFilePath(this.ClassID, this.AETQID, this.UserID, out paddedUserID, true);
    break;

    // Write information to sql server
    SqlCommand sqlCommand = new SqlCommand("AlterAETAnswer", this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
    sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = this.ClassID;
    sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar, AETLimitations.AETType);
    sqlCommand.Parameters["@aetQType"].Value = this.AETType.ToString();
    sqlCommand.Parameters.Add("@userID", SqlDbType.BigInt);
    sqlCommand.Parameters["@userID"].Value = this.UserID;
    sqlCommand.Parameters.Add("@aetASubmittedDate", SqlDbType.DateTime);
    sqlCommand.Parameters["@aetASubmittedDate"].Value = this.SubmittedDateTime;
    sqlCommand.Parameters.Add("@aetAMarked", SqlDbType.Bit);
    sqlCommand.Parameters["@aetAMarked"].Value = this.Marked;
    this.DataBridge.SqlConnection.Open();
    sqlCommand.ExecuteNonQuery();
    this.DataBridge.SqlConnection.Close();

    // Write information to xml file
    XmlSerializer xsSerializer = new XmlSerializer(this.GetType());
    XmlTextWriter xwTextWriter;
    xsSerializer.Serialize(xwTextWriter = new XmlTextWriter(aetAFilePath, Encoding.UTF8), this);
    xwTextWriter.Close();

    } // <summary>
    /// Uses the current instance’s properties and deletes the stored data.
    /// </summary>
    public void Delete()
    {
        string aetAFilePath = null;
        string paddedUserID = null;

        // Get the filepath
        switch (this.AETType)
        {
        }
case AETType.Assignment:
    aetAFilePath = this.DataBridge.GetClassAssignmentAnswerFilePath(this.ClassID, this.AETQID, this.UserID, out paddedUserID, true);
    break;

case AETType.Exam:
    aetAFilePath = this.DataBridge.GetClassExamAnswerFilePath(this.ClassID, this.AETQID, this.UserID, out paddedUserID, true);
    break;

case AETType.Tutorial:
    aetAFilePath = this.DataBridge.GetClassTutorialAnswerFilePath(this.ClassID, this.AETQID, this.UserID, out paddedUserID, true);
    break;
}

// Write information to sql server
SqlCommand sqlCommand = new SqlCommand("DeleteAETAnswer", this.DataBridge.SqlConnection);
sqlCommand.CommandType = CommandType.StoredProcedure;
sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt); sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
sqlCommand.Parameters["@classID"].Value = this.ClassID;
sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar, AETLimitations.AETType);
sqlCommand.Parameters["@aetQType"].Value = this.AETType.ToString();
sqlCommand.Parameters.Add("@userID", SqlDbType.BigInt); sqlCommand.Parameters["@userID"].Value = this.UserID;
this.DataBridge.SqlConnection.Open(); sqlCommand.ExecuteNonQuery(); this.DataBridge.SqlConnection.Close();

// Delete xml file
File.Delete(aetAFilePath);

/// <summary>
/// An inherited class of AETAnswer for Information answers.
/// </summary>
[XmlRoot("AETAnswer")]
public sealed class AETInformationAnswer : AETAnswer
{
    #region Properties
    /// <summary>
    /// The filename of where the answer exists.
    /// </summary>
private string _aetAFilename;

[XmlElement("AnswerFilename")]

public string AETAFilename
{
    get { return this._aetAFilename; }
    set { this._aetAFilename = value; }
}

#region Constructors

/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>

public AETInformationAnswer() : base(null) {}

/// <summary>
/// Main constructor.
/// </summary>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>

public AETInformationAnswer(DataBridge dataBridge) : base(dataBridge) {}

#endregion

#region Static Data Retrieving Methods

/// <summary>
/// Finds an AETInformationAnswer.
/// </summary>
/// <param name="aetQID">The associated AETQuestion ID.</param>
/// <param name="classID">The associated Class ID.</param>
/// <param name="userId">The owner of the AETAnswer.</param>
/// <param name="aetType">The type of the AETAnswer (Assignment, Exam, or Tutorial).</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns the found class AETInformationAnswer. Null if not found.</returns>

public static AETInformationAnswer FindAETInformationAnswer(
    int aetQID, string classID, int userId, AETType aetType, DataBridge dataBridge)
{
    string aetAFilePath = null;
    string paddedUserID = null;

    try
    {
        // Get the filepath
        switch (aetType)
        {
            case AETType.Assignment:
                aetAFilePath = dataBridge.GetClassAssignmentAnswerFilePath(classID, aetQID, userId, out paddedUserID, true);
                break;
        }
    }

    return new AETInformationAnswer(aetAFilePath, paddedUserID);

}
case AETType.Exam:
    aetAFilePath = dataBridge.
    GetClassExamAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
    break;

case AETType.Tutorial:
    aetAFilePath = dataBridge.
    GetClassTutorialAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
    break;

try
{
    // Not found
    return null;
}

// Retrieve the information from the xml file
XmlSerializer xmlSerializer = new XmlSerializer(typeof(AETInformationAnswer));
XmlTextReader xmlTextReader;
AETInformationAnswer newAETA = xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(aetAFilePath)) as AETInformationAnswer;
xmTextReader.Close();

    return newAETA;
#endregion

/// <summary>
/// An inherited class of AETAnswer for Multiple Choice answers.
/// </summary>
[XmlRoot("AETAnswer")]
public sealed class AETMultipleChoiceAnswer : AETAnswer
{
    #region Properties
    /// <summary>
    /// An array of answers.
    /// </summary>
    [XmlArray("Answers")]
    [XmlArrayItem("Answer")]
    public MultipleChoiceAnswer[] answers;
    #endregion

    #region Constructors
    /// <summary>
    /// Used for XmlSerializer purposes.
    /// </summary>
    public AETMultipleChoiceAnswer() : base(null) {}
    /// <summary>
    /// Main constructor.
    /// </summary>

/// <summary>
/// The DataBridge that connects to the data.</summary>

public AETMultipleChoiceAnswer(DataBridge dataBridge) : base(dataBridge) {}

#region Static Data Retrieving Methods

/// <summary>
/// Finds an AETMultipleChoiceAnswer.
/// </summary>
/// <param name="aetQID">The associated AETQuestion ID.</param>
/// <param name="classID">The associated Class ID.</param>
/// <param name="userID">The owner of the AETAnswer.</param>
/// <param name="aetType">The type of the AETAnswer (Assignment, Exam, or Tutorial).</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns the found class AETMultipleChoiceAnswer. Null if not found.</returns>

public static AETMultipleChoiceAnswer FindAETMultipleChoiceAnswer(int aetQID, string classID, int userID, AETType aetType, DataBridge dataBridge)
{
    string aetAFilePath = null;
    string paddedUserID = null;

    try
    {
        // Get the filepath
        switch (aetType)
        {
            case AETType.Assignment:
                aetAFilePath = dataBridge.GetClassAssignmentAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
                break;
            case AETType.Exam:
                aetAFilePath = dataBridge.GetClassExamAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
                break;
            case AETType.Tutorial:
                aetAFilePath = dataBridge.GetClassTutorialAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
                break;
        }
    }
    catch
    {
        // Not found
        return null;
    }

#endregion
// Retrieve the information from the xml file
XmlSerializer xmlSerializer = new XmlSerializer(typeof(AETMultipleChoiceAnswer));
XmlTextReader xmlTextReader;
AETMultipleChoiceAnswer newAETA = xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(aetAFilePath)) as AETMultipleChoiceAnswer;
xmlTextReader.Close();

return newAETA;

/// <summary>
/// An inherited class of AETAnswer for Short Response answers.
/// </summary>
public sealed class AETShortResponseAnswer : AETAnswer
{
    #region Properties
    /// <summary>
    /// An array of answers.
    /// </summary>
    [XmlArray("Answers")]
    [XmlArrayItem("Answer")]
    public ShortResponseAnswer[] answers;
    #endregion

    #region Constructors
    /// <summary>
    /// Used for XmlSerializer purposes.
    /// </summary>
    public AETShortResponseAnswer() : base(null) {}

    /// <summary>
    /// Main constructor.
    /// </summary>
    /// <param name="dataBridge">The DataBridge that connects to the data.</param>
    public AETShortResponseAnswer(DataBridge dataBridge) : base(dataBridge) {}
    #endregion

    #region Static Data Retrieving Methods
    /// <summary>
    /// Finds an AETShortResponseAnswer.
    /// </summary>
    /// <param name="aetQID">The associated AETQuestion ID.</param>
    /// <param name="classID">The associated Class ID.</param>
    /// <param name="userID">The owner of the AETAnswer.</param>
    /// <param name="aetType">The type of the AETAnswer (Assignment, Exam, or Tutorial).</param>
    }
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns the found class AETShortResponseAnswer. Null if not found.</returns>

public static AETShortResponseAnswer
FindAETShortResponseAnswer(int aetQID, string classID, int userID, AETType aetType, DataBridge dataBridge)
{
    string aetAFilePath = null;
    string paddedUserID = null;

    try
    {
        // Get the filepath
        switch (aetType)
        {
        case AETType.Assignment:
            aetAFilePath = dataBridge.GetClassAssignmentAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
            break;
        case AETType.Exam:
            aetAFilePath = dataBridge.GetClassExamAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
            break;
        case AETType.Tutorial:
            aetAFilePath = dataBridge.GetClassTutorialAnswerFilePath(classID, aetQID, userID, out paddedUserID, true);
            break;
        }
    }
    catch
    {
        return null;
    }

    // Retrieve the information from the xml file
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(AETShortResponseAnswer));
    XmlTextReader xmlTextReader;
    AETShortResponseAnswer newAETA = xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(aetAFilePath)) as AETShortResponseAnswer;
    xmlTextReader.Close();

    return newAETA;
}

Listing H.95: AETQuestion.cs

using System;
namespace OnlineLearningEnvironment
{
    #region Enums
    /// <summary>
    /// The type of the AETQuestion.
    /// </summary>
    public enum AETType
    {
        /// <summary>
        /// An assignment.
        /// </summary>
        Assignment,
        /// <summary>
        /// An exam.
        /// </summary>
        Exam,
        /// <summary>
        /// A tutorial.
        /// </summary>
        Tutorial,
        /// <summary>
        /// Enum hasn’t be set.
        /// </summary>
        NotSet
    }
    /// <summary>
    /// The AETQuestion’s question type.
    /// </summary>
    public enum QuestionType
    {
        /// <summary>
        /// Information on the assessment questions.
        /// </summary>
        Information,
        /// <summary>
        /// Short response assessment questions.
        /// </summary>
        ShortResponse,
        /// <summary>
        /// Multiple choice assessment questions.
        /// </summary>
        MultipleChoice
    }
    #endregion
    #region Structures
    /// <summary>

/// Properties that belong to an InformationQuestion.
/// </summary>
public struct InformationQuestion
{
    /// <summary>
    /// The question's unique ID.
    /// </summary>
    [XmlAttribute("questionID")]
    public int questionID;
    /// <summary>
    /// The question's title.
    /// </summary>
    [XmlElement("Title")]
    public string title;
    /// <summary>
    /// The question's question.
    /// </summary>
    [XmlElement("Question")]
    public string question;
}
/// Properties that belong to an ShortResponseQuestion.
/// </summary>
public struct ShortResponseQuestion
{
    /// <summary>
    /// The question's unique ID.
    /// </summary>
    [XmlAttribute("questionID")]
    public int questionID;
    /// <summary>
    /// The question's title.
    /// </summary>
    [XmlElement("Title")]
    public string title;
    /// <summary>
    /// The question's question.
    /// </summary>
    [XmlElement("Question")]
    public string question;
}
/// Properties that belong to an MultipleChoiceQuestion.
/// </summary>
public struct MultipleChoiceQuestion
{
    /// <summary>
    /// The question's unique ID.
    /// </summary>
    [XmlAttribute("questionID")]
    public int questionID;
    /// <summary>
    /// The question's title.
    /// </summary>
    [XmlElement("Title")]
    public string title;
    /// <summary>
    /// The question's question.
    /// </summary>
    [XmlElement("Question")]
    public string question;
}
```csharp
public string title;
/// <summary>
/// The question's question.
/// </summary>
[XmlElement("Question")]
public string question;
/// <summary>
/// True if the answer is correct, false if otherwise.
/// </summary>
[XmlAttribute("correctChoice")]
public int correctChoice;
/// <summary>
/// An array of choices to accompany the question.
/// </summary>
[XmlArray("Choices")]
[XmlArrayItem("Choice")]
public MultipleChoiceChoice[] choices;
}
/// <summary>
/// Properties that belong to an MultipleChoiceQuestion's choice
/// </summary>
public struct MultipleChoiceChoice
{
    /// <summary>
    /// The choice's unique ID.
    /// </summary>
    [XmlAttribute("choiceID")]
    public int choiceID;
    /// <summary>
    /// The choice's text.
    /// </summary>
    [XmlText()]
    public string choiceText;
}
#endregion

/// <summary>
/// A structure similar to AETQuestion, but without the question
/// properties.
/// </summary>
[XmlRoot("AETQuestion")]
public struct ShortAETQuestion
{
    #region Properties
    /// <summary>
    /// The unique ID.
    /// </summary>
    private int _aetQID;
    [XmlAttribute("aetQID")]
    public int AETQID
    {
        get { return this._aetQID; }
        set { this._aetQID = value; }
    }
```
private string _classID;
[XmlAttribute("classID")]
public string ClassID
{
    get { return this._classID; }
    set { this._classID = value; }
}

private string _title;
[XmlElement("Title")]
public string Title
{
    get { return this._title; }
    set { this._title = value; }
}

private string _description;
[XmlElement("Description")]
public string Description
{
    get { return this._description; }
    set { this._description = value; }
}

private AETType _aetType;
[XmlAttribute("aetType")]
public AETType AETType
{
    get { return this._aetType; }
    set { this._aetType = value; }
}

private QuestionType _questionType;
[XmlAttribute("questionType")]
public QuestionType QuestionType
{
    get { return this._questionType; }
}
```csharp
    set { this._questionType = value; }

    /// <summary>
    /// The date & time students can view the question.
    /// </summary>
    private DateTime _viewQuestionDateTime;
    [XmlAttribute("viewQuestionDateTime")]
    public DateTime ViewQuestionDateTime
    {
        get { return this._viewQuestionDateTime; }
        set { this._viewQuestionDateTime = value; }
    }

    /// <summary>
    /// The date & time the answers are due.
    /// </summary>
    private DateTime _dueDateTime;
    [XmlAttribute("dueDateTime")]
    public DateTime DueDateTime
    {
        get { return this._dueDateTime; }
        set { this._dueDateTime = value; }
    }

    /// <summary>
    /// The date & time no more answers are accepted.
    /// </summary>
    private DateTime _haltSubmissionsDateTime;
    [XmlAttribute("haltSubmissionsDateTime")]
    public DateTime HaltSubmissionsDateTime
    {
        get { return this._haltSubmissionsDateTime; }
        set { this._haltSubmissionsDateTime = value; }
    }

    /// <summary>
    /// The date & time students are able to see their marked answers.
    /// </summary>
    private DateTime _viewMarkedAnswersDateTime;
    [XmlAttribute("viewMarkedAnswersDateTime")]
    public DateTime ViewMarkedAnswersDateTime
    {
        get { return this._viewMarkedAnswersDateTime; }
        set { this._viewMarkedAnswersDateTime = value; }
    }

    /// <summary>
    /// The amount of questions.
    /// </summary>
    private int _amountOfQuestions;
    [XmlAttribute("amountOfQuestions")]
    public int AmountOfQuestions
    {
```
```csharp
get { return this._amountOfQuestions; }
set { this._amountOfQuestions = value; }
}
#endregion

#region Static Data Retrieving Methods
/// <summary>
/// Finds a specific ShortAETQuestion.
/// </summary>
/// <param name="aetQID">The associated AETQuestion ID.</param>
/// <param name="classID">The associated Class ID.</param>
/// <param name="aetType">The type of the AETAnswer (Assignment, Exam, or Tutorial).</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns the found ShortAETQuestion. Null if not found.</returns>

public static ShortAETQuestion FindShortAETQuestion(int aetQID, string classID, AETType aetType, DataBridge dataBridge)
{
    SqlCommand sqlCommand = new SqlCommand("FindShortAETQuestion", dataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
    sqlCommand.Parameters["@aetQID"].Value = aetQID;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = classID;
    sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar, AETLimitations.AETType);
    sqlCommand.Parameters["@aetQType"].Value = aetType;

    dataBridge.SqlConnection.Open();
    XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.ExecuteXmlReader();
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(ShortAETQuestion));

    ShortAETQuestion shortAETQuestion = new ShortAETQuestion();
    xmlSerializer.Deserialize(xmlTextReader);

    xmlTextReader.Close();
    dataBridge.SqlConnection.Close();

    if (shortAETQuestion.AETQID == 0)
        throw new Exception("ShortAETQuestion not found!");

    return shortAETQuestion;
```
/// <summary>
/// Finds the properties of AETQuestions for a particular Class ID (without the questions).
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="aetType">The associated AETType.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns an array of ShortAETQuestions that were found.</returns>

public static ShortAETQuestion[] FindShortAETQuestions(string classID, AETType aetType, DataBridge dataBridge) {
    SqlCommand sqlCommand = new SqlCommand("FindShortAETQuestions", dataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = classID;
    sqlCommand.Parameters.Add("@aetQType", SqlDbType.Char, AETLimitations.AETType);
    sqlCommand.Parameters["@aetQType"].Value = aetType.ToString();

    dataBridge.SqlConnection.Open();
    XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.ExecuteXmlReader();

    // Deserialise questions
    ArrayList shortAETQuestions = new ArrayList();
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(ShortAETQuestion));
    while (xmlTextReader.Read()) {
        Console.WriteLine("∗");
        while (xmlSerializer.CanDeserialize(xmlTextReader))
            shortAETQuestions.Add(xmlSerializer.Deserialize(xmlTextReader));
    }

    xmlTextReader.Close();
    dataBridge.SqlConnection.Close();

    if (shortAETQuestions.Count == 0)
        throw new Exception("ShortAETQuestions not found!");

    return (ShortAETQuestion[])shortAETQuestions.ToArray(typeof(ShortAETQuestion));
}
/// An abstract class which provides shared properties and methods for AETQuestions.
/// </summary>
public abstract class AETQuestion
{
    #region Properties
    /// <summary>
    /// The unique ID.
    /// </summary>
    private int _aetQID;
    [XmlAttribute("aetQID")]
    public int AETQID
    {
        get { return this._aetQID; }
        set { this._aetQID = value; }
    }

    /// <summary>
    /// The associated class ID.
    /// </summary>
    private string _classID;
    [XmlAttribute("classID")]
    public string ClassID
    {
        get { return this._classID; }
        set { this._classID = value; }
    }

    /// <summary>
    /// The associated title.
    /// </summary>
    private string _title;
    [XmlElement("Title")]
    public string Title
    {
        get { return this._title; }
        set { this._title = value; }
    }

    /// <summary>
    /// The associated description (usually 'Assignment', 'Exam', or 'Tutorial').
    /// </summary>
    private string _description;
    [XmlElement("Description")]
    public string Description
    {
        get { return this._description; }
        set { this._description = value; }
    }

    /// <summary>
    /// The associated type.
    /// </summary>
    private AETType _aetType;
[XmlAttribute("aetType")]
public AETType AETType
{
    get { return this._aetType; }
    set { this._aetType = value; }
}

/// <summary>
/// The associated question type.
/// </summary>
private QuestionType _questionType;
[XmlAttribute("questionType")]
public QuestionType QuestionType
{
    get { return this._questionType; }
    set { this._questionType = value; }
}

/// <summary>
/// The date & time students can view the question.
/// </summary>
private DateTime _viewQuestionDateTime;
[XmlAttribute("viewQuestionDateTime")]
public DateTime ViewQuestionDateTime
{
    get { return this._viewQuestionDateTime; }
    set { this._viewQuestionDateTime = value; }
}

/// <summary>
/// The date & time the answers are due.
/// </summary>
private DateTime _dueDateTime;
[XmlAttribute("dueDateTime")]
public DateTime DueDateTime
{
    get { return this._dueDateTime; }
    set { this._dueDateTime = value; }
}

/// <summary>
/// The date & time no more answers are accepted.
/// </summary>
private DateTime _haltSubmissionsDateTime;
[XmlAttribute("haltSubmissionsDateTime")]
public DateTime HaltSubmissionsDateTime
{
    get { return this._haltSubmissionsDateTime; }
    set { this._haltSubmissionsDateTime = value; }
}

/// <summary>
/// The date & time students are able to see their marked answers.
/// </summary>
private DateTime _viewMarkedAnswersDateTime;

    [XmlAttribute("viewMarkedAnswersDateTime")]
    public DateTime ViewMarkedAnswersDateTime
    {
        get { return this._viewMarkedAnswersDateTime; }
        set { this._viewMarkedAnswersDateTime = value; }
    }

    /// <summary>
    /// The amount of questions.
    /// </summary>
    private int _amountOfQuestions;
    [XmlAttribute("amountOfQuestions")]
    public int AmountOfQuestions
    {
        get { return this._amountOfQuestions; }
        set { this._amountOfQuestions = value; }
    }

    #endregion

    #region Constructors
    /// <summary>
    /// Abstract constructor.
    /// </summary>
    /// <param name="questionType">The associated question type</param>
    public AETQuestion(QuestionType questionType)
    {
        this._questionType = questionType;
    }

    /// <summary>
    /// Abstract constructor.
    /// </summary>
    /// <param name="aetType">The associated AET type.</param>
    /// <param name="questionType">The associated question type</param>
    /// <param name="dataBridge">The DataBridge that connects to the data.</param>
    public AETQuestion(AETType aetType, QuestionType questionType, DataBridge dataBridge)
    {
        this._aetType = aetType;
        this._questionType = questionType;
        this._dataBridge = dataBridge;
    }

    #endregion

    #region Data Management Properties
    /// <summary>
    /// The DataBridge that connects to the data.
    /// </summary>
    private DataBridge _dataBridge = null;
    [XmlIgnore()]
    public DataBridge DataBridge
    {
    }
get { return this.datavBridge; }
set { this.datavBridge = value; }

#endregion

#region Data Management Methods
/// <summary>
/// Uses current instance’s properties and stores the data.
/// </summary>
public int Create()
{
    string aetQFilePath = null;
    string aetAnswerDirectory = null;
    int numericAETQID = 0;
    string paddedAETQID = null;

    switch (this.AETType)
    {
    case AETType.Assignment:
        aetQFilePath = this.datavBridge.
            GetNextClassAssignmentFilePath(this.ClassID,
            out numericAETQID, out paddedAETQID);
        aetAnswerDirectory = this.datavBridge.
            GetClassAssignmentAnswersDirectory(this.ClassID,
            numericAETQID, false);
        break;
    case AETType.Exam:
        aetQFilePath = this.datavBridge.
            GetNextClassExamFilePath(this.ClassID,
            out numericAETQID, out paddedAETQID);
        aetAnswerDirectory = this.datavBridge.
            GetClassExamAnswersDirectory(this.ClassID,
            numericAETQID, false);
        break;
    case AETType.Tutorial:
        aetQFilePath = this.datavBridge.
            GetNextClassTutorialFilePath(this.ClassID,
            out numericAETQID, out paddedAETQID);
        aetAnswerDirectory = this.datavBridge.
            GetClassTutorialAnswersDirectory(this.ClassID,
            numericAETQID, false);
        break;
    }

    this.AETQID = numericAETQID;

    // Write information to sql server
    SqlCommand sqlCommand = new SqlCommand("CreateAETQuestion", this.datavBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt).Value = numericAETQID;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID).Value = this.ClassID;
    sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt).Value = numericAETQID;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID).Value = this.ClassID;
schema.Parameters.AddWithValue("@aetQTitle", SqlDbType.VarChar, AETLimitations.Title);
schema.Parameters["@aetQTitle"].Value = this.Title;
schema.Parameters.AddWithValue("@aetQDescription", SqlDbType.VarChar, AETLimitations.Description);
schema.Parameters["@aetQDescription"].Value = this.Description;
schema.Parameters.AddWithValue("@aetQType", SqlDbType.VarChar, AETLimitations.AETType);
schema.Parameters["@aetQType"].Value = this.AETType.ToString();
schema.Parameters.AddWithValue("@aetQQuestionType", SqlDbType.VarChar, AETLimitations.AETQuestionType);
schema.Parameters["@aetQQuestionType"].Value = this.QuestionType.ToString();
schema.Parameters.AddWithValue("@aetQViewQuestionDateTime", SqlDbType.DateTime);
schema.Parameters["@aetQViewQuestionDateTime"].Value = this.ViewQuestionDateTime;
schema.Parameters.AddWithValue("@aetQDueDateTime", SqlDbType.DateTime);
schema.Parameters["@aetQDueDateTime"].Value = this.DueDateTime;
schema.Parameters.AddWithValue("@aetQHaltSubmissionDateTime", SqlDbType.DateTime);
schema.Parameters["@aetQHaltSubmissionDateTime"].Value = this.HaltSubmissionsDateTime;
schema.Parameters.AddWithValue("@aetQAmountOfQuestions", SqlDbType.Int);
schema.Parameters["@aetQAmountOfQuestions"].Value = this.AmountOfQuestions;
this.DataBridge.SqlConnection.Open();
schema.ExecuteNonQuery();
this.DataBridge.SqlConnection.Close();

// Create Answer directory for AETQuestion
if (!Directory.Exists(aetAnswerDirectory))
    Directory.CreateDirectory(aetAnswerDirectory);

// Write information to xml file
XmlSerializer xmlSerializer = new XmlSerializer(this.GetType());
XmlTextWriter xmlTextWriter;
xmlSerializer.Serialize(xmlTextWriter = new
    XmlTextWriter(aetQFilePath, Encoding.UTF8), this);
xmlTextWriter.Close();

return this.AETQID;
}
/// Uses current instance’s properties and re-stores the data.
/// </summary>
public void Alter()
{
    string aetQFilePath = null;
    string aetAnswerDirectory = null;
    string paddedAETQID = null;

    switch (this.AETType)
    {
    case AETType.Assignment:
        aetQFilePath = this.DataBridge.GetClassAssignmentFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
        aetAnswerDirectory = this.DataBridge.GetClassAssignmentAnswersDirectory(this.ClassID, this.AETQID, true);
        break;
    case AETType.Exam:
        aetQFilePath = this.DataBridge.GetClassExamFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
        aetAnswerDirectory = this.DataBridge.GetClassExamAnswersDirectory(this.ClassID, this.AETQID, true);
        break;
    case AETType.Tutorial:
        aetQFilePath = this.DataBridge.GetClassTutorialFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
        aetAnswerDirectory = this.DataBridge.GetClassTutorialAnswersDirectory(this.ClassID, this.AETQID, true);
        break;
    }

    // Write information to sql server
    SqlCommand sqlCommand = new SqlCommand("AlterAETQuestion", this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
    sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, 7);
    sqlCommand.Parameters["@classID"].Value = this.ClassID;
    sqlCommand.Parameters.Add("@aetQTitle", SqlDbType.VarChar, 100);
    sqlCommand.Parameters["@aetQTitle"].Value = this.Title;
    sqlCommand.Parameters.Add("@aetQDescription", SqlDbType.VarChar, 30);
    sqlCommand.Parameters["@aetQDescription"].Value = this.Description;
    sqlCommand.Parameters.Add("@aetQType", SqlDbType.VarChar, 20);
sqlCommand.Parameters["@aetQType"].Value = this.AETType.ToString();
sqlCommand.Parameters.Add("@aetQQuestionType", SqlDbType.VarChar, 20);
sqlCommand.Parameters["@aetQQuestionType"].Value = this.QuestionType.ToString();
sqlCommand.Parameters.Add("@aetQViewQuestionDateTime", SqlDbType.DateTime);
sqlCommand.Parameters["@aetQViewQuestionDateTime"].Value = this.ViewQuestionDateTime;
sqlCommand.Parameters.Add("@aetQDueDateTime", SqlDbType.DateTime);
sqlCommand.Parameters["@aetQDueDateTime"].Value = this.DueDateTime;
sqlCommand.Parameters.Add("@aetQHaltSubmissionDateTime", SqlDbType.DateTime);
sqlCommand.Parameters["@aetQHaltSubmissionDateTime"].Value = this.HaltSubmissionsDateTime;
sqlCommand.Parameters.Add("@aetQViewMarkedAnswersDateTime", SqlDbType.DateTime);
sqlCommand.Parameters["@aetQViewMarkedAnswersDateTime"].Value = this.ViewMarkedAnswersDateTime;
sqlCommand.Parameters.Add("@aetQAmountOfQuestions", SqlDbType.Int);
sqlCommand.Parameters["@aetQAmountOfQuestions"].Value = this.AmountOfQuestions;
this.DataBridge.SqlConnection.Open();
sqlCommand.ExecuteNonQuery();
this.DataBridge.SqlConnection.Close();

XmlSerializer xmlSerializer = new XmlSerializer(this.GetType());
XmlTextWriter xmlTextWriter;
xmlSerializer.Serialize(xmlTextWriter = new XmlTextWriter(aetQFilePath, Encoding.UTF8), this);
xmlTextWriter.Close();

/// <summary>
/// Uses the current instance's properties and deletes the stored data.
/// </summary>
public void Delete()
{
    string aetQFilePath = null;
    string aetAnswerDirectory = null;
    string paddedAETQID = null;

    switch (this.AETType)
    {
    case AETType.Assignment:
        aetQFilePath = this.DataBridge.GetClassAssignmentFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
        break;
    // Other cases...
    }
}
aetAnswerDirectory = this.DataBridge.GetClassAssignmentAnswersDirectory(this.ClassID, this.AETQID, true);
break;
case AETType.Exam:
aetQFilePath = this.DataBridge.GetClassExamFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
aetAnswerDirectory = this.DataBridge.GetClassExamAnswersDirectory(this.ClassID, this.AETQID, true);
break;
case AETType.Tutorial:
aetQFilePath = this.DataBridge.GetClassTutorialFilePath(this.ClassID, this.AETQID, out paddedAETQID, true);
aetAnswerDirectory = this.DataBridge.GetClassTutorialAnswersDirectory(this.ClassID, this.AETQID, true);
break;
}

// Delete record from sql server
SqlCommand sqlCommand = new SqlCommand("DeleteAETQuestion", this.DataBridge.SqlConnection);
sqlCommand.CommandType = CommandType.StoredProcedure;
sqlCommand.Parameters.Add("@aetQID", SqlDbType.BigInt);
sqlCommand.Parameters["@aetQID"].Value = this.AETQID;
sqlCommand.Parameters.Add("@classID", SqlDbType.Char, 7);
sqlCommand.Parameters["@classID"].Value = this.ClassID;
this.DataBridge.SqlConnection.Open();
sqlCommand.ExecuteNonQuery();
this.DataBridge.SqlConnection.Close();

// Delete xml file
File.Delete(aetQFilePath);
// Delete folder with answers
Directory.Delete(aetAnswerDirectory, true);

#endregion

/// <summary>
/// A sealed class that contains properties and methods of a Information Question.
/// </summary>
[XmlRoot("AETInformationQuestion")]
public sealed class AETInformationQuestion : AETQuestion
{
    #region Properties
    /// <summary>
    /// An array of information questions.
    /// </summary>
    private InformationQuestion[] _questions;
public InformationQuestion[] Questions
{
    get { return this._questions; }
    set { this._questions = value; }
}

/// <summary>
/// The filename of the provided solutions file.
/// </summary>
private string _aetSFilename;

public string AETSFilename
{
    get { return this._aetSFilename; }
    set { this._aetSFilename = value; }
}

#endregion

#region Constructors
/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>
public AETInformationQuestion() : base(QuestionType.Information) {}
/// <summary>
/// Main Constructor.
/// </summary>
public AETInformationQuestion(AETType aetType, DataBridge dataBridge) : base(aetType, QuestionType.Information, dataBridge) {}

#endregion

#region Static Data Retrieving Methods
/// <summary>
/// Find a specific AETInformationQuestion.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="aetQID">The associated AETQuestion</param>
/// <param name="aetType">The type of the AETAnswer (Assignment, Exam, or Tutorial).</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Return the found AETInformationQuestion. Null if not found.</returns>
public static AETInformationQuestion FindAETInformationQuestion(string classID, int aetQID, AETType aetType, DataBridge dataBridge)
{
    string aetQFilePath = null;
    string paddedAETQID = null;
}
try
{
    // Find path
    switch (aetType)
    {
        case AETType.Assignment:
            aetQFilePath = dataBridge.
                           GetClassAssignmentFilePath(classID, aetQID, out paddedAETQID, true);
            break;
        case AETType.Exam:
            aetQFilePath = dataBridge.
                           GetClassExamFilePath(classID, aetQID, out paddedAETQID, true);
            break;
        case AETType.Tutorial:
            aetQFilePath = dataBridge.
                           GetClassTutorialFilePath(classID, aetQID, out paddedAETQID, true);
            break;
    }

    catch
    {
        // Not found
        return null;
    }

    // De serialise the question
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(AETInformationQuestion));
    XmlTextReader xmlTextReader;
    AETInformationQuestion newAETQ = xmlSerializer.
                           Deserialize(xmlTextReader = new XmlTextReader(aetQFilePath)) as AETInformationQuestion;
    xmlTextReader.Close();

    return newAETQ;
}
#endregion

/// <summary>
/// A sealed class that contains properties and methods of a Multiple Choice Question.
/// </summary>
[XmlRoot("AETQuestion")]
public sealed class AETMultipleChoiceQuestion : AETQuestion
{
    #region Properties
    /// <summary>
    /// An array of multiple choice questions.
    /// </summary>
    private MultipleChoiceQuestion[] _questions;
public MultipleChoiceQuestion[] Questions
{
    get { return this._questions; }
    set { this._questions = value; }
}
#endregion

#region Constructors
/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>
public AETMultipleChoiceQuestion() : base(QuestionType.
    MultipleChoice) {}  
/// <summary>
/// Main Constructor.
/// </summary>
/// <param name="aetType">The associated AET type.</param>
/// <param name="dataBridge">The DataBridge that connects to
the data.</param>
public AETMultipleChoiceQuestion(AETType aetType , DataBridge
    dataBridge) : base(aetType , QuestionType.Information
, dataBridge) {}
#endregion

#region Static Data Retrieving Methods
/// <summary>
/// Find a specific AETMultipleChoiceQuestion.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="aetQID">The associated AETQuestion</param>
/// <param name="aetType">The type of the AETAnswer (Assignment, Exam, or Tutorial).</param>
/// <param name="dataBridge">The DataBridge that connects to
the data.</param>
/// <returns>Return the found AETMultipleChoiceQuestion. Null if not found.</returns>
public static AETMultipleChoiceQuestion
FindAETMultipleChoiceQuestion(string classID , int aetQID ,
    AETType aetType , DataBridge dataBridge)  
{
    string aetQFilePath = null;    
    string paddedAETQID = null;
    try
    {
        // Find path
        switch (aetType)  
        {
            case AETType.Assignment:
                aetQFilePath = dataBridge.
                    GetClassAssignmentFilePath(classID ,
                        aetQID , out paddedAETQID , true);
                break;
            }  
    }  
}
```csharp
    case AETType.Exam:
        aetQFilePath = dataBridge.
            GetClassExamFilePath(classID, aetQID, out
                paddedAETQID, true);
        break;
    case AETType.Tutorial:
        aetQFilePath = dataBridge.
            GetClassTutorialFilePath(classID, aetQID, 
                out paddedAETQID, true);
        break;

    catch
    {
        // Not found
        return null;
    }

    // Deserialise the question
    XmlSerializer xmlSerializer = new XmlSerializer(typeof( 
        AETMultipleChoiceQuestion));
    XmlTextReader xmlTextReader;
    AETMultipleChoiceQuestion newAETQ = xmlSerializer.
        Deserialize(xmlTextReader = new XmlTextReader( 
            aetQFilePath)) as AETMultipleChoiceQuestion;
    xmlTextReader.Close();

    return newAETQ;
}

/// <summary>
/// A sealed class that contains properties and methods of a 
/// Short Response Question.
/// </summary>
[XmlRoot("AETQuestion")]
public sealed class AETShortResponseQuestion : AETQuestion
{
    #region Properties
    /// <summary>
    /// An array of short response questions.
    /// </summary>
    private ShortResponseQuestion[] _questions;
    [XmlArray("Questions")]
    [XmlArrayItem("Question")]
    public ShortResponseQuestion[] Questions
    {
        get { return this._questions; }
        set { this._questions = value; }
    }

    /// <summary>
    /// An array of short response question's answers (entered 
    /// by Instructors).
    /// </summary>
    [XmlArray("Answers")]
    [XmlArrayItem("Answer")]
    public ShortResponseAnswer[] Answers
    {
        get { return this._answers; }
        set { this._answers = value; }
    }

    #endregion
```
private ShortResponseAnswer[] _answers;

[XmlArray("Answers")]
[XmlArrayItem("Answer")]
public ShortResponseAnswer[] Answers
{
    get { return this._answers; }
    set { this._answers = value; }
}
#endregion

#region Constructors
/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>
public AETShortResponseQuestion() : base(QuestionType.ShortResponse) {}

/// <summary>
/// Main Constructor.
/// </summary>
public AETShortResponseQuestion(AETType aetType, DataBridge dataBridge) : base(aetType, QuestionType.Information, dataBridge) {}
#endregion

#region Static Data Retrieving Methods
/// <summary>
/// Find a specific AETShortResponseQuestion.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="aetQID">The associated AETQuestion.</param>
/// <param name="aetType">The type of the AETAnswer (Assignment, Exam, or Tutorial).</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Return the found AETShortResponseQuestion. Null if not found.</returns>
public static AETShortResponseQuestion FindAETShortResponseQuestion(string classID, int aetQID, AETType aetType, DataBridge dataBridge)
{
    string aetQFilePath = null;
    string paddedAETQID = null;

    try
    {
        // Find path
        switch (aetType)
        {
            case AETType.Assignment:
                aetQFilePath = dataBridge.GetClassAssignmentFilePath(classID,
                GetClassAssignmentFilePath(classID,
            case AETType.Exam:
                aetQFilePath = dataBridge.GetClassExamFilePath(classID,
                GetClassExamFilePath(classID,
            case AETType.Tutorial:
                aetQFilePath = dataBridge.GetClassTutorialFilePath(classID,
                GetClassTutorialFilePath(classID,
                break;
        }
    }
}
break;

break;

break;

break;

catch
{
    // Not found
    return null;
}

// Deserialise the question
XmlSerializer xmlSerializer = new XmlSerializer(typeof(AETShortResponseQuestion));
XmlTextReader xmlTextReader;
AETShortResponseQuestion newAETQ = xmlSerializer.
    Deserialise(xmlTextReader = new XmlTextReader(
aetQFilePath)) as AETShortResponseQuestion;
xmlTextReader.Close();

return newAETQ;

} // endregion

Listing H.96: Authentication.cs

using System;
using System.Data;
using System.Data.SqlClient;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A sealed class that contains static methods to check authentication.
    /// </summary>
    public sealed class Authentication
    {
        #region Constructors
        /// <summary>
        /// Disabled constructor.
        /// </summary>
        /// <summary>
        private Authentication() {}
        } // endregion

        using System;
        using System.Data;
        using System.Data.SqlClient;

        namespace OnlineLearningEnvironment
        {
            /// <summary>
            /// A sealed class that contains static methods to check authentication.
            /// </summary>
            public sealed class Authentication
            {
                #region Constructors
                /// <summary>
                /// Disabled constructor.
                /// </summary>
                /// <summary>
                private Authentication() {}
                } // endregion
/// <summary>
/// The associated question type.
/// </summary>
public Authentication(DataBridge dataBridge)
{
    this._dataBridge = dataBridge;
}
#endregion

#region Data Management Properties
/// <summary>
/// The DataBridge that connects to the data.
/// </summary>
private DataBridge _dataBridge = null;
public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}
#endregion

#region Static Data Retrieving Methods
/// <summary>
/// Check username and password.
/// </summary>
/// <param name="username">The associated username.</param>
/// <param name="password">The associated password (in non-encrypted string form).</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns true if password is correct. False if otherwise.</returns>
public static bool CheckAuthentication(int username, string password, DataBridge dataBridge)
{
    SqlCommand sqlCommand = new SqlCommand("Authentication", dataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
    sqlCommand.Parameters["@userID"].Value = username;
    sqlCommand.Parameters.Add("@password", SqlDbType.VarChar, UserLimitations.Password);
    sqlCommand.Parameters["@password"].Value = MD5Encryption.EncryptStringToByteString(password);
    sqlCommand.Parameters.Add("RT_VAL", SqlDbType.Int);
    sqlCommand.Parameters["RT_VAL"].Direction = ParameterDirectionReturnValue;
dataBridge.SqlConnection.Open();
    sqlCommand.ExecuteNonQuery();
dataBridge.SqlConnection.Close();

    // Check returned value from SQL
    if ((int)sqlCommand.Parameters["RT_VAL"].Value == 0)
        return false;
    else

```csharp
using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// A sealed class that contains methods and properties for classes.
    /// </summary>
    [XmlRoot("Class")]
    public sealed class Class
    {
        #region Properties
        /// The unique ID.
        /// </summary>
        private string _classID;
        [XmlAttribute("classID")]
        public string ClassID
        {
            get { return this._classID; }
            set { this._classID = value; }
        }

        /// The associated class name.
        /// </summary>
        private string _className;
        [XmlElement("Name")]
        public string ClassName
        {
            get { return this._className; }
            set { this._className = value; }
        }

        /// Whether or not non-enrolled users can visit the class’ section.
        /// </summary>
        private bool _denyOtherUsers;
        [XmlAttribute("denyOtherUsers")]
        public bool DenyOtherUsers
        {
```
```csharp
get { return this._denyOtherUsers; }
set { this._denyOtherUsers = value; }
}
#endregion

#region Constructors
/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>
public Class() {}
/// <summary>
/// Main Constructor.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
public Class(string classID, DataBridge dataBridge)
{
    this._classID = classID;
    this._dataBridge = dataBridge;
}
#endregion

#region Data Management Properties
/// <summary>
/// The DataBridge that connects to the data.
/// </summary>
private DataBridge _dataBridge = null;
[XmlIgnore()]
public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}
#endregion

#region Data Management Methods
/// <summary>
/// Uses current instance’s properties and stores the data.
/// </summary>
public string Create()
{
    if (Directory.Exists(this.DataBridge.GetClassDirectory( this.ClassID, false)))
        throw new Exception("ClassID exists!");

    // Create SQL Server DB entry
    SqlCommand sqlCommand = new SqlCommand("CreateClass", this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@classID", SqlDbType.VarChar, ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = this.ClassID;
    sqlCommand.Parameters.Add("@className", SqlDbType.VarChar, ClassLimitations.ClassName);
```
sqlCommand.Parameters["@className"].Value = this.ClassName;
sqlCommand.Parameters.Add("@denyOtherUsers", SqlDbType.Bit);
sqlCommand.Parameters["@denyOtherUsers"].Value = this.DenyOtherUsers;
this.DataBridge.SqlConnection.Open();
sqlCommand.ExecuteNonQuery();
this.DataBridge.SqlConnection.Close();

    // Create Class folders
    Directory.CreateDirectory(this.DataBridge.GetClassDirectory(this.ClassID, false));
    Directory.CreateDirectory(this.DataBridge.GetClassAnnouncementsDirectory(this.ClassID, false));
    Directory.CreateDirectory(this.DataBridge.GetClassAssignmentsDirectory(this.ClassID, false));
    Directory.CreateDirectory(this.DataBridge.GetClassExamsDirectory(this.ClassID, false));
    Directory.CreateDirectory(this.DataBridge.GetClassThreadsDirectory(this.ClassID, false));
    Directory.CreateDirectory(this.DataBridge.GetClassNotesDirectory(this.ClassID, false));
    Directory.CreateDirectory(this.DataBridge.GetClassTutorialsDirectory(this.ClassID, false));

    return this.ClassID;
};

    /// <summary>
    /// Uses current instance's properties and re-stores the data.
    /// </summary>
    public void Alter()
    {
        SqlCommand sqlCommand = new SqlCommand("AlterClass", this.DataBridge.SqlConnection);
        sqlCommand.CommandType = CommandType.StoredProcedure;
        sqlCommand.Parameters.Add("@classID", SqlDbType.Char, 0, ClassLimitations.ClassID);
        sqlCommand.Parameters["@classID"].Value = this.ClassID;
        sqlCommand.Parameters.Add("@className", SqlDbType.VarChar, 0, ClassLimitations.ClassName);
        sqlCommand.Parameters["@className"].Value = this.ClassName;
        sqlCommand.Parameters.Add("@denyOtherUsers", SqlDbType.Bit);
        sqlCommand.Parameters["@denyOtherUsers"].Value = this.DenyOtherUsers;
        this.DataBridge.SqlConnection.Open();
        sqlCommand.ExecuteNonQuery();
        this.DataBridge.SqlConnection.Close();
    }

    /// <summary>
/// Uses the current instance’s properties and deletes the stored data.
/// <summary>
public void Delete()
{
    SqlCommand sqlCommand = new SqlCommand("DeleteClass", this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.AddWithValue("@classID", ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = this.ClassID;
    this.DataBridge.SqlConnection.Open();
    sqlCommand.ExecuteNonQuery();
    this.DataBridge.SqlConnection.Close();
    Directory.Delete(this.DataBridge.GetClassDirectory(this.ClassID, true), true);
}
#endregion
#region Data Retrieving Methods
/// <summary>
/// Enrol a user to the class.
/// </summary>
/// <param name="userID">The associated user’s ID.</param>
/// <param name="roleID">The associated role ID.</param>
public void EnrolUser(int userID, RoleID roleID)
{
    Role role = new Role(roleID, this.DataBridge);
    role.AssignUserToClass(userID, this.ClassID);
}
#endregion
/// <summary>
/// DeEnrol a user to the class.
/// </summary>
/// <param name="userID">The associated user’s ID.</param>
/// <param name="roleID">The associated role ID.</param>
public void DeEnrolUser(int userID, RoleID roleID)
{
    Role role = new Role(roleID, this.DataBridge);
    role.UnassignUserToClass(userID, this.ClassID);
}
#endregion
/// <summary>
/// Find all roles a user is enrolled in.
/// </summary>
/// <param name="userID">The associated user’s ID.</param>
/// <returns>An array of associated roles. Null if none found.</returns>
public RoleID[] IsInClass(int userID)
{
    ArrayList roleIDs = new ArrayList();
    SqlCommand sqlCommand = new SqlCommand("IsInClass", this.DataBridge.SqlConnection);
    sqlCommand.Parameters.AddWithValue("@userID", userID);
    sqlCommand.Parameters["@userID"].Value = userID;
    sqlCommand.ExecuteReader();
    while (sqlCommand.Read())
    {
        RoleID roleID = (RoleID)sqlCommand["RoleID"];
sqlCommand.CommandType = CommandType.StoredProcedure;
sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
sqlCommand.Parameters["@userID"].Value = userID;
sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
    ClassLimitations.ClassID);
sqlCommand.Parameters["@classID"].Value = this.ClassID;
this.DataBridge.SqlConnection.Open();
SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

while (sqlDataReader.Read())
{
    switch (sqlDataReader["roleID"] as string)
    {
        case "Administrators":
            roleIDs.Add(RoleID.Administrators);
            break;
        case "Instructors":
            roleIDs.Add(RoleID.Instructors);
            break;
        case "Students":
            roleIDs.Add(RoleID.Students);
            break;
    }
}
this.DataBridge.SqlConnection.Close();

// Check what roles
if (roleIDs.Count == 0)
    return null;
else
    return (RoleID[])roleIDs.ToArray(typeof(RoleID));

#endregion

#region Static Data Retrieving Methods
/// <summary>
/// Find a class.
/// </summary>
/// <param name="classID">The class ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns the found class. Null if not found.</returns>
public static Class FindClass(string classID, DataBridge dataBridge)
{
    SqlCommand sqlCommand = new SqlCommand("FindClass",
        dataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char,
        ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = classID;
dataBridge.SqlConnection.Open();
    XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.
        ExecuteXmlReader();

    return xmlTextReader.Read();

#endregion
Class c = null;
XmlSerializer xmlSerializer = new XmlSerializer(typeof(Class));
  // Deserialise class
while (xmlTextReader.Read())
  if (xmlSerializer.CanDeserialize(xmlTextReader))
    c = xmlSerializer.Deserialize(xmlTextReader) as Class;

xmlTextReader.Close();
dataBridge.SqlConnection.Close();

if (c == null)
  return null;
else
  {
    c.DataBridge = dataBridge;
    return c;
  }

/// <summary>
/// Find the classes associated with a user.
/// </summary>
/// <param name="userID">The associated user ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns the found classes. Null if not found.</returns>
public static Class[] FindClasses(int userID, DataBridge dataBridge)
{
  SqlCommand sqlCommand = new SqlCommand("FindClasses",
      dataBridge.SqlConnection);
  sqlCommand.CommandType = CommandType.StoredProcedure;
  sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
  sqlCommand.Parameters["@userID"].Value = userID;
  dataBridge.SqlConnection.Open();
  XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.ExecuteXmlReader();
  ExecuteXmlReader();

  ArrayList classes = new ArrayList();
  XmlSerializer xmlSerializer = new XmlSerializer(typeof(Class));
  // Deserialise classes
  while (xmlTextReader.Read())
  {
    while (xmlSerializer.CanDeserialize(xmlTextReader))
    {
      Class tempClass = xmlSerializer.Deserialize(
          xmlTextReader) as Class;
      tempClass.DataBridge = dataBridge;
      classes.Add(tempClass);
Listing H.98: Configuration.cs

```csharp
using System;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A sealed class that contains methods and properties for configuration.
    /// </summary>
    [XmlRoot("Configuration")]
    public sealed class Configuration
    {
        #region Properties
        #region InstitutionConfiguration
        /// <summary>
        /// The institution name.
        /// </summary>
        private string _institutionName = "Online Learning Environment";
        [XmlElement("InstitutionName")]
        public string InstitutionName
        {
            get { return this._institutionName; }
            set { this._institutionName = value; }
        }
        #endregion
        #endregion
        #region InstitutionNameAbbreviation
        /// <summary>
        /// The institution name’s abbreviation.
        /// </summary>
        private string _institutionNameAbbreviation = "OLE";
        [XmlElement("InstitutionNameAbbreviation")]
        public string InstitutionNameAbbreviation
        {
            get { return this._institutionNameAbbreviation; }
            set { this._institutionNameAbbreviation = value; }
        }
        #endregion
    }
}
```
/// <summary>
/// The institution's logo image for the website header.
/// </summary>
private string _institutionLogoImage = "Images\header.jpg";
[XmlElement("InstitutionLogoImage")]
public string InstitutionLogoImage
{
    get { return _institutionLogoImage; }
    set { this._institutionLogoImage = value; }
}
#endregion

#region TerminologyConfiguration
/// <summary>
/// Institution terminology.
/// </summary>
private string _institution = "Institution";
[XmlElement("Institution")]
public string Institution
{
    get { return this._institution; }
    set { this._institution = value; }
}
/// <summary>
/// Class terminology.
/// </summary>
private string _class = "Class";
[XmlElement("Class")]
public string Class
{
    get { return this._class; }
    set { this._class = value; }
}
/// <summary>
/// Assignment terminology.
/// </summary>
private string _assignment = "Assignment";
[XmlElement("Assignment")]
public string Assignment
{
    get { return this._assignment; }
    set { this._assignment = value; }
}
/// <summary>
/// Exam terminology.
/// </summary>
private string _exam = "Exam";
[XmlElement("Exam")]
public string Exam
{
    get { return this._exam; }
    set { this._exam = value; }
}
private string _tutorial = "Tutorial";
[XmlElement("Tutorial")]
public string Tutorial
{
    get { return this._tutorial; }
    set { this._tutorial = value; }
}

private string _note = "Note";
[XmlElement("Note")]
public string Note
{
    get { return this._note; }
    set { this._note = value; }
}

public Configuration ( ) {}

public Configuration ( DataBridge dataBridge )
{
    this._dataBridge = dataBridge;
}

private DataBridge _dataBridge = null;
[XmlIgnore( )]
public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}

public Configuration ( ) {}

public Configuration ( DataBridge dataBridge )
{
    this._dataBridge = dataBridge;
}

private DataBridge _dataBridge = null;
[XmlIgnore( )]
public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}

public Configuration ( ) {}

public Configuration ( DataBridge dataBridge )
{
    this._dataBridge = dataBridge;
}
public void Create()
{
    string configurationFilePath = this.DataBridge.
        GetConfigurationFilePath(false);

    // Serialise the data
    XmlSerializer xmlSerializer = new XmlSerializer(this.
        GetType());
    XmlTextWriter xmlTextWriter;
    xmlSerializer.Serialize(xmlTextWriter = new
        XmlTextWriter(configurationFilePath, Encoding.UTF8),
        this);
    xmlTextWriter.Close();
}

/// <summary>
/// Uses current instance’s properties and re-stores the
/// data.
/// </summary>
public void Alter()
{
    string configurationFilePath = this.DataBridge.
        GetConfigurationFilePath(false);

    // Serialise the data
    XmlSerializer xmlSerializer = new XmlSerializer(this.
        GetType());
    XmlTextWriter xmlTextWriter;
    xmlSerializer.Serialize(xmlTextWriter = new
        XmlTextWriter(configurationFilePath, Encoding.UTF8),
        this);
    xmlTextWriter.Close();
}
#endregion

#region Static Data Retrieving Methods
/// <summary>
/// Find configuration
/// </summary>
/// <param name="dataBridge">The DataBridge that connects to
/// the data.</param>
/// <returns>Returns a data filled Configuration class. Null
/// if not found.</returns>
public static Configuration FindConfiguration(DataBridge
    dataBridge)
{
    string configurationFilePath;
    try
    {
        configurationFilePath = dataBridge.
            GetConfigurationFilePath(true);
    }
    catch
    {
        // Not found
    }
#endregion
Listing H.99: DataBridge.cs

```csharp
using System;
using System.Data.SqlClient;
using System.IO;

namespace OnlineLearningEnvironment
{
    /// The class that provides properties and methods to connect to XML files and an SQL Server database.
    public sealed class DataBridge
    {
        #region Properties
        private SqlConnection _sqlConnection = null;  
        /// The connection to an SQL Server database.
        public SqlConnection SqlConnection
        {
            get { return this._sqlConnection; } 
        }

        private string _classesDirectory = null;  
        /// The directory that holds all class critical XML files.
        public string ClassesDirectory
        {
            get { return this._classesDirectory; } 
        }
    }
}
```
private string _institutionDirectory = null;
/// <summary>
/// The directory that holds all institution critical Xml files.
/// </summary>
public string InstitutionDirectory
{
    get { return this._institutionDirectory; }
}
#endregion
#region Methods
/// <summary>
/// Used to pad a numeric number by a certain length (FileLimitations.FilenameLength).
/// It also pads the numeric number by a certain character (FileLimitations.PadCharacter).
/// </summary>
/// <param name="numericID">The numeric number to pad.</param>
/// <returns>Returns a padded number string.</returns>
private string PadNumericID(int numericID)
{
    return numericID.ToString().PadLeft(FileLimitations.FilenameLength, FileLimitations.PadCharacter);
}
#endregion
/// <summary>
/// Gets the root data file directory (the directory before the class and institution directory).
/// </summary>
/// <returns>Returns a string with the root directory.</returns>
private string GetRootDirectory()
{
    string institutionDirectory = this.InstitutionDirectory;
    return institutionDirectory.TrimEnd("Institution").ToCharArray();
}
#endregion
/// <summary>
/// Retrieves a file path with a certain name for a file.
/// </summary>
/// <param name="directory">The directory where it belongs</param>
/// <param name="nameID">The file’s name.</param>
/// <param name="fileExtension">The extension of the file.</param>
/// <param name="checkIfExists">If true, and file is not found, an exception is throw.</param>
/// <returns>Returns the file path of the file.</returns>
private string GetFilePath(string directory, string nameID, string fileExtension, bool checkIfExists)
{
    string filePath = directory + "\" + nameID + "." +
    fileExtension;
    if (!File.Exists(filePath) && checkIfExists)
        throw new Exception("File does not exist!");

    return filePath;
}
/// <summary>
/// Retrieves a file path with a certain number.
/// </summary>
/// <param name="directory">The directory where it belongs</param>
/// <param name="numericID">The numeric number of the file</param>
/// <param name="paddedID">The padded string number.</param>
/// <param name="checkIfExists">If true, and file is not found, an exception is throw.</param>
/// <returns>Returns the file path of the file.</returns>
private string GetFilePath(string directory, int numericID, out string paddedID, bool checkIfExists)
{
    paddedID = PadNumericID(numericID);
    string filePath = directory + "\" + paddedID + ".xml";
    if (!File.Exists(filePath) && checkIfExists)
        throw new Exception("File does not exist!");

    return filePath;
}
/// <summary>
/// Retrieves a 'new' file path with a numeric number above the last in the directory (+1).
/// </summary>
/// <param name="directory">The directory where it belongs</param>
/// <param name="numericID">The numeric number of the file</param>
/// <param name="paddedID">The padded string number.</param>
/// <returns>Returns the file path of the file.</returns>
private string GetNextFilePath(string directory, out int numericID, out string paddedID)
{
    string[] filenames = Directory.GetFiles(directory, "*.xml");
    if (filenames.Length > 0)
    {
        string[] tempStrings = filenames[filenames.Length - 1].Split("\".ToCharArray(), 100);
        numericID = Int32.Parse(tempStrings[tempStrings.Length - 1].TrimEnd(".xml".ToCharArray())) + 1;
    }
    else
    {
        numericID = 1;
    }
paddedID = numericID.ToString().PadLeft(FileLimitations.FilenameLength, FileLimitations.PadCharacter);

    return (directory + "\\" + paddedID + ".xml");
} #endregion

#region Constructors
/// <summary>
/// Disabled Constructor.
/// </summary>
private DataBridge() {}
/// <summary>
/// Main Constructor.
/// </summary>
/// <param name="sqlConnection">The shared connection to an SQL Server database.</param>
/// <param name="classesDirectory">The directory to the class Xml files.</param>
/// <param name="institutionDirectory">The directory to the institution Xml files.</param>
/// <param name="checkIfExists">If true, and directories are not found, an exception is thrown.</param>
public DataBridge(SqlConnection sqlConnection, string classesDirectory, string institutionDirectory, bool checkIfExists)
{
    this._sqlConnection = sqlConnection;

    if (!Directory.Exists(this._classesDirectory = classesDirectory) && classesDirectory != null && checkIfExists)
        throw new Exception("Classes\Directory\does\not\exist!");

    if (!Directory.Exists(this._institutionDirectory = institutionDirectory) && institutionDirectory != null && checkIfExists)
        throw new Exception("Institution\Directory\does\not\exist!");
} #endregion

#region Other Methods
/// <summary>
/// Get configuration file path.
/// </summary>
/// <param name="checkIfExists">If true, and file is not found, an exception is thrown.</param>
/// <returns>Returns a string of the configuration’s file path.</returns>
public string GetConfigurationFilePath(bool checkIfExists)
{
    string filePath = this.GetRootDirectory() + "Configuration.xml";

if (!File.Exists(filePath) & checkIfExists)
    throw new Exception("File does not exist!");
return filePath;
}
#endregion

#region Institution Enums
/// <summary>
/// Components within the Institution.
/// </summary>
private enum InstitutionComponent
{
    Announcements,
    ApplicationErrors,
    SecurityErrors
}
#endregion

#region Institution Methods
/// <summary>
/// Gets an institution component’s directory.
/// </summary>
/// <param name="institutionComponent">The institution component.</param>
/// <param name="checkIfExists">If true, and directory not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>
private string GetInstitutionComponentDirectory(InstitutionComponent institutionComponent, bool checkIfExists)
{
    string directory = this.InstitutionDirectory + "\\" + institutionComponent;
    if (!Directory.Exists(directory))
        throw new Exception("Directory does not exist!");
return directory;
}
/// <summary>
/// Gets the institution’s announcement directory.
/// </summary>
/// <param name="checkIfExists">If true, and file is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>
public string GetInstitutionAnnouncementsDirectory(bool checkIfExists)
{
    return this.GetInstitutionComponentDirectory(InstitutionComponent.Announcements, checkIfExists);
}
/// <summary>
/// Gets a file path within the institution announcement directory.
/// </summary>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <param name="checkIfExists">If true, and file is not found, an exception is throw.</param>
/// <returns>Returns a string with the file path.</returns>
public string GetInstitutionAnnouncementFilePath(int numericID, out string paddedID, bool checkIfExists)
{
    return this.GetFilePath(this.GetInstitutionComponentDirectory(InstitutionComponent.Announcements, true), numericID, out paddedID, checkIfExists);
}

/// <summary>
/// Get next file path to exist within the institution announcement directory.
/// </summary>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <returns>Returns a string of the 'new' file path.</returns>
public string getNextInstitutionAnnouncementFilePath(out int numericID, out string paddedID)
{
    return this.GetNextFilePath(this.GetInstitutionComponentDirectory(InstitutionComponent.Announcements, true), out numericID, out paddedID);
}

/// <summary>
/// Gets the institution's application errors directory.
/// </summary>
/// <param name="checkIfExists">If true, and file is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>
public string GetInstitutionApplicationErrorsDirectory(bool checkIfExists)
{
    return this.GetInstitutionComponentDirectory(InstitutionComponent.ApplicationErrors, checkIfExists);
}

/// <summary>
/// Gets a file path within the institution application errors directory.
/// </summary>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <param name="checkIfExists">If true, and file is not found, an exception is throw.</param>
/// <returns>Returns a string with the file path.</returns>
public string GetInstitutionApplicationErrorFilePath(int numericID, out string paddedID, bool checkIfExists)
{
**H.2 OLEBusinessLogic Project**

240  return this.GetFilePath(this.
241  GetInstitutionComponentDirectory(InstitutionComponent.
242  ApplicationErrors, true), numericID, out paddedID,
243  checkIfExists);
244
245  }  /// <summary>
246  /// Get next file path to exist within the application
247  /// errors directory.
248  /// </summary>
249  /// <param name="numericID">The numeric number ID.</param>
250  /// <param name="paddedID">The padded number ID.</param>
251  /// <returns>Returns a string of the 'new' file path.</returns>
252  public string GetNextInstitutionApplicationErrorFilePath(out int numericID, out string paddedID)
253  {
254    return this.GetNextFilePath(this.
255      GetInstitutionComponentDirectory(InstitutionComponent.
256        ApplicationErrors, true), out numericID, out paddedID);
257  }
258
259  }  /// <summary>
260  /// Gets the institution's security errors directory.
261  /// </summary>
262  /// <param name="checkIfExists">If true, and file is not
263    found, an exception is throw.</param>
264  /// <returns>Returns a string of the path.</returns>
265  public string GetInstitutionSecurityErrorsDirectory(bool checkIfExists)
266  {
267    return this.GetInstitutionComponentDirectory(
268      InstitutionComponent.SecurityErrors, checkIfExists);
269  }
270
271  }  /// <summary>
272  /// Gets a file path within the institution security errors
directory.
273  /// </summary>
274  /// <param name="numericID">The numeric number ID.</param>
275  /// <param name="paddedID">The padded number ID.</param>
276  /// <param name="checkIfExists">If true, and file is not
277    found, an exception is throw.</param>
278  /// <returns>Returns a string with the file path.</returns>
279  public string GetInstitutionSecurityErrorFilePath(int numericID, out string paddedID, bool checkIfExists)
280  {
281    return this.GetFilePath(this.
282      GetInstitutionComponentDirectory(InstitutionComponent.
283        SecurityErrors, true), numericID, out paddedID,
284        checkIfExists);
285  }
286
287  }  /// <summary>
288  /// Get next file path to exist within the institution
289  /// security errors directory.
290  /// </summary>
/// <param name="numericID">The numeric number ID.</param>  
/// <param name="paddedID">The padded number ID.</param>  
/// <returns>Returns a string of the 'new' file path.</returns>

public string GetNextInstitutionSecurityErrorFilePath(out int numericID, out string paddedID)  
{
    return this.GetNextFilePath(this.GetInstitutionComponentDirectory(InstitutionComponent.SecurityErrors, true), out numericID, out paddedID);
}

#region Class Enums
/// <summary>  
/// Components within Classes.  
/// </summary>
private enum ClassComponent
{
    Announcements,  
    Assignments,  
    Exams,  
    Tutorials,  
    Notes,  
    Forum
}
#endregion

#region Class Methods
/// <summary>  
/// Get a class' files directory.  
/// </summary>
/// <param name="classID">The associated class ID.</param>  
/// <param name="checkIfExists">If true, and directory is not found, an expection is throw.</param>  
/// <returns>Returns a string of the path.</returns>
public string GetClassDirectory(string classID, bool checkIfExists)
{
    string directory = this.ClassesDirectory + "\\" + classID;
    if (!Directory.Exists(directory) && checkIfExists)
        throw new Exception("Directory does not exist!");

    return directory;
}
#endregion
private string GetClassComponentDirectory(string classID, ClassComponent classComponent, bool checkIfExists)
{
    return (this.GetClassDirectory(classID, checkIfExists) + "\" + classComponent);
}

/// <summary>
/// Gets a class component’s sub-directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="paddedID">The padded number ID to be the sub-directory name.</param>
/// <param name="classComponent">The class component.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>

private string GetClassSubComponentDirectory(string classID, string paddedID, ClassComponent classComponent, bool checkIfExists)
{
    return (this.GetClassDirectory(classID, checkIfExists) + "\" + classComponent + "\" + paddedID);
}

/// <summary>
/// Get the class’ announcements directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>

public string GetClassAnnouncementsDirectory(string classID, bool checkIfExists)
{
    return this.GetClassComponentDirectory(classID, ClassComponent.Announcements, checkIfExists);
}

/// <summary>
/// Get class announcement’s file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the file path.</returns>

public string GetClassAnnouncementFilePath(string classID, int numericID, out string paddedID, bool checkIfExists)
{
    return this.GetFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Announcements, true), numericID, out paddedID, checkIfExists);
}

/// <summary>
/// Get a 'new' class announcement's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <returns>Returns a string of the 'new' file path.</returns>

public string GetNextClassAnnouncementFilePath(string classID, out int numericID, out string paddedID)
{
  return this.GetNextFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Announcements, true), out numericID, out paddedID);
}

/// Get the class' assignments directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>

public string GetClassAssignmentsDirectory(string classID, bool checkIfExists)
{
  return this.GetClassComponentDirectory(classID, ClassComponent.Assignments, checkIfExists);
}

/// Get class assignment's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the file path.</returns>

public string GetClassAssignmentFilePath(string classID, int numericID, out string paddedID, bool checkIfExists)
{
  return this.GetFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Assignments, true), numericID, out paddedID, checkIfExists);
}

/// Get a 'new' class assignment's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <returns>Returns a string of the 'new' file path.</returns>

public string GetNextClassAssignmentFilePath(string classID, out int numericID, out string paddedID)
{
return this.GetNextFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Assignments, true), out numericID, out paddedID);
}

/// <summary>
/// Get the class’ assignment answer directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="aetQID">The associated AETQuestion ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>
public string GetClassAssignmentAnswersDirectory(string classID, int aetQID, bool checkIfExists)
{
    return this.GetClassSubComponentDirectory(classID, this.PadNumericID(aetQID), ClassComponent.Assignments, checkIfExists);
}

/// <summary>
/// Get a 'new' class assignment answer's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericUserID">The associated numeric user ID.</param>
/// <param name="paddedUserID">The associated padded user ID</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the 'new' file path.</returns>
public string GetClassAssignmentAnswerFilePath(string classID, int aetQID, int numericUserID, out string paddedUserID, bool checkIfExists)
{
    return this.GetFilePath(this.GetClassSubComponentDirectory(classID, this.PadNumericID(aetQID), ClassComponent.Assignments, true), numericUserID, out paddedUserID, checkIfExists);
}

/// <summary>
/// Get the class' exams directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>
public string GetClassExamsDirectory(string classID, bool checkIfExists)
{
return this.GetClassComponentDirectory(classID, ClassComponent.Exams, checkIfExists);
}
/// <summary>
/// Get class exam's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the file path.</returns>
public string GetClassExamFilePath(string classID, int numericID, out string paddedID, bool checkIfExists)
{
    return this.GetFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Exams, true), numericID, out paddedID, checkIfExists);
}
/// <summary>
/// Get a 'new' class exam's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <returns>Returns a string of the 'new' file path.</returns>
public string GetNextClassExamFilePath(string classID, out int numericID, out string paddedID)
{
    return this.GetNextFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Exams, true), out numericID, out paddedID);
}
/// <summary>
/// Get the class' exam answer directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="aetQID">The associated AETQuestion ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>
public string GetClassExamAnswersDirectory(string classID, int aetQID, bool checkIfExists)
{
    return this.GetClassSubComponentDirectory(classID, this.PadNumericID(aetQID), ClassComponent.Exams, checkIfExists);
}
/// <summary>
/// Get a 'new' class exam answer's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="aetQID">The associated AETQuestion ID.</param Producto de la traducción natural.
/// <param name="numericUserID">The associated numeric user ID.</param>
/// <param name="paddedUserID">The associated padded user ID</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the 'new' file path.</returns>

public string GetClassExamAnswerFilePath(string classID, int actQID, int numericUserID, out string paddedUserID, bool checkIfExists)
{
    return this.GetFilePath(this.GetClassSubComponentDirectory(classID, this.PadNumericID(actQID), ClassComponent.Exams, true), numericUserID, out paddedUserID, checkIfExists);
}

/// <summary>
/// Get the class' tutorials directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>

public string GetClassTutorialsDirectory(string classID, bool checkIfExists)
{
    return this.GetClassComponentDirectory(classID, ClassComponent.Tutorials, checkIfExists);
}

/// <summary>
/// Get class tutorial's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the file path.</returns>

public string GetClassTutorialFilePath(string classID, int numericID, out string paddedID, bool checkIfExists)
{
    return this.GetFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Tutorials, true), numericID, out paddedID, checkIfExists);
}

/// <summary>
/// Get a 'new' class tutorial's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <returns>Returns a string of the 'new' path.</returns>
public string GetNextClassTutorialFilePath(string classID, out int numericID, out string paddedID)
{
    return this.GetNextFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Tutorials, true), numericID, paddedID);
}

/// <summary>
/// Get the class' tutorial answer directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="aetQID">The associated AETQuestion ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>
public string GetClassTutorialAnswersDirectory(string classID, int aetQID, bool checkIfExists)
{
    return this.GetClassSubComponentDirectory(classID, this.PadNumericID(aetQID), ClassComponent.Tutorials, checkIfExists);
}

/// <summary>
/// Get a 'new' class tutorial answer's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="aetQID">The associated AETQuestion ID.</param>
/// <param name="numericUserID">The associated numeric user ID.</param>
/// <param name="paddedUserID">The associated padded user ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the 'new' file path.</returns>
public string GetClassTutorialAnswerFilePath(string classID, int aetQID, int numericUserID, out string paddedUserID, bool checkIfExists)
{
    return this.GetFilePath(this.GetClassSubComponentDirectory(classID, this.PadNumericID(aetQID), ClassComponent.Tutorials, true), numericUserID, out paddedUserID, checkIfExists);
}

/// <summary>
/// Get the class' notes directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>
public string GetClassNotesDirectory(string classID, bool checkIfExists)
checkIfExists)
{
    return this.GetClassComponentDirectory(classID, ClassComponent.Notes, checkIfExists);
}

/// <summary>
/// Get class note's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the file path.</returns>
public string GetClassNoteFilePath(string classID, int numericID, out string paddedID, bool checkIfExists)
{
    return this.GetFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Notes, true), numericID, out paddedID, checkIfExists);
}

/// <summary>
/// Get a 'new' class note's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <returns>Returns a string of the 'new' file path.</returns>
public string GetNextClassNoteFilePath(string classID, out int numericID, out string paddedID)
{
    return this.GetNextFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Notes, true), out numericID, out paddedID);
}

/// <summary>
/// Get the class' threads directory.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="checkIfExists">If true, and directory is not found, an exception is throw.</param>
/// <returns>Returns a string of the path.</returns>
public string GetClassThreadsDirectory(string classID, bool checkIfExists)
{
    return this.GetClassComponentDirectory(classID, ClassComponent.Forum, checkIfExists);
}

/// <summary>
/// Get class thread's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="numericID">The numeric number ID.</param>
public string GetClassThreadFilePath (string classID, int numericID, out string paddedID, bool checkIfExists)
{
    return this.GetFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Forum, true), numericID, out paddedID, checkIfExists);
}

public string GetNextClassThreadFilePath(string classID, out int numericID, out string paddedID)
{
    return this.GetNextFilePath(this.GetClassComponentDirectory(classID, ClassComponent.Forum, true), out numericID, out paddedID);
}

public string GetClassThreadPostsDirectory(string classID, int threadID, bool checkIfExists)
{
    return this.GetClassSubComponentDirectory(classID, this.PadNumericID(threadID), ClassComponent.Forum, checkIfExists);
}

public string GetClassThreadPostFilePath(string classID, int threadID, int numericID, out string paddedID, bool checkIfExists)
{
return this.GetFilePath(this).
    GetClassSubComponentDirectory(classID, this.
    PadNumericID(threadID), ClassComponent.Forum, true),
    numericID, out paddedID, checkIfExists);

/// <summary>
/// Get a 'new' class thread post's file path.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="threadID">The associated thread ID.</param>
/// <param name="numericID">The numeric number ID.</param>
/// <param name="paddedID">The padded number ID.</param>
/// <returns>Returns a string of the 'new' file path.</returns>

public string GetNextClassThreadPostFilePath(string classID, int threadID, out int numericID, out string paddedID)
{
    return this.GetNextFilePath(this.
        GetClassSubComponentDirectory(classID, this.
        PadNumericID(threadID), ClassComponent.Forum, true),
        out numericID, out paddedID);
}

#endregion
#endif

Listing H.100: ErrorAndSecurity.cs
/// The ID of the associated user.
/// <summary>
private int _userID;
[XmlElement("UserID")]
public int UserID
{
    get { return this._userID; }
    set { this._userID = value; }
}

/// The value of the associated IP address.
/// <summary>
private string _ipAddress;
[XmlElement("IPAddress")]
public string IPAddress
{
    get { return this._ipAddress; }
    set { this._ipAddress = value; }
}

/// The date and time the error was created.
/// <summary>
private DateTime _createdDateTime;
[XmlAttribute("createdDateTime")]
public DateTime CreatedDateTime
{
    get { return this._createdDateTime; }
    set { this._createdDateTime = value; }
}

#region Constructors
/// Used for XmlSerializer purposes.
/// <summary>
public SecurityError() {}

/// Main constructor.
/// <summary>
/// <param name="dataBridge">The data bridge to the data.</param>
public SecurityError(DataBridge dataBridge)
{
    this._dataBridge = dataBridge;
}
#endregion

#region Data Management Properties
/// The DataBridge that connects to the data.
/// <summary>
private DataBridge _dataBridge = null;

[XmlIgnore()]

public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}
#endregion

#region Data Management Methods
/// <summary>
/// Creates an instance of the error as an xml file.
/// </summary>
/// <returns>Returns the ID of the newly created error.</returns>
public int Create()
{
    int numericID;
    string paddedID;

    // Get next file path
    string securityErrorFilePath = this.DataBridge.GetNextInstitutionSecurityErrorFilePath(out numericID, out paddedID);

    // Assign ID
    this.SecurityID = numericID;

    // Create the file by serializing it to the xml file
    XmlSerializer xmlSerializer = new XmlSerializer(this.GetType());
    XmlTextWriter xmlTextWriter;
    xmlSerializer.Serialize(xmlTextWriter = new XmlTextWriter(securityErrorFilePath, Encoding.UTF8), this);
    xmlTextWriter.Close();

    return numericID;
}
#endregion

/// <summary>
/// Delete the specified error ID.
/// </summary>
public void Delete()
{
    int numericID = this.SecurityID;
    string paddedID;

    // Get file path of the ID
    string securityErrorFilePath = this.DataBridge.GetInstitutionSecurityErrorFilePath(numericID, out paddedID, true);

    // Delete the file
    File.Delete(securityErrorFilePath);
}
#endregion

#region Static Data Retrieving Methods

/// <summary>
/// Find a specific security error.
/// </summary>
/// <returns>
/// Returns a SecurityError object or null if not found.
/// </returns>
public static SecurityError FindSecurityError(int securityID, DataBridge dataBridge)
{
    string paddedID;

    string securityFilePath;
    try
    {
        // Get file path
        securityFilePath = dataBridge.
            GetInstitutionSecurityErrorFilePath(securityID, out paddedID, true);
    }
    catch
    {
        // File not found
        return null;
    }

    // Deserialize the object from the xml file
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(SecurityError));
    XmlTextReader xmlTextReader;
    SecurityError newSE = (SecurityError)xmlSerializer.
        Deserialize(xmlTextReader = new XmlTextReader(securityFilePath));
    xmlTextReader.Close();

    return newSE;
}

/// <summary>
/// Find all security errors.
/// </summary>
/// <returns>
/// Returns an array of SecurityError objects or null if not found.
/// </returns>
public static SecurityError[] FindSecurityErrors(DataBridge dataBridge)
{
    string xmlDirectory = dataBridge.
        GetInstitutionSecurityErrorsDirectory(true);

    return null;
}
// Get files to be deserialized
string[] securityFilePaths = Directory.GetFiles(xmlDirectory, "*.xml");

ArrayList securityErrors = new ArrayList(securityFilePaths.Length);
XmlSerializer xmlSerializer = new XmlSerializer(typeof(SecurityError));
XmlTextReader xmlTextReader;
// Deserialize files
foreach (string securityFilePath in securityFilePaths)
{
    securityErrors.Add(xmlSerializer.Deserialize(
        xmlTextReader = new XmlTextReader(securityFilePath)));
    xmlTextReader.Close();
}

if (securityErrors.Count == 0)
    return null;
else
    return (SecurityError[])securityErrors.ToArray(typeof(SecurityError));

#endregion

/// <summary>
/// A class that holds information about application errors.
/// </summary>
[XmlRoot("Error")]
public sealed class ApplicationError
{
    #region Properties
    /// <summary>
    /// The ID of the error.
    /// </summary>
    [XmlAttribute("errorID")]
    public int ErrorID
    {
        get { return this._errorID; }
        set { this._errorID = value; }
    }
    
    /// <summary>
    /// The date and time the error was created
    /// </summary>
    [XmlAttribute("createdDateTime")]
    public DateTime CreatedDateTime
    {
        get { return this._createdDateTime; }
        set { this._createdDateTime = value; }
    }
    
    #endregion
}
/// <summary>
/// Information about the error.
/// </summary>
private string _information;
[XmlElement("Information")]
public string Information
{
    get { return this._information; }
    set { this._information = value; }
}

#region Constructors
/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>
public ApplicationError() {}

/// <summary>
/// Main constructor.
/// </summary>
/// <param name="dataBridge">The data bridge to the data.</param>
public ApplicationError(DataBridge dataBridge)
{
    this._dataBridge = dataBridge;
}

#endregion

#region Data Management Properties
/// <summary>
/// The DataBridge that connects to the data.
/// </summary>
private DataBridge _dataBridge = null;
[XmlIgnore()]
public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}

#endregion

#region Data Management Methods
/// <summary>
/// Creates an instance of the error as an xml file.
/// </summary>
/// <returns>Returns the ID of the newly created error.</returns>
public int Create()
{
    int numericID;
    string paddedID;
// Get next file path
string applicationErrorFilePath = this.DataBridge.
GetNextInstitutionApplicationErrorFilePath(out numericID, out paddedID);

// Assign ID
this.ErrorID = numericID;

// Create the file by serializing it to the xml file
XmlSerializer xmlSerializer = new XmlSerializer(this.GetType());
XmlTextWriter xmlTextWriter;
xmlSerializer.Serialize(xmlTextWriter = new XmlTextWriter(applicationErrorFilePath, Encoding.UTF8), this);
xmTextWriter.Close();

return numericID;
}

/// <summary>
/// Delete the specified error ID.
/// </summary>
public void Delete()
{
    int numericID = this.ErrorID;
    string paddedID;

    // Get file path of the ID
    string applicationErrorFilePath = this.DataBridge.
        GetInstitutionApplicationErrorFilePath(numericID, out paddedID, true);

    // Delete the file
    File.Delete(applicationErrorFilePath);
}

#region Static Data Retrieving Methods
/// <summary>
/// Find a specific application error.
/// </summary>
/// <param name="applicationErrorID">The ID to find.</param>
/// <param name="dataBridge">The bridge to the data.</param>
/// <returns>
/// Returns a ApplicationError object or null if not found.
/// </returns>
public static ApplicationError FindApplicationError(int applicationErrorID, DataBridge dataBridge)
{
    string paddedID;

    string securityFilePath;
    try
    {

public static ApplicationError[] FindApplicationErrors(DataBridge dataBridge) {
    string xmlDirectory = dataBridge.
        GetInstitutionApplicationErrorsDirectory(true);

    // Get files to be deserialized
    string[] applicationErrorFilePaths = Directory.GetFiles(
        xmlDirectory, "*.xml");

    ArrayList applicationErrors = new ArrayList(
        applicationErrorFilePaths.Length);
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(
        ApplicationError));
    XmlTextReader xmlTextReader;

    // Deserialize files
    foreach (string applicationErrorFilePath in applicationErrorFilePaths) {
        applicationErrors.Add(xmlSerializer.Deserialize(
            xmlTextReader = new XmlTextReader(
                applicationErrorFilePath)));
        xmlTextReader.Close();
    }
}

/// Get file path
securityFilePath = dataBridge.
    GetInstitutionApplicationErrorFilePath(
        applicationErrorID, out paddedID, true);

} catch

    // File not found
    return null;

} // Deserialize the object from the xml file
XmlSerializer xmlSerializer = new XmlSerializer(typeof(
    ApplicationError));
XmlTextReader xmlTextReader;
ApplicationError newAE = (ApplicationError)xmlSerializer
    .Deserialize(xmlTextReader = new XmlTextReader(
        securityFilePath));
xmlTextReader.Close();

    return newAE;

/// <summary>
/// Find all application errors.
/// </summary>
/// <param name="dataBridge">The bridge to the data.</param>
/// <returns>
/// Returns an array of ApplicationError objects or null if not
/// found.
/// </returns>
public static ApplicationError[] FindApplicationErrors(DataBridge dataBridge) {
    string xmlDirectory = dataBridge.
        GetInstitutionApplicationErrorsDirectory(true);

    // Get files to be deserialized
    string[] applicationErrorFilePaths = Directory.GetFiles(
        xmlDirectory, "*.xml");

    ArrayList applicationErrors = new ArrayList(
        applicationErrorFilePaths.Length);
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(
        ApplicationError));
    XmlTextReader xmlTextReader;

    // Deserialize files
    foreach (string applicationErrorFilePath in applicationErrorFilePaths) {
        applicationErrors.Add(xmlSerializer.Deserialize(
            xmlTextReader = new XmlTextReader(
                applicationErrorFilePath)));
        xmlTextReader.Close();
if (applicationErrors.Count == 0)
    return null;
else
    return (ApplicationError[]) applicationErrors.ToArray
        (typeof(ApplicationError));
#endregion
}
}

Listing H.101: Limitations.cs

using System;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// Property limitations for classes, AETQuestion and AETAnswer.
    /// </summary>
    public struct AETLimitations
    {
        public const int Title = 100;
        public const int Description = 30;
        public const int AETType = 20;
        public const int AETQuestionType = 20;
    }

    /// <summary>
    /// Property limitations for class, Role.
    /// </summary>
    public struct RoleLimitations
    {
        public const int RoleID = 50;
    }

    /// <summary>
    /// Property limitations for class, User.
    /// </summary>
    public struct UserLimitations
    {
        public const int Password = 47;
        public const int FirstName = 50;
        public const int MiddleName = 50;
        public const int LastName = 50;
        public const int PhoneNumber = 15;
        public const int EmailAddress = 50;
    }

    /// <summary>
    /// Property limitations for structure, AddressStructure.
    /// </summary>
    public struct AddressLimitations
    {
        public const int Street = 50;
    }
public const int Suburb = 50;
public const int State = 50;
public const int Postcode = 10;
public const int Country = 50;
}

/// <summary>
/// Property limitations for class, Class.
/// </summary>
public struct ClassLimitations
{
    public const int ClassID = 7;
    public const int ClassName = 50;
}

/// <summary>
/// Limitations for filenames used within the OLE.
/// </summary>
public struct FileLimitations
{
    public const int FilenameLength = 8;
    public const char PadCharacter = '0';
}

Listing H.102: MD5Encryption.cs

using System;
using System.Text;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// This class is used to encrypt strings of data to an MD5 hash string.
    /// </summary>
    public sealed class MD5Encryption
    {
        /// <summary>
        /// Encrypts a string to a MD5 hash in the form of a byte array.
        /// </summary>
        /// <param name="s">The string to encrypt.</param>
        /// <returns>Returns an encrypted MD5 hash (byte array).</returns>
        private static byte[] EncryptString(string s)
        {
            return new MD5CryptoServiceProvider().ComputeHash(new ASCIIEncoding().GetBytes(s));
        }
    }
}
/// <returns>Returns a readable MD5 hash string.</returns>

public static string EncryptStringToByteString(string s)
{
    return BitConverter.ToString(MD5Encryption.EncryptString(s));
}

Listing H.103: Post.cs

using System;
using System.Collections;
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// An abstract class which provides shared properties and methods for posts.
    ///
    /// The unique ID.
    ///
    private int _postID;
    [XmlAttribute("postID")]
    public int PostID
    {
        get { return this._postID; }
        set { this._postID = value; }
    }

    protected string _classID;
    protected int _threadID;
    protected string _title;
    protected bool _withhold = false;

    /// The name of the creator.
    ///
    protected string _creatorName;
    [XmlAttribute("creatorName")]
    public string CreatorName
    {
        get { return this._creatorName; }
        set { this._creatorName = value; }
    }

    /// The date & time created.
/// <summary>
private DateTime _createdDateTime;
[XmlAttribute("createdDateTime")]
public DateTime CreatedDateTime
{
    get { return this._createdDateTime; }
    set { this._createdDateTime = value; }
}
/// <summary>
/// The date & time updated.
/// </summary>
private DateTime _updatedDateTime;
[XmlAttribute("updatedDateTime")]
public DateTime UpdatedDateTime
{
    get { return this._updatedDateTime; }
    set { this._updatedDateTime = value; }
}
/// <summary>
/// The associated information.
/// </summary>
private string _information;
[XmlElement("Information")]
public string Information
{
    get { return this._information; }
    set { this._information = value; }
}
/// <summary>
/// An array of attachment filenames.
/// </summary>
private string[] _attachmentFilenames = null;
[XmlArray("Attachments")]
[XmlArrayItem("Filename")]
public string[] AttachmentFilenames
{
    get { return this._attachmentFilenames; }
    set { this._attachmentFilenames = value; }
}
#endregion
#region Constructors
/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>
public Post() {}
/// <summary>
/// Abstract constructor.
/// </summary>
public Post(DataBridge dataBridge)
{ 
    this._createdDateTime = DateTime.Now;
    this._dataBridge = dataBridge;
} #endregion

#region Data Management Properties
/// <summary>
/// The DataBridge that connects to the data.
/// </summary>
private DataBridge _dataBridge = null;
[XmlIgnore( )]
public DataBridge DataBridge
{
    get { return this._dataBridge; } 
    set { this._dataBridge = value; }
}
#endregion

#region Data Management Methods
/// <summary>
/// Uses current instance’s properties and stores the data.
/// </summary>
public int Create()
{
    string postFilePath = null;
    int numericID = 0;
    string paddedID = null;

    // Get filepath
    if (this is InstitutionAnnouncement)
    {
        // Is an InstitutionAnnouncement
        DataBridge newDataBridge = this.DataBridge as DataBridge;
        postFilePath = newDataBridge.
            GetNextInstitutionAnnouncementFilePath(out numericID, out paddedID);
    }
    else
    {
        DataBridge newDataBridge = this.DataBridge as DataBridge;

        if (this is ClassAnnouncement)
            postFilePath = newDataBridge.
                GetNextClassAnnouncementFilePath(this._classID, out numericID, out paddedID);
        else if (this is ClassNote)
            postFilePath = newDataBridge.
                GetNextClassNoteFilePath(this._classID, out numericID, out paddedID);
        else if (this is ClassThreadPost)
            postFilePath = newDataBridge.
                GetNextClassThreadPostFilePath(this._classID, this._threadID, out numericID, out paddedID);

        }
    }
this.PostID = numericID;

XmlSerializer xmlSerializer = new XmlSerializer(this.GetType());
XmlTextWriter xmlTextWriter;
    // Serialize
xmlSerializer.Serialize(xmlTextWriter = new XmlTextWriter(postFilePath, Encoding.UTF8), this);
xmlTextWriter.Close();

return this.PostID;

/// <summary>
/// Uses current instance's properties and re-stores the data.
/// </summary>
public void Alter()
{
    string postFilePath = null;
    string paddedID;

    // Get filepath
    if (this is InstitutionAnnouncement)
    {
        // Is InstitutionAnnouncement
        DataBridge newDataBridge = this.DataBridge as DataBridge;
        postFilePath = newDataBridge.GetInstitutionAnnouncementFilePath(this.PostID, out paddedID, true);
    }
    else
    {
        DataBridge newDataBridge = this.DataBridge as DataBridge;
        if (this is ClassAnnouncement)
            postFilePath = newDataBridge.GetClassAnnouncementFilePath(this._classID, this.PostID, out paddedID, true);
        else if (this is ClassNote)
            postFilePath = newDataBridge.GetClassNoteFilePath(this._classID, this.PostID, out paddedID, true);
        else if (this is ClassThreadPost)
            postFilePath = newDataBridge.GetClassThreadPostFilePath(this._classID, this._threadID, this.PostID, out paddedID, true);

    }

XmlSerializer xmlSerializer = new XmlSerializer(this.
GetType();
XmlTextWriter xmlTextWriter;

// Serialise
xmlSerializer.Serialize(xmlTextWriter = new
XmlTextWriter(postFilePath, Encoding.UTF8), this);
xmlTextWriter.Close();

/// <summary>
/// Uses the current instance's properties and deletes the
/// stored data.
/// </summary>
public void Delete()
{
    string postFilePath = null;
    string paddedID;

    // Get filepath
    if (this is InstitutionAnnouncement)
    {
        DataBridge newDataBridge = this.DataBridge as
            DataBridge;
        postFilePath = newDataBridge.
            GetInstitutionAnnouncementFilePath(this.PostID,
            out paddedID, true);
    }
    else
    {
        DataBridge newDataBridge = this.DataBridge as
            DataBridge;
        if (this is ClassAnnouncement)
        {
            postFilePath = newDataBridge.
                GetClassAnnouncementFilePath(this._classID,
                    this.PostID, out paddedID, true);
        }
        else if (this is ClassNote)
        {
            postFilePath = newDataBridge.
                GetClassNoteFilePath(this._classID, this.
                    PostID, out paddedID, true);
        }
        else if (this is ClassThreadPost)
        {
            postFilePath = newDataBridge.
                GetClassThreadPostFilePath(this._classID,
                    this._threadID, this.PostID, out paddedID,
                    true);
        }

        // Delete the file
        File.Delete(postFilePath);
    }
    #endregion
    
    /// <summary>
    /// A sealed class that contains properties and methods of a
    /// institution announcement.
    /// </summary>
public sealed class InstitutionAnnouncement : Post
{
    #region Properties
    /// <summary>
    /// The associated title.
    /// </summary>
    [XmlElement("Title")]
    public string Title
    {
        get { return base._title; }
        set { base._title = value; }
    }
    #endregion

    #region Constructors
    /// <summary>
    /// Used for XmlSerializer purposes.
    /// </summary>
    public InstitutionAnnouncement() {}
    /// <summary>
    /// Main Constructor.
    /// </summary>
    /// <param name="dataBridge">The DataBridge that connects to the data.</param>
    public InstitutionAnnouncement(DataBridge dataBridge) : base(dataBridge) {}
    #endregion

    #region Static Data Retrieving Methods
    /// <summary>
    /// Find a specific institution announcement.
    /// </summary>
    /// <param name="postID">The associated post ID.</param>
    /// <param name="dataBridge">The DataBridge that connects to the data.</param>
    /// <returns>Returns a InstitutionAnnouncement class. Null if not found.</returns>
    public static InstitutionAnnouncement FindInstitutionAnnouncement(int postID, DataBridge dataBridge)
    {
        string paddedPostID;
        string postFilePath;
        try
        {
            postFilePath = dataBridge.GetInstitutionAnnouncementFilePath(postID, out paddedPostID, true);
        }
        catch
        {
            // Not found
            return null;
        }
    }
}
XML Serializer

```csharp
new XmlSerializer(typeof(InstitutionAnnouncement));
```

```csharp
XmlTextReader xmlTextReader;
// Deserialise
InstitutionAnnouncement newIA = (InstitutionAnnouncement)xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(postFilePath));
xmTextReader.Close();
newIA.DataBridge = dataBridge;

return newIA;
```

```csharp
/// <summary>
/// Finds all institution announcements.
/// </summary>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns an array of InstitutionAnnouncement classes. Null if none found.</returns>
public static InstitutionAnnouncement[] FindInstitutionAnnouncements(DataBridge dataBridge)
{
  string xmlDirectory = dataBridge.GetInstitutionAnnouncementsDirectory(true);
  // Get files
  string[] postFilePaths = Directory.GetFiles(xmlDirectory, "*.xml");
  ArrayList posts = new ArrayList(postFilePaths.Length);
  XmlSerializer xmlSerializer = new XmlSerializer(typeof(InstitutionAnnouncement));
  XmlTextReader xmlTextReader;
  // Deserialise
  foreach (string postFilePath in postFilePaths)
  {
    InstitutionAnnouncement tempPost = xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(postFilePath)) as InstitutionAnnouncement;
    tempPost.DataBridge = dataBridge;
    posts.Add(tempPost);
    xmlTextReader.Close();
  }

  // Any found?
  if (posts.Count == 0)
    return null;
  else
    return (InstitutionAnnouncement[])posts.ToArray(typeof(InstitutionAnnouncement));
}
/// <summary>
/// A sealed class that contains properties and methods of a class announcement.
/// </summary>
[XmlRoot("Announcement")]
public sealed class ClassAnnouncement : Post
{
    #region Properties
    /// <summary>
    /// The associated class ID.
    /// </summary>
    [XmlAttribute("classID")]
    public string ClassID
    {
        get { return base._classID; }
        set { base._classID = value; }
    }
    /// <summary>
    /// The associated title.
    /// </summary>
    [XmlElement("Title")]
    public string Title
    {
        get { return base._title; }
        set { base._title = value; }
    }
    #endregion

    #region Constructors
    /// <summary>
    /// Used for XmlSerializer purposes.
    /// </summary>
    public ClassAnnouncement() {}
    /// <summary>
    /// Main Constructor.
    /// </summary>
    /// <param name="classID">The associated class ID.</param>
    /// <param name="dataBridge">The DataBridge that connects to the data.</param>
    public ClassAnnouncement(string classID, DataBridge dataBridge) : base(dataBridge)
    {
        this.ClassID = classID;
    }
    #endregion

    #region Static Data Retrieving Methods
    /// <summary>
    /// Find a specific class announcement.
    /// </summary>
    /// <param name="classID">The associated class ID.</param>
    }
/// <param name="postID">The associated post ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns a ClassAnnouncement class. Null if not found.</returns>
public static ClassAnnouncement FindClassAnnouncement(string classID, int postID, DataBridge dataBridge)
{
    string paddedPostID;
    string postFilePath;
    try
    {
        postFilePath = dataBridge.GetClassAnnouncementFilePath(classID, postID, out paddedPostID, true);
    }
    catch
    {
        // Not found
        return null;
    }

    XmlSerializer xmlSerializer = new XmlSerializer(typeof(ClassAnnouncement));
    XmlTextReader xmlTextReader;
    // Deserialise
    ClassAnnouncement newCA = (ClassAnnouncement)xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(postFilePath));
    xmlTextReader.Close();

    newCA.DataBridge = dataBridge;

    return newCA;
}

/// <summary>
/// Finds all class announcements for a specific class.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns an array of ClassAnnouncement classes. Null if none found.</returns>
public static ClassAnnouncement[] FindClassAnnouncements(string classID, DataBridge dataBridge)
{
    string xmlDirectory = dataBridge.GetClassAnnouncementsDirectory(classID, true);

    // Get files
    string[] postFilePaths = Directory.GetFiles(xmlDirectory, "*.xml");
ArrayList posts = new ArrayList(postFilePaths.Length);
XmlSerializer xmlSerializer = new XmlSerializer(typeof(ClassAnnouncement));
XmlTextReader xmlTextReader;
    // Deserialise
    foreach (string postFilePath in postFilePaths)
    {
        ClassAnnouncement tempPost = xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(postFilePath)) as ClassAnnouncement;
        tempPost.DataBridge = dataBridge;
        posts.Add(tempPost);
        xmlTextReader.Close();
    }

    // Any found?
    if (posts.Count == 0)
        return null;
    else
        return (ClassAnnouncement[])posts.ToArray(typeof(ClassAnnouncement));
    #endregion

    /// <summary>
    /// A sealed class that contains properties and methods of a class note.
    /// </summary>
    [XmlRoot("Note")]
    public sealed class ClassNote : Post
    {
        #region Properties
        /// <summary>
        /// The associated title.
        /// </summary>
        [XmlElement("Title")]
        public string Title
        {
            get { return base._title; }
            set { base._title = value; }
        }

        /// <summary>
        /// The associated class ID.
        /// </summary>
        [XmlAttribute("classID")]
        public string ClassID
        {
            get { return base._classID; }
            set { base._classID = value; }
        }

        /// <summary>
        /// The associated description.
        /// </summary>
    }
/// <summary>
private string _description;
[XmlElement("Description")]
public string Description
{
    get { return this._description; }
    set { this._description = value; }
}
#endregion

#region Constructors
/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>
public ClassNote() {}
/// <summary>
/// Main Constructor.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
public ClassNote(string classID, DataBridge dataBridge) : base(dataBridge)
{
    this.ClassID = classID;
}
#endregion

#region Static Data Retrieving Methods
/// <summary>
/// Find a specific class note.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="postID">The associated post ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns a ClassNote class. Null if not found.</returns>
public static ClassNote FindClassNote(string classID, int postID, DataBridge dataBridge)
{
    string paddedPostID;

    string postFilePath;
    try
    {
        postFilePath = dataBridge.GetClassNoteFilePath(classID, postID, out paddedPostID, true);
    }
    catch
    {
        // Not found
        return null;
    }
}
XmlSerializer xmlSerializer = new XmlSerializer(typeof(ClassNote));
XmlTextReader xmlTextReader;
    // Deserialise
    ClassNote newCN = (ClassNote)xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(postFilePath));
    xmlTextReader.Close();
    newCN.DataBridge = dataBridge;
    return newCN;
}

/// <summary>
/// Finds all class notes for a specific class.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns an array of ClassNote classes. Null if none found.</returns>
public static ClassNote[] FindClassNotes(string classID, DataBridge dataBridge)
{
    string xmlDirectory = dataBridge.GetClassNotesDirectory(classID, true);
    string[] postFilePaths = Directory.GetFiles(xmlDirectory, "*.xml");
    ArrayList posts = new ArrayList(postFilePaths.Length);
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(ClassNote));
    XmlTextReader xmlTextReader;
    // Deserialise
    foreach (string postFilePath in postFilePaths)
    {
        ClassNote tempPost = xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(postFilePath)) as ClassNote;
        tempPost.DataBridge = dataBridge;
        posts.Add(tempPost);
        xmlTextReader.Close();
    }
    // Any found?
    if (posts.Count == 0)
        return null;
    else
        return (ClassNote[])posts.ToArray(typeof(ClassNote));
}

#endregion
/// <summary>
/// A sealed class that contains properties and methods of a class thread post.
/// </summary>
[XmlRoot("ThreadPost")]
public sealed class ClassThreadPost : Post
{
    #region Properties
    /// <summary>
    /// The associated class ID.
    /// </summary>
    [XmlAttribute("classID")]
    public string ClassID
    {
        get { return base._classID; } 
        set { base._classID = value; } 
    }

    /// <summary>
    /// The associated thread ID.
    /// </summary>
    [XmlAttribute("threadID")]
    public int ThreadID
    {
        get { return base._threadID; } 
        set { base._threadID = value; } 
    }

    /// <summary>
    /// Whether or not to show this thread post.
    /// </summary>
    [XmlAttribute("withhold")]
    public bool Withhold
    {
        get { return base._withhold; } 
        set { base._withhold = value; } 
    }
    #endregion

    #region Constructors
    /// <summary>
    /// Used for XmlSerializer purposes.
    /// </summary>
    public ClassThreadPost() {}
    /// <summary>
    /// Main Constructor.
    /// </summary>
    /// <param name="classID">The associated class ID.</param>
    /// <param name="threadID">The associated thread ID.</param>
    /// <param name="dataBridge">The DataBridge that connects to the data.</param>
    public ClassThreadPost(string classID, int threadID, DataBridge dataBridge) : base(dataBridge)
    {
    }
}
{ this.ClassID = classID;
    this.ThreadID = threadID;
}

#region Static Data Retrieving Methods
/// <summary>
/// Find a specific class thread post.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="threadID">The associated thread ID.</param>
/// <param name="postID">The associated post ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
/// <returns>Returns a ClassThreadPost class. Null if not found.</returns>
public static ClassThreadPost FindClassThreadPost(string classID, int threadID, int postID, DataBridge dataBridge)
{
    string paddedPostID;
    string postFilePath;
    try
    {
        postFilePath = dataBridge.GetClassThreadPostFilePath(classID, threadID, postID, out paddedPostID, true);
    }
    catch
    {
        // Not found
        return null;
    }

    XmlSerializer xmlSerializer = new XmlSerializer(typeof(ClassThreadPost));
    XmlTextReader xmlTextReader;
    // Deserialise
    ClassThreadPost newCTP = (ClassThreadPost)xmlSerializer.Deserialize(xmlTextReader = new XmlTextReader(postFilePath));
    xmlTextReader.Close();

    newCTP.DataBridge = dataBridge;

    return newCTP;
}

/// <summary>
/// Finds all class thread posts for a specific class.
/// </summary>
/// <param name="classID">The associated class ID.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
Returns an array of ClassThreadPost classes.
Null if none found.

```csharp
public static ClassThreadPost[] FindClassThreadPosts(string classID, int threadID, DataBridge dataBridge)
{
    string xmlDirectory = dataBridge.
        GetClassThreadPostsDirectory(classID, threadID, true);

    // Get files
    string[] postFilePaths = Directory.GetFiles(xmlDirectory
        , "*.xml");

    ArrayList posts = new ArrayList(postFilePaths.Length);
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(ClassThreadPost));
    XmlTextReader xmlTextReader;
    // Deserialise
    foreach (string postFilePath in postFilePaths)
    {
        ClassThreadPost tempPost = xmlSerializer.Deserializer
            (xmlTextReader = new XmlTextReader(postFilePath))
            as ClassThreadPost;
        tempPost.DataBridge = dataBridge;
        posts.Add(tempPost);
    }

    // Any found?
    if (posts.Count == 0)
        return null;
    else
        return (ClassThreadPost[]) posts.ToArray(typeof(ClassThreadPost));
}
```

Listing H.104: Role.cs
/// The highest authenticated users within the OLE.
/// Administrators,

/// Users that controls Students within the OLE.
/// Instructors,

/// The lowest authenticated users within the OLE.
/// Students

/// A class which holds properties and methods that define a role.
///
public sealed class Role
{
    #region Properties

    /// A string ID that defines a role.
    ///
    private RoleID _roleID;
    public RoleID RoleID
    {
        get { return this._roleID; }
        set { this._roleID = value; }
    }
    #endregion

    #region Constructors

    /// Used for XmlSerializer purposes.
    ///
    public Role() {}
    #endregion

    #region Data Management Properties

    public Role(RoleID roleID, DataBridge dataBridge)
    {
        this._roleID = roleID;
        this._dataBridge = dataBridge;
    }
    #endregion
/// <summary>
/// The DataBridge that connects to the data.
/// </summary>
private DataBridge _dataBridge = null;

[XmlIgnore()]
public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}

#region Data Retrieving Methods

/// <summary>
/// Assign a specific user to a role within a specific class.
/// </summary>

/// <param name="userID">The ID of the user.</param>
/// <param name="classID">The ID of the class.</param>
public void AssignUserToClass(int userID, string classID)
{
    SqlCommand sqlCommand = new SqlCommand("AssignUserToClass", this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters["@userID"].Value = userID;
    sqlCommand.Parameters.Add("@roleID", SqlDbType.VarChar, RoleLimitations.RoleID);
    sqlCommand.Parameters["@roleID"].Value = this.RoleID.ToString();
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = classID;
    this.DataBridge.SqlConnection.Open();
    // Execute the stored procedure
    sqlCommand.ExecuteNonQuery();
    this.DataBridge.SqlConnection.Close();
}

/// <summary>
/// Unassign a specific user from a role within a specific class.
/// </summary>

/// <param name="userID">The ID of the user.</param>
/// <param name="classID">The ID of the class.</param>
public void UnassignUserToClass(int userID, string classID)
{
    SqlCommand sqlCommand = new SqlCommand("UnassignUserToClass", this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
    sqlCommand.Parameters["@userID"].Value = userID;
    sqlCommand.Parameters.Add("@roleID", SqlDbType.VarChar, RoleLimitations.RoleID);
    sqlCommand.Parameters["@roleID"].Value = this.RoleID.
/// Assign a specific user to a role.
/// 
/// <param name="userID">The ID of the user.</param>
public void AssignUser(int userID)
{
    SqlCommand sqlCommand = new SqlCommand("AssignUser",
            this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
    sqlCommand.Parameters["@userID"].Value = userID;
    sqlCommand.Parameters.Add("@roleID", SqlDbType.VarChar,
            RoleLimitations.RoleID);
    sqlCommand.Parameters["@roleID"].Value = this.RoleID.
            ToString();
    this.DataBridge.SqlConnection.Open();
    sqlCommand.ExecuteNonQuery();
    this.DataBridge.SqlConnection.Close();
}

/// Unassign a specific user to a role.
/// 
/// <param name="userID">The ID of the user.</param>
public void UnassignUser(int userID)
{
    SqlCommand sqlCommand = new SqlCommand("UnassignUser",
            this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
    sqlCommand.Parameters["@userID"].Value = userID;
    sqlCommand.Parameters.Add("@roleID", SqlDbType.VarChar,
            RoleLimitations.RoleID);
    sqlCommand.Parameters["@roleID"].Value = this.RoleID.
            ToString();
    this.DataBridge.SqlConnection.Open();
    sqlCommand.ExecuteNonQuery();
    this.DataBridge.SqlConnection.Close();
}
/// Find all roles associated with a user’s ID.
/// <param name="userID">The ID of the user.</param>
/// <param name="dataBridge">The bridge to the data.</param>
/// <returns>Returns a Role object array of the associated roles.</returns>

class OLEBusinessLogic
{
    public static Role[] FindRoles(int userID, DataBridge dataBridge)
    {
        // Get roles
        RoleID[] rolesIDs = FindRolesIDs(userID, dataBridge);
        if (rolesIDs == null)
            return null;

        ArrayList roles = new ArrayList(rolesIDs.Length);
        // Add Roles to an arraylist
        foreach (RoleID roleID in rolesIDs)
        {
            roles.Add(new Role(roleID, dataBridge));
        }

        if (roles.Count == 0)
            return null;
        else
            return (Role[]) roles.ToArray(typeof(Role));
    }
}

class DataBridge
{
    public static RoleID[] FindRolesIDs(int userID, string dataBridge)
    {
        ArrayList rolesArray = new ArrayList();

        SqlCommand sqlCommand = new SqlCommand("FindRolesIDs", dataBridge.SqlConnection);
        sqlCommand.CommandType = CommandType.StoredProcedure;
        sqlCommand.Parameters.AddWithValue("@userID", SqlDbType.Int);
        sqlCommand.Parameters["@userID"].Value = userID;
        dataBridge.SqlConnection.Open();
        // Execute stored procedure and retrieve data
        SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();
        // Read data from DataReader and add to an arraylist
        while (sqlDataReader.Read())
        {
            switch (sqlDataReader["roleID"] as string)
            {
                case "Administrators":
                    rolesArray.Add(RoleIDAdministrators);
                    break;
            }
        }
        return (RoleID[]) rolesArray.ToArray(typeof(RoleID));
    }
}
break;
case "Instructors":
rolesArray.Add(RoleID.Instructors);
break;
case "Students":
rolesArray.Add(RoleID.Students);
break;
}
}
dataBridge.SqlConnection.Close();

if (rolesArray.Count == 0)
return null;
else
return (RoleID[])rolesArray.ToArray(typeof(RoleID));

#endregion

using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A class which holds properties and methods that define a thread.
    /// </summary>
    [XmlRoot("Thread")]
    public sealed class Thread
    {
    #region Properties
    /// <summary>
    /// The ID associated to the thread.
    /// </summary>
    private int _threadID;
    [XmlAttribute("threadID")]
    public int ThreadID
    {
        get { return this._threadID; }
        set { this._threadID = value; }
    }
    
    /// <summary>
    /// The ID of the class associated to the thread.
    /// </summary>
    private string _classID;
```csharp
public string ClassID
{
    get { return this._classID; }
    set { this._classID = value; }
}

/// <summary>
/// The title of the thread.
/// </summary>
private string _title;

public string Title
{
    get { return this._title; }
    set { this._title = value; }
}

/// <summary>
/// The thread creator's full name.
/// </summary>
private string _creatorName;

public string CreatorName
{
    get { return this._creatorName; }
    set { this._creatorName = value; }
}

/// <summary>
/// The date and time the thread was created.
/// </summary>
private DateTime _createdDateTime;

public DateTime CreatedDateTime
{
    get { return this._createdDateTime; }
    set { this._createdDateTime = value; }
}

#endregion

#region Constructors
/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>
public Thread() {}

/// <summary>
/// Main constructor.
/// </summary>
/// <param name="dataBridge">The bridge to the data.</param>
public Thread(DataBridge dataBridge)
{
    this._dataBridge = dataBridge;
}
```
#endregion

#region Data Management Properties

/// <summary>
/// The DataBridge that connects to the data.
/// </summary>

private DataBridge _dataBridge = null;

[XmlIgnore()]
public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}

#endregion

#region Data Management Methods

/// <summary>
/// Creates an xml file entry of a thread.
/// </summary>

/// <returns>Returns the numeric ID of the new thread.</returns>

public int Create()
{
    int numericThreadID;
    string paddedThreadID;

    // Get next filepath and thread's directory ready for storage
    string threadFilePath = this.DataBridge.GetNextClassThreadFilePath(this.ClassID, out numericThreadID, out paddedThreadID);

    string threadPostsDirectory = this.DataBridge.GetClassThreadPostsDirectory(this.ClassID, numericThreadID, false);

    // Assign ID
    this.ThreadID = numericThreadID;

    // Serialize thread to the xml file
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(Thread));
    XmlTextWriter xmlTextWriter = new XmlTextWriter(threadFilePath, Encoding.UTF8);
    xmlSerializer.Serialize(xmlTextWriter, this);
    xmlTextWriter.Close();

    // Create directory for posts
    Directory.CreateDirectory(threadPostsDirectory);

    return numericThreadID;
}

/// <summary>
/// Alters the thread's xml file information.
/// </summary>
public void Alter()
{
    string paddedThreadID;

    // Get the filepath of the thread
    string threadFilePath = this.DataBridge.GetDataBridge().GetClassThreadFilePath(this.ClassID, this.ThreadID, out paddedThreadID, true);

    // Serialize the thread back to its xml file with the changes
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(Thread));
    XmlTextWriter xmlTextWriter;
    xmlSerializer.Serialize(xmlTextWriter = new XmlTextWriter(threadFilePath, Encoding.UTF8), this);
    xmlTextWriter.Close();
}

public void Delete()
{
    string paddedThreadID;

    // Get the filepath
    string threadFilePath = this.DataBridge.GetDataBridge().GetClassThreadFilePath(this.ClassID, this.ThreadID, out paddedThreadID, true);

    // Delete the xml file
    File.Delete(threadFilePath);
}

#endregion

#region Data Retrieving Methods
/// <summary>
/// Creates a subscription to a thread for a user.
/// </summary>

public void CreateThreadSubscription(string classID, int userID)
{
    SqlCommand sqlCommand = new SqlCommand("CreateThreadSub", this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
    sqlCommand.Parameters["@userID"].Value = classID;
    sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
    sqlCommand.Parameters["@userID"].Value = userID;
    sqlCommand.Parameters.Add("@threadID", SqlDbType.Int);
    sqlCommand.Parameters["@threadID"].Value = this.ThreadID;
    this.DataBridge.SqlConnection.Open();

    // Execute SQL Stored Procedure

/// Deletes a subscription to a thread for a user.
/// <param name="classID">The associated class ID.</param>
/// <param name="userID">The associated user ID.</param>
public void DeleteThreadSubscription(string classID, int userID)
{
    SqlCommand sqlCommand = new SqlCommand("DeleteThreadSub", this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@classID", SqlDbType.Char, ClassLimitations.ClassID);
    sqlCommand.Parameters["@classID"].Value = classID;
    sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
    sqlCommand.Parameters["@userID"].Value = userID;
    sqlCommand.Parameters.Add("@threadID", SqlDbType.Int);
    sqlCommand.Parameters["@threadID"].Value = this.ThreadID;
    this.DataBridge.SqlConnection.Open();
    // Execute SQL Stored Procedure
    sqlCommand.ExecuteNonQuery();
    this.DataBridge.SqlConnection.Close();
}

#region Static Data Retrieving Methods
/// <summary>
/// Find a specific thread that belongs within a specific class.
/// </summary>
/// <param name="classID">The associated class’ ID.</param>
/// <param name="threadID">The ID of the thread.</param>
/// <param name="dataBridge">The bridge to the data.</param>
/// <returns></returns>
public static Thread FindThread(string classID, int threadID, DataBridge dataBridge)
{
    string paddedThreadId;
    string threadFilePath;
    try
    {
        // Get the filepath of the xml file
        threadFilePath = dataBridge.GetClassThreadFilePath(classID, threadID, out paddedThreadId, true);
    }
    catch
    {
        // Does not exist
        return null;
    }

    // Further processing...
}
#endregion
XMLSerializer xmlSerializer = new XMLSerializer(typeof(Thread));

XmlNodeReader xmlTextReader;

Thread newThread = (Thread)xmlSerializer.Deserialize(
    xmlTextReader = new XmlNodeReader(threadFilePath));

xmlTextReader.Close();

if (newThread == null)
    return null;
else
    return newThread;
}

/// <summary>
/// Finds all threads that belong to a specific class ID.
/// </summary>
/// <param name="classID">The ID of the associated class.</param>
/// <param name="dataBridge">The bridge to the data.</param>
/// <returns>Returns an array of all threads associated with the class.</returns>
public static Thread[] FindAllThreads(string classID,
DataBridge dataBridge)
{
    // Retrieve its directory
    string xmlDirectory = dataBridge.
GetClassThreadsDirectory(classID, true);

    ArrayList threads = new ArrayList();

    XMLSerializer xmlSerializer = new XMLSerializer(typeof(Thread));

    XmlNodeReader xmlTextReader;

    // Get file paths and deserialize them into a thread arraylist
    foreach (string filename in Directory.GetFiles( xmlDirectory, "*.xml") )
    {
        threads.Add(xmlSerializer.Deserialize(xmlTextReader = new XmlNodeReader(filename)));

        xmlTextReader.Close();
    }

    if (threads.Count == 0)
        return null;
else
    return (Thread[])threads.ToArray(typeof(Thread));
using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;
using System.Security;
using System.Security.Principal;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace OnlineLearningEnvironment
{
    /// <summary>
    /// A class which holds properties and methods that define a user.
    /// </summary>
    [XmlRoot("User")]
    public sealed class User : IPrincipal
    {
        #region IPrincipal Properties
        /// <summary>
        /// Implemented as part of the IPrincipal interface.
        /// </summary>
        private Identity _identity;
        [XmlIgnore()]
        public Identity Identity
        {
            get { return this._identity; }
        }
        #endregion

        #region IPrincipal Methods
        /// <summary>
        /// Implemented as part of the IPrincipal interface.
        /// </summary>
        /// <param name="roleID">The Role ID.</param>
        /// <returns> Returns true if part of a role and false if otherwise.</returns>
        public bool IsInRole(string roleID)
        {
            if (this._userID == 0)
                throw new Exception("UserID not specified!");

            // Search for role
            if (this.RoleIds != null)
                switch (roleID)
                {
                    case "Administrators":
                        return (Array.BinarySearch(this.RoleIds, RoleID.Administrators) >= 0) ? true : false;
                    case "Instructors":
                        return (Array.BinarySearch(this.RoleIds, RoleID.Instructors) >= 0) ? true : false;
                    default:
                        return false;
                }
        }
    }
}
RoleID.Instructors) >= 0) ? true : false;
case "Students":
    return (Array.BinarySearch(this.RoleIDs,
        RoleID.Students) >= 0) ? true : false;
}

    return false;
}
#endregion
#region Properties
/// <summary>
/// The user’s numeric ID. Also used as username.
/// </summary>
private int _userID = 0;
[XmlAttribute("userID")]
public int UserID
{
    get { return this._userID; }
    set { this._userID = value; }
}

/// <summary>
/// The user’s password in MD5 encryption form.
/// </summary>
private string _password;
[XmlElement("Password")]
public string Password
{
    get { return this._password; }
    set { this._password = value; }
}

/// <summary>
/// Used to set and convert the user’s password from a
/// string to an MD5 encryption string.
/// </summary>
public string SetEncryptedPassword
{
    set { this._password = MD5Encryption.EncryptStringToByteString(value); }
}

private RoleID[] _roleIDs = null;
/// <summary>
/// The user’s roles.
/// </summary>
[XmlIgnore()]
public RoleID[] RoleIDs
{
    get { return this._roleIDs; }
    set
    {
        RoleID[] roles = value;
        if (roles != null)
            Array.Sort(roles);
this._roleIDs = roles;

private string _firstName;
/// <summary>
/// The user's first name.
/// </summary>
[XmlElement("FirstName")]
public string FirstName
{
    get { return this._firstName; }
    set { this._firstName = value; }
}

private string _middleName;
/// <summary>
/// The user's middle name.
/// </summary>
[XmlElement("MiddleName")]
public string MiddleName
{
    get { return this._middleName; }
    set { this._middleName = value; }
}

private string _lastName;
/// <summary>
/// The user's last name.
/// </summary>
[XmlElement("LastName")]
public string LastName
{
    get { return this._lastName; }
    set { this._lastName = value; }
}

/// <summary>
/// Gets the user's full name from the first, middle, and last name properties.
/// </summary>
[XmlIgnore()]
public string FullName
{
    get
    {
        if (this.MiddleName == "")
            return (this.FirstName + " " + this.LastName);
        else
            return (this.FirstName + " " + this.MiddleName + " " + this.LastName);
    }
}

private DateTime _dob;
/// <summary>
/// The user’s date of birth.
/// </summary>
[XmlAttribute("dob")]
public DateTime DOB
{
    get { return this._dob; }
    set { this._dob = value; }
}

private AddressStructure _postalAddress;
/// <summary>
/// The user’s postal address.
/// </summary>
[XmlElement("PostalAddress")]
public AddressStructure PostalAddress
{
    get { return this._postalAddress; }
    set { this._postalAddress = value; }
}

private AddressStructure _residentAddress;
/// <summary>
/// The user’s residential address.
/// </summary>
[XmlElement("ResidentAddress")]
public AddressStructure ResidentAddress
{
    get { return this._residentAddress; }
    set { this._residentAddress = value; }
}

private string _phoneNumber;
/// <summary>
/// The user’s home phone number.
/// </summary>
[XmlElement("PhoneNumber")]
public string PhoneNumber
{
    get { return _phoneNumber; }
    set { this._phoneNumber = value; }
}

private string _mobileNumber;
/// <summary>
/// The user’s mobile phone number.
/// </summary>
[XmlElement("MobileNumber")]
public string MobileNumber
{
    get { return _mobileNumber; }
    set { this._mobileNumber = value; }
}

private string _emailAddress;
/// <summary>
/// The user’s email address.
/// </summary>
[XmlElement("EmailAddress")]
public string EmailAddress
{
    get { return _emailAddress; }
    set { this._emailAddress = value; }
}
#endregion

#region Constructors
/// <summary>
/// Used for XmlSerializer purposes.
/// </summary>
public User()
{}

/// <summary>
/// Main constructor.
/// </summary>
/// <param name="dataBridge">The data bridge to the data.</param>
public User(DataBridge dataBridge)
{
    this._dataBridge = dataBridge;
}

/// <summary>
/// Main constructor.
/// </summary>
/// <param name="iIdentity">Used for the IPrincipal interface.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
public User(IIdentity iIdentity, DataBridge dataBridge)
{
    this._identity = iIdentity;
    this._dataBridge = dataBridge;
}

/// <summary>
/// Main constructor.
/// </summary>
/// <param name="userID">The user’s ID.</param>
/// <param name="iIdentity">Used for the IPrincipal interface.</param>
/// <param name="dataBridge">The DataBridge that connects to the data.</param>
public User(int userID, IIdentity iIdentity, DataBridge dataBridge)
{
    this._userId = userID;
    this._identity = iIdentity;
    this._dataBridge = dataBridge;
}
#endregion

#region Data Management Properties

/// <summary>
/// The DataBridge that connects to the data.
/// </summary>
private DataBridge _dataBridge = null;

[XmlIgnore()]
public DataBridge DataBridge
{
    get { return this._dataBridge; }
    set { this._dataBridge = value; }
}
#endregion

#region Data Management Methods

/// <summary>
/// Creates an entry in the database of a user.
/// </summary>
/// <returns>Returns the user's newly assigned ID.</returns>
public int Create()
{
    SqlCommand sqlCommand = new SqlCommand("CreateUser", this.DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
    sqlCommand.Parameters["@userID"].Direction = ParameterDirection.Output;
    sqlCommand.Parameters.Add("@password", SqlDbType.Char, UserLimitations.Password);
    sqlCommand.Parameters["@password"].Value = this.Password;
    sqlCommand.Parameters.Add("@firstName", SqlDbType.VarChar, UserLimitations.FirstName);
    sqlCommand.Parameters["@firstName"].Value = this.FirstName;
    sqlCommand.Parameters.Add("@middleName", SqlDbType.VarChar, UserLimitations.MiddleName);
    sqlCommand.Parameters["@middleName"].Value = this.MiddleName;
    sqlCommand.Parameters.Add("@lastName", SqlDbType.VarChar, UserLimitations.LastName);
    sqlCommand.Parameters["@lastName"].Value = this.LastName;
    sqlCommand.Parameters.Add("@dob", SqlDbType.DateTime);
    sqlCommand.Parameters["@dob"].Value = this.DOB;
    sqlCommand.Parameters.Add("@postalStreet", SqlDbType.VarChar, AddressLimitations.Street);
    sqlCommand.Parameters["@postalStreet"].Value = this.PostalAddress.Street;
    sqlCommand.Parameters.Add("@postalSuburb", SqlDbType.VarChar, AddressLimitations.Suburb);
    sqlCommand.Parameters["@postalSuburb"].Value = this.PostalAddress.Suburb;
    sqlCommand.Parameters.Add("@postalState", SqlDbType.VarChar,
```csharp
    SqlCommand.Parameters["@postalState"].Value = this.PostalAddress.State;
    SqlCommand.Parameters.Add("@postalPostcode", SqlDbType.VarChar, AddressLimitations.Postcode);
    SqlCommand.Parameters["@postalPostcode"].Value = this.PostalAddress.Postcode;
    SqlCommand.Parameters.Add("@postalCountry", SqlDbType.VarChar, AddressLimitations.Country);
    SqlCommand.Parameters["@postalCountry"].Value = this.PostalAddress.Country;
    SqlCommand.Parameters.Add("@residentStreet", SqlDbType.VarChar, AddressLimitations.Street);
    SqlCommand.Parameters["@residentStreet"].Value = this.ResidentAddress.Street;
    SqlCommand.Parameters.Add("@residentSuburb", SqlDbType.VarChar, AddressLimitations.Suburb);
    SqlCommand.Parameters["@residentSuburb"].Value = this.ResidentAddress.Suburb;
    SqlCommand.Parameters.Add("@residentState", SqlDbType.VarChar, AddressLimitations.State);
    SqlCommand.Parameters["@residentState"].Value = this.ResidentAddress.State;
    SqlCommand.Parameters.Add("@residentPostcode", SqlDbType.VarChar, AddressLimitations.Postcode);
    SqlCommand.Parameters["@residentPostcode"].Value = this.ResidentAddress.Postcode;
    SqlCommand.Parameters.Add("@residentCountry", SqlDbType.VarChar, AddressLimitations.Country);
    SqlCommand.Parameters["@residentCountry"].Value = this.ResidentAddress.Country;
    SqlCommand.Parameters.Add("@phoneNumber", SqlDbType.VarChar, PhoneNumber);
    SqlCommand.Parameters["@phoneNumber"].Value = this.PhoneNumber;
    SqlCommand.Parameters.Add("@mobileNumber", SqlDbType.VarChar, MobileNumber);
    SqlCommand.Parameters["@mobileNumber"].Value = this.MobileNumber;
    SqlCommand.Parameters.Add("@emailAddress", SqlDbType.VarChar, UserLimitations.EmailAddress);
    SqlCommand.Parameters["@emailAddress"].Value = this.EmailAddress;
    this.DataBridge SqlConnection.Open();
    // Execute SQL Stored Procedure
    sqlCommand.ExecuteNonQuery();
    this setUserID = (int)sqlCommand.Parameters["@userID"].Value;
    return this.UserID;
} // <summary>
```
/// Alters an entry of a user within the database.
/// </summary>
public void Alter()
{
    SqlCommand sqlCommand = new SqlCommand("AlterUser", this .DataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
    sqlCommand.Parameters["@userID"].Value = this.UserID;
    sqlCommand.Parameters.Add("@password", SqlDbType.VarChar, UserLimitations.Password);
    sqlCommand.Parameters["@password"].Value = this.Password;
    sqlCommand.Parameters.Add("@firstName", SqlDbType.VarChar, UserLimitations.FirstName);
    sqlCommand.Parameters["@firstName"].Value = this.FirstName;
    sqlCommand.Parameters.Add("@middleName", SqlDbType.VarChar, UserLimitations.MiddleName);
    sqlCommand.Parameters["@middleName"].Value = this.MiddleName;
    sqlCommand.Parameters.Add("@lastName", SqlDbType.VarChar, UserLimitations.LastName);
    sqlCommand.Parameters["@lastName"].Value = this.LastName;
    sqlCommand.Parameters.Add("@dob", SqlDbType.Date);
    sqlCommand.Parameters["@dob"].Value = this.DOB;
    sqlCommand.Parameters.Add("@postalStreet", SqlDbType.VarChar, AddressLimitations.Street);
    sqlCommand.Parameters["@postalStreet"].Value = this.PostalAddress.Street;
    sqlCommand.Parameters.Add("@postalSuburb", SqlDbType.VarChar, AddressLimitations.Suburb);
    sqlCommand.Parameters["@postalSuburb"].Value = this.PostalAddress.Suburb;
    sqlCommand.Parameters.Add("@postalState", SqlDbType.VarChar, AddressLimitations.State);
    sqlCommand.Parameters["@postalState"].Value = this.PostalAddress.State;
    sqlCommand.Parameters.Add("@postalPostcode", SqlDbType.VarChar, AddressLimitations.Postcode);
    sqlCommand.Parameters["@postalPostcode"].Value = this.PostalAddress.Postcode;
    sqlCommand.Parameters.Add("@postalCountry", SqlDbType.VarChar, AddressLimitations.Country);
    sqlCommand.Parameters["@postalCountry"].Value = this.PostalAddress.Country;
    sqlCommand.Parameters.Add("@residentStreet", SqlDbType.VarChar, AddressLimitations.Street);
    sqlCommand.Parameters["@residentStreet"].Value = this.ResidentAddress.Street;
    sqlCommand.Parameters.Add("@residentSuburb", SqlDbType.VarChar, AddressLimitations.Suburb);
    sqlCommand.Parameters["@residentSuburb"].Value = this.ResidentAddress.Suburb;
}
sqlCommand.Parameters.Add("@residentState", SqlDbType.VarChar, AddressLimitations.State);
sqlCommand.Parameters["@residentState"].Value = this.ResidentAddress.State;
sqlCommand.Parameters.Add("@residentPostcode", SqlDbType.VarChar, AddressLimitations.Postcode);
sqlCommand.Parameters["@residentPostcode"].Value = this.ResidentAddress.Postcode;
sqlCommand.Parameters.Add("@residentCountry", SqlDbType.VarChar, AddressLimitations.Country);
sqlCommand.Parameters["@residentCountry"].Value = this.ResidentAddress.Country;
sqlCommand.Parameters.Add("@phoneNumber", SqlDbType.VarChar, UserLimitations.PhoneNumber);
sqlCommand.Parameters["@phoneNumber"].Value = this.PhoneNumber;
sqlCommand.Parameters.Add("@mobileNumber", SqlDbType.VarChar, UserLimitations.PhoneNumber);
sqlCommand.Parameters["@mobileNumber"].Value = this.MobileNumber;
sqlCommand.Parameters.Add("@emailAddress", SqlDbType.VarChar, UserLimitations.EmailAddress);
sqlCommand.Parameters["@emailAddress"].Value = this.EmailAddress;
this.DataBridge.SqlConnection.Open();
// Execute SQL Stored Procedure
sqlCommand.ExecuteNonQuery();
this.DataBridge.SqlConnection.Close();
}

/// <summary>
/// Delete a user from within the database.
/// </summary>
public void Delete()
{
SqlCommand sqlCommand = new SqlCommand("DeleteUser", this.DataBridge.SqlConnection);
sqlCommand.CommandType = CommandType.StoredProcedure;
sqlCommand.Parameters.Add("@userID", SqlDbType.Int);
sqlCommand.Parameters["@userID"].Value = this.UserID;
this.DataBridge.SqlConnection.Open();
// Execute SQL Stored Procedure
sqlCommand.ExecuteNonQuery();
this.DataBridge.SqlConnection.Close();
}
#endregion

#region Static Data Retrieving Methods
/// <summary>
/// Find a specific user.
/// </summary>
/// <param name="userID">The ID of the user to find.</param>
/// <param name="dataBridge">The bridge to the data.</param>
/// <returns>Returns a User class or null if not found.</returns>
public static User FindUser(int userID, DataBridge dataBridge)
{
    SqlCommand sqlCommand = new SqlCommand("FindUser", dataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.AddWithValue("@userID", SqlDbType.Int).Value = userID;

    dataBridge.SqlConnection.Open();
    XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.ExecuteXmlReader();

    User user = null;
    XmlSerializer xmlSerializer = new XmlSerializer(typeof(User));
    // Read and deserialize xml to a User object
    while (xmlTextReader.Read())
    {
        if (xmlSerializer.CanDeserialize(xmlTextReader))
            user = xmlSerializer.Deserialize(xmlTextReader) as User;
    }

    xmlTextReader.Close();
    dataBridge.SqlConnection.Close();

    if (user == null)
        return null;
    else
    {
        user.DataBridge = dataBridge;
        return user;
    }
}

/// <summary>
/// Find all users within a specific class.
/// </summary>
/// <param name="classID">The ID of the class to search for users.</param>
/// <param name="dataBridge">The bridge to the data.</param>
/// <returns>Returns an array of all users or null if not found.</returns>
public static User[] FindUsers(string classID, DataBridge dataBridge)
{
    SqlCommand sqlCommand = new SqlCommand("FindUsers", dataBridge.SqlConnection);
    sqlCommand.CommandType = CommandType.StoredProcedure;
    sqlCommand.Parameters.AddWithValue("@classID", SqlDbType.Char, ClassLimitations.ClassID).Value = classID;
    dataBridge.SqlConnection.Open();
    // Execute stored procedure and retrieve as XML
XmlTextReader xmlTextReader = (XmlTextReader)sqlCommand.
   ExecuteXmlReader();

ArrayList users = new ArrayList();
XmlSerializer xmlSerializer = new XmlSerializer(typeof(User));
   // Read and deserialize xml to a User object
   arraylist
while (xmlTextReader.Read())
{
    while (xmlSerializer.CanDeserialize(xmlTextReader))
    {
       User tempUser = xmlSerializer.Deserialize(
       xmlTextReader) as User;
       tempUser.DataBridge = dataBridge;
       users.Add(tempUser);
    }
}

xmlTextReader.Close();
dataBridge.SqlConnection.Close();

if (users == null)
    return null;
else
    return (User[])users.ToArray(typeof(User));

/* Generated by Web Data Administrator on 5/08/2004 4:08:22 PM */
/* Options selected: database drop—commands table—schema table—data
   stored—procedures comments */
******/
Object: Database oledb    Script Date: 5/08/2004 4:08:22 PM
******
IF EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'oledb')
    DROP DATABASE [oledb]
GO
CREATE DATABASE [oledb]
GO
exec sp_dboption N'oledb', N'autoclose', N'true'
GO
exec sp_dboption 'oledb', 'bulkcopy', 'false'
GO
exec sp_dboption 'oledb', 'trunc_log', 'true'
GO
exec sp_dboption 'oledb', 'torn_page_detection', 'true'
GO
exec sp_dboption 'oledb', 'read_only', 'false'
GO
exec sp_dboption 'oledb', 'dbo_use', 'false'
GO
exec sp_dboption 'oledb', 'single', 'false'
GO
exec sp_dboption 'oledb', 'autoshrink', 'true'
GO
exec sp_dboption 'oledb', 'ansi_null_default', 'false'
GO
exec sp_dboption 'oledb', 'recursive_triggers', 'false'
GO
exec sp_dboption 'oledb', 'ansi_nulls', 'false'
GO
exec sp_dboption 'oledb', 'concat_null_yields_null', 'false'
GO
exec sp_dboption 'oledb', 'cursor_close_on_commit', 'false'
GO
exec sp_dboption 'oledb', 'default_to_local_cursor', 'false'
GO
exec sp_dboption 'oledb', 'quoted_identifier', 'false'
GO
exec sp_dboption 'oledb', 'ansi_warnings', 'false'
GO
exec sp_dboption 'oledb', 'auto_create_statistics', 'true'
GO
exec sp_dboption 'oledb', 'auto_update_statistics', 'true'
GO
if ( ( @@microsoftversion / power(2, 24) = 8) and (@@microsoftversion & 0xffff >= 724) ) or ( ( @@microsoftversion / power(2, 24) = 7) and ( @@microsoftversion & 0xffff >= 1082) )
exec sp_dboption 'oledb', 'db_chaining', 'false'

use [oledb]
exec sp_adduser 'NT\AUTHORITY\Network\Service', 'NT\AUTHORITY\Network\Service', 'db_owner'
exec sp_grantlogin 'NT\AUTHORITY\Network\Service'

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO

GO
GO
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Authentication]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[Authentication]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateAETAnswer]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateAETAnswer]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateAETQuestion]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateAETQuestion]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateClass]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateClass]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateRole]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateRole]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateThreadSub]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateThreadSub]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[CreateUser]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[CreateUser]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[DeleteAETAnswer]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[DeleteAETAnswer]
if exists (select * from dbo.sysobjects where id = object_id(N' [dbo].[DeleteAETQuestion]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
    drop procedure [dbo].[DeleteAETQuestion]
GO

if exists (select * from dbo.sysobjects where id = object_id(N' [dbo].[DeleteClass]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
    drop procedure [dbo].[DeleteClass]
GO

if exists (select * from dbo.sysobjects where id = object_id(N' [dbo].[DeleteRole]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
    drop procedure [dbo].[DeleteRole]
GO

if exists (select * from dbo.sysobjects where id = object_id(N' [dbo].[DeleteThreadSub]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
    drop procedure [dbo].[DeleteThreadSub]
GO

if exists (select * from dbo.sysobjects where id = object_id(N' [dbo].[DeleteUser]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
    drop procedure [dbo].[DeleteUser]
GO

if exists (select * from dbo.sysobjects where id = object_id(N' [dbo].[FindClass]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
    drop procedure [dbo].[FindClass]
GO

if exists (select * from dbo.sysobjects where id = object_id(N' [dbo].[FindClasses]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
    drop procedure [dbo].[FindClasses]
GO

if exists (select * from dbo.sysobjects where id = object_id(N' [dbo].[FindRolesIDs]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
    drop procedure [dbo].[FindRolesIDs]
GO
if exists ( select * from dbo.sysobjects where id = object_id(N'[dbo].[FindRolesIDs]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindRolesIDs]
GO

/****** Object:  Stored Procedure dbo.FindShortAETAnswer  Script Date: 5/08/2004 4:08:23 PM ******/
if exists ( select * from dbo.sysobjects where id = object_id(N'[dbo].[FindShortAETAnswer]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindShortAETAnswer]
GO

/****** Object:  Stored Procedure dbo.FindShortAETAnswers  Script Date: 5/08/2004 4:08:23 PM ******/
if exists ( select * from dbo.sysobjects where id = object_id(N'[dbo].[FindShortAETAnswers]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindShortAETAnswers]
GO

/****** Object:  Stored Procedure dbo.FindShortAETQuestion  Script Date: 5/08/2004 4:08:23 PM ******/
if exists ( select * from dbo.sysobjects where id = object_id(N'[dbo].[FindShortAETQuestion]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindShortAETQuestion]
GO

/****** Object:  Stored Procedure dbo.FindShortAETQuestions  Script Date: 5/08/2004 4:08:23 PM ******/
if exists ( select * from dbo.sysobjects where id = object_id(N'[dbo].[FindShortAETQuestions]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[FindShortAETQuestions]
GO

/****** Object:  Stored Procedure dbo.IsInClass  Script Date: 5/08/2004 4:08:23 PM ******/
if exists ( select * from dbo.sysobjects where id = object_id(N'[dbo].[IsInClass]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[IsInClass]
GO
drop procedure [dbo].[IsInClass]
GO

/***** Object: Table [dbo].[AETAnswers] Script Date: 5/08/2004 4:08:23 PM *****/
if exists (select * from dbo.sysobjects where id = object_id(N'[AETAnswers]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
  drop table [AETAnswers]
GO

/***** Object: Table [dbo].[AETQuestions] Script Date: 5/08/2004 4:08:23 PM *****/
if exists (select * from dbo.sysobjects where id = object_id(N'[AETQuestions]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
  drop table [AETQuestions]
GO

/***** Object: Table [dbo].[Classes] Script Date: 5/08/2004 4:08:23 PM *****/
if exists (select * from dbo.sysobjects where id = object_id(N'[Classes]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
  drop table [Classes]
GO

/***** Object: Table [dbo].[ClassUsers] Script Date: 5/08/2004 4:08:23 PM *****/
if exists (select * from dbo.sysobjects where id = object_id(N'[ClassUsers]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
  drop table [ClassUsers]
GO

/***** Object: Table [dbo].[OtherUsers] Script Date: 5/08/2004 4:08:23 PM *****/
if exists (select * from dbo.sysobjects where id = object_id(N'[OtherUsers]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
  drop table [OtherUsers]
GO

/***** Object: Table [dbo].[Roles] Script Date: 5/08/2004 4:08:23 PM *****/
if exists (select * from dbo.sysobjects where id = object_id(N'[Roles]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
  drop table [Roles]
GO

/***** Object: Table [dbo].[ThreadSubscriptions] Script Date: 5/08/2004 4:08:23 PM *****/
if exists (select * from dbo.sysobjects where id = object_id(N'[ThreadSubscriptions]') and OBJECTPROPERTY(id, N'IsUserTable') =
drop table [ThreadSubscriptions]
GO

/* **** Object: Table [dbo].[Users]     Script Date: 5/08/2004 4:08:23 PM *******/
if exists (select * from dbo.sysobjects where id = object_id(N'[Users]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [Users]
GO

/* **** Object: Table [dbo].[AETAnswers]     Script Date: 5/08/2004 4:08:23 PM *******/
CREATE TABLE [AETAnswers] (  
aetQID bigint NOT NULL ,
    classID char (7) COLLATE Latin1_General_CI_AS NOT NULL ,
    userID int NOT NULL ,
    aetQType varchar (20) COLLATE Latin1_General_CI_AS NOT NULL ,
    aetASubmittedDate datetime NULL ,
    aetAMarked bit NULL ) ON [PRIMARY]
GO

/* **** Object: Table [dbo].[AETQuestions]     Script Date: 5/08/2004 4:08:24 PM *******/
CREATE TABLE [AETQuestions] (  
aetQID bigint NOT NULL ,
    classID char (7) COLLATE Latin1_General_CI_AS NOT NULL ,
    aetQType varchar (20) COLLATE Latin1_General_CI_AS NOT NULL ,
    aetQTitle varchar (100) COLLATE Latin1_General_CI_AS NULL ,
    aetQDescription varchar (30) COLLATE Latin1_General_CI_AS NULL ,
    aetQQuestionType varchar (20) COLLATE Latin1_General_CI_AS NULL ,
    aetQViewQuestionDateTime datetime NULL ,
    aetQDueDateTime datetime NULL ,
    aetQHaltSubmissionDateTime datetime NULL ,
    aetQViewMarkedAnswersDateTime datetime NULL ,
    aetQAmountOfQuestions int NULL ) ON [PRIMARY]
GO

/* **** Object: Table [dbo].[Classes]       Script Date: 5/08/2004 4:08:24 PM *******/
CREATE TABLE [Classes] (  
    classID char (7) COLLATE Latin1_General_CI_AS NOT NULL ,
    className varchar (50) COLLATE Latin1_General_CI_AS NULL ,
    denyOtherUsers bit NULL ) ON [PRIMARY]
GO
CREATE TABLE [ClassUsers] (
    [classID] [char] (7) COLLATE Latin1_General_CI_AS NOT NULL,
    [userID] [int] NOT NULL,
    [roleID] [varchar] (50) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [OtherUsers] (
    [userID] [int] NOT NULL,
    [roleID] [varchar] (50) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [Roles] (
    [roleID] [varchar] (50) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [ThreadSubscriptions] (
    [userID] [int] NOT NULL,
    [threadID] [int] NOT NULL,
    [classID] [char] (7) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [Users] (
    [userID] [int] IDENTITY (1, 1) NOT NULL,
    [password] [char] (47) COLLATE Latin1_General_CI_AS NOT NULL,
    [firstName] [varchar] (50) COLLATE Latin1_General_CI_AS NULL,
    [middleName] [varchar] (50) COLLATE Latin1_General_CI_AS NULL,
    [lastName] [varchar] (50) COLLATE Latin1_General_CI_AS NULL,
    [dob] [datetime] NULL,
    [postalStreet] [varchar] (50) COLLATE Latin1_General_CI_AS NULL,
    [postalSuburb] [varchar] (50) COLLATE Latin1_General_CI_AS NULL,
    [postalState] [varchar] (50) COLLATE Latin1_General_CI_AS NULL,
)
[postalPostcode] [varchar] (10) COLLATE Latin1_General_CI_AS,
[postalCountry] [varchar] (50) COLLATE Latin1_General_CI_AS
NULL,
[residentStreet] [varchar] (50) COLLATE Latin1_General_CI_AS
NULL,
[residentSuburb] [varchar] (50) COLLATE Latin1_General_CI_AS
NULL,
[residentState] [varchar] (50) COLLATE Latin1_General_CI_AS
NULL,
[residentPostcode] [varchar] (10) COLLATE Latin1_General_CI_AS
NULL,
[residentCountry] [varchar] (50) COLLATE Latin1_General_CI_AS
NULL,
[phoneNumber] [varchar] (15) COLLATE Latin1_General_CI_AS
NULL,
[mobileNumber] [varchar] (15) COLLATE Latin1_General_CI_AS
NULL,
[emailAddress] [varchar] (50) COLLATE Latin1_General_CI_AS
NULL
) ON [PRIMARY]
GO

/* Data for table AETAnswers */
/* Data for table AETQuestions */
/* Data for table Classes */
/* Data for table ClassUsers */
/* Data for table OtherUsers */
INSERT [OtherUsers] ([userID], [roleID]) VALUES (1, 'Administrators')
/* Data for table Roles */
INSERT [Roles] ([roleID]) VALUES ('Administrators')
INSERT [Roles] ([roleID]) VALUES ('Instructors')
INSERT [Roles] ([roleID]) VALUES ('Students')
/* Data for table ThreadSubscriptions */
/* Data for table Users */
SET identity_insert [Users] on

INSERT [Users] ([userID], [password], [firstName], [middleName], [lastName], [dob], [postalStreet], [postalSuburb], [postalState],
[postalPostcode], [postalCountry], [residentStreet], [residentSuburb], [residentState], [residentPostcode],
[residentCountry], [phoneNumber], [mobileNumber], [emailAddress])
VALUES (1, '64−4B−45−7A−92−E4−66−A1−EB−14−33−0E−CE−BE−A6−B9',
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
SET identity_insert [Users] off
GO

<<<<<<<<<<< Object: Table [dbo].[AETAnswers] Script Date: 5/08/2004 4:08:28 PM >>>>>>>>>>
ALTER TABLE [AETAnswers] WITH NOCHECK ADD
CONSTRAINT [PK_AETAnswers] PRIMARY KEY CLUSTERED
(
[netQID], [classID], [userID],
```sql
/* Table [dbo].[AETQuestions] */
ALTER TABLE [dbo].[AETQuestions] WITH NOCHECK ADD
    CONSTRAINT [PK_AETQuestions] PRIMARY KEY CLUSTERED
    ([aetQID], [classID], [aetQType])
GO

/* Table [dbo].[Classes] */
ALTER TABLE [dbo].[Classes] WITH NOCHECK ADD
    CONSTRAINT [PK_Classes] PRIMARY KEY CLUSTERED
    ([classID])
GO

/* Table [dbo].[ClassUsers] */
ALTER TABLE [dbo].[ClassUsers] WITH NOCHECK ADD
    CONSTRAINT [PK_ClassUsers] PRIMARY KEY CLUSTERED
    ([classID], [userID], [roleID])
GO

/* Table [dbo].[OtherUsers] */
ALTER TABLE [dbo].[OtherUsers] WITH NOCHECK ADD
    CONSTRAINT [PK_OtherUsers] PRIMARY KEY CLUSTERED
    ([userID], [roleID])
GO

/* Table [dbo].[Roles] */
ALTER TABLE [dbo].[Roles] WITH NOCHECK ADD
    CONSTRAINT [PK_Roles] PRIMARY KEY CLUSTERED
    ( )
GO
```
H.3 Database Source Code Project

```sql
[roleID]
) ON [PRIMARY]
GO

/******** Object: Table [dbo].[ThreadSubscriptions]    Script Date:
5/08/2004 4:08:28 PM******/
ALTER TABLE [ThreadSubscriptions] WITH NOCHECK ADD
    CONSTRAINT [PK_ThreadSubscriptions] PRIMARY KEY CLUSTERED
    (
        [userID],
        [threadID],
        [classID]
    ) ON [PRIMARY]
GO

/******** Object: Table [dbo].[Users]    Script Date: 5/08/2004
4:08:28 PM******/
ALTER TABLE [Users] WITH NOCHECK ADD
    CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED
    (
        [userID]
    ) ON [PRIMARY]
GO

/******** Object: Table [dbo].[AETAnswers]    Script Date: 5/08/2004
4:08:28 PM******/
ALTER TABLE [AETAnswers] ADD
    CONSTRAINT [FK_AETAnswers_AETQuestions1] FOREIGN KEY
    (
        [aetQID],
        [classID],
        [aetQType]
    ) REFERENCES [AETQuestions] (
        [aetQID],
        [classID],
        [aetQType]
    ),
    CONSTRAINT [FK_AETAnswers_Classes] FOREIGN KEY
    (
        [classID]
    ) REFERENCES [Classes] (
        [classID]
    ) ON DELETE CASCADE ON UPDATE CASCADE ,
    CONSTRAINT [FK_AETAnswers_Users] FOREIGN KEY
    (
        [userID]
    ) REFERENCES [Users] (
        [userID]
    ) ON DELETE CASCADE ON UPDATE CASCADE
GO
```
/*----- Object: Table [dbo].[AETQuestions]  Script Date: 5/08/2004 4:08:28 PM -----*/
ALTER TABLE [AETQuestions] ADD
  CONSTRAINT [FK_AETQuestions_Classes] FOREIGN KEY
  ( [classID] ) 
  REFERENCES [Classes] ( [classID] ) 
  ON DELETE CASCADE  ON UPDATE CASCADE
GO

/*----- Object: Table [dbo].[ClassUsers]  Script Date: 5/08/2004 4:08:28 PM -----*/
ALTER TABLE [ClassUsers] ADD
  CONSTRAINT [FK_ClassUsers_Classes] FOREIGN KEY
  ( [classID] ) 
  REFERENCES [Classes] ( [classID] ) 
  ON DELETE CASCADE  ON UPDATE CASCADE ,
  CONSTRAINT [FK_ClassUsers_Roles] FOREIGN KEY
  ( [roleID] ) 
  REFERENCES [Roles] ( [roleID] ) 
  ON DELETE CASCADE  ON UPDATE CASCADE ,
  CONSTRAINT [FK_ClassUsers_Users] FOREIGN KEY
  ( [userID] ) 
  REFERENCES [Users] ( [userID] ) 
  ON DELETE CASCADE  ON UPDATE CASCADE
GO

/*----- Object: Table [dbo].[OtherUsers]  Script Date: 5/08/2004 4:08:29 PM -----*/
ALTER TABLE [OtherUsers] ADD
  CONSTRAINT [FK_OtherUsers_Roles] FOREIGN KEY
  ( [roleID] ) 
  REFERENCES [Roles] ( [roleID] ) 
  ON DELETE CASCADE  ON UPDATE CASCADE ,
  CONSTRAINT [FK_OtherUsers_Users] FOREIGN KEY
  ( [userID] ) 
  REFERENCES [Users] ( [userID] ) 
  ON DELETE CASCADE  ON UPDATE CASCADE
GO
/****** Object: Table [dbo].[ThreadSubscriptions] Script Date: 5/08/2004 4:08:29 PM******/
ALTER TABLE [ThreadSubscriptions] ADD
    CONSTRAINT [FK_ThreadSubscriptions_Classes] FOREIGN KEY
        ( [classID] ) REFERENCES [Classes] ( [classID] )
    ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT [FK_ThreadSubscriptions_Users] FOREIGN KEY
        ( [userID] ) REFERENCES [Users] ( [userID] )
    ON DELETE CASCADE ON UPDATE CASCADE
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object: Stored Procedure dbo.AlterAETAnswer Script Date: 5/08/2004 4:08:29 PM******/
CREATE PROCEDURE dbo.AlterAETAnswer
    (@aetQID BIGINT, @classID CHAR(7), @userID BIGINT,
        @aetQType VARCHAR(20), @aetASubmittedDate DATETIME,
        @aetAMarked BIT)
AS
    UPDATE AETAnswers
    SET aetASubmittedDate=@aetASubmittedDate, aetAMarked=@aetAMarked
WHERE aetQID=@aetQID AND classID=@classID AND userID=@userID AND aetQType=@aetQType
GO

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_NULLS ON
GO

CREATE PROCEDURE dbo.AlterAETQuestion
(
    @aetQID BIGINT,
    @classID CHAR(7),
    @aetQType VARCHAR(20),
    @aetQTitle VARCHAR(100),
    @aetQDescription VARCHAR(30),
    @aetQQuestionType VARCHAR(20),
    @aetQViewQuestionDateTime DATETIME,
    @aetQDueDateTime DATETIME,
    @aetQHaltSubmissionDateTime DATETIME,
    @aetQViewMarkedAnswersDateTime DATETIME,
    @aetQAmountOfQuestions INT
)
AS

    UPDATE AETQuestions
    SET aetQTitle=@aetQTitle, aetQDescription=@aetQDescription,
    aetQType=@aetQType, aetQQuestionType=@aetQQuestionType,
    aetQViewQuestionDateTime=@aetQViewQuestionDateTime,
    aetQDueDateTime=@aetQDueDateTime, aetQHaltSubmissionDateTime=
    @aetQHaltSubmissionDateTime, aetQViewMarkedAnswersDateTime=
    @aetQViewMarkedAnswersDateTime, aetQAmountOfQuestions=
    @aetQAmountOfQuestions
WHERE aetQID=@aetQID AND classID=@classID AND aetQType=@aetQType
GO

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_NULLS ON
GO

CREATE PROCEDURE dbo.AlterClass

/***** Object: Stored Procedure dbo.AlterAETQuestion Script Date: 5/08/2004 4:08:29 PM ******/

/***** Object: Stored Procedure dbo.AlterClass Script Date: 5/08/2004 4:08:29 PM *******/
(  
    @classID CHAR(7),
    @className VARCHAR(50),
    @denyOtherUsers BIT
  )
AS
UPDATE Classes
SET className=@className, denyOtherUsers=@denyOtherUsers
WHERE classID=@classID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

CREATE PROCEDURE dbo.AlterUser
(  
    @userID INT,
    @password CHAR(47),
    @firstName VARCHAR(50),
    @middleName VARCHAR(50),
    @lastName VARCHAR(50),
    @dob DATETIME,
    @postalStreet VARCHAR(50),
    @postalSuburb VARCHAR(50),
    @postalState VARCHAR(50),
    @postalPostcode VARCHAR(10),
    @postalCountry VARCHAR(50),
    @residentStreet VARCHAR(50),
    @residentSuburb VARCHAR(50),
    @residentState VARCHAR(50),
    @residentPostcode VARCHAR(10),
    @residentCountry VARCHAR(50),
    @phoneNumber VARCHAR(15),
    @mobileNumber VARCHAR(15),
    @emailAddress VARCHAR(50)
) AS
UPDATE Users
SET password=@password, firstName=@firstName, middleName=@middleName, lastName=@lastName, dob=@dob, postalStreet=@postalStreet, postalSuburb=@postalSuburb, postalState=@postalState, postalPostcode=@postalPostcode, postalCountry=@postalCountry, residentStreet=@residentStreet, residentSuburb=@residentSuburb, residentState=@residentState, residentPostcode=@residentPostcode, residentCountry=@residentCountry,
@userId=

GO

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_NULLS ON
GO

W H E R E  

userID=@userId

GO

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_NULLS ON
GO

---

CREATE PROCEDURE dbo.AssignUser

( @userId INT,
  @roleID VARCHAR(50)
 )

AS

INSERT INTO OtherUsers

( userID , roleID )

VALUES

( @userId , @roleID )

GO

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_NULLS ON
GO

---

CREATE PROCEDURE dbo.AssignUserToClass

( @userId INT,
  @classID CHAR(7),
  @roleID VARCHAR(50)
 )

AS

INSERT INTO ClassUsers

( userID , classID , roleID )

VALUES

( @userId , @classID , @roleID )

GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/

****** Object: Stored Procedure dbo.Authentication  Script Date:
5/08/2004 4:08:29 PM *****/
CREATE PROCEDURE dbo.Authentication
(
    @userID INT,
    @password CHAR(47)
)
AS
SELECT userID
FROM Users
WHERE userID=@userID AND password=@password
IF @@ROWCOUNT < 1
    RETURN 0
ELSE
    RETURN 1
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/

****** Object: Stored Procedure dbo.CreateAETAnswer  Script Date:
5/08/2004 4:08:29 PM *****/
CREATE PROCEDURE dbo.CreateAETAnswer
(
    @aetQID BIGINT,
    @classID CHAR(7),
    @userID BIGINT,
    @aetQType VARCHAR(20),
    @aetASubmittedDate DATETIME,
    @aetAMarked BIT
)
AS
    INSERT INTO AETAnswers
    (aetQID, classID, userID, aetQType, aetASubmittedDate, aetAMarked)
    VALUES
    (@aetQID, @classID, @userID, @aetQType, @aetASubmittedDate, @aetAMarked)
GO

SET QUOTED_IDENTIFIER OFF

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

SET ANSI_NULLS ON

GO

/* ------- Object: Stored Procedure dbo.CreateAETQuestion Script Date: 5/08/2004 4:08:29 PM -------*/

CREATE PROCEDURE dbo.CreateAETQuestion (@aetQID BIGINT, @classID CHAR(7), @aetQTitle VARCHAR(100), @aetQDescription VARCHAR(30), @aetQType VARCHAR(20), @aetQQuestionType VARCHAR(20), @aetQViewQuestionDateTime DATETIME, @aetQDueDateTime DATETIME, @aetQHaltSubmissionDateTime DATETIME, @aetQViewMarkedAnswersDateTime DATETIME, @aetQAmountOfQuestions INT)

AS

INSERT INTO AETQuestions (aetQID, classID, aetQTitle, aetQDescription, aetQType, aetQQuestionType, aetQViewQuestionDateTime, aetQDueDateTime, aetQHaltSubmissionDateTime, aetQViewMarkedAnswersDateTime, aetQAmountOfQuestions)

VALUES (@aetQID, @classID, @aetQTitle, @aetQDescription, @aetQType, @aetQQuestionType, @aetQViewQuestionDateTime, @aetQDueDateTime, @aetQHaltSubmissionDateTime, @aetQViewMarkedAnswersDateTime, @aetQAmountOfQuestions)

GO

SET QUOTED_IDENTIFIER OFF

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

SET ANSI_NULLS ON

GO

/* ------- Object: Stored Procedure dbo.CreateClass Script Date: 5/08/2004 4:08:29 PM -------*/

CREATE PROCEDURE dbo.CreateClass (}
H.3 Database Source Code Project

```
CREATE PROCEDURE dbo.CreateRole
(
    @roleID VARCHAR(50)
) AS
    INSERT INTO Roles
    (roleID)
    VALUES
    (@roleID)
GO

CREATE PROCEDURE dbo.CreateThreadSub
(
    @classID CHAR(7),
    @userID INT,
    @threadID INT
) AS
    INSERT INTO ThreadSubscriptions
    (classID, userID, threadID)
GO
```
VALUES
    (@classID, @userID, @threadID)
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/

Object: Stored Procedure dbo.CreateUser

CREATE PROCEDURE dbo.CreateUser
(
    @userID INT OUTPUT,
    @password CHAR(47),
    @firstName varchar(50),
    @middleName varchar(50),
    @lastName varchar(50),
    @dob DATETIME,
    @postalStreet varchar(50),
    @postalSuburb varchar(50),
    @postalState varchar(50),
    @postalPostcode varchar(10),
    @postalCountry varchar(50),
    @residentStreet varchar(50),
    @residentSuburb varchar(50),
    @residentState varchar(50),
    @residentPostcode varchar(10),
    @residentCountry varchar(50),
    @phoneNumber varchar(15),
    @mobileNumber varchar(15),
    @emailAddress varchar(50)
) AS

    INSERT INTO Users
    (password, firstName, middleName, lastName, dob, postalStreet, postalSuburb, postalState, postalPostcode, postalCountry, residentStreet, residentSuburb, residentState, residentPostcode, residentCountry, phoneNumber, mobileNumber, emailAddress)
VALUES
    (@password, @firstName, @middleName, @lastName, @dob, @postalStreet, @postalSuburb, @postalState, @postalPostcode, @postalCountry, @residentStreet, @residentSuburb, @residentState, @residentPostcode, @residentCountry, @phoneNumber, @mobileNumber, @emailAddress)

SET @userID=@@IDENTITY
GO
---

**Object: Stored Procedure dbo.DeleteAETAnswer  Script Date:** 5/08/2004 4:08:29 PM

```sql
CREATE PROCEDURE dbo.DeleteAETAnswer
    (@aetQID BIGINT, 
     @classID CHAR(7), 
     @userID BIGINT, 
     @aetQType VARCHAR(20))
AS
    DELETE FROM AETAnswers
    WHERE aetQID = @aetQID AND classID = @classID AND userID = @userID AND aetQType = @aetQType
GO
```

**Object: Stored Procedure dbo.DeleteAETQuestion  Script Date:** 5/08/2004 4:08:29 PM

```sql
CREATE PROCEDURE dbo.DeleteAETQuestion
    (@aetQID BIGINT, 
     @classID CHAR(7))
AS
    DELETE FROM AETQuestions
    WHERE aetQID = @aetQID AND classID = @classID
GO
```
SET ANSI_NULLS ON
GO

/* ******** Object: Stored Procedure dbo.DeleteClass  Script Date: 5/08/2004 4:08:29 PM ********/
CREATE PROCEDURE dbo. DeleteClass
    (@classID CHAR(10))
AS
    DELETE FROM Classes
    WHERE classID=@classID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/* ******** Object: Stored Procedure dbo.DeleteRole  Script Date: 5/08/2004 4:08:29 PM ********/
CREATE PROCEDURE dbo. DeleteRole
    (@roleID VARCHAR(50))
AS
    DELETE FROM Roles
    WHERE roleID=@roleID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/* ******** Object: Stored Procedure dbo.DeleteThreadSub  Script Date: 5/08/2004 4:08:29 PM ********/
CREATE PROCEDURE dbo. DeleteThreadSub
    (@classID CHAR(7),
     @userID INT,
     @threadID INT)
AS
    DELETE FROM ThreadSubscriptions
WHERE classID=@classID AND userID=@userID AND threadID=@threadID
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
CREATE PROCEDURE dbo.DeleteUser
AS
    @userId INT
)
DELETE FROM Users
WHERE userId=@userId
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/***** Object: Stored Procedure dbo.FindClass Script Date: 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.FindClass
AS
    @classId CHAR(10)
)
SELECT
1 AS Tag,
    NULL AS Parent,
    NULL AS [Classes!1],
    NULL AS [Class!2!classID],
    NULL AS [Class!2!Name!element],
    NULL AS [Class!2!denyOtherUsers]
UNION ALL
SELECT
2 AS Tag,
    1 AS Parent,
    NULL,
    Classes.classID,
    Classes.className,
Classes . denyOtherUsers
FROM Classes
WHERE classID=@classID
FOR XML EXPLICIT
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/
****** Object:  Stored Procedure dbo.FindClasses  Script Date: 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.FindClasses
(
    @userID INT
)
AS
SELECT
    1 AS Tag,
    NULL AS Parent,
    NULL AS [Classes!1],
    NULL AS [Class!2!classID],
    NULL AS [Class!2!Name!element],
    NULL AS [Class!2!denyOtherUsers]
UNION ALL
SELECT
    2 AS Tag,
    1 AS Parent,
    NULL,
    Classes . classID ,
    Classes . className ,
    Classes . denyOtherUsers
FROM ClassUsers CROSS JOIN Classes
    WHERE ClassUsers . userID=@userID AND ClassUsers . classID=Classes . classID
FOR XML EXPLICIT
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
CREATE PROCEDURE dbo.FindRolesIDs
(
    @userID INT
)
AS
    SELECT roleID
    FROM ClassUsers
    WHERE userID=@userID
UNION
    SELECT roleID
    FROM OtherUsers
    WHERE userID=@userID
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

CREATE PROCEDURE dbo.FindShortAETAnswer
(
    @aetQID INT,
    @userID BIGINT,
    @classID CHAR(7),
    @aetQType VARCHAR(20)
)
AS
    SELECT 1 AS Tag,
        NULL AS Parent,
        NULL AS [ShortAETAnswers!1],
        NULL AS [ShortAETAnswer!2!aetQID],
        NULL AS [ShortAETAnswer!2!classID],
        NULL AS [ShortAETAnswer!2!userID],
        NULL AS [ShortAETAnswer!2!submittedDate],
        NULL AS [ShortAETAnswer!2!marked]
UNION ALL
    SELECT 2 AS Tag,
        1 AS Parent,
        NULL,
        AETAnswers.aetQID,
        AETAnswers.classID,
        AETAnswers.userID,
        AETAnswers.aetASubmittedDate,
        AETAnswers.aetAMarked
**Object: Stored Procedure dbo.FindShortAETAnswers Script**

Date: 5/08/2004 4:08:29 PM

```
CREATE PROCEDURE dbo.FindShortAETAnswers

    @aetQID INT,
    @classID CHAR(7),
    @aetQType VARCHAR(20)

AS

SELECT
    1 AS Tag,
    NULL AS Parent,
    NULL AS [ShortAETAnswers!1],
    NULL AS [ShortAETAnswer!2!aetQID],
    NULL AS [ShortAETAnswer!2!classID],
    NULL AS [ShortAETAnswer!2!userID],
    NULL AS [ShortAETAnswer!2!submittedDate],
    NULL AS [ShortAETAnswer!2!marked]
UNION ALL

SELECT
    2 AS Tag,
    1 AS Parent,
    NULL,
    AETAnswers.aetQID,
    AETAnswers.classID,
    AETAnswers.userID,
    AETAnswers.aetASubmittedDate,
    AETAnswers.aetAMarked
FROM AETAnswers

    WHERE AETAnswers.aetQID=@aetQID AND AETAnswers.classID=@classID
    AND AETAnswers.userID=@userID AND aetQType=@aetQType
FOR XML EXPLICIT
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

FROM AETAnswers
WHERE AETAnswers.aetQID=@aetQID AND AETAnswers.classID=@classID
AND AETAnswers.userID=@userID AND aetQType=@aetQType
FOR XML EXPLICIT
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
CREATE PROCEDURE dbo.FindShortAETQuestion
(
    @aetQID INT,
    @classID CHAR(7),
    @aetQType VARCHAR(20)
)
AS
SELECT
    1 AS Tag,
    NULL AS Parent,
    NULL AS [ShortAETQuestions!1],
    NULL AS [ShortAETQuestion!2!aetQID],
    NULL AS [ShortAETQuestion!2!classID],
    NULL AS [ShortAETQuestion!2!Title!element],
    NULL AS [ShortAETQuestion!2!Description!element],
    NULL AS [ShortAETQuestion!2!aetType],
    NULL AS [ShortAETQuestion!2!questionType],
    NULL AS [ShortAETQuestion!2!viewQuestionDateTime],
    NULL AS [ShortAETQuestion!2!dueDateTime],
    NULL AS [ShortAETQuestion!2!haltSubmissionDateTime],
    NULL AS [ShortAETQuestion!2!viewMarkedAnswersDateTime],
    NULL AS [ShortAETQuestion!2!amountOfQuestions]
UNION ALL
SELECT
    2 AS Tag,
    1 AS Parent,
    NULL,
    AETQuestions.aetQID,
    AETQuestions.classID,
    AETQuestions.aetQTitle,
    AETQuestions.aetQDescription,
    AETQuestions.aetQType,
    AETQuestions.aetQQuestionType,
    AETQuestions.aetQViewQuestionDateTime,
    AETQuestions.aetQDueDateTime,
    AETQuestions.aetQHaltSubmissionDateTime,
    AETQuestions.aetQViewMarkedAnswersDateTime,
    AETQuestions.aetQAmountOfQuestions
FROM AETQuestions
WHERE aetQID=@aetQID AND classID=@classID AND aetQType=@aetQType
FOR XML EXPLICIT
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/***** Object: Stored Procedure dbo.FindShortAETQuestions
Script Date: 5/08/2004 4:08:29 PM ******/
CREATE PROCEDURE dbo.FindShortAETQuestions
(
    @classID CHAR(7),
    @aetQType VARCHAR(20)
)
AS
SELECT 1 AS Tag,
    NULL AS Parent,
    NULL AS [ShortAETQuestions!1],
    NULL AS [AETQuestion!2!aetQID],
    NULL AS [AETQuestion!2!classID],
    NULL AS [AETQuestion!2!Title!element],
    NULL AS [AETQuestion!2!Description!element],
    NULL AS [AETQuestion!2!aetType],
    NULL AS [AETQuestion!2!questionType],
    NULL AS [AETQuestion!2!viewQuestionDateTime],
    NULL AS [AETQuestion!2!dueDateTime],
    NULL AS [AETQuestion!2!haltSubmissionDateTime],
    NULL AS [AETQuestion!2!viewMarkedAnswersDateTime],
    NULL AS [AETQuestion!2!amountOfQuestions]
UNION ALL
SELECT 2 AS Tag,
    1 AS Parent,
    NULL,
    AETQuestions.aetQID,
    AETQuestions.classID,
    AETQuestions.aetQTitle,
    AETQuestions.aetQDescription,
    AETQuestions.aetQType,
    AETQuestions.aetQQuestionType,
    AETQuestions.aetQViewQuestionDateTime,
    AETQuestions.aetQDueDateTime,
    AETQuestions.aetQHaltSubmissionDateTime,
    AETQuestions.aetQViewMarkedAnswersDateTime,
    AETQuestions.aetQAmountOfQuestions
FROM AETQuestions
WHERE classID=@classID AND aetQType=@aetQType
FOR XML EXPLICIT
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/*∗∗∗∗∗ Object: Stored Procedure dbo.FindUser Script Date: 5/08/2004 4:08:29 PM ∗∗∗∗∗*/
CREATE PROCEDURE dbo.FindUser
(
    @userID INT
)
AS
SELECT
  1 AS Tag,
  NULL AS Parent,
  NULL AS [Users!1],
  NULL AS [User!2!userID],
  NULL AS [User!2!Password!element],
  NULL AS [User!2!FirstName!element],
  NULL AS [User!2!MiddleName!element],
  NULL AS [User!2!LastName!element],
  NULL AS [User!2!dob],
  NULL AS [PostalAddress!3],
  NULL AS [PostalAddress!3!Street!element],
  NULL AS [PostalAddress!3!Suburb!element],
  NULL AS [PostalAddress!3!State!element],
  NULL AS [PostalAddress!3!Postcode!element],
  NULL AS [PostalAddress!3!Country!element],
  NULL AS [ResidentAddress!4],
  NULL AS [ResidentAddress!4!Street!element],
  NULL AS [ResidentAddress!4!Suburb!element],
  NULL AS [ResidentAddress!4!State!element],
  NULL AS [ResidentAddress!4!Postcode!element],
  NULL AS [ResidentAddress!4!Country!element],
  NULL AS [User!2!PhoneNumber!element],
  NULL AS [User!2!MobileNumber!element],
  NULL AS [User!2!EmailAddress!element]
UNION ALL
SELECT
  2 AS Tag,
  1 AS Parent,
  NULL,
  Users.userID,
  Users.password,
  Users.firstName,
  Users.middleName,
  Users.lastName,
  Users.dob,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
CREATE PROCEDURE dbo.FindUsers
(
    @classID CHAR(7)
)
AS

SELECT

  1 AS Tag,
  NULL AS Parent,
  NULL AS [Users!1],
  NULL AS [User!2!userID],
  NULL AS [User!2!Password!element],
  NULL AS [User!2!FirstName!element],
  NULL AS [User!2!MiddleName!element],
  NULL AS [User!2!LastName!element],
  NULL AS [User!2!dob],
  NULL AS [PostalAddress!3],
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL AS [ResidentAddress!3],
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL AS [User!2!PhoneNumber!element],
  NULL AS [User!2!MobileNumber!element],
  NULL AS [User!2!EmailAddress!element]

UNION ALL

SELECT

  2 AS Tag,
  1 AS Parent,
  NULL,
  Users . userID,
  Users . password,
  Users . firstName,
  Users . middleName,
  Users . lastName,
  Users . dob,
  NULL,
  NULL AS [PostalAddress!3!Street!element],
  NULL AS [PostalAddress!3!Suburb!element],
  NULL AS [PostalAddress!3!State!element],
  NULL AS [PostalAddress!3!Postcode!element],
  NULL AS [PostalAddress!3!Country!element],
  NULL,
  NULL AS [ResidentAddress!3!Street!element],
  NULL AS [ResidentAddress!3!Suburb!element],
  NULL AS [ResidentAddress!3!State!element],
  NULL AS [ResidentAddress!3!Postcode!element],
  NULL AS [ResidentAddress!3!Country!element],
  Users . phoneNumber,
  Users . mobileNumber,
  Users . emailAddress

FROM Classes CROSS JOIN Users

WHERE Classes . classID=@classID
UNION ALL

SELECT

3 AS Tag,
2 AS Parent,
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,
NULL,

Users.postalStreet,
Users.postalSuburb,
Users.postalState,
Users.postalPostcode,
Users.postalCountry,
NULL,
Users.residentStreet,
Users.residentSuburb,
Users.residentState,
Users.residentPostcode,
Users.residentCountry,
NULL,
NULL,
NULL,

FROM Classes CROSS JOIN Users
WHERE Classes.classID=@classID
FOR XML EXPLICIT
GO

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS OFF
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_NULLS ON
GO

/***** Object: Stored Procedure dbo.IsInClass  Script Date: 5/08/2004 4:08:29 PM ******/

CREATE PROCEDURE dbo.IsInClass

( @userID INT,
  @classID CHAR(7)
 )

AS

SELECT roleID
FROM ClassUsers
WHERE userID=@userID AND classID=@classID
ORDER BY roleID
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO