2012

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

IMPLEMENTING MINI QUIZZES TO INCREASE STUDENT LEARNING FOR A CLASS VIA A MOBILE LEARNING APPLICATION

Andrew Montanaro

A Capstone Project Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems and Operations Management

University of North Carolina Wilmington

2012

Approved By

Advisory Committee

_____	_____
Dr. Laurie Patterson	Dr. Ron Vetter

_____
Dr. Thomas Janicki, Chair

Abstract

Implementing Mobile Devices to Increase Student Learning for a Class via a Mobile Learning Application. Montanaro, Andrew, 2012. Capstone Paper, University of North Carolina Wilmington.

The ever increasing number of people who own mobile devices capable of browsing the internet creates an opportunity to provide today's students with an additional learning resource in the form of a short quiz. Creating web pages or applications for the ever growing number of mobile devices available to the public is a challenge but in a school setting it is important to allow for the largest percent of students to have availability. This project researches Learning Theories to better understand how individuals learn to design quizzes to have a positive impact on a student. Development Methodologies and Mobile Development were researched to plan appropriately before creating the Mobile Learning Application. The paper concludes with the design and development of a user interface for instructors to create quizzes and an interface that students can access via a mobile device or desktop computer and have the quiz display in a reasonable size to read and take the quiz.

Table of Contents

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: UNDERSTANDING LEARNING THEORIES 3

2.1 BEHAVIORISM 3
2.2 COGNITIVISM 4
2.3 RESOURCE-BASED LEARNING 6

CHAPTER 3: DEVELOPMENT METHODOLOGIES 8

3.1 WATERFALL APPROACH 8
3.2 SPIRAL APPROACH 11
3.3 AGILE APPROACH: RAPID APPLICATION DEVELOPMENT (RAD) 12

CHAPTER 4: SELECTION OF METHODOLOGY 14

CHAPTER 5: SYSTEM DESIGN IMPLEMENTATION 14

5.1 EXAMPLE RAD IMPLEMENTATION 14
5.2 USE CASE ACTOR DIAGRAM 17
5.3 USE CASE EXAMPLE 17
5.4 TECHNOLOGIES SELECTED 19
5.5 ASP.NET TEXTBOX CONTROL EXTENDED 20

CHAPTER 6: MOBILE PLATFORM ALTERNATIVES 22

6.1 ANDROID 23
6.2 IPHONE 24
6.3 WINDOWS MOBILE 26
6.4 BLACKBERRY 27
6.5 WEB 28

CHAPTER 7: MOBILE DEVELOPMENT GUIDE 30

7.1 CATEGORIES OF DEVICES 30
7.2 DETECTION 31
7.3 DEVICE FUNCTIONALITY 32
7.4 MOBILE WEB STYLES DEVICE CONSIDERATIONS 33

CHAPTER 8: EXAMPLE OF A POTENTIAL 'MINI QUIZ' 35

8.1 MINI QUIZ ON A SMART PHONE SYSTEM OVERVIEW 35
8.2 EXAMPLES OF MINI QUIZ 35

CHAPTER 9: HELP GUIDE 41

9.1 ADDING A COURSE 42
9.2 ADDING A TOPIC 43
9.3 ADDING A QUESTION 44
9.4 ADDING QUESTIONS TO A QUIZ 47
9.5 LAUNCH QUIZ 48

CHAPTER 10: COMPLETED DELIVERABLES 49

10.1 PROJECT TIMELINE 50
10.2 ACTUAL TIMELINE 51
10.3 LESSONS LEARNED AS THE PROJECT TIMELINE WAS EXTENDED 51
10.4 POSSIBLE FUTURE ENHANCEMENTS 53

REFERENCES 55

APPENDIXES **57**

APPENDIX A - USE CASE DIAGRAM 57
APPENDIX B - USE CASES 58
Courses *58*
Topics *61*
Question *62*
Quiz *64*
Admin Tools *65*
APPENDIX C - DATABASE DESIGN 69
APPENDIX D - TEXTBOXT EXTEDED CLASS REFERENCE 70
APPENDIX E - STUDENTQUIZ CLASS REFERENCE 76
APPENDIX E - STUDENTQUESTION CLASS REFERENCE 79

Table of Figures

FIGURE 1 - SAMPLE WATERFALL STAGES (TSUI & KARAM, 2011) 9
FIGURE 2 - SAMPLE SPIRAL (TSUI & KARAM, 2011) 10
FIGURE 3 - TOPICS PAGE AFTER 2ND ITERATION 15
FIGURE 4 - QUESTION PAGE AFTER 2ND ITERATION 15
FIGURE 5 - TOPICS PAGE AFTER 3RD ITERATION 16
FIGURE 6 - QUESTIONS PAGE AFTER 3RD ITERATION 16
FIGURE 7 - EXAMPLE SCROLL BALL 33
FIGURE 8 - EXAMPLE TOUCH SCREEN ..... **ERROR! BOOKMARK NOT DEFINED.**
FIGURE 9 - EXAMPLE DUAL - ORIENTATION ..... **ERROR! BOOKMARK NOT DEFINED.**
FIGURE 10 - DISPLAY A QUESTION 36
FIGURE 11 - EXAMPLE ACTUAL SCREEN DISPLAYING A QUESTION 36
FIGURE 12 - SCREEN SHOWING A CORRECT ANSWER 37
FIGURE 13 - EXAMPLE ACTUAL IMAGE FOR CORRECT ANSWER 37
FIGURE 14 - EXAMPLE ACTUAL SCREEN SHOWING A CORRECT ANSWER 38
FIGURE 15 - SCREEN SHOWING QUESTION AFTER CORRECT ANSWER 38
FIGURE 16 - EXAMPLE ACTUAL SCREEN SHOWING QUESTION AFTER CORRECT ANSWER 39
FIGURE 17 - SCREEN SHOWING AN INCORRECT ANSWER 39
FIGURE 18 - EXAMPLE ACTUAL SCREEN SHOWING AN INCORRECT ANSWER 40
FIGURE 19 - SCREEN SHOWING TRUE/FALSE QUESTION 40
FIGURE 20 - EXAMPLE ACTUAL SCREEN SHOWING TRUE/FALSE QUESTION 41
FIGURE 21 - EXAMPLE OF HELP BUTTON ON SCREEN 42
FIGURE 22 - EXAMPLE COURSE PAGE 43
FIGURE 23 - EXAMPLE TOPICS PAGE SHOWING COURSE PROMPT 43
FIGURE 24 - EXAMPLE TOPICS PAGE WITH COURSE SELECTED 44
FIGURE 25 - EXAMPLE OF QUESTIONS PAGE WITH A COURSE SELECTED 44
FIGURE 26 - EXAMPLE QUESTION PAGE WITH TOPIC SELECTED 45
FIGURE 27 - EXAMPLE OF ADD NEW QUESTION 46
FIGURE 28 - EXAMPLE QUIZ PAGE WITH QUIZ SELECTED 47
FIGURE 29 - EXAMPLE OF SELECTING QUESTION FOR QUIZ 48
FIGURE 30 - EXAMPLE LAUNCH QUIZ PAGE 48

Table of Tables

TABLE 1 - EXAMPLE USE CASE 18
TABLE 2 - MARKET PERCENTAGE BY OPERATING SYSTEM 23
TABLE 3 - PROJECT TIMELINE 50

Chapter 1: Introduction

The overall goal of this capstone project is to assist students in their learning and provide them a fun and easy way to increase their retention of class materials via smart phones and tablet devices. The project involved researching learning theories, and investigating the complexities of system software development for different operating platforms found in smart devices. It concludes with building a prototype for a smart phone. It is hoped to provide students an additional and fun way to prepare for future exams and tests on their smart devices via the development of flexible mini quizzes.

The application was designed to enable students to be able to take short quizzes quickly using handheld devices such as tablets and smart phones. To encourage more participation students are able to take these quizzes without having to login or to enter any information (other than the course and section) allowing the student to take the quiz easily and quickly. The application also has a web component that gives instructors and administration an interface to manage the quizzes. This tool is designed so that the instructor can quickly and easily create and manage quizzes for different classes.

The reason for developing a system to enable short quizzes and then allowing the student to start taking the quiz in a short period of time is so that the student can take the quiz when they have a few minutes of downtime (i.e. waiting for a bus, 5 minutes before class, or while walking around on campus). A key consideration is that the instructor must also be able to create, edit, and delete quizzes easily in order to gain the instructor's acceptance of an additional tool on top of their already busy class work.

Learning theory concepts are applied so that the student experiences a different style of learning from that of what they get in class (i.e. a typical YES you are correct, or

NO you are incorrect). The use of positive reinforcement is used for correct answers with congratulatory images that appear on the screen as well as instant feedback for incorrect answers. This instant feedback is the most important part of the quiz because the students can learn which answer was the correct answer, why it is the correct answer, and potential reference links to the chapter and the section where the answer can be found.

The key stakeholders for this project are the students and the instructors. Students will benefit from this application because it is another tool that can be used to gain a better understanding of class material or to simply revisit information before a test. A student feature is the ability to take a quiz in minutes anywhere and anytime from a handheld device, freedom from having to create and carry note cards, and the benefit from simply seeing how the questions might be worded or presented on upcoming tests.

The instructors benefit because the application will assist traditional, web enhanced and all web courses as another vehicle to present the class material. The quizzes themselves could also act as a great way to deliver a study guide for an upcoming tests and quizzes in class. The system will be able to track student correct and incorrect answers (anonymously) so that the instructor can also use these results as a barometer of the class's understanding about the current material by going over questions that students found difficult and that can lead to healthy discussions that gives the students a broader understanding of the material.

Chapter 2: Understanding Learning Theories

This section briefly reviews three of the different theories of how people learn. Included in this section is an overview of the more historically considered learning theories of Behaviorism and Cognitivism. Another more recent learning theory. Resource Based Theory, was also selected because of its higher application to computerized or online learning.

2.1 Behaviorism

Behaviorism assumes a learner is essentially passive, responding to environmental stimuli. The learner's behavior is shaped through positive reinforcement and negative reinforcement which is why that Behaviorism is sometimes referred to as the stimulus and response psychology.

> "J.B. Watson, the father of Behaviorism, defined learning as a sequence of stimulus and response actions in observable cause and effect relationships." (Forrester & Jantize, 1998)

Positive reinforcement increases the probability that the action will happen again as negative will decrease the probability that the action will happen again. An example of how positive reinforcement that can be used in the mobile learning application could be an amusing image or short clip. Behaviorism's roots began as testing with animals; an example being Pavlov's dogs (Amsel, 1989).

According to Hannafin and Peck (1988) there are four principles to guide the design of instruction that are taken from behavioral learning theory:

1. Contiguity: The response should follow the stimulus without delay
2. Repetition: Practice strengthens learning and improves retention.
3. Feedback and reinforcement: Knowledge concerning the correctness of the response contributes to learning
4. Prompting and fading: Learning may be achieved by leading the student to the desired response under decreasingly cued conditions.

A criticism against behaviorism is that it does not account for all types of learning, since it disregards the activities of the mind and that testing on animals would not be as complex as testing on humans (Hannafin & Peck, 1988).

2.2 Cognitivism

Cognitivism focuses on the inner mental activities of the human mind and considers these activities valuable for understand how people learn (Leonard, 2002). The concepts of Cognitivism research are information processing, knowledge types and knowledge representations, and human memory (McGilly, 1994). In many ways the researchers in this field compare the human brain to a computer and suggest that information processing is no different between the two. Information is processed in the following steps:

1. Information in the form of symbols enters the system
2. Activates particular processes
3. Result in mental actions (McGilly, 1994).

Knowledge types and knowledge representations are split into two types: declarative and procedural. Declarative is the knowledge of the world and its properties and procedural is the knowledge of how to do things (McGilly, 1994).

How knowledge is represented differs from novice and experts in a field. Novices tend to group ideas in a field by surface or lower level features while experts group ideas by deeper level features which leads to the differences in the abilities to reason and solve problems in that field (McGilly, 1994). It is thought that there are two different types of memory: working memory and long-term memory. Working memory is short term, enters through our senses and leaves very fast unless continual efforts to keep it active. Long-term is considered always there and can be accessed anytime (McGilly, 1994). When information goes from working memory to long-term memory is when it is actually learned. Again, according the Hannafin and Peck (1988) there are three additional principles to guide the design of instruction that are taken from cognitive learning theory.

1. Orientation and recall: Learning involves the synthesis of prior information that must be recalled to active memory.
2. Intellectual skills: Learning is facilitated by the use of existing processes or strategies.
3. Individualization: Learning may be more efficient when the instruction is adapted to the needs and the profiles of individual learners.

Theses principals are used in the design and development of the application used by instructors and students. The Orientation and Recall and Intellectual skills principles are used when the student is taking a quiz. The quiz should be on information that has been

already introduced in a class or reading and the process of taking a quiz is something all students should have done since grade school. The Individualization principle is used when the instructor reviews the results of the quizzes. An example of how the quiz program will comply with the Individualization principle could be when ten percent of the class does not understand a subject like Network Subnetting and does not make lack of understanding known to the instructor, a quiz on Network Subnetting will allow for the instructor to have this information that there is a percentage of students not comfortable with the current chapter.

2.3 Resource-based learning

"Resource-based learning is defined as an integrated set of strategies to promote student-centered learning in a mass education context, through a combination of specially designed learning resources and interactive media and technologies" (Ryan, Scott, Freeman, & Patel, 2000). Resource based learning looks to modify the central role of the actual instructor to one that is more to help students as another resource or facilitator rather than to teach them (Ryan et al., 2000). The instructor's role as a resource may be seen in on-line classes where actual meeting between instructor and student either don't occur or are infrequent. The student is presented with materials which can be of different media and formats. Examples of these may be study guides, tutorials, videos and audio clips, and assessments. In most cases the instructor substitutes the time traditionally spent in the classroom with more time to provide individualized feedback and assessments of individual students on their strengths and weaknesses (Ryan et al., 2000).

Advantages of using resource-based learning are increased interaction, administrative financial strengths and cost effectiveness, motivation, immediate feedback, ease of record keeping, and lesson integrity (Hannafin & Peck, 1988).

There may be increased personal interaction in comparison to a normal class where a student can sit passively because the student must respond to each and every problem the computer presents to them versus just answering one question in the classroom periodically. If students are busy thinking of future plans, daydreaming, or answering text messages, the learning system will wait for the their responses or actions. In some studies, students with ADHD have had positive results when using computer assisted instruction (Hannafin & Peck, 1988). Students also benefit from the personal interaction of different types of media and resources which cannot be as easily duplicated in a classroom.

At the K-12 level, administrative strengths and cost effectiveness of using resource-based learning systems are advantages because the school can have many students studying different courses at the same time with no teacher supervision and low reproduction costs.

Students motivation in using a computer assisted instruction vary from a lower threat of a computer versus a human instructor to the ability to control one's learning process (Hannafin & Peck, 1988). The ability for a program to give instant feedback is a significant advantage because in most cases it is almost instant and is personalized to that student. The last two advantages ease of record keeping, and lesson integrity all fall in line with the ability to keep information when using program. A student’s selection can

be recorded for statistics because of the lesson integrity that the students will receive the same questions and material.

Chapter 3: Development Methodologies

A major software development project should encompass the adoption of a systems analysis and design methodology. A goal of the capstone was to provide the developer more experience with a specific methodology to enable learning through the use of a specific methodology.

This section reviews three popular development methodologies: the Waterfall approach, the Spiral approach, and the Agile approach (specifically the Rapid Application Development method).

3.1 Waterfall approach

The waterfall approach was proposed by Winston Royce in 1970 and was the answer to the commonly used method of "code and fix" (Awad, 2005). The waterfall approach is called a linear or a sequential method because it is comprised of different phases with each one leading into to the next. The different phases are named different depending on your source but all are relatively similar to the naming convention shown in **Figure 1**. The stages are: Requirements Analysis, Design, Code, Unit Testing, and System Testing & Integration.

The basic principles of the waterfall development methodology are an emphasis on planning, time schedules, target dates, budgets, and implementation of an entire system at one time and that tight control is maintained throughout the entire project with sign offs and deliverables at the end of each stage (CMS, 2008). The waterfall approach

has three different types that are differentiated by the transition from phase to phase. The first is No Overlap which it typically what one thinks of when talking about the waterfall approach meaning that one phase must be completed before the next one can start and once one starts no going back to the last phase. The second is One Phase Overlap with allows a phase to be overlapped by only one other phase and it must be an adjacent phase. The third is Overlapping Phases which allows each phase to overlap the other and is difficult to coordinate the deliverables (Charvat, 2003).

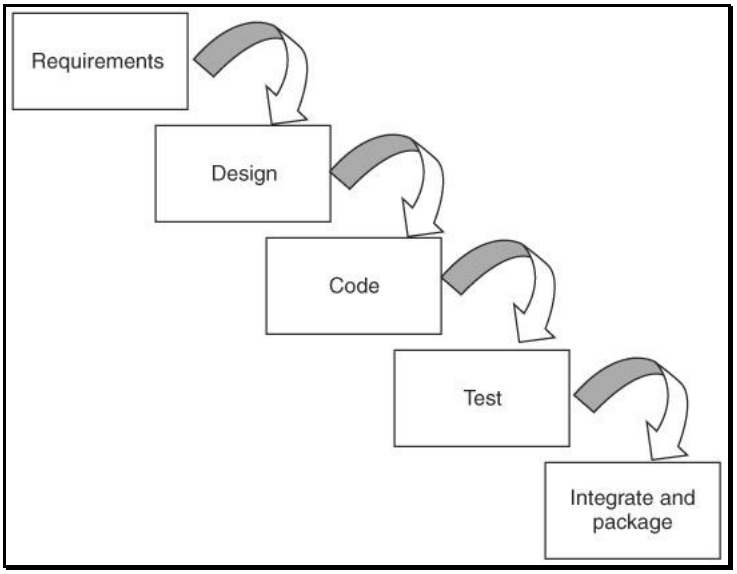


Figure 1 - Sample Waterfall Stages (Tsui & Karam, 2011)

Advantages

- Progress of development is easily measurable
- Checkpoints and deliverables at the end of each phase
- For projects that have clear objectives and solutions
- Can be used as part of iterative approaches

Disadvantages

- Slow
- Depends on clear specifications early which is often hard to get
- Hard to respond to changes
- Results are not seen until late in the cycle
- Excessive documentation is time consuming
- Distinct gap between the users and developers
- Not Ideal for Web projects which usually have time pressures

(Charvat, 2003)

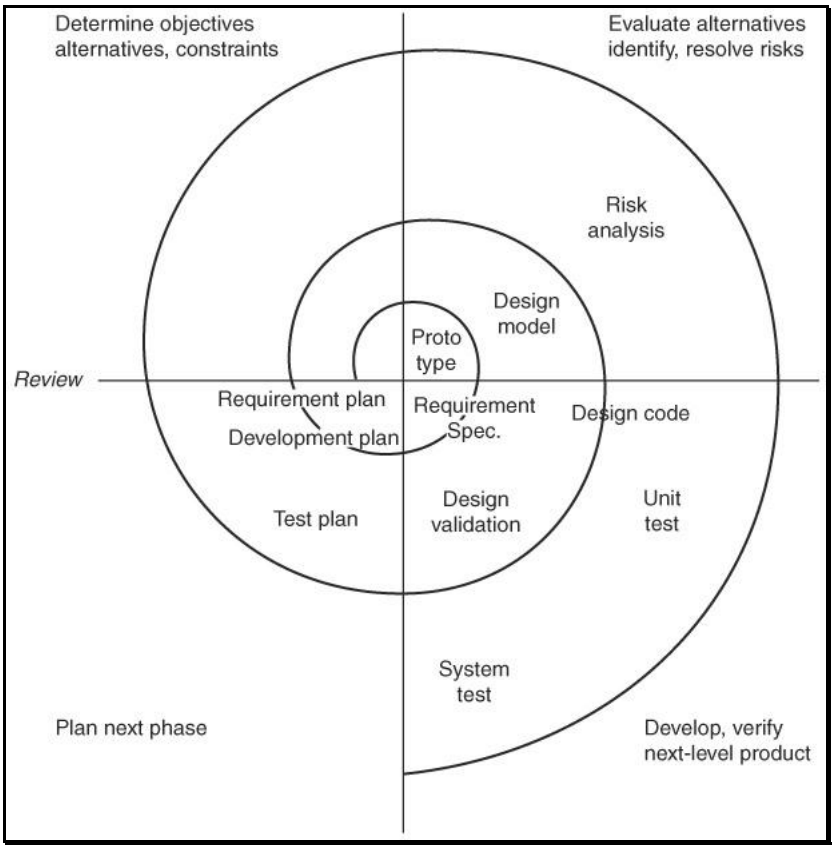


Figure 2 - Sample Spiral (Tsui & Karam, 2011)

3.2 Spiral approach

The spiral approach was developed to be a refinement of the waterfall model (Awad, 2005) and address the lack of flexibility in the waterfall method. The name spiral comes from the model which consists of quadrants (Tsui & Karam, 2011):

1. Identify the objectives, alternatives, or constraints for each cycle of the spiral.
2. Evaluate the alternatives relative to the objectives and constraints.
3. Depending on the amount of and the type of identified risks, develop a prototype, more detailed evaluation, an evolutionary development. If low risk requirements, design, or code.
4. Validate the achievement of the objective and plan for the next cycle.

The spiral approach is often thought of iterating through waterfall approaches for each project phase. One major difference between spiral and waterfall is that spiral has a focus on risk. The focus on risk and minimizing that risk is done by breaking a project into smaller segments which allows for a change to be done easier (CMS, 2008). The reason for change from waterfall to spiral is that often detail of what the final system should be and do will not be known on the first attempt. The spiral methodology takes the focus on risk into account and starts with the highest priority or risk features. An iteration is completed when the user feedback is obtained and then another iteration begins. The user feedback part is another important feature that is different from the waterfall approach because the spiral allows for the users to be involved in each iteration and apart of the project as a whole giving the resemblance of an in house development.

Advantages

- Risk avoidance and early detection
- Each iteration could have a different approach like waterfall for one and extreme programming for another
- Large project broken into smaller segments is easier to do cost calculations.
- Flexible to changes at different stages
- Client involvement, knowledge, and ownership
- Battles against poor testing since testing starts at first iteration

Disadvantages

- Customized to each project which limits reusability
- No firm deadlines
- Usually requires experienced a project manager to implement
- Resource heavy
- Each iteration could cause more work for the next

(Charvat, 2003)

3.3 Agile Approach: Rapid Application Development (RAD)

There are many different agile approaches such as Extreme Programming (XP), Scrum, and Crystal. The one I chose to study more is Rapid Application Development (RAD). Rapid Application Development is similar to the spiral approach in that it goes through iterations. The difference is that at the end of iteration using RAD the deliverable is an actual usable working system with proper documentation which allows for the users or clients to realize immediate benefits. RAD recognizes that changes are more often than

not going to be a part of the development process and the entire approach is based on incremental changes ending with a quality product (CMS, 2008). The incremental changes approach allows for faster development which also results in lower investment costs. The lower investment costs is ensured because of a principal of: if the project start to slip, emphasis is on reducing requirements to fit the time constraint rather than increasing the deadline (CMS, 2008). Systems are quickly produced with the aid of prototypes and tools such as CASE, GUI builders, and code generators.

Advantages

- Identify risks early
- Early operational versions
- Can be rapidly changed
- Saves time, money, and human effort

Disadvantages

- Could lead to low quality
- Scope creep or feature creep
- Large stakeholder involvement (can be seen as both) - high cost
- Potential for a design with a lack of flexibility
- Difficult problems could be hidden until future iterations to demo early success
- Interfaces must be defined early and well

(Charvat, 2003)

Chapter 4: Selection of Methodology

The decision to implement the RAD methodology was made because it would be the best fit for this project (especially since there is one developer). It is a methodology that the developer would like to learn more about how it assists the developer and the end users. The main reason why RAD is the best fit for this project is because this project is highly interactive (CMS, 2008) with two clearly defined user groups: instructors and students. RAD is also a great fit for the way the developer prefers to work on a project in that the project will be broken up into manageable iterations. Some concerns in selecting RAD is that it lends its self to feature or scope creep and the risk of the end product not being flexible to future changes or versioning. The risk associated with these concerns can be reduced by proper identification of project scope and proper planning during each iteration to keep from taking away from future flexibility.

Chapter 5: System Design Implementation

5.1 Example RAD Implementation

This section will briefly describe an example of the three iterations I used in the development of the user interface across multiple pages. Again the RAD permitted me to develop a working prototype at each stage and demonstrate it for an end user. The demonstration of the prototype allowed users to provide feedback for enhancement during the next stage.

The first iteration encompassed the requirements and created a working prototype. When this prototype was first shown to an end user it was shown along with the functionality of the "Topics" page. The stakeholder suggested some changes to the wording of prompts and spacing on the layout. In addition the flow of tabs/labels and

instructions for the user were modified. The second iteration the suggested changes were implemented along with a prototype for the development of new and revised "Questions" for the mini quizzes. **Figure 3** and **Figure 4** are examples of the prototype pages on initial load after the second iteration.

Figure 3 - Topics Page After 2nd Iteration

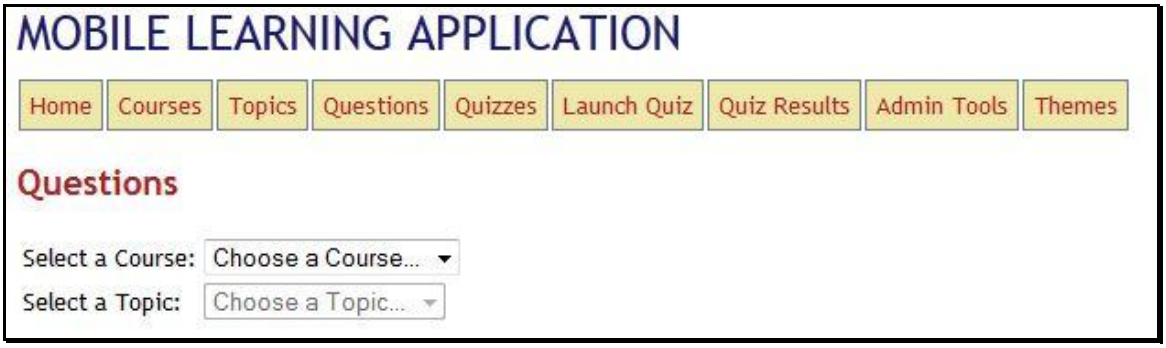


Figure 4 - Question Page After 2nd Iteration

After the end user interacted with the second iteration it was discussed the repetitive nature of having to pick a course on multiple different pages. The prototypes where then redesigned for the third iteration so that when a user selected a Course that selection will be saved and used on other pages, but still allow the user to change this selection if desired. **Figure 5** shows an example of the "Topics" and **Figure 6** page initial load when a user already selected a Course.

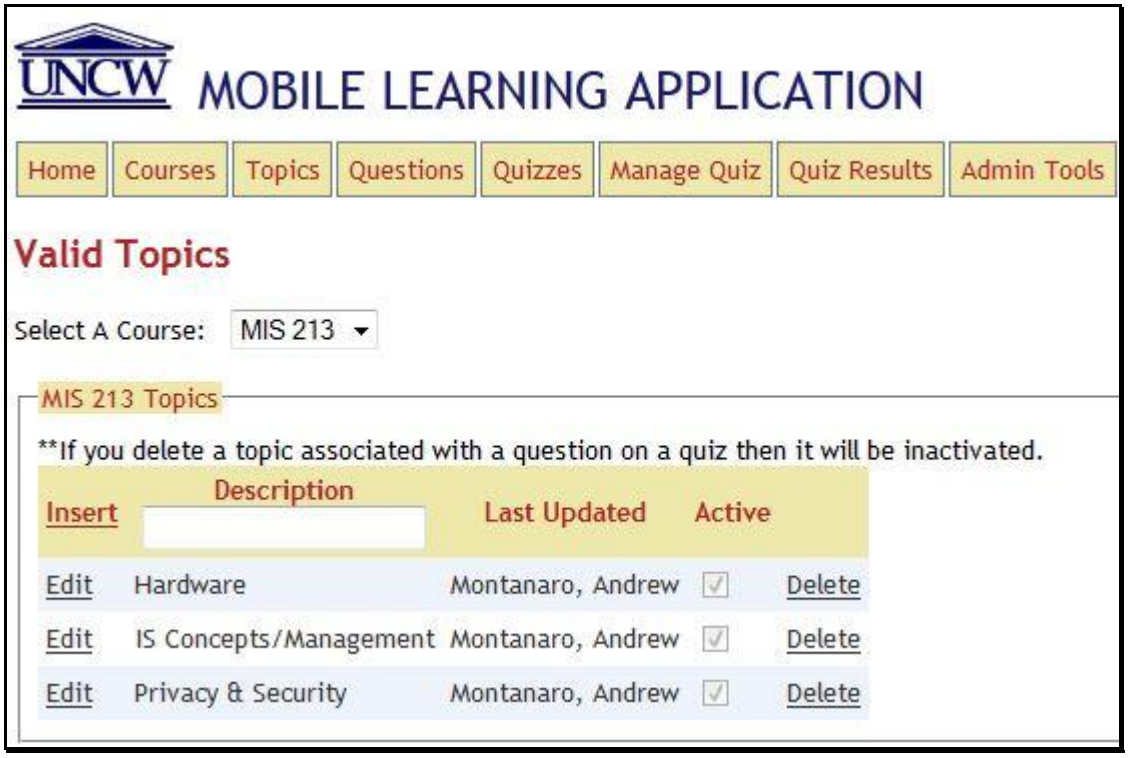


Figure 5 - Topics Page After 3rd Iteration

Figure 6 - Questions Page After 3rd Iteration

This simple example shows how the iterative style that RAD provides influenced the enhancement and usability of this project. These changes took a prototype that was working based on the requirements and helped it evolve into a more user friendly design.

5.2 Use Case Actor Diagram

The Use Case Actor Diagram was created to answer two questions, what are the use cases needed for this project and who are the actors and what permissions will they need? When creating the Use Case Actor Diagram the process involved starting with the actor with the most permissions moving down to the actor with the least permissions was used. To create this diagram a list of use cases must be made for each actor and then it is made to a diagram with an actor symbol and use cases in bubbles. The Use Case Actor Diagram may be found in Appendix A. The creating of the Use Case Actor Diagram does not entirely answer the question what permissions will each actor need but does create a starting off point because it provides visual representation of what each actor needs to do in the system.

5.3 Use Case Example

The following is the use case to "Update a Question" in the instructors question bank. Additional Use Cases can be found in Appendix B.

Table 1 - Example Use Case

<table>
<tr><td>Use Case Name:</td><td>Update a Question</td></tr>
<tr><td>Scenario:</td><td>User wants to Update a Question and its Answers</td></tr>
<tr><td>Triggering Event:</td><td>Admin or Instructor wants to Update a Question or it's Answers</td></tr>
<tr><td>Brief Description</td><td>When a Admin or Instructor wants to Update a Question they go to the "Questions" page and selects the proper course and topic to display the question. The Actor will then be able to edit if they were the creator of the question.</td></tr>
<tr><td>Actors:</td><td>Admin or Instructor</td></tr>
<tr><td>Related Use Case:</td><td>None</td></tr>
<tr><td>Stakeholders:</td><td>Admin, Instructors, and Students</td></tr>
<tr><td>Preconditions:</td><td>Course & Topic must exist, User must have access to the application, and the Actor must have created the question</td></tr>
<tr><td>Post conditions:</td><td>None</td></tr>
<tr><td>Flow of Events</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Actor goes to the Questions page.</td><td>1.1 Displays a list of all Courses</td></tr>
<tr><td>2. Actor selects a Course</td><td>2.1 Displays a list of all Topics for the Course</td></tr>
<tr><td>3. Actor selects a Topic</td><td>3.1 Displays a list of all Questions for the Topic.
3.2 Show Edit & Delete button if Actor Created Question</td></tr>
<tr><td>4. Actor selects "Edit"</td><td>4.1 Display the question and answers if Edit Mode</td></tr>
<tr><td>5. Actor Selects "Update"</td><td>5.1 Check that there is an Answer and 1 correct
5.2 Update the Question
5.3 Update existing Answers if there is text
5.4 Delete existing Answers if empty
5.5 Insert new Answers if there is text
5.6 Close Question in edit mode and display questions</td></tr>
<tr><td>6. Actor Selects "Cancel"</td><td>6.1 Close Question in edit mode and display questions</td></tr>
</table>
</td></tr>
<tr><td>Exception Conditions:</td><td></td></tr>
</table>

5.4 Technologies Selected

For this project, the Microsoft duo of SQL Server 2008 and Visual Studio 2010 was used for development. The choice of using the language of Visual Basic was chosen because the MIS program currently teaches this language to undergraduate students. Giving a larger number of students familiar with this language makes this application easier to maintain and upgrade if this application is used by the University of North Carolina Wilmington (UNCW) staff.

When developing the pages that display the quiz to the students the system incorporates an API developed by 51Degress.mobi. The reason for using 51Degrees.mobi is because it is an easy to use shell on top of WURFL. Developers are provided with some device information for free but for more detailed information they are required to purchase a license. After the framework is installed the mobile device information is access using the Request.Browser object. An example of some useful information about a mobile device available using the free framework are: Screen Pixels Height and Width, detects if a mobile device, the mobile device manufacturer, and the mobile device model. MSDN also supports 51Degrees.mobi on their website which reassured that the API would work properly in Visual Studio. 51Degrees.mobi also offers licenses that come along with a framework of ASP.Net controls to use in mobile devices. The list of controls are similar to what an ASP.Net developer uses on regular website, for an example button, image, listbox, dropdownlist, and datalist. All of these controls are similar to their ASP.Net counter parts but are styled for mobile devices and have properties to aid in the development for mobile web pages. An example of a property that the 51Degrees.mobi offers to aid in development is the ability to have multiple image sources for a single

image. The mobile image has a property called CalculateSizeMode in which this can be set to calculate the size of the mobile screen and then pick from the list of image sources to find the closest match and then shrinks it to size. This is a wonderful addition because in mobile development it is important to keep image sizes small to reduce the amount of loading. The normal Asp.Net image control allows the developer to set the size of the image as displayed but the image still loads on the page as full size, an example is an image 500px x 500px put in an Asp.Net Image control with the size of 100px x 100px will load the full 500px x 500px image and scale it down. The mobile image control in the same example with the proper properties set will scale down the image and only load an image with the size of 100px x 100px. These controls were not free and so are not used in this project.

5.5 ASP.Net Textbox Control Extended

For this project, a class was created to extend the asp.net textbox control. The extended class was done to aid in the development of the instructor side of the project where textboxes will be used. The reason for this extended class was for each textbox on a webpage there was normally an associated validation control or two. With the creation of this extended class it allowed the developer to just add a textbox control and change the textbox's properties to allow for the different required fields and at runtime these other controls would be added. One major limitation to using the extended textbox would be that since these additional controls to the textbox are created at runtime you cannot refer to them in server side code. If using the additional controls in the server side code is a requirement then you will have to add the controls the original way.

To use controls in the project:

- TextboxControl.vb (class attached in Appendix section) in App_Code folder.
- Register Tag on the aspx page

Current uses of the Textbox Control

- All Properties from ASP Textbox Control
- Adds Required field validator
- Adds RegExpression validator for:
    - Date
        - Includes JavaScript for dashes (attached below)
    - Email
    - Phone
- Adds a compare validator for:
    - Integer
- Adds a CalendarExtender

Properties that were added to the textbox control are:

- RequiredFieldValidator
    - IsRequired
    - RequiredMessage (optional – has default)
- DateValidator
    - IsDate
    - DateValidationMessage (optional – has default)
- IntegerValidator
    - IsInteger
    - IntegerValidationMessage (optional – has default)
- EmailValidator
    - IsEmail
    - EmailValidationMessage (optional – has default)
- PhoneValidator
    - IsPhone
    - PhoneValidationMessage
- AJAX Calendar Extendar
    - IsCalendar
    - CalendarPopupButtonID (optional)

Examples of how the extended textbox control are:

Example Register Tag

```
<%@ Register TagPrefix="ASM" Namespace="ASMTextboxControl" %>
```

Example Required field validator

```
<ASM:ASMTextboxControl ID="txtRoleDescInsert" runat="server" Text=""
IsRequired="true" ValidationGroup="ValidRoleInsert" />
```

Example Date With CalendarExtendar (*Note to use ajax there must be a scriptmanager on page)

```
<ASM:ASMTextboxControl ID="txtDate" runat="server" Text="" IsDate="true"
IsCalendar="true" />
```

Example Email

```
<ASM:ASMTextboxControl ID="txtEmailInsert" runat="server" Text=""
IsRequired="true" IsEmail="true" ValidationGroup="AdminUserInsert" />
```

Example Phone

```
<ASM:ASMTextboxControl ID="txtPhoneNumberInsert" runat="server" Text=""
Width="100px" IsPhone="true" ValidationGroup="AdminUserInsert" />
```

Example Integer

```
<ASM:ASMTextboxControl ID="txtRoleIDInsert" runat="server" Text=""
IsRequired="true" IsInteger="true" ValidationGroup="ValidRoleInsert" />
```

Chapter 6: Mobile Platform Alternatives

Part of the development process is to understand the different requirements based on the uniqueness of mobile devices. Although this project does not have within its scope to publish an ‘app’, the author desired to understand the process and requirements as the project was being development. Understanding these requirements helped to form some of the decisions made.

There are currently four popular mobile platforms for development of applications (apps). These are: droid, iPhone, windows mobile and blackberry. **Table 2** shows the market share by mobile operating system as of March 29, 2012. In addition to building ‘apps’ for specific devices, another option is for web to create web pages that can be

displayed on any handheld or desktop computer rather than create applications that run locally on one particular device using HTML 5 and CSS 3 techniques

Table 2 - Market Percentage by Operating System

Name	Phone	Market%
iOS	Apple	60.50
Windows Phone 7	HTC & Samsung	.83
RIM	BlackBerry	2.70
Android	HTC, Samsung, LG	35.00

Source: http://getclicky.com (retrieved 3/29/2012)

This next section will list the procedures to develop and publish new applications on these devices.

6.1 Android

- Development
  - Java, Eclipse (preferred) with the ADT plug-in.
  - Set up Virtual Devices or connect hardware devices (phone)
  - Create, Build, and Test
    1. Test your application extensively on an actual device
    2. Consider adding an End User License Agreement in you application
    3. consider adding licensing support
    4. Specify an icon and label in the application's manifest
    5. Turn of logging and debugging and clean up data/files
    6. Version your application
    7. Obtain a suitable cryptographic key

8. Register for a Maps API Key, if your application is using MapView elements
9. Sign your application
10. Test your compiled applications

- Publish on Android Market- Get Google account. Upload application and publish
    1. Your app must be signed with cryptographic private key whose validity period ends after 22 Oct. 2033
    2. Your app must define both an "android:versionCode" and an "android:versionName" attribute in the <manifest> element.
    3. Your app must define both an "android:icon" and an "android:label" attribute in the <application> element.

- You can publish your application and allow users to install it any way you choose. (http://developer.android.com/guide/publishing/publishing.html)

6.2 iPhone

- Development
    - iPhone Software Development Kit (free) - Xcode from Apple (Objective C)
    - Create a Development Certificate and a Development Provisioning Profile
- These allow the developer to run the application on a specific device.
- Publish To the App Store
- The App Store - Apple will list your application in the App Store, and take care of credit-card processing, hosting, downloading, and notifying users of updates.

Developers name their own prices for their creations; Apple gets 30% of the sales price, with the developer getting the rest.

- iPhone developer program - To get your app into the store you have to pay $99 to join the program. No hidden charges.
- Once registered
    - A Distribution Certificate - associates a digital identity with the developer
    - A Distribution Provisioning Profile - code elements that Xcode builds into your application, creating a kind of "code fingerprint" that acts as a unique digital signature.
- A sample checklist
    - No crashes
    - No (big) memory leaks
    - Test on several devices with different (minimum and maximum!) OS versions
    - Test with different region and language settings
    - Consistent UI (use appropriate icons/buttons and expected behavior)
    - Artwork: Check default screens, icons. Both in various resolutions
    - Check icon pre-rendered state
    - Screenshots (without status bar), maybe in several languages
    - Description for AppStore, "What's new", maybe in several languages
    - If new app: decide on price and categories
    - Select Distribution build configuration
    - Check signing (distribution profile still valid etc.)

- Check base SDK and deployment target, supported devices
- Submit everything
  (Goldstein, 2009)

6.3 Windows Mobile

- Development
  - Visual Studio 2010 VB or C
  - Expression Studio used with VS2010,
  - Silverlight - XAML
- Publish Deploy to Device (Testing)
  - Create a Windows Phone developer account - Annual fee of $99
  - Get Zune software and have it running
    - Used for computer to communicate with the phone
  - Register the device with Windows Phone Developer Registration tool
  - Launch Application Deployment tool and follow wizard
- Publish to Windows Phone Marketplace
  - Create a Windows Phone developer account - Annual fee of $99
  - Developer keeps 70% of all proceeds
  - Set up a App Hub membership - Annual fee of $99
  - Needs application icon, application title icon, one screenshot
  - Five major categories of policies and requirements
    - Application policies
    - Content policies
    - Application submission requirements
    - Application certification requirements

        - Additional requirements for specific application types
    - Five steps to submit
        1. Upload the application
        2. Provide the application description
        3. Upload the artwork
        4. Set the application pricing
        5. Submit the application

        (Faucher, 2011)

6.4 Blackberry

- Development
    - Register for the BlackBerry Developer Community (Free)
    - BlackBerry Java Development Environment (JDE) or BlackBerry JDE Component Plug-in for Eclipse.
- Publish through BlackBerry App World
    - Register for the BlackBerry Developer Community (Free)
    - Sign up and register as a vendor with BlackBerry App World ($20 per application must buy ten credits - Total $200)
    - Select License
        - Static - Same as no license key at all.
        - Single - Uses the same key for all users.
        - Pool - Randomly retrieves a key from a pool of keys that the developer generates and maintains

        - Dynamic - Makes an HTTP request to a URL the developer specifies to retrieve the key.
    - Submit the application
        - Approval process deducts a application credit
        - If supports more than one OS and device target it still only takes one credit

    (Foust, 2010)

6.5 Web

After investigating the previous options of developing an application for a specific mobile device the choice was made to use the web to design for multiple devices. below are the results of the web investigation.

- Allows users to access the web application regardless of which hand held device they are using or if they are using a computer
- Allows developers to create one application to support all instead of developing for each possible device
- Does not use local resources
    - Development does not have to concern with efficient use of device resources
    - Less strain on the handheld device vs. local apps
- Rendering the correct content
    - Use HTTP request headers to identify mobile devices
        - User-Agent - Identifies the mobile browser
        - X-Wap-Profile - Provides the URL to a User Agent Profile

        - Accept - Provides a list of MIME types for content supported in the browser or device.
    - Use a device database to obtain device capabilities
        - Wireless Universal Resource File (WURFL)
    - Create Device Groups
    - Choose Adaptation Points - ex. Scale images for different size screens
    - Write Adaptation rules

    (Frederick, 2009)

HTML 5 Changes

- Doctype for HTML5
    - <!DOCTYPE html>
- Character Encoding
    - <meta charset = "UTF-8" />
- Dividing a Document
    - <section> - Logical division of the document.
        - Similar to div but more descriptive and content-sensitive way of dividing the document.
- Make Parts of Document Distributable
    - <article> - Used to identify the portions of the document that you want to be independent and distributable from the rest of the document

    (Lowery & Fletcher, 2011)

Chapter 7: Mobile Development Guide

Mobile devices may be grouped into categories to assist in the development of mobile applications. These categories are important because it is not practical to develop a website design for each of the thousands of devices. Proper categorization will be equally important to give the mobile user the proper experience on the website. The process of categorization and proper display is one that is in constant adjustment to get it right. As this process continues there will be more categories of phones to develop styles for to insure the correct display. There are many open source tools that can help in this process (Warner & Lafontaine, 2010). In general to accomplish the ability to design a website for multiple devices is to have device detection, find what a device can do, and redirection to proper page. This section will discuss the categories of devices, device detection, device functionality storage, and mobile device considerations when creating master pages.

7.1 Categories of Devices

Setting up different categories to which each device must fall into is an important step if creating a style sheet for different types of devices is needed. These categories will then be used to create a style sheet for that range of devices. The categories that will be used in this project are:

- Mob1 - Minicomputer or Pads. Most support similar features as desktop computers but allow for touch screen and orientation changing.
- Mob2 - Feature filled phones. These include iPhones, iPods, Androids, Windows Phone 7 devices, and other similar devices.

- Mob3 - Feature less phones. These include phones that do not support HTML5, CSS#, or JavaScript.

7.2 Detection

The detection of a mobile device is accomplished through a User Agent. The User Agent is a string of information containing what browser the client is using, browser version, and computer/device information. The website http://whatsmyuseragent.com/ shows an example of this information using your computer/device (Warner & Lafontaine, 2010). Below are examples from this website.

- iPad - Mozilla/5.0 (iPad; U; CPU OS 3_2 like Mac OS X; en-us) AppleWebKit/531.21.10 (KHTML, like Gecko) Version/4.0.4 Mobile/7B334b Safari/531.21.102011-10-16 20:23:50
- BlackBerry - Mozilla/5.0 (BlackBerry; U; BlackBerry 9800; en) AppleWebKit/534.1+ (KHTML, like Gecko) Version/6.0.0.337 Mobile Safari/534.1+2011-10-16 20:21:10
- Desktop Win 7 - Mozilla/5.0 (Windows NT 6.1; WOW64; rv:11.0) Gecko/20100101 Firefox/11.0

When a user opens a website the user's browser sends this information to the website's server. Client-side detection is also possible but since not all devices support JavaScript, especially older phones, there is a risk of not supporting a percentage of possible users. (Harrel 2011)

Another option would be to simply give the user the option to choose which version of your website they wish to see on the initial page of a website. This project will not be taking this approach.

The approach taken in this project is server-side detection. The reasons for picking this option is that client side detection does not work for users that do not allow or support JavaScript and to make a user simply pick to use a desktop version or mobile version is not an acceptable approach for the given application. Another reason for using the server side detection is once this method is used it allows the programmer more flexibility on programming business logic depending on the type of device and the device's attributes. An example of business logic depending on the type of device is if a site wanted to handle the shopping cart part of a website different for mobile users versus desktop users.

7.3 Device Functionality

There are two possibilities to determine a devices functionality, a) create a custom database filled with this information or b) use someone else's. It is extremely labor and time extensive to create a custom database filled with phone information that is important to a particular web design process. For that reason alone this method was not chosen for this project. It should be noted for a large corporation creating a custom database filled with device functionality information might be a reasonable option if using a 3rd party database's does not provide enough information for each device for their design. (Harrel 2011)

Wireless Universal Resource File (WURFL) is the most popular resource when finding device functionality. WURFL is an online database that contains information for a large amount of mobile devices based on user agents.

7.4 Mobile Web Styles Device Considerations

Differences in traditional web pages and mobile versions are in some ways obvious to users that use mobile devices quite regularly but never less are important when designing the different mobile master pages. The following is a list of features that should be considered. These considerations will be taken when a master page is created with a style sheet for each category.

Scroll Balls

Figure 7 - Example Scroll Ball

Figure 7 shows an example of a mobile device with a scroll ball. There no alternate tool tips to provide extra information to users. There are roll over events and these should be used to help the user know that certain objects are clickable links.

Touch Screens

Error! Reference source not found. shows an example of a mobile device that is a touch screen. Touch screens will obviously not have either the roll over events or tool tips so it is very important that the design of the object that is a clickable link appears to be something that should be clicked like a button.

Dual-Orientation

Error! Reference source not found. shows an example of a mobile device that has the dual orientation feature. Many handheld devices offer the ability to turn the device and use it in landscape or portrait view. These means depending on the criteria in the categories selected for a project, the same device might fall into different categories depending on orientation.

Figure 8 - Example Touch Screen

Figure 9 - Example Dual - Orientation

Chapter 8: Example of a potential ‘mini quiz’

8.1 Mini Quiz on a smart phone system overview

The system's main goal is to allow for instructors to access a web site where they can create a quiz and for students to access and take the quiz and get instant feedback on their answers. The instructor's website was designed to allow an instructor to create questions based on topics to allow for questions to be used on multiple quizzes and by different instructors. The students will be able to get to these quizzes by answering a few questions: the course, the instructor (if multiple), and the quiz. When the student selects an answer(s) before moving on to the next answer the student will be presented with the correct answer and a help tip that is provided by the instructor.

8.2 Examples of Mini Quiz

Figure 10, **Figure 12**, **Figure 15**, **Figure 17**, and **Figure 19** , demonstrate the ‘pre-implementation screens’. **Figure 11**, **Figure 13**, **Figure 14**, **Figure 16**, **Figure 18**, and **Figure 20** show final screens as implemented.

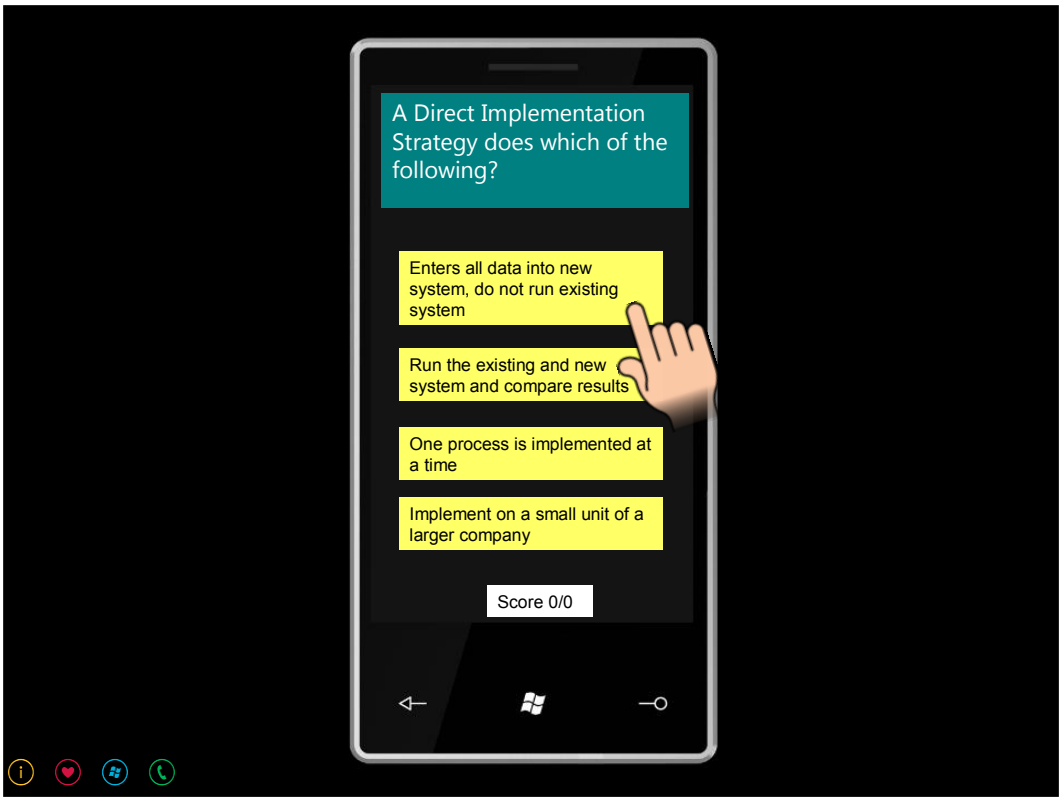


Figure 10 - Display a Question

Figure 11 - Example Actual Screen Displaying a Question

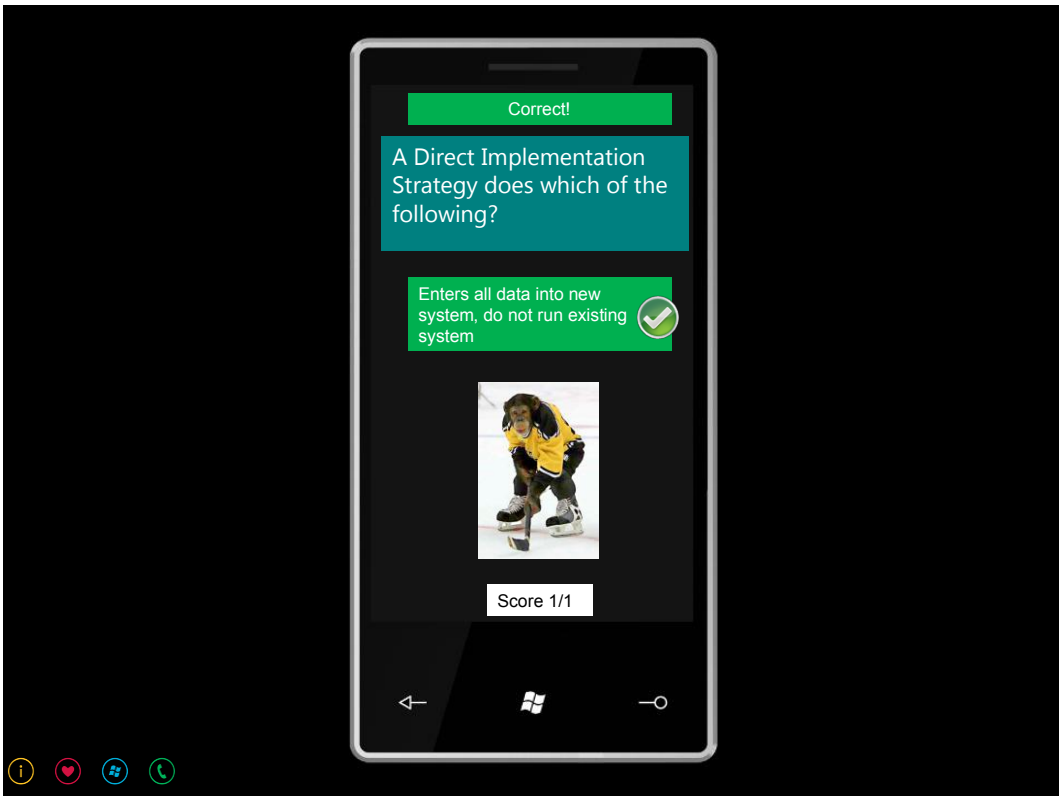


Figure 12 - Screen Showing a Correct Answer

Figure 13 - Example Actual Image for Correct Answer

Figure 14 - Example Actual Screen Showing a Correct Answer

Figure 15 - Screen Showing Question After Correct Answer

Figure 16 - Example Actual Screen Showing Question After Correct Answer

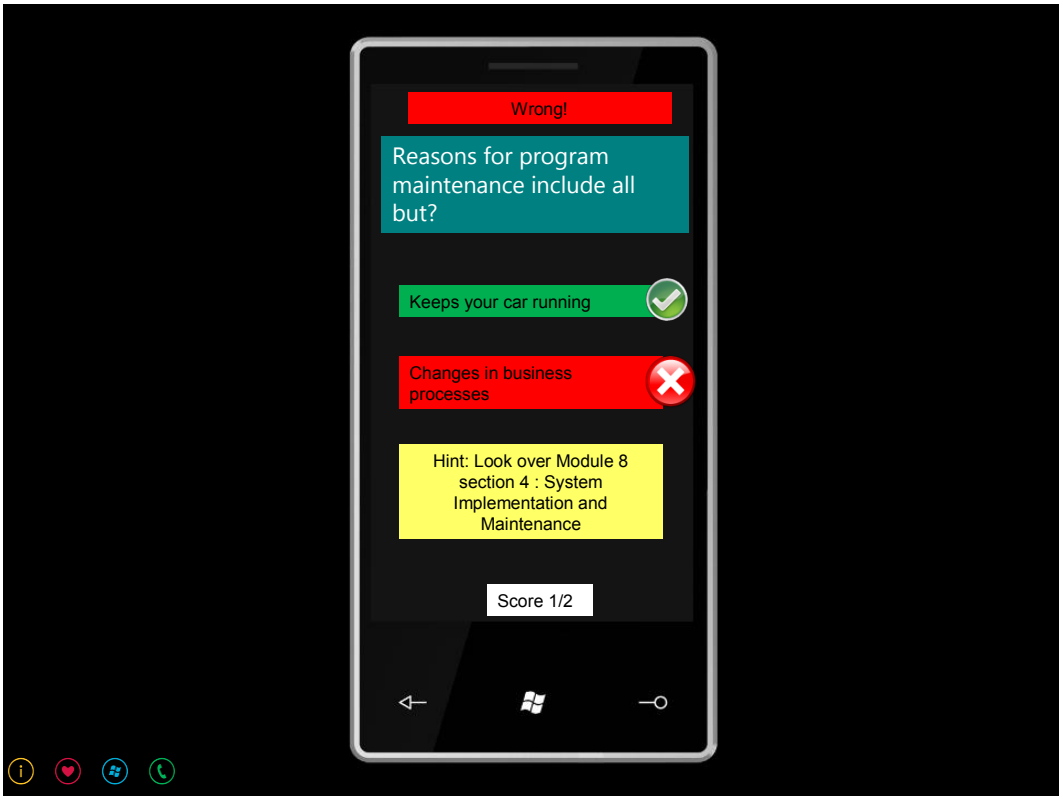


Figure 17 - Screen Showing an Incorrect Answer

Figure 18 - Example Actual Screen Showing an Incorrect Answer

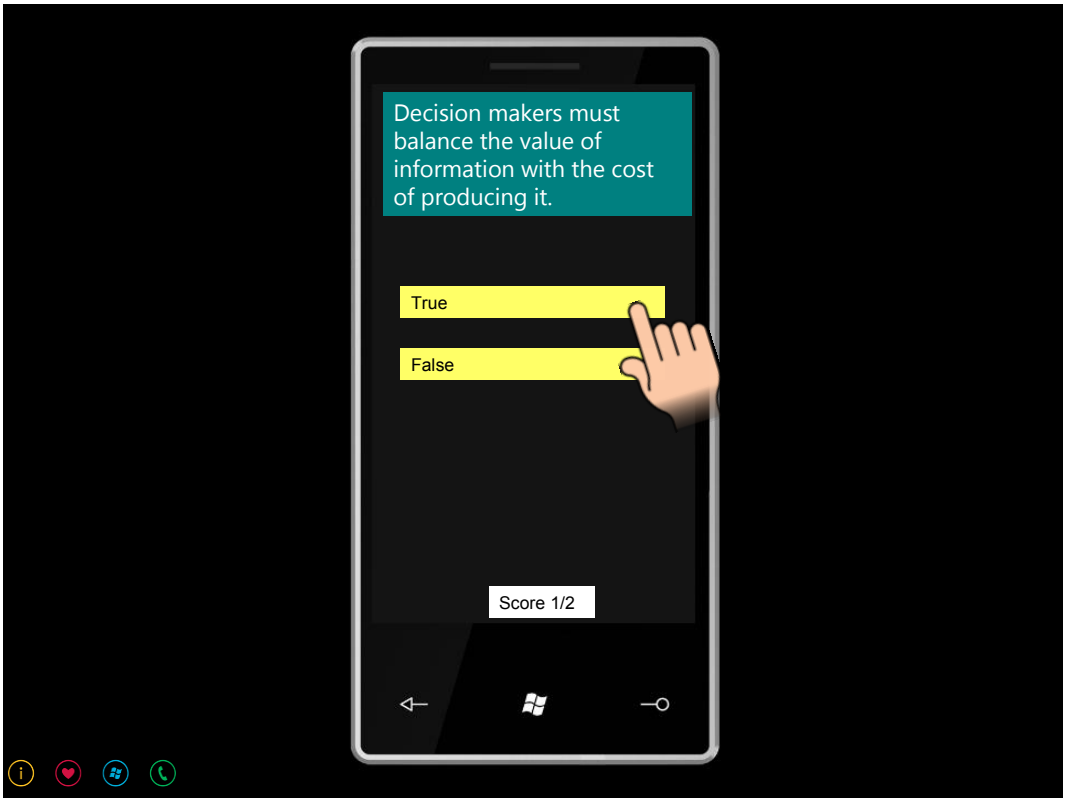


Figure 19 - Screen Showing True/False Question

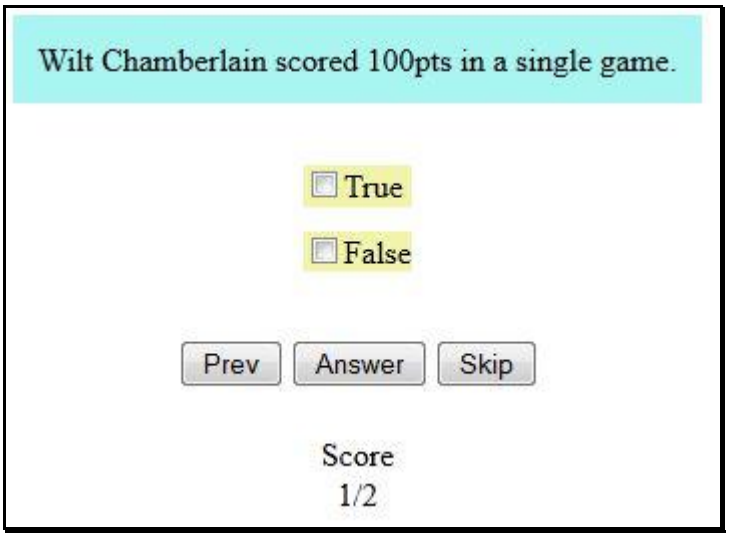


Figure 20 - Example Actual Screen Showing True/False Question

Chapter 9: Help Guide

This section is a help guide to using the instructors side of the Mobile Learning Application. The goals of this guide are to inform users where on the application to find help, understand the underlying methods to creating a quiz, and the steps to create and launch a quiz.

The Help icon, a blue circle with a blue question mark on a green page, can be found on each page on the right underneath the Log Out button as shown next to the black arrow in **Figure 21**. Clicking on the Help icon will give the user information on what the page is used for and how to use the controls on the page.

Figure 21 - Example of Help Button on Screen

The Mobile Learning Application is designed to allow instructors to create Quizzes by selecting from question either made by them or other instructors. Instructors will first assign themselves their Course from a populated list. Instructors will then create their own Topic for a given Course. They can then add a Question to a Topic that they created or that another instructor created. A Quiz is created by selecting Questions from any Topic for a given Course. Once the Quiz is created the instructor will launch the Quiz by selecting dates in which the Quiz will be available to the public. Finally the instructor will be able to see the unanimous results for each Quiz created.

9.1 Adding a Course

By clicking the Courses button on the top navigation instructors will see a list of Courses they are currently assigned. The instructors can assign a new Course by picking one from the list under the Header "Course" and then clicking on the "Insert" button to the left. An example of the Course page is shown in **Figure 22**.

Figure 22 - Example Course Page

9.2 Adding a Topic

By clicking the Topics button on the top navigation the instructors will first be prompted to select a Course that they are currently assigned shown in **Figure 23**.

Figure 23 - Example Topics Page Showing Course Prompt

The instructors will then be shown the current Topics that exist for the selected Course. Examples of topics for a particular course might be: IF statements, Do-While Loops, Subroutines and Functions. The instructors can add a new Topic by inputting the Topic description underneath the header "Description" and then clicking on the "Insert" button on the left. The instructors can also edit or delete each Topic by selecting the "Edit" button on the left or "Delete" button on the right of each Topic. Note that if an instructor tries to delete a Topic that has associated questions the Topic will not be deleted but it will be inactivated and removed from all lists. **Figure 24** is an example of the Topics page with a Course selected.

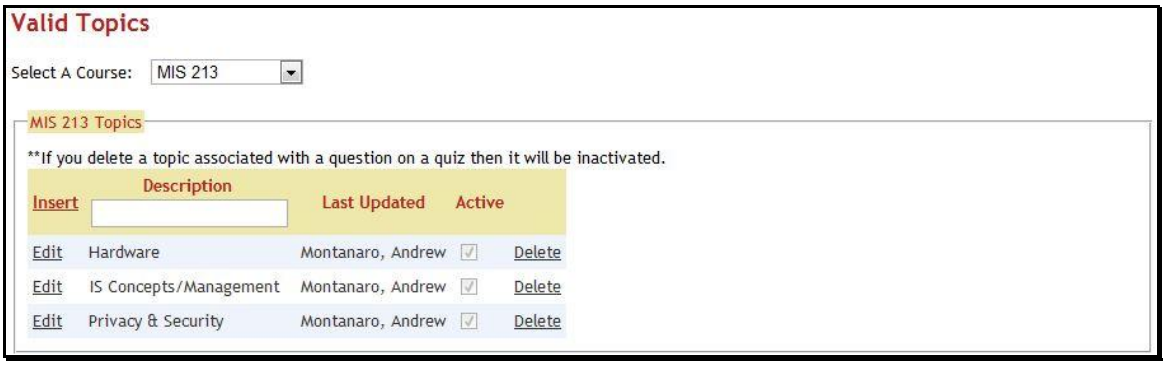


Figure 24 - Example Topics Page with Course Selected

9.3 Adding a Question

By clicking the Questions button on the top navigation the instructor will be first prompted to select a Course they are assigned and then a Topic as shown in **Figure 25**.

Figure 25 - Example of Questions Page with a Course Selected

After selecting a Topic the instructor will be shown a list of Questions associated with the selected Course and Topic. The instructor can then add a Question by clicking on the "Add Question" button as shown in **Figure 26**.

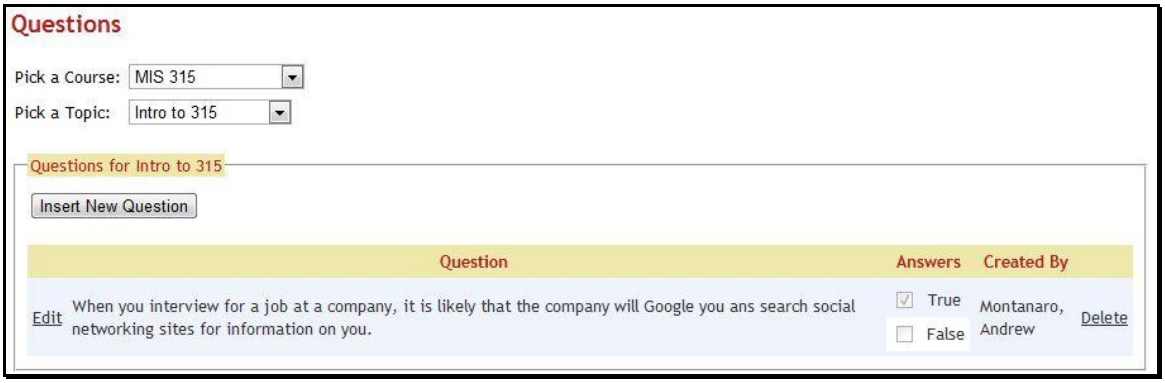


Figure 26 - Example Question Page with Topic Selected

Clicking on the "Add Question" button will then give the instructor areas to add the question, answers, and a hint to be shown to students after answering the Question. **Figure 27** is an example of inserting a Question.

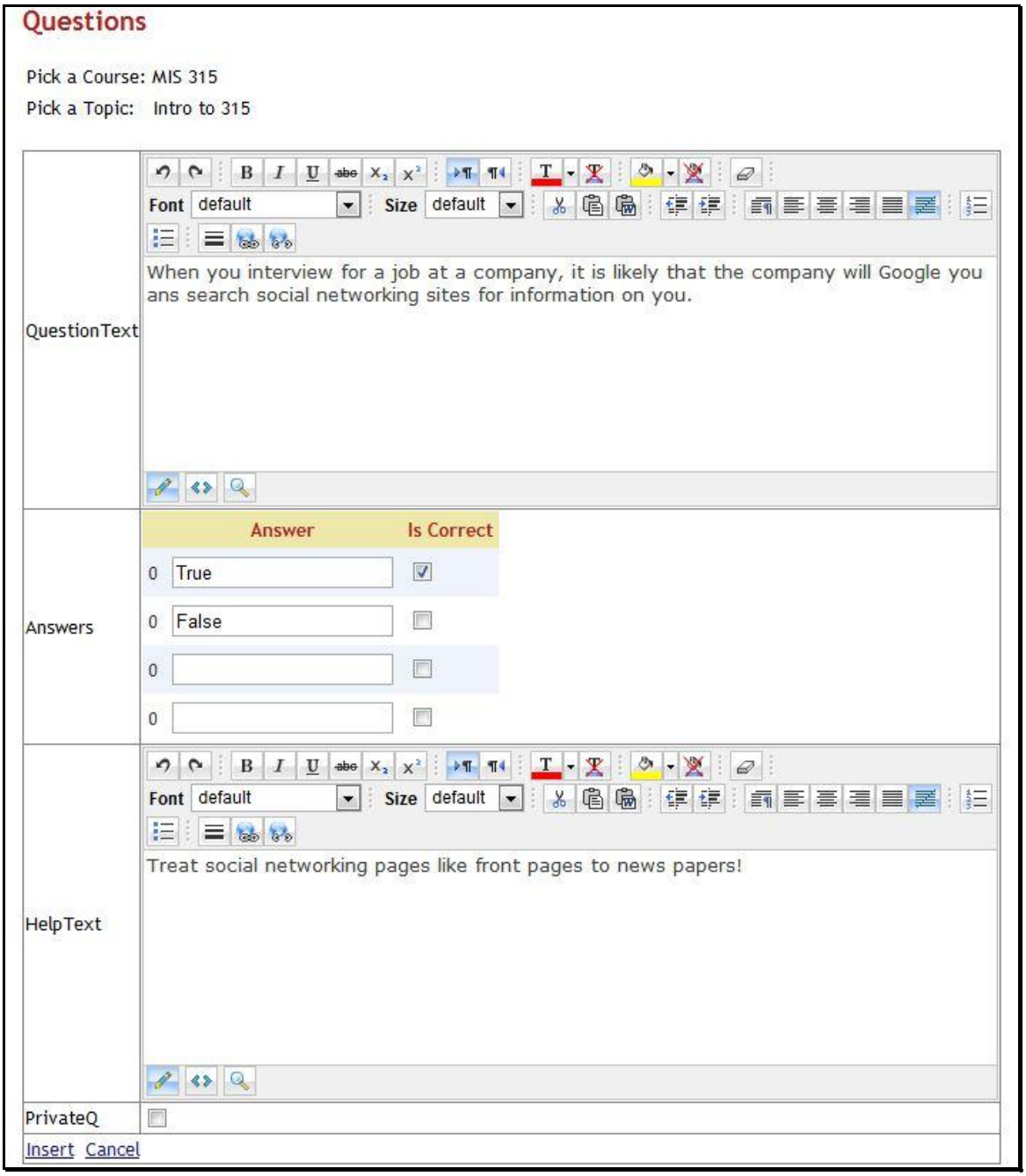


Figure 27 - Example of Add New Question

9.4 Adding Questions to a Quiz

By clicking the Quizzes button on the top navigation the instructor will be first prompted to select a Course and then a Quiz. When a Quiz is selected the current Questions on the Quiz will be displayed along with a button to add a Question as shown in **Figure 28**.

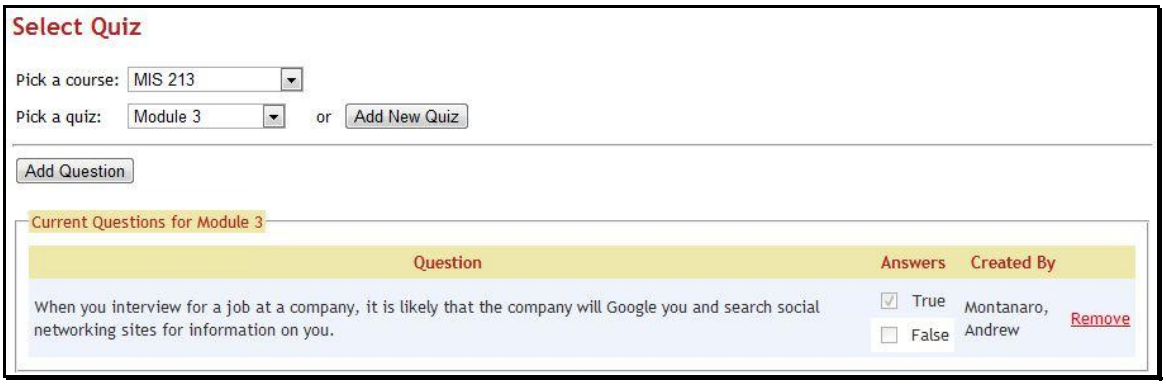


Figure 28 - Example Quiz Page with Quiz Selected

When the instructor clicks on the button "Add Question" a they are prompted to selected a Topic of Questions to select from. When the instructor has selected all the Questions for the Quiz they select the button "Add Questions to Quiz". **Figure 29** is an example of the Question selection screen.

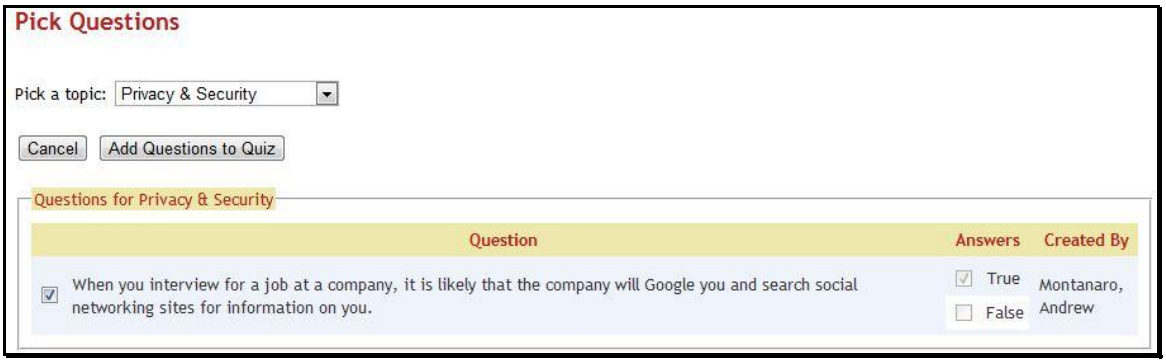


Figure 29 - Example of Selecting Question for Quiz

9.5 Launch Quiz

By clicking the Launch Quiz button on the top navigation the instructor will be first prompted to select a Course. Selecting a Course will then display a list of all Quizzes the instructor has made for the Course with start and end dates of when the Quiz will be active and available to students. **Figure 30** is an example of the Launch Quiz page.

Figure 30 - Example Launch Quiz Page

Chapter 10:Completed Deliverables

There were four main deliverables for this project. There must be a working prototype in a web and smart phone environment so students can take a quiz. There needs to be a web based site for instructors to be able to create and launch quizzes for the students to take. The instructor site must also have a reporting mechanism to allow the instructor to review the results of quizzes that are taken. A how to guide in this paper and implemented on the instructor site must also be completed to help with new instructor training.

10.1 Project Timeline

Table 3 - Project Timeline

Task Name	Duration	Start	Finish	Predecessors
Analysis	**17 days**	**Mon 4/18/11**	**Tue 5/10/11**	
Investigate & Learn from Similar Applications	2 days	Mon 4/18/11	Tue 4/19/11	
Investigate Uploading Quizzes from Book	2 days	Wed 4/20/11	Thu 4/21/11	2
Create Faculty Survey	1 day	Fri 4/22/11	Fri 4/22/11	3,2
Faculty Survey	4 days	Mon 4/25/11	Thu 4/28/11	4
Create Project Charter	2 days	Fri 4/29/11	Mon 5/2/11	5
Create Actor diagram	3 days	Tue 5/3/11	Thu 5/5/11	6
Create Use Cases	3 days	Tue 5/3/11	Thu 5/5/11	6
Review software specifications with Committee	4 hrs	Fri 5/6/11	Fri 5/6/11	5,6,7,8
Incorporate feedback on software specifications	1 day	Fri 5/6/11	Mon 5/9/11	9
Develop delivery timeline	1 day	Mon 5/9/11	Tue 5/10/11	10
Obtain approvals to proceed (concept, timeline)	4 hrs	Tue 5/10/11	Tue 5/10/11	11
Analysis complete	0 days	Tue 5/10/11	Tue 5/10/11	12
Design	**13.5 days**	**Wed 5/11/11**	**Mon 5/30/11**	**13**
Review preliminary software specifications	1 day	Wed 5/11/11	Wed 5/11/11	13
Develop Data Model	3 days	Thu 5/12/11	Mon 5/16/11	15
Develop System Diagram	3 days	Tue 5/17/11	Thu 5/19/11	16
Develop prototype based on functional specifications	4 days	Fri 5/20/11	Wed 5/25/11	17
Review functional specifications with committee	1 day	Thu 5/26/11	Thu 5/26/11	18
Incorporate feedback into functional specifications	1 day	Fri 5/27/11	Fri 5/27/11	19
Obtain approval to proceed	4 hrs	Mon 5/30/11	Mon 5/30/11	20
Design complete	0 days	Mon 5/30/11	Mon 5/30/11	21
Development	**31 days**	**Mon 5/30/11**	**Tue 7/12/11**	**22**
Review functional specifications	1 day	Mon 5/30/11	Tue 5/31/11	22
Identify target device	1 day	Tue 5/31/11	Wed 6/1/11	24
Develop Admin User Interface	7 days	Wed 6/1/11	Fri 6/10/11	25
Develop Mobile UI	15 days	Fri 6/10/11	Fri 7/1/11	26
Develop Mobile quiz code	7 days	Fri 7/1/11	Tue 7/12/11	27
Developer testing (primary debugging)	5 days	Tue 7/5/11	Tue 7/12/11	28FS-75%
Development complete	0 days	Tue 7/12/11	Tue 7/12/11	29
Documentation	**3 days**	**Tue 7/12/11**	**Fri 7/15/11**	**30**
Develop Admin Help specification	1 day	Tue 7/12/11	Wed 7/13/11	22
Review Admin Help documentation	1 day	Tue 7/12/11	Wed 7/13/11	
Incorporate Help documentation feedback	1 day	Wed 7/13/11	Thu 7/14/11	33
Develop user Help	1 day	Tue 7/12/11	Wed 7/13/11	28FS-50%
Review all user documentation	1 day	Wed 7/13/11	Thu 7/14/11	35
Incorporate user documentation feedback	1 day	Thu 7/14/11	Fri 7/15/11	36
Documentation complete	0 days	Fri 7/15/11	Fri 7/15/11	37,34
Software development template complete	0 days	Fri 7/15/11	Fri 7/15/11	38

10.2 Actual Timeline

The project was completed on of 4/17/2012. There were many reasons in which the projected timeline, as shown in Table 3, and actual completed date were greatly different. When the projected timeline was first drafted I was working at UNCW as a student developer and could only work up to 30 hours a week. With that schedule I would work on the project for at least 12 hours a week to complete a full work week. I had planned to work through the summer 2011 to reach the projected goal until I paid for my continuing credit course in Fall instead of the Summer to give some leeway. Shortly after that I was lucky enough to find a job in Wilmington in which I work 40 plus hours a week and with other obligations and lots of procrastination I continued to stretch out the timeline through the spring semester.

10.3 Lessons learned as the project timeline was extended

Since this is a project for educational purposes and not for commercial purposes one might think that the lack of a budget would remove problems from going past dates on milestones. I would agree that my lively hood was never in jeopardy but there were some negatives to dragging out a project like this. I will discuss the negatives of dragging out a project, things that worked well during the project, and what I would do differently on future projects.

One negative was the longer the project would linger the more moral can dwindle. It went from being a project that I can't wait to do to one that I can't wait to finish. Another negative would be consistency and familiarly within my project. I found it hard to pick up where I left off if there was substantial time in between and it was difficult to handle similar user interactions with consistent code without taking more time to look at

already coded milestones. One last negative would be simply the miss use of the timeline itself. Instead of keeping track of when milestones were completed I started to simply use it as a list of remaining items for me to work on here and there.

Using RAD was a decision that went particularly well in that I was able to make a lot of progress through each iteration of the design of the database and the UI pages. The 51Degrees.mobi framework worked fairly simply and gave me the desired results I was looking for about mobile devices. I would use this framework again and test the premium product for funded projects.

Things that I will do differently on future projects that I learned from this project are proper project management, proper documentation, use tools and volunteers for testing. I would use past experiences and techniques like page counting to help in the time estimation milestones as well as be realistic as possible to hours of developer availability. Documenting interviews with end users using formal interview sheets with questions and related answers as well as document what decisions that might have taken place during the interview. It is important to know where important decisions about the application came from and why they are made and have that information documented to produce to the users upon request. I would also take advantage of user interface tools like Selenium to test the user interface as well as unit testing to test classes that were created. I also would ask for volunteers to gain their opinions of the user interface on if the pages, prompts, and tasks make sense and to determine how much training a future user will need when creating quizzes.

10.4 Possible Future enhancements

Possible future enhancements for this project are to improve on the design for scalability, allow for the capability to import test banks, more mobile device testing and style sheet creation, and the ability to track unique devices.

When this project was designed it was designed for one department of a college to use. To allow for scalability to grow to all departments in a university or to even allow for many universities to use the design will have to change. In both cases a table should be added to track the university and department of the instructor. This would fix the problem of lists of instructors, courses, topics, and questions becoming overwhelmingly long. When scaling to as large as multiple universities it could be beneficial to allow for the sharing of topics and questions as universities have similar courses but this should probably be regulated to only questions that are imported out of textbooks.

As stated just before many instructors would like to import questions from textbooks and since this is standard on many other programs that create quizzes this should be a future enhancement. The easier it is for the instructor to create the quizzes the larger the possible acceptance would be. The program would have to be able to accept the normal formats such as xml or csv and create the questions from the imported files.

The more testing the better especially when it comes to mobile devices. The more devices that get tested will allow for the more styles sheets that can be created to fit different resolutions.

The ability to track unique devices would allow for the instructors to have a little more information on the students taking the quizzes but still allow for students to take quizzes and not have their name associated with any results. The idea is to find out if

there were 100 responses to question number 1 how many students actually answered the question. In the current system you will not know if there were 100 different students or 10 different students taking it 10 times each. A way to track the unique number of students would be to obtain and store the user's mac address. This would only take minor changes to the database and the "Quiz Results" page.

References

Amsel, Abram (1989). Behaviorism, Neobehaviorism, and Cognitivism in Learning Theory: Historical and Contemporary Perspectives. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

Awad, M.A. (2005). A Comparison between Agile and Traditional Software Development Methodologies. The University of Western Australia.

Charvat, Jason (2003). Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects. John Wiley & Sons, Inc. Hoboken, New Jersey.

CMS - Centers for Medicare & Medicaid Services (2008). Selecting a Development Approach. Publishing in CMS. March 27, 2008.

Faucher, Brian (2011). Windows Phone 7 Application Development: 24 Hour Trainer. Wiley Publishing, Inc. Indianapolis, IN.

Forrester, D., & Jantzie, N. (1998). Learning Theories. University of Calgary. Retrieved January 1, 2011 from *http://www.acs.ucalgary.ca/%7Egnjantzi/learning_theories.htm*

Foust, Bill (2010). BlackBerry Java Application Development - Beginner's Guide. Packt Publishing. Birmingham, UK.

Frederick, G. & Lal, R.(2009). Beginning Smartphone Web Development-Building JavaScript, CSS, HTML and Ajax-Based Applications for iPhone, Android, Palm Pre, BlackBerry, Windows Mobile, and Nokia S60. Apress. New York, New York.

Goldstein, Neal (2009). iPhone Application Development for Dummies. John Wiley & Sons, Inc. Hoboken, New Jersey.

Hannafin, M., & Peck, K. (1988). The Design, Development, and Evaluation of Instructional Software. Macmillan Publishing Company. New York, New York.

Harrel, William (2011). HTML, CSS & JavaScript Mobile Development for Dummies. Wiley Publishing, Inc. Hoboken, NJ.

Leonard, David (2002). Learning Theories, A to Z. Greenwood Press Westport, Ct.

Lowery J & Fletcher (2011). HTML 5: 24-Hour Trainer. Wiley Publishing, Inc. Indianapolis, Indiana.

McGilly, Kate (1994). Classroom Lessons: Integrating Cognitive Theory and Classroom. Massachusetts Institute of Technology Press.

Rosenberg, M. Westing, D., & Mcleskey J (2006). Using Computer-Assisted Instruction with Students Identified with ADHD. Retrieved March 21, 2011 from http://www.education.com/reference/article/computer-assisted-instruction-ADHD/

Ryan, S., Scott, B., Freeman, H., & Patel, D (2000). The Virtual University: The Internet and Resource-Based Learning. Stylus Publishing Inc. Sterling, VA.

Tsui, F. & Karam O. (2011). Essentials of Software Engineering. Jones and Barlett Publishers, LLC. Sudbury, MA.

Warner, J., & Lafontaine, D. (2010). Mobile Web Design For Dummies. Wiley Publishing, Inc. Hoboken, NJ.

Appendixes

Appendix A - Use Case Diagram

Appendix B - Use Cases

Courses

<table>
<tr><td>Use Case Name:</td><td>Add Course to Instructor</td></tr>
<tr><td>Scenario:</td><td>Instructor wants to add a current course</td></tr>
<tr><td>Triggering Event:</td><td>Instructor wants to add a current course for them self</td></tr>
<tr><td>Brief Description</td><td>When an Instructor wants to Add a current course for them self they will go to the "Courses" page and select a new course to add.</td></tr>
<tr><td>Actors:</td><td>Instructor</td></tr>
<tr><td>Related Use Case:</td><td>None</td></tr>
<tr><td>Stakeholders:</td><td>Admin, Instructors, and Students</td></tr>
<tr><td>Preconditions:</td><td>User must have a role of Instructor</td></tr>
<tr><td>Post conditions:</td><td>None</td></tr>
<tr><td>Flow of Events</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Instructor goes to the Courses page.</td><td>1.1 Displays a list of all current Courses</td></tr>
<tr><td>2. Instructor selects the course name they wish to insert.</td><td></td></tr>
<tr><td>3. Instructor clicks Insert.</td><td>3.1 Insert the selected course as a current course for the instructor</td></tr>
</table>
</td></tr>
<tr><td>Exception Conditions:</td><td></td></tr>
<tr><td>Use Case Name:</td><td>Edit a Quiz/Launch Quiz</td></tr>
<tr><td>Scenario:</td><td>User wants to Edit a Quiz</td></tr>
<tr><td>Triggering Event:</td><td>Instructor wants to a Edit a Quiz</td></tr>
<tr><td>Brief Description</td><td>When an Instructor wants to Edit a Quiz they go to the "Launch Quiz" page, selects a course and changes the quizzes info</td></tr>
<tr><td>Actors:</td><td>Instructor</td></tr>
<tr><td>Related Use Case:</td><td>None</td></tr>
<tr><td>Stakeholders:</td><td>Admin, Instructors, and Students</td></tr>
<tr><td>Preconditions:</td><td>User must have Role of Instructor</td></tr>
<tr><td>Post conditions:</td><td>None</td></tr>
<tr><td>Flow of Events</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Actor goes to the Launch Quiz page.</td><td>1.1 Displays a list of all Courses</td></tr>
<tr><td>2. Actor selects a Course</td><td>2.1 Displays a list of all Quizzes</td></tr>
</table>
</td></tr>
</table>

		in Edit Mode
	3. Actor inputs dates or changes name	
	4. Actor selects "Update"	4.1 Updates each row with a change
Exception Conditions:		

Use Case Name:	Update a course
Scenario:	User wants to Update a course
Triggering Event:	Admin or Instructor wants to Update a course
Brief Description	When a Admin or Instructor wants to Update a course they go to the "Courses" page and clicks on the "Edit" button next to the course they want to update
Actors:	Admin or Instructor
Related Use Case:	None
Stakeholders:	Admin, Instructors, and Students
Preconditions:	Course must exist and User must have access to the application
Post conditions:	None

Flow of Events

Actor	System
1. Admin or Instructor goes to the Courses page.	1.1 Displays a list of all Courses
2. Admin or Instructor clicks edit next to the course they want to update.	
3. Admin or Instructor inputs the new course name and selects if the course is active or not	
3. Admin or Instructor clicks Update.	3.1 The course is updated with the inputted course name and active check

Exception Conditions:	

Use Case Name:	Delete a course
Scenario:	User wants to delete a course
Triggering Event:	Admin or Instructor wants to delete a course
Brief Description	When a Admin or Instructor wants to delete a course they go to the "Courses" page and click on the "Delete" button next to the course they

<table>
<tr><td></td><td>wish to delete</td></tr>
<tr><td>Actors:</td><td>Admin or Instructor</td></tr>
<tr><td>Related Use Case:</td><td>None</td></tr>
<tr><td>Stakeholders:</td><td>Admin, Instructors, and Students</td></tr>
<tr><td>Preconditions:</td><td>Course must exist and User must have access to the application</td></tr>
<tr><td>Post conditions:</td><td>None</td></tr>
<tr><td>Flow of Events</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Admin or Instructor goes to the Courses page.</td><td>1.1 Displays a list of all Courses</td></tr>
<tr><td>2. Admin or Instructor finds the Course they wish to delete</td><td></td></tr>
<tr><td>3. Admin or Instructor clicks delete for the Course.</td><td>3.1 Check that a Topic doesn't have the course and a Class doesn't have a Class.
3.2 No Topics or Classes have course so delete from table</td></tr>
</table>
</td></tr>
<tr><td>Exception Conditions:</td><td>3.1 If Topic has a course or if a Class has course.
3.1b Update the Course to in-active</td></tr>
</table>

<table>
<tr><td>Use Case Name:</td><td>Insert a course</td></tr>
<tr><td>Scenario:</td><td>User wants to Insert a course</td></tr>
<tr><td>Triggering Event:</td><td>Admin or Instructor wants to Insert a course</td></tr>
<tr><td>Brief Description</td><td>When a Admin or Instructor wants to Insert a course they go to the "Courses" page, inputs the name of the course, and clicks on the "Insert" button</td></tr>
<tr><td>Actors:</td><td>Admin or Instructor</td></tr>
<tr><td>Related Use Case:</td><td>None</td></tr>
<tr><td>Stakeholders:</td><td>Admin, Instructors, and Students</td></tr>
<tr><td>Preconditions:</td><td>User must have access to the application</td></tr>
<tr><td>Post conditions:</td><td>None</td></tr>
<tr><td>Flow of Events</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Admin or Instructor goes to the Courses page.</td><td>1.1 Displays a list of all Courses</td></tr>
<tr><td>2. Admin or Instructor inputs the course name they wish to insert.</td><td></td></tr>
<tr><td>3. Admin or Instructor clicks Insert.</td><td>3.1 Insert the course with the inputted course name and as</td></tr>
</table>
</td></tr>
</table>

		active
Exception Conditions:		

Topics

Use Case Name:	Insert a Topic
Scenario:	User wants to Insert a Topic
Triggering Event:	Instructor wants to Insert a Topic
Brief Description	When an Instructor wants to Insert a Topic they go to the "Topics" page, selects a course, inputs the name of the Topic, and clicks on the "Insert" button
Actors:	Instructor
Related Use Case:	None
Stakeholders:	Admin, Instructors, and Students
Preconditions:	User must have a Role of Instructor
Post conditions:	None
Flow of Events	Actor: 1. Instructor goes to the Topics page. System: 1.1 Displays a list of all Courses Actor: 2. Instructor selects a course. System: 2.1 Displays a list of all Topics for Course Actor: 3. Instructor inputs a Topic description Actor: 4. Instructor clicks insert System: 3.1 Insert Topic for the Course
Exception Conditions:	

Use Case Name:	Edit a Topic
Scenario:	User wants to Edit a Topic
Triggering Event:	Instructor wants to Edit a Topic
Brief Description	When an Instructor wants to Edit a Topic they go to the "Topics" page, selects a course, clicks edit next to the name of the Topic
Actors:	Instructor
Related Use Case:	None
Stakeholders:	Admin, Instructors, and Students

Preconditions:	User must have a Role of Instructor
Post conditions:	None
Flow of Events	

Actor	System
1. Instructor goes to the Topics page.	1.1 Displays a list of all Courses
2. Instructor selects a course.	2.1 Displays a list of all Topics for Course
3. Instructor clicks edit next to Topic	3.1 Display Topic in Edit mode
3. Instructor changes a Topic description or active check	
4. Instructor clicks update	3.1 Update Topic for the Course

Exception Conditions:	

Question

Use Case Name:	Delete a Question
Scenario:	User wants to delete a question for a topic
Triggering Event:	Admin or Instructor wants to question for a topic
Brief Description	When a Admin or Instructor wants to delete a question they go to the "Questions" page and selects the proper course and topic to display the question. The Actor will then be able to delete if they were the creator of the question.
Actors:	Admin or Instructor
Related Use Case:	None
Stakeholders:	Admin, Instructors, and Students
Preconditions:	Course & Topic must exist, User must have access to the application, and the Actor must have created the question
Post conditions:	None
Flow of Events	

Actor	System
1. Actor goes to the Questions page.	1.1 Displays a list of all Courses
2. Actor selects a Course	2.1 Displays a list of all Topics for the Course
3. Actor selects a Topic	3.1 Displays a list of all Questions for the Topic. 3.2 Show Edit & Delete button if Actor Created Question
4. Actor selects "Delete"	4.1 Display confirmation box

	5. Actor selects "OK" to confirmation	5.1 Delete Question
Exception Conditions:	5.1 No Action if another user is using question on a quiz 5.2 Sets Inactive if Actor is using the question on a quiz	

Use Case Name:	Insert a Question	
Scenario:	User wants to Insert a Question for a Topic	
Triggering Event:	Admin or Instructor wants to a Question for a Topic	
Brief Description	When a Admin or Instructor wants to Insert a Question they go to the "Question" page, inputs the question, help, private status, answers for the question, and clicks on the "Insert" button	
Actors:	Admin or Instructor	
Related Use Case:	None	
Stakeholders:	Admin, Instructors, and Students	
Preconditions:	User must have access to the application	
Post conditions:	None	
Flow of Events	Actor	System
	1. Actor goes to the Questions page.	1.1 Displays a list of all Courses
	2. Actor selects a Course	2.1 Displays a list of all Topics for the Course
	3. Actor selects a Topic	3.1 Display the "Insert New Question" button
	4. Actor selects "Insert New Question"	4.1 Display question in insert mode.
	5. Actor selects "Insert"	5.1 Check that there is an Answer and 1 correct 5.2 Insert the Question 5.3 Insert Answers if there is text
	6. Actor selects "Cancel"	6.1 Close Question in insert mode and display questions
Exception Conditions:		

Use Case Name:	Update a Question
Scenario:	User wants to Update a Question or it's Answers
Triggering	Admin or Instructor wants to Update a Question or it's Answers

<table>
<tr><td>Event:</td><td></td></tr>
<tr><td>Brief Description</td><td>When a Admin or Instructor wants to Update a Question they go to the "Questions" page and selects the proper course and topic to display the question. The Actor will then be able to edit if they were the creator of the question.</td></tr>
<tr><td>Actors:</td><td>Admin or Instructor</td></tr>
<tr><td>Related Use Case:</td><td>None</td></tr>
<tr><td>Stakeholders:</td><td>Admin, Instructors, and Students</td></tr>
<tr><td>Preconditions:</td><td>Course & Topic must exist, User must have access to the application, and the Actor must have created the question</td></tr>
<tr><td>Post conditions:</td><td>None</td></tr>
<tr><td>Flow of Events</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Actor goes to the Questions page.</td><td>1.1 Displays a list of all Courses</td></tr>
<tr><td>2. Actor selects a Course</td><td>2.1 Displays a list of all Topics for the Course</td></tr>
<tr><td>3. Actor selects a Topic</td><td>3.1 Displays a list of all Questions for the Topic.
3.2 Show Edit & Delete button if Actor Created Question</td></tr>
<tr><td>4. Actor selects "Edit"</td><td>4.1 Display the question and answers if Edit Mode</td></tr>
<tr><td>5. Actor Selects "Update"</td><td>5.1 Check that there is an Answer and 1 correct
5.2 Update the Question
5.3 Update existing Answers if there is text
5.4 Delete existing Answers if empty
5.5 Insert new Answers if there is text
5.6 Close Question in edit mode and display questions</td></tr>
<tr><td>6. Actor Selects "Cancel"</td><td>6.1 Close Question in edit mode and display questions</td></tr>
</table>
</td></tr>
<tr><td>Exception Conditions:</td><td></td></tr>
</table>

Quiz

Use Case Name:	Insert a Quiz

<table>
<tr><td>Scenario:</td><td>User wants to Insert a Quiz</td></tr>
<tr><td>Triggering Event:</td><td>Instructor wants to a Insert a Quiz</td></tr>
<tr><td>Brief Description</td><td>When an Instructor wants to Insert a Quiz they go to the "Quizzes" page, selects a course and inserts a new Quiz name</td></tr>
<tr><td>Actors:</td><td>Instructor</td></tr>
<tr><td>Related Use Case:</td><td>None</td></tr>
<tr><td>Stakeholders:</td><td>Admin, Instructors, and Students</td></tr>
<tr><td>Preconditions:</td><td>User must have access to the application</td></tr>
<tr><td>Post conditions:</td><td>None</td></tr>
<tr><td>Flow of Events</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Actor goes to the Quizzes page.</td><td>1.1 Displays a list of all Courses</td></tr>
<tr><td>2. Actor selects a Course</td><td>2.1 Displays a list of all Quizzes and a option to add new quiz</td></tr>
<tr><td>3. Actor Click on "Add New Quiz"</td><td>3.1 Display textbox to input a quiz name</td></tr>
<tr><td>4. Actor inputs a new quiz name</td><td>4.1 Check that a name is entered</td></tr>
<tr><td>5. Actor selects "Insert"</td><td>5.1 Insert a new Quiz for that Instructor's Course</td></tr>
<tr><td>6. Actor selects "Cancel"</td><td>6.1 Close the enter new quiz mode</td></tr>
</table>
</td></tr>
<tr><td>Exception Conditions:</td><td></td></tr>
</table>

Admin Tools

Use Case Name:	Update a User
Scenario:	Admin or user wants to Update User info
Triggering Event:	Admin or user wants to Update User info
Brief Description	When a Admin wants to update a users info they go to the "Admin Tools" page and select Edit. A User can change their info by clicking edit next to their info on the "Profile"
Actors:	Admin or User
Related Use Case:	None
Stakeholders:	Admin, Instructors, and Students
Preconditions:	User must exists and Actor must get to the "Profile" or "Admin Tools" pages.
Post	None

conditions:	
Flow of Events	<table><tr><th>Actor</th><th>System</th></tr><tr><td>1. Actor clicks on edit next to the users information.</td><td>1.1 Displays a list of all Users Information</td></tr><tr><td>2. Actor changes the Users current info</td><td></td></tr><tr><td>3. Actor clicks Update.</td><td>3.1 The User is updated with the inputted info</td></tr></table>
Exception Conditions:	

Use Case Name:	Insert a User
Scenario:	Admin wants to Insert a new User
Triggering Event:	Admin wants to Insert a new User
Brief Description	When a Admin wants to Insert a User they go to the "Admin Tools" page, inputs the required information, and clicks on the "Insert" button
Actors:	Admin
Related Use Case:	None
Stakeholders:	Admin, Instructors, and Students
Preconditions:	User must have Role of Admin
Post conditions:	None
Flow of Events	<table><tr><th>Actor</th><th>System</th></tr><tr><td>1. Actor goes to the Admin Tools page.</td><td>1.1 Displays a list of all Users</td></tr><tr><td>2. Actor Inputs all required information</td><td>2.1 Displays error message if incorrectly entered</td></tr><tr><td>3. Actor selects "Insert"</td><td>3.1 Check all inputs required are entered and all inputs are entered correctly</td></tr></table>
Exception Conditions:	

Use Case Name:	Delete a Role
Scenario:	User wants to delete a Role
Triggering Event:	Admin wants to delete a Role
Brief Description	When a Admin wants to delete a Role they go to the "Admin Tools" page and click on the "Delete" button next to the Role they wish to

<table>
<tr><td></td><td>delete</td></tr>
<tr><td>Actors:</td><td>Admin</td></tr>
<tr><td>Related Use Case:</td><td>None</td></tr>
<tr><td>Stakeholders:</td><td>Admin, Instructors, and Students</td></tr>
<tr><td>Preconditions:</td><td>Role must exist and User must have Role of Admin</td></tr>
<tr><td>Post conditions:</td><td>None</td></tr>
<tr><td>Flow of Events</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Admin or Instructor goes to the Courses page.</td><td>1.1 Displays a list of all Courses</td></tr>
<tr><td>2. Admin or Instructor finds the Course they wish to delete</td><td></td></tr>
<tr><td>3. Admin or Instructor clicks delete for the Course.</td><td>3.1 Check that a Topic doesn't have the course and a Class doesn't have a Class.
3.2 No Topics or Classes have course so delete from table</td></tr>
</table>
</td></tr>
<tr><td>Exception Conditions:</td><td>3.1 If Topic has a course or if a Class has course.
3.1b Update the Course to in-active</td></tr>
</table>

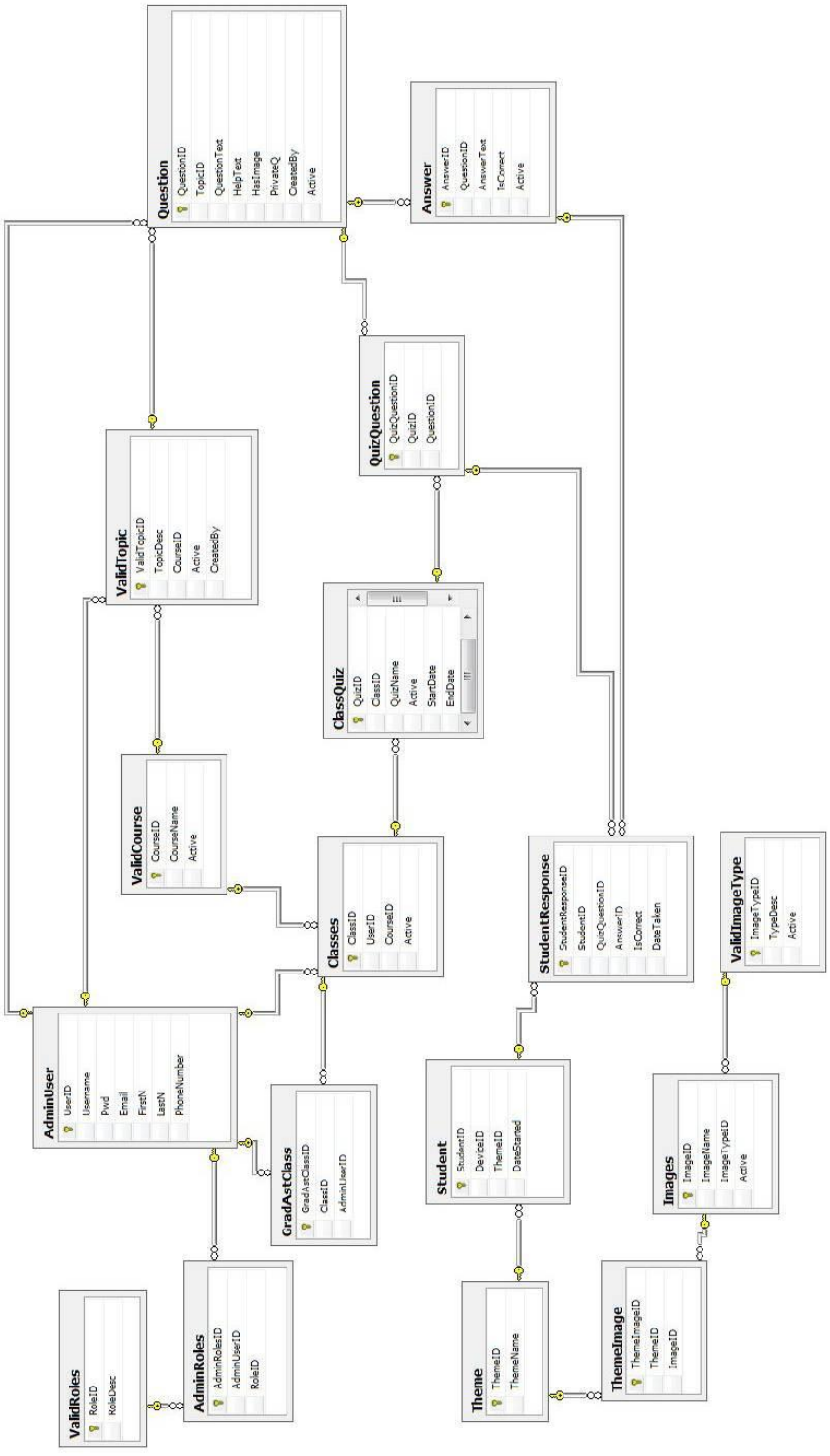
<table>
<tr><td>Use Case Name:</td><td>Insert a Role</td></tr>
<tr><td>Scenario:</td><td>Admin wants to Insert a Role</td></tr>
<tr><td>Triggering Event:</td><td>Admin wants to Insert a Role</td></tr>
<tr><td>Brief Description</td><td>When an Admin wants to Insert a Role they go to the "Admin Tools" page, inputs the name of the Role, and clicks on the "Insert" button</td></tr>
<tr><td>Actors:</td><td>Admin or Instructor</td></tr>
<tr><td>Related Use Case:</td><td>None</td></tr>
<tr><td>Stakeholders:</td><td>Admin, Instructors, and Students</td></tr>
<tr><td>Preconditions:</td><td>User must have a Role of Admin</td></tr>
<tr><td>Post conditions:</td><td>None</td></tr>
<tr><td>Flow of Events</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Admin goes to the Admin Tools page.</td><td>1.1 Displays a list of all Roles</td></tr>
<tr><td>2. Admin inputs the Role name and RoleID they wish to insert.</td><td></td></tr>
<tr><td>3. Admin clicks Insert.</td><td>3.1 Insert the Role with the inputted Role name, RoleID and as active</td></tr>
</table>
</td></tr>
</table>

Exception Conditions:	

Use Case Name:	Update a Role
Scenario:	User wants to Update a Role
Triggering Event:	Admin wants to Update a Role
Brief Description	When a Admin wants to Update a Question they go to the "Admin Tools" page and clicks "Edit" next to the Role they wish to update.
Actors:	Admin
Related Use Case:	None
Stakeholders:	Admin, Instructors, and Students
Preconditions:	Role must exist, User must have Admin Role
Post conditions:	None
Flow of Events	
Exception Conditions:	

Actor	System
1. Actor goes to the Admin Tools page.	1.1 Displays a list of all Roles
2. Actor selects "Edit"	2.1 Display Role in Edit Mode
3. Actor Selects "Update"	3.1 Check that there is a description. 3.2 Update the Role
4. Actor Selects "Cancel"	4.1 Close Question in edit mode and display questions

Appendix C - Database Design



Appendix D - Textboxt Exteded Class Reference

Asp:TextBox Control Extended

To use this control you will need

- TextboxControl.vb (class attachted below) in App_Code.
- Register Tag on the aspx page

Current uses of the Textbox Control

- All Properties from ASP Textbox Control
- Adds Required field validator
- Adds RegExpression validator for:
    - Date
        - Includes javascript for dashes (attached below)
    - Email
    - Phone
- Adds a compare validator for:
    - Integer
- Adds a CalendarExtender

Added Properties

- RequiredFieldValidator
    - IsRequired (Boolean)
    - RequiredMessage (optional – has default)
- DateValidator
    - IsDate (Boolean)
    - DateValidationMessage (optional – has default)
- IntegerValidator
    - IsInteger (Boolean)
    - IntegerValidationMessage (optional – has default)
- EmailValidator
    - IsEmail (Boolean)
    - EmailValidationMessage (optional – has default)
- PhoneValidator
    - IsPhone (Boolean)
    - PhoneValidationMessage (optional – has default)
- AJAX Calendar Extendar
    - IsCalendar (Boolean)
    - CalendarPopupButtonID (optional)

Examples

Example Register Tag

```
<%@ Register TagPrefix="ASM" Namespace="ASMTextboxControl" %>
```

Example Required field validator

```
<ASM:ASMTextboxControl ID="txtRoleDescInsert" runat="server" Text=""
IsRequired="true" ValidationGroup="ValidRoleInsert" />
```

Example Date With CalendarExtendar (*Note to use ajax there must be a scriptmanager on page)

```
<ASM:ASMTextboxControl ID="txtDate" runat="server" Text="" IsDate="true"
IsCalendar="true" />
```

Example Email

```
<ASM:ASMTextboxControl ID="txtEmailInsert" runat="server" Text=""
IsRequired="true" IsEmail="true" ValidationGroup="AdminUserInsert" />
```

Example Phone

```
<ASM:ASMTextboxControl ID="txtPhoneNumberInsert" runat="server" Text=""
Width="100px" IsPhone="true" ValidationGroup="AdminUserInsert" />
```

Example Integer

```
<ASM:ASMTextboxControl ID="txtRoleIDInsert" runat="server" Text=""
IsRequired="true" IsInteger="true" ValidationGroup="ValidRoleInsert" />
```

Class

```
Imports Microsoft.VisualBasic

Namespace ASMTextboxControl
    Public Class ASMTextboxControl

#Region "Properties"
        Inherits TextBox

#Region "RequiredFieldValidator"
        '/////////////////////////////////////////////////
        '' This will add a required field validator to
        '' the textbox
        '/////////////////////////////////////////////////
        Private _rfv As RequiredFieldValidator
        'set default value of properties
        Private _IsRequired As Boolean = False
        Private _RequiredMessage As String = ""

        Public Property IsRequired As Boolean
            Get
                Return _IsRequired
            End Get
            Set(ByVal value As Boolean)
                _IsRequired = value
            End Set
        End Property
        Public Property RequiredMessage As String
            Get
                Return _RequiredMessage
            End Get
            Set(ByVal value As String)
                _RequiredMessage = value
            End Set
        End Property

        Private Sub CheckIfRequired()
            If Me.IsRequired = True Then
                'create new instance of RequiredFieldValidator
                _rfv = New RequiredFieldValidator()

                'Set properties of RequiredFieldValidator
                _rfv.ControlToValidate = Me.ID
                _rfv.ForeColor = Drawing.Color.Red
```

```
                _rfv.EnableClientScript = "true"
                _rfv.Display = ValidatorDisplay.Dynamic
                _rfv.SetFocusOnError = True
                _rfv.ValidationGroup = Me.ValidationGroup

                'check if message was set
                If Me.RequiredMessage Is Nothing Or Me.RequiredMessage = "" Then
                    _rfv.ErrorMessage = "Required"
                Else
                    _rfv.ErrorMessage = Me.RequiredMessage
                End If

                'add control to render
                Controls.Add(_rfv)
            End If
        End Sub
#End Region
#Region "DateValidator"
        '////////////////////////////////////////////////////
        '' This will add a regular expression validator to
        '' to check for a date format (DD/MM/YYYY)
        '////////////////////////////////////////////////////
        Private _rev As RegularExpressionValidator
        'set default value of properties
        Private _IsDate As Boolean = False
        Private _DateValidationMessage As String = ""

        Public Property IsDate As Boolean
            Get
                Return _IsDate
            End Get
            Set(ByVal value As Boolean)
                _IsDate = value
            End Set
        End Property
        Public Property DateValidationMessage As String
            Get
                Return _DateValidationMessage
            End Get
            Set(ByVal value As String)
                _DateValidationMessage = value
            End Set
        End Property

        Private Sub CheckIfDate()
            If Me.IsDate = True Then
                'create new instance of RegularExpressionValidator
                _rev = New RegularExpressionValidator

                'Set properties of RegularExpressionValidator
                _rev.ControlToValidate = Me.ID
                _rev.ForeColor = Drawing.Color.Red
                _rev.EnableClientScript = "true"
                _rev.Display = ValidatorDisplay.Dynamic
                _rev.SetFocusOnError = True
                _rev.ValidationExpression = "((^(10|12|0?[13578])([/])(3[01]|[12][0-9]|0?[1-9])([/])((1[8-9]\d{2})|([2-
9]\d{3}))$)|(^(11|0?[469])([/])(30|[12][0-9]|0?[1-9])([/])((1[8-9]\d{2})|([2-9]\d{3}))$)|(^(0?2)([/])(2[0-8]|1[0-9]|0?[1-
9])([/])((1[8-9]\d{2})|([2-
9]\d{3}))$)|(^(0?2)([/])(29)([/])([2468][048]00)$)|(^(0?2)([/])(29)([/])([3579][26]00)$)|(^(0?2)([/])(29)([/])([1][89][0][48])$)|(
^(0?2)([/])(29)([/])([2-9][0-9][0][48])$)|(^(0?2)([/])(29)([/])([1][89][2468][048])$)|(^(0?2)([/])(29)([/])([2-9][0-
9][2468][048])$)|(^(0?2)([/])(29)([/])([1][89][13579][26])$)|(^(0?2)([/])(29)([/])([2-9][0-9][13579][26])$))"
                _rev.ValidationGroup = Me.ValidationGroup

                'check if message was set
                If Me.DateValidationMessage Is Nothing Or Me.DateValidationMessage = " " Then
                    _rev.ErrorMessage = "(DD/MM/YYYY)"
                Else
                    _rev.ErrorMessage = Me.DateValidationMessage
                End If

                'add control to render
                Controls.Add(_rev)
            End If

        End Sub
#End Region
#Region "IntegerValidator"
        '////////////////////////////////////////////////////
        '' This will add a compare validator to the textbox
        '' to check for integer
        '////////////////////////////////////////////////////
        Private _cv As CompareValidator
        'set default value of properties
        Private _IsInteger As Boolean = False
        Private _IntegerValidationMessage As String = ""

        Public Property IsInteger As Boolean
            Get
                Return _IsInteger
            End Get
            Set(ByVal value As Boolean)
                _IsInteger = value
```

```
            End Set
        End Property
        Public Property IntegerValidationMessage As String
            Get
                Return _IntegerValidationMessage
            End Get
            Set(ByVal value As String)
                _IntegerValidationMessage = value
            End Set
        End Property

        Private Sub CheckIfInteger()
            'create new instance of CompareValidator
            _cv = New CompareValidator()

            'Set properties of CompareValidator
            _cv.ControlToValidate = Me.ID
            _cv.ForeColor = Drawing.Color.Red
            _cv.EnableClientScript = "true"
            _cv.Display = ValidatorDisplay.Dynamic
            _cv.SetFocusOnError = True
            _cv.Type = ValidationDataType.Integer
            _cv.Operator = ValidationCompareOperator.DataTypeCheck
            _cv.ValidationGroup = Me.ValidationGroup

            'check if message was set
            If Me.IntegerValidationMessage Is Nothing Or Me.IntegerValidationMessage = "" Then
                _cv.ErrorMessage = "Whole Numbers Only"
            Else
                _cv.ErrorMessage = IntegerValidationMessage
            End If

            'add control to render
            Controls.Add(_cv)
        End Sub
#End Region
#Region "EmailValidator"
        '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        '' This will add a regular expression validator to
        '' to check for a email format (xxx@xxx.xxx)
        '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Private _revEmail As RegularExpressionValidator
        'set default value of properties
        Private _IsEmail As Boolean = False
        Private _EmailValidationMessage As String = ""

        Public Property IsEmail As Boolean
            Get
                Return _IsEmail
            End Get
            Set(ByVal value As Boolean)
                _IsEmail = value
            End Set
        End Property
        Public Property EmailValidationMessage As String
            Get
                Return _EmailValidationMessage
            End Get
            Set(ByVal value As String)
                _EmailValidationMessage = value
            End Set
        End Property

        Private Sub CheckIfEmail()
            If Me.IsEmail = True Then
                'create new instance of RegularExpressionValidator
                _revEmail = New RegularExpressionValidator

                'Set properties of CompareValidator
                _revEmail.ControlToValidate = Me.ID
                _revEmail.ForeColor = Drawing.Color.Red
                _revEmail.EnableClientScript = "true"
                _revEmail.Display = ValidatorDisplay.Dynamic
                _revEmail.SetFocusOnError = True
                _revEmail.ValidationExpression = "\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"
                _revEmail.ValidationGroup = Me.ValidationGroup

                'check if message was set
                If Me.EmailValidationMessage Is Nothing Or Me.EmailValidationMessage = "" Then
                    _revEmail.ErrorMessage = "(xxx@xxx.xxx)"
                Else
                    _revEmail.ErrorMessage = Me.EmailValidationMessage
                End If

                'add control to render
                Controls.Add(_revEmail)
            End If

        End Sub
#End Region
#Region "PhoneValidator"
        '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
        '' This will add a regular expression validator to
        '' to check for a Phone format (XXX-XXX-XXXX)
        '' and add javascript to mask the textbox
        '////////////////////////////////////////////////////
        Private _revPhone As RegularExpressionValidator
        'set default value of properties
        Private _IsPhone As Boolean = False
        Private _PhoneValidationMessage As String = ""

        Public Property IsPhone As Boolean
            Get
                Return _IsPhone
            End Get
            Set(ByVal value As Boolean)
                _IsPhone = value
            End Set
        End Property
        Public Property PhoneValidationMessage As String
            Get
                Return _PhoneValidationMessage
            End Get
            Set(ByVal value As String)
                _PhoneValidationMessage = value
            End Set
        End Property

        Private Sub CheckIfPhone()
            If Me.IsPhone = True Then
                'Set javascript mask for textbox
                Me.Attributes.Add("onKeyUp", "javascript:AutoPhone(this)")

                'Set properties of textbox
                Me.MaxLength = "12"

                'create new instance of RegularExpressionValidator
                _revPhone = New RegularExpressionValidator

                'Set properties of CompareValidator
                _revPhone.ControlToValidate = Me.ID
                _revPhone.ForeColor = Drawing.Color.Red
                _revPhone.EnableClientScript = "true"
                _revPhone.Display = ValidatorDisplay.Dynamic
                _revPhone.SetFocusOnError = True
                _revPhone.ValidationExpression = "\d{3}-\d{3}-\d{4}"
                _revPhone.ValidationGroup = Me.ValidationGroup

                'check if message was set
                If Me.PhoneValidationMessage Is Nothing Or Me.PhoneValidationMessage = "" Then
                    _revPhone.ErrorMessage = "(XXX-XXX-XXXX)"
                Else
                    _revPhone.ErrorMessage = Me.PhoneValidationMessage
                End If

                'add control to render
                Controls.Add(_revPhone)
            End If

        End Sub
    #End Region
    #Region "AJAX Calendar Extendar"
        '////////////////////////////////////////////////////
        '' This will add a AJAAX calendar Extender to
        '' the textbox
        '////////////////////////////////////////////////////
        Private _cal As AjaxControlToolkit.CalendarExtender
        'set default value of property
        Private _IsCalendar As Boolean = False
        Private _CalendarPopupButtonID As String = ""

        Public Property IsCalendar As Boolean
            Get
                Return _IsCalendar
            End Get
            Set(ByVal value As Boolean)
                _IsCalendar = value
            End Set
        End Property
        Public Property CalendarPopupButtonID As String
            Get
                Return _CalendarPopupButtonID
            End Get
            Set(ByVal value As String)
                _CalendarPopupButtonID = value
            End Set
        End Property
        Private Sub CheckIfCalendar()
            If Me.IsCalendar = True Then
                'create new instance of AjaxControlToolkit.CalendarExtender
                _cal = New AjaxControlToolkit.CalendarExtender()

                'Set properties of CompareValidator
                _cal.TargetControlID = Me.ID
```

```
                _cal.Enabled = True

                'check if PopupButtonID was set
                If Me.CalendarPopupButtonID Is Nothing Or Me.CalendarPopupButtonID = "" Then
                    '
                Else
                    _cal.PopupButtonID = Me.CalendarPopupButtonID
                End If

                'add control to render
                Controls.Add(_cal)
            End If
        End Sub

#End Region

#End Region

#Region "PreRender - Render"
        Protected Overrides Sub OnPreRender(ByVal e As System.EventArgs)
            'PreRender you check if controls are nessessary and build
            MyBase.OnPreRender(e)
            'Check Scenarios
            CheckIfRequired()
            CheckIfDate()
            CheckIfInteger()
            CheckIfEmail()
            CheckIfPhone()
            '/////////////////
            '' AJAXToolKit
            '/////////////////
            CheckIfCalendar()

        End Sub

        Protected Overrides Sub Render(ByVal w As HtmlTextWriter)
            'Render you check if controls are nessessary and render
            If Not Me.DesignMode Then
                'Render if not in Design Mode - So the control will render in Design view with out error
                MyBase.Render(w)
                If IsRequired = True Then
                    _rfv.RenderControl(w)
                End If
                If IsDate = True Then
                    _rev.RenderControl(w)
                End If
                If IsInteger = True Then
                    _cv.RenderControl(w)
                End If
                If IsEmail = True Then
                    _revEmail.RenderControl(w)
                End If
                If IsPhone = True Then
                    _revPhone.RenderControl(w)
                End If

                '/////////////////
                '' AJAXToolKit
                '/////////////////
                If IsCalendar = True Then
                    _cal.RenderControl(w)
                End If


            End If
        End Sub
#End Region

    End Class
End Namespace
```

JavaScript – js/JScript.js (*Note code points to folder js, item JScript.js)

```
// Formats the text as the following phone format:
// ddd-ddd-dddd
function AutoPhone(e) {
    var dl = e.value.length
    switch (dl) {
        case 3:
            e.value += "-"
            break

        case 7:
            e.value += "-"
            break
    }
}
```

Appendix E - StudentQuiz Class Reference

```
Imports Microsoft.VisualBasic
Imports System.Data

Public Class StudentQuiz

#Region "Local Varaibles"
    Dim _QuizID As Integer
    Dim _ClassID As Integer
    Dim _QuizName As String
    Dim _MasterQuizQuestionList As New ArrayList
    Dim _QuizQuestionList As New ArrayList
    Dim _QNumber As Integer
    Dim _QRight As Integer
    Dim _CurIndex As Integer
#End Region

#Region "Constructors"

    Sub New()
        'this sets the initial values for a new instance of the object
        _QuizID = 0
        _ClassID = 0
        _QuizName = ""
        _QNumber = 0
        _QRight = 0
        _CurIndex = 0

    End Sub
    Sub New(ByVal QuizID As Integer)
        'this sets the initial values for a new instance of the object

        'This sets the quiz info
        Dim quizAdapt As New mssqlDataSetTableAdapters.uspStuQuizSelectByIDTableAdapter
        Dim quizTbl As mssqlDataSet.uspStuQuizSelectByIDDataTable = quizAdapt.GetData(QuizID)
        Dim quizRow As mssqlDataSet.uspStuQuizSelectByIDRow = quizTbl.Rows(0)

        _QuizID = quizRow.QuizID
        _ClassID = quizRow.ClassID
        _QuizName = quizRow.QuizName
        _QNumber = 0
        _QRight = 0
        _CurIndex = 0

        'this get the question info
        Dim questAdapt As New mssqlDataSetTableAdapters.StuQuizQuestionIDListTableAdapter
        Dim questTbl As mssqlDataSet.StuQuizQuestionIDListDataTable = questAdapt.GetData(quizRow.QuizID)
        Dim questRow As mssqlDataSet.StuQuizQuestionIDListRow

        'Loop through questions and save in list
        For Each questRow In questTbl
            Dim myQuest As New StudentQuestion(questRow.QuestionID, questRow.QuizQuestionID)

            'Set both lists
            _MasterQuizQuestionList.Add(myQuest)
            _QuizQuestionList.Add(myQuest)
        Next

    End Sub

#End Region


#Region "Functions"

    Public Function GetQuestion() As StudentQuestion
        Dim retQ As StudentQuestion = Nothing

        If _QuizQuestionList.Count = 0 Then
            'There are no more questions
            retQ = Nothing

        ElseIf _QuizQuestionList.Count = 1 Then
            'return the only question left
            retQ = _QuizQuestionList.Item(0)

        Else
            'Get current Index
            retQ = _QuizQuestionList.Item(_CurIndex)
        End If

        Return retQ
    End Function

    Public Function SkipQuestion(ByVal FwdDirection As Boolean) As StudentQuestion
        Dim retQ As StudentQuestion = Nothing

        If FwdDirection = True Then
```

```
            'Check if it is the last question
            If _CurIndex + 1 = _QuizQuestionList.Count Then

                'Set current index to begining
                _CurIndex = 0
            Else

                'Set current index to Next
                _CurIndex = _CurIndex + 1
            End If

        Else
            'Check if it is the first question
            If _CurIndex = 0 Then

                'Set current index to last
                _CurIndex = (_QuizQuestionList.Count - 1)

            Else

                'set index as prev
                _CurIndex = _CurIndex - 1

            End If
        End If

        'return next question
        Try
            retQ = GetQuestion()
        Catch ex As Exception

        End Try

        Return retQ
    End Function

    Public Function AnswQuestion(ByVal StuId As Integer, ByVal QuizQuestID As Integer, AnswerTable As DataTable) As
StudentQuestion
        Dim retQ As StudentQuestion = Nothing

        'set adapter
        Dim query As New mssqlDataSetTableAdapters.QueriesTableAdapter

        'Create Student response
        Dim studentResponseID As Integer = query.uspStuResponse(QuizQuestID, StuId)

        'Loop through and Insert Answers
        Dim dr As DataRow

        For Each dr In AnswerTable.Rows
            query.uspStuRespAnsInsert(studentResponseID, dr.Item("AnswerID"), dr.Item("IsCorrect"))

            'If One is correct mark answer as correct
            If dr.Item("IsCorrect") = True Then
                _QRight = _QRight + 1
            End If
        Next

        'Set Question Number
        _QNumber = _QNumber + 1

        'Check if answered the last question and set curent index to begining
        'Check if it is the last question
        If _CurIndex + 1 = _QuizQuestionList.Count Then

            'remove current question from list
            _QuizQuestionList.RemoveAt(_CurIndex)

            'Set current index to begining
            _CurIndex = 0

        Else
            'remove current question from list
            _QuizQuestionList.RemoveAt(_CurIndex)
        End If

        'return next question
        Try
            retQ = GetQuestion()
        Catch ex As Exception

        End Try


        Return retQ
    End Function
#End Region

#Region "Properties"

    Public ReadOnly Property MasterQuizQuestionList() As ArrayList
```

```
        Get
            Return _MasterQuizQuestionList
        End Get
    End Property

    Public ReadOnly Property QuizQuestionList() As ArrayList
        Get
            Return _QuizQuestionList
        End Get
    End Property

    Public ReadOnly Property QuizID() As Integer
        Get
            Return _QuizID
        End Get
    End Property

    Public Property QNumber() As Integer
        Get
            Return _QNumber
        End Get
        Set(ByVal value As Integer)
            _QNumber = value
        End Set
    End Property

    Public Property QRight() As Integer
        Get
            Return _QRight
        End Get
        Set(ByVal value As Integer)
            _QRight = value
        End Set
    End Property

#End Region


End Class
```

Appendix E - StudentQuestion Class Reference

```
Imports Microsoft.VisualBasic
Imports System.Data

Public Class StudentQuestion

#Region "Local Varaibles"
    Dim _QuizQuestionID As Integer
    Dim _QuestionID As Integer
    Dim _QText As String
    Dim _QHelp As String
    Dim _AnswerTable As DataTable
#End Region

#Region "Constructors"
    Sub New()

        'this sets the initial values for a new instance of the object
        _QuizQuestionID = 0
        _QuestionID = 0
        _QText = ""
        _QHelp = ""

    End Sub


    Sub New(QuestionID As Integer, QuizQuestionID As Integer)

        'this get the question info
        Dim questAdapt As New mssqlDataSetTableAdapters.uspQuestionSelectByIDTableAdapter
        Dim questTbl As mssqlDataSet.uspQuestionSelectByIDDataTable = questAdapt.GetData(QuestionID)
        Dim questRow As mssqlDataSet.uspQuestionSelectByIDRow = questTbl.Rows(0)

        _QuizQuestionID = QuizQuestionID
        _QuestionID = QuestionID
        _QText = questRow.QuestionText
        _QHelp = questRow.HelpText


        'create table
        Dim dt As New DataTable
        dt.Columns.Add("AnswerID", GetType(Integer))
        dt.Columns.Add("AnswerText", GetType(String))
        dt.Columns.Add("IsCorrect", GetType(Boolean))

        'get answers
        Dim ansAdapt As New mssqlDataSetTableAdapters.AnswersByQuestionIDTableAdapter
        Dim ansTbl As mssqlDataSet.AnswersByQuestionIDDataTable = ansAdapt.GetData(questRow.QuestionID)
        Dim ansRow As mssqlDataSet.AnswersByQuestionIDRow

        For Each ansRow In ansTbl

            Dim dr As DataRow = dt.NewRow
            dr.Item("AnswerID") = ansRow.AnswerID
            dr.Item("AnswerText") = ansRow.AnswerText
            dr.Item("IsCorrect") = ansRow.IsCorrect
            dt.Rows.Add(dr)
        Next

        _AnswerTable = dt

    End Sub

#End Region

#Region "Properties"

    Public Property QuizQuestionID() As Integer
        Get
            Return _QuizQuestionID
        End Get
        Set(ByVal value As Integer)
            _QuizQuestionID = value
        End Set
    End Property


    Public Property QuestionID() As Integer
        Get
            Return _QuestionID
        End Get
        Set(ByVal value As Integer)
            _QuestionID = value
        End Set
    End Property

    Public Property QText() As String
        Get
```

```
            Return _QText
        End Get
        Set(ByVal value As String)
            _QText = value
        End Set
    End Property

    Public Property QHelp() As String
        Get
            Return _QHelp
        End Get
        Set(ByVal value As String)
            _QHelp = value
        End Set
    End Property

    Public Property AnswerTable() As DataTable
        Get
            Return _AnswerTable
        End Get
        Set(ByVal value As DataTable)
            _AnswerTable = value
        End Set
    End Property
#End Region

End Class
```