University of Leicester
Department of Computer Science
CO3016 Computing Project
Dissertation

Project Title: Multiple Choice Question and Answer Web App

Author: Jack Charlesworth

Supervisor: Roy L. Crole

Date of Submission: May 2015

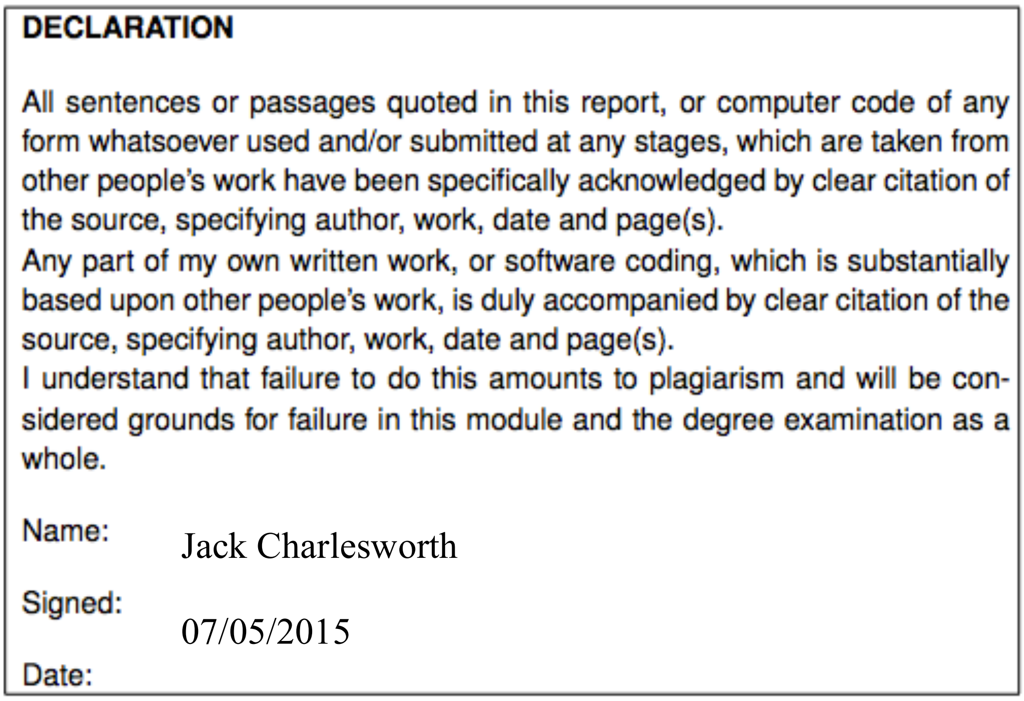
Contents

Abstract

This reports provides a detailed account of Jack Charlesworth's development of Quiz Master (question and answer web application) software system.
There was a problem area identified in higher education institutions, paper based tests were becoming more and more obsolete in that they offer little flexibility for lecturers when building test papers as well as being timely and costly to produce. Completed test scripts are sometimes misplaced, leading to negative implications for both students and teaching staff within the department. Higher education teaching institutions are in need of a simple to use, yet rich in features web application that allows students to sit multiple choice tests to which academic data can be retrieved from.

To solve this problem, a web application that allows lecturers greater creative control over how quizzes are created, maintained and set for students was created.

Therefore the goal of this project was to assist in the creation of test based coursework within teaching institutions by building a highly usable system offering lectures a number of benefits they would not receive by setting pen and paper bases tests. The project calls for a system that could be rolled out across a number of institutions and within departments, calling for a system that is both reliable and accessible to the masses.



DECLARATION

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s).
Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s).
I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Jack Charlesworth

Signed:

07/05/2015

Date:

Introduction

This document will provide a detailed account of the Quiz Master question and answer web application project, detailing the construction process as well as a critical evaluation and reflection of the project as a whole. Outlining the problem that has been solved and the effectiveness of this solution.

Semester one focused on the planning and designing of the software system, eliciting requirements for the project in line with the needs of the target audience. Once drawn up, construction began leading up to an initial prototype demonstration in December for which to build upon in Semester two.

Semester two carried on from where the previous semester left, the second phase of construction building upon the features implemented already and working on the more advanced and timely features of the system.

Aims

The aim of this project is to research the requirements needed in order to build a web application that supports the creation and maintenance of question and answer tests aiding module lecturers and conveners within teaching institutions.
There is a need for a software application that supports modality of users, both lecturer and student to be used in higher education teaching institutions whereby the method of using paper based tests is becoming more obsolete by the year. In practice the system must be accessible and usable, drawing the institution away from the use of redundant paper based tests.
The problem being that the options of similar software available to lecturers is limited in terms of test construction as well as a lack of gaining meaningful analytical data from completed tests. The question and answer web application will allow module conveners the ability to create, edit and set tests for students enrolled on a module which they run, similarly it will allow students of a module to play through the quizzes they have been set by their lecturers as part of the coursework component for a module they are taking.

Objectives

- One of the first objectives during the initial phases was to research and collect data on the most suitable technologies that could be utilized for implementing such a project.

- To research current software solutions in the marketplace, critically evaluating them to find areas where they lack and areas where they perform badly.

- To make informed design decisions on how the QuizMaster system will be implemented, designing the top-level structure and how to allocate and prioritise time as well as how to best modularise the system into tiers.

- System functionality will be split into appropriate classes and directories to allow for accessible maintenance whilst creating seamless integration between technologies, components and external libraries used.

- To evaluate the delivered quiz application as a tool within a learning environment both quantitatively and qualitatively to understand its value and limitations.

Continuing on from top-level design decisions, we must consider factors such as scalability, portability and maintainability. Design choices to build upon a client server style whereby a majority of the work is done on the server side, thus allowing for secure computing, higher performance and superior maintainability.

To ensure the technologies and design decisions made now will allow for QuizMaster to expand should the number of users and participating teaching institutions grow. Decisions made at this stage should not cause restrictions in the future life of the system in terms of expanding user base and the load of quiz data the system can handle.

When moving on the implementation phase it was a case of producing and often trialing prototypes of system branches gathering ideas on what worked and more importantly what didn't have a place in the final system.

During the project I will follow agile methodologies, this choice offers flexibility and allows for requirements to evolve yet still keeping the timescale fixed. Testing is integrated throughout the project lifecycle meaning errors are often dealt with along the way.
Other focuses and advantages of following agile methodologies include:

- Continuous attention to technical excellence and good design
- Working software prototypes are delivered frequently
- The allowance given for regular adaptation to changing circumstances

Requirements

In this section I will present the system requirements, I will make use of the Volere requirements specification template as well as a number of templates learnt from CO2006 Software Engineering and System Development. We will also use the MoSCoW requirements prioritisation method in an attempt to understand the importance placed on each requirement by stakeholders.

Purpose of the System

Purpose: To produce a question and answer web application that allows lectures to produce quizzes and allows students to sit these quizzes.
Advantage: Provide a platform for lecturers and module conveyers to create quizzes for their students, offering lecturers greater control and flexibility over paper based quizzes.
Measurement: The success of the system can be measured through participation, if within a University department more than 50% of all class tests are performed using the Quiz Master system then it can be deemed a valuable tool for the department.
Reasonable: If the advantage of having the system available is greater than the design, implementation and construction effort

Glossary

Student: A user enrolled onto a module at the teaching institution, these are the ones who will be playing the set quizzes.

Lecturer: A teaching member of a department within an institution, they will be running the module and creating the quizzes to be set. Throughout this document and software system lecturer, instructor and convener are referring to the same.

Score: In the context of this system a score is the final cumulative mark achieved at the end of a successful quiz submission, this is essentially the users grade.

Result: In the context of this system a result refers to the mark achieved by a student on an individual question, it logs their posted answer against the correct answer for that question.

Quiz: Throughout this document a quiz refers to a short examination of a person's proficiency in a certain topic area. In context of the software application, this is the part of the system whereby students are tested on their ability by being required to give their answers to a number of questions. Throughout this document quiz and test both refer to the same meaning described above.

Background and Experience of Users

User group: University lecturers and students
Technical experience: Novice to master

Technical experience throughout the University will differ, it is assumed that users will have a basic knowledge of how to access and navigate to a web application.

Description of the Problem Domain

This section will present and explain the context of the system as well as the environment in which the system is intended for.

A number of departments or lecturers within Universities choose to set class tests as part of the coursework component for a module which they oversee.
The majority of lecturers at the University of Leicester set class tests to be completed on paper, this comes with a number of drawbacks:

Inaccessibility: Administering paper-based evaluations in-class may exclude certain students from participating. Students who are absent from class, or who have impairments, may be at a disadvantage.

Limited Flexibility: Using traditional paper-based assessments often does not allow for the customization of questions. The questions used may be standard, university set queries without instructors being able to add tailored course-specific questions. Using a paper-based process also doesn't afford much flexibility in terms of the reports that can be generated for instructors.

Lengthy Process: The process of administering paper-based assessments is a lengthy one. It can take several months to complete the cycle from administering the forms to collecting and analysing the results, to sharing reports and acting on the feedback obtained.

High Cost: A major disadvantage of using paper assessments is the high cost associated with the process. The number of personnel involved as well as the printing, distributing, scanning, rekeying, filing and archiving can be very costly.

Project Constraints

General Constraints

The QuizMaster system must be user friendly, users should not be required to know any of the technical workings of the system. After initial set up within an institution, users will not require any formal training to use the system to its full potential. The system will be error preventing thus reducing the chances of a fatal error occurring. Should an error occur, the system is designed to recover by taking a step back and reversing the action that caused the error.

Time Constraints

The deadline of completion of the software system is 7th May 2015.

Environmental Constraints

The system is accessible from a web browser, the application logic and database all resides on the server side making the system widely available regardless of the processing power of users PC.

Hardware Environment

The PC of the user must be able to connect to the Internet and be able to render the systems graphical user interface smoothly.

Software Environment

The system is developed primarily in PHP but also making use of JavaScript, MySQL and HTML. The server will be running the database server containing all relevant tables and data.

Functional Requirements

In this section I will formalise the requirements of the project.

The Scope of the Work

The current situation: Lecturers currently set class tests to be completed on paper or use an existing question and answer tool that is very basic in features.

The context of the work: University module, tests to be completed as part of the coursework component of a module that is assessed towards their final grade.

The Scope of the Product

Product Boundary: The Use Case Diagram found below is a representation of a users interaction with the Quiz Master system, showing the relationship between the users and the different use cases in which the user is involved.

Functional and Data Requirements

Functional Requirement: Lecturers have the ability to create, edit, delete and maintain quizzes.
Rationale: Users that are lecturers are able to create a library of quizzes that are stored for publishing or editing at a later date, allowing lecturers the ability to reuse their created data and quiz components. Creating a quiz populated with questions to be played by students.
Fit criterion: Upon successful creation of their profile as a lecturer they are given the capability to begin creating quizzes using the appropriate navigation tools and text fields, this data is then stored in the appropriate tables in the database.

Functional Requirement: Lecturers are able to set a quiz for students to complete. This is true if the lecturer is the module convener for which the quiz belongs and the students they are setting the quiz for are enrolled on the module to which the quiz belongs.
Rationale: Provides lecturers the ability to notify students enrolled on their module that they have outstanding quizzes to complete.
Fit criterion: Once a lecturer has created a quiz they are able to set this quiz for their students to complete as part of their coursework component. When a quiz is created it is only viewable by the creator, a quiz is then playable by students when the lecturer chooses to set this quiz for specified students.

Functional Requirement: Students are able to play through a quiz they have been set to complete as part of the coursework component of a module they are enrolled on by their lecturer.
Rationale: Students enrolled onto a module that have a quiz set should be able to sit the quiz.
Fit criterion: When a student logs into the Quiz Master system they will be notified if they have been set a quiz to complete for a module they are enrolled on, they are then able to play through the set quiz.

Functional Requirement: Lecturers have the ability to create a number of question types to insert into their quiz.
Rationale: Lectures have at their disposal a greater number of question types in their toolkit to build their quizzes.
Fit criterion: When creating a question the lecturer is given the option to create either a true and false, multiple choice or blank answer question.

Functional Requirement: Lecturers have the ability to set a time allowance for a quiz.
Rationale: Giving lecturers more control over the quizzes they produce for students, restricting the time allowed for students to complete the quiz.
Fit criterion: When creating a quiz the lecturer is asked to input the length of time allowed for the quiz. When a student then logs in to play through this quiz they are allocated that amount of time to complete the quiz.

Functional Requirement: Lecturers have the ability to set a minimum mark that must be achieved by students in order to achieve a pass for a quiz.
Rationale: This feature gives lecturers more control over their quizzes, providing feedback in the form of whether a student passed of failed the quiz.
Fit criterion: When creating a quiz, the lecturer is prompted to input the percentage pass mark for that quiz.

Functional Requirement: The system will allow those that are eligible for extra time to be automatically given it.
Rationale: A number of the student body are eligible for extra time, this holds for all examinations and class tests whilst on their degree. This is done to conform to University policy regarding exam support arrangements.
Fit criterion: When adding a new student, they user is prompted to select a box if they are eligible for 20% extra time. If checked, the system will give this user an extra 20% time on all tests thy attempt.

Functional Requirement: Lecturers will receive the results of students that have taken their quiz, showing statistics as meaningful data.
Rationale: Lecturers can use this statistical data to view class averages and provide meaningful detailed feedback to the class.
Fit criterion: Once a student has completed a quiz, the lecturer who created that quiz can log into the results panel and view the whether that student passed of failed as well as the how many questions they got correct and their percentage score.

Functional Requirement: Once a student has completed a quiz they will receive instant feedback with results and score.
Rationale: This feature lets the student know how they faired on the quiz they just sat.
Fit criterion: Upon successful completion and submission of a quiz, a results page is shown to the user displaying their percentage score, how many questions they got correct and whether they passed or failed.

Functional Requirement: A lecturer has the ability to create new users into the system.
Rationale: When new lecturers join the department or new students enrol, an existing lecturer has the ability to register them onto the system.
Fit criterion: When an existing lecturer is logged into the Quiz Master system they can navigate to the users page to register new users onto the system.

Functional Requirement: A lecturer has the ability to manually sort the order in which questions of a quiz appear to the students when playing the quiz.
Rationale: A lecturer may wish to order questions by how difficult they deem them to be, or to have the questions appear in a specified order. This gives lecturers greater control over how their quizzes are displayed to students.
Fit criterion: When a quiz is created initially, questions have no order to them and are retrieved in a random order. A lecturer has the ability to sort the order of questions in a quiz they have created given the quiz have two or more questions. Upon order form submission each question is allocated the order in which it will appear according to the order in which the lecturer submitted the ordering form.

Functional Requirement: Students will be able to view a detailed breakdown and feedback of how they faired on the quiz after submission, displayed to them their answer compared to the correct answer for each question.
Rationale: The QuizMaster system is a tool for creating quizzes but also a learning tool, lecturers will use the system to help the students learn the module content and thus should be able to know where they went wrong.
Fit criterion: When viewing past quiz results students will be able to see on request a break down of their answers against the correct answers for that quiz.

MoSCoW Prioritisation

We will now prioritise the requirements using the MoSCoW method learnt during ‘CO3095 Software Measures and Quality Assurance’.

Must
(High Priority):

- Create Quiz
- Create Question
- View Results
- Play Quiz

Should
(Medium Priority):

- Set Quiz
- Edit Question
- Create Module
- Create User
- Set Quiz Pass mark
- Set Quiz Timer

Could
(Low Priority):

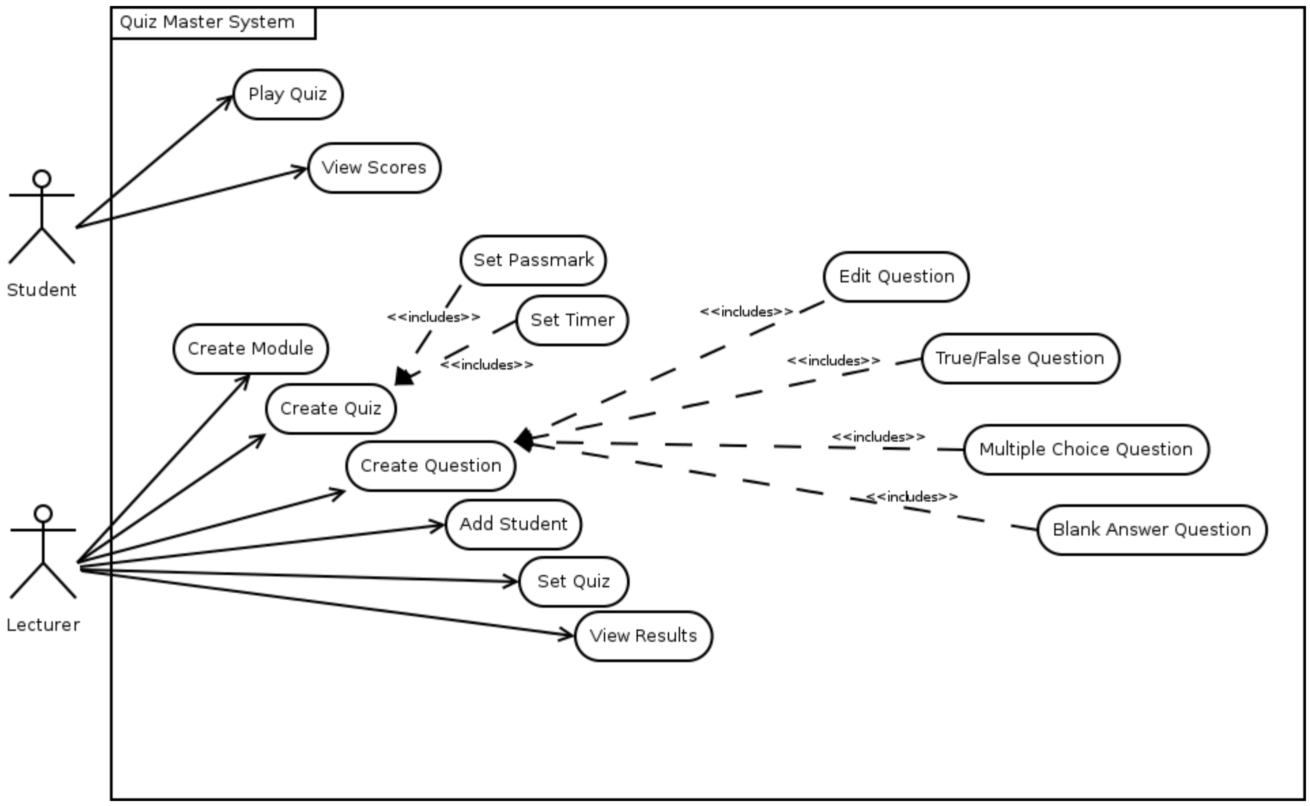
- Order Questions in a Quiz
- View Score Data in Graphical Form
- Allow for multiple correct answers to a quiz question
- Implement a time machine type backlog of past quizzes set for a module over the years, a type of question bank

Won’t
(Will not implement during this release, but may be included in future deliveries):

- Message notifications when a student has been set a quiz to complete
- Expanding create question to included a question type with more than one correct answers.

Product Functions

Use Case Diagram



Description of Use Cases

Name	***Create Quiz***
Primary Actor	Lecturer
Goal of the User	Add a quiz to the Quiz Master database
Precondition	Lecturer is successfully logged into the system and a quiz with the chosen name does not already exist
Post condition	Quiz is successfully added to the database in the appropriate tables
Flow of events	- User is logged in as a lecturer - Navigates to 'Add Quiz' page - Fills in details (Quiz name, quiz code, pass mark, timer) - Submits form - If all fields are correctly filled in, data is sent and stored in the database - New Quiz is created

Name	***Create Question***
Primary Actor	Lecturer
Goal of the User	Add a question to the Quiz Master database
Precondition	Lecturer is successfully logged into the system and all fields are submitted containing valid data
Post condition	The question is added to the database and is allocated to the chosen quiz
Flow of events	- User is logged in as a lecturer - Navigates to 'Add Question' page - Selects from the drop down menu the following: quiz subject, question type and if the question type is 'Multiple choice' the user is prompted to select the number of answer choices - Drop down menu form is submit - Question and answer fields are displayed according to the question type selected - User prompted to enter question, question mark and answers, specifying which answer is correct - Form is successfully submit, splitting the data up and stored in the database - User is notified of their successfully created question - Redirected to the 'Questions' page where the user can view their created question

Name	***Set Quiz***
Primary Actor	Lecturer
Goal of the User	Set a quiz for the students enrolled on the module the quiz is associated with
Precondition	Lecturer is successfully logged into the system and valid parameters are selected upon submission
Post condition	The quiz is set for the students enrolled onto that module, when a student from that module next logs in they will be notified they have an outstanding quiz to complete
Flow of events	- User is logged in as a lecturer - Navigates to 'Set Quiz' page - Lecturer is prompted to choose from drop down menus the following: quiz code, module code and the users they wish to set the quiz for (Users menu populated with users from the database enrolled on the specified module) - Form is successfully submit, user notified on their quiz being set

Name	***Edit Question***
Primary Actor	Lecturer
Goal of the User	Edit a question in pre-existing quiz
Precondition	Lecturer is successfully logged into the system and is attempting to edit a question of which they are the creator of the quiz it belongs
Post condition	Question is updated according to the sections edited, these can include the module the answers and the quiz for which the question belongs to
Flow of events	- User logged in as lecturer - Navigates to 'Question' page - Clicks 'edit' next to the question they wish to change - Current data on the chosen question is displayed and available to be edited including the quiz to which the question belongs, the question and the answers - Once finished editing, user clicks 'Save changes' - Database is updated accordingly - User notified that their question has been successfully edited

Name	***Order Questions in a Quiz***
Primary Actor	Lecturer
Goal of the User	To sort the order in which the questions of a quiz appear when sat
Precondition	Lecturer is successfully logged into the system and is attempting to order the questions of a quiz to which they are the creator
Post condition	Quiz question order is successfully updated according to the form submission
Flow of events	- User logged in as a lecturer - Navigates to 'Quizzes' page - Clicks the 'Order' button next to the quiz they wish to order the questions of - Questions within that quiz are displayed in their current order - Questions can be dragged and dropped to their new positions - Once finished, the form is submit - Question order for that quiz is updated in the database according to the state the user submit the form

Name	***View Results***
Primary Actor	Lecturer
Goal of the User	Quiz results of students are displayed
Precondition	Lecturer is successfully logged into the system and at least one student has completed a quiz whereby the lecturer is the quiz creator
Post condition	Quiz results and statistics are displayed to the lecturer for quizzes whereby the logged in lecturer is the quiz creator
Flow of events	- User is logged in as a lecturer - Navigates to 'Results' page - Result data from every student that has successfully submit one of the lecturers' quizzes is displayed, ordered by quiz code. - The following data is shown: username, module, correct marks, total marks and percentage achieved

Name	***Create Module***
Primary Actor	Lecturer
Goal of the User	Creation of a new module into the Quiz Master database
Precondition	Lecturer is successfully logged into the system and input fields are submitted with valid data
Post condition	A new module is created and stored in the appropriate tables in the database
Flow of events	- User is logged in as a lecturer - Navigates to 'Add Module' page - User prompted to enter the module name and module code - Form is submit notifying the user of the successful creation of the module - User is redirected to 'Modules' page where they can view their created modules

Name	***Play Quiz***
Primary Actor	Student
Goal of the User	To play through a quiz the student has been set
Precondition	Student is successfully logged into the system and they have at least one outstanding quiz they have been set to complete
Post condition	The relevant quiz data is retrieved from the database and displayed to the user on screen, the student is able to select their answers and play through the quiz
Flow of events	- User is logged in as a student - User has been set at least one quiz to complete by one of their module lecturers - Student chooses the quiz he wishes to play by the drop down menu which is populated with the quizzes he has been set - Once chosen, the quiz is started by clicking 'Start Quiz' - The page scripts retrieve question and answer data for the quiz, formatting the data in question order. The Quiz timer is retrieved and begins as soon as the quiz is started - Once the user has finished inputting their answers OR the timer reaches 0 the quiz form is submit

Name	***Create User***
Primary Actor	Lecturer
Goal of the User	Add a new user into the Quiz Master database
Precondition	User is successfully logged into the system as a lecturer
Post condition	The new user is successfully added to the database and their profile is created
Flow of events	- User logged in as a lecturer - Navigates to 'Add Student' page - User is prompted to enter information on the new student including username and the module to which they will be enrolled onto. Module data is retrieved from the database, populating a drop down menu with modules the currently logged in lecturer is the module convener for - Form is submit, user notified on the successful creation of the student - User redirected to 'Students' page

Name	***View Score Data in Graphical Form***
Primary Actor	Lecturer
Goal of the User	To view student score data displayed on a statistically beneficial graph
Precondition	Lecturer is successfully logged in and attempting to view graphical data for a quiz they created whereby at least one student has completed the quiz and has a saved score
Post condition	The system displays to the lecturer a graph showing all of the students who have submitted the quiz adjacent to their score. The scores are displayed in ascending order as to show a line of best fit
Flow of events	- User is logged in as a lecturer - Navigates to 'Results' page - Page is populated with existing module results - User clicks the 'Show' button next to a quiz code they wish to view graphical data for - A graph is drawn using database data, the graph shows every user that has successfully submit that quiz, ordered from lowest to highest score

Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. Look and feel is a term used in respect of a graphical user interface and comprises aspects of its design. It is important to be able to quantitatively measure the success of the implementation of a non-functional requirement.

Usability

Usability Requirement: The system will be easy and simple to use for lecturers from all departments.
Fit criteria: A focus group of lecturers shall be able to create a module, quiz and 3 questions within 8 minutes

Usability Requirement: Using the system should be the preferred choice over a lecturer setting paper based quiz.
Fit criteria: An anonymous survey shall show that more than 50% of lecturers prefer Quiz Master to a paper quiz.

Usability Requirement: The system will provide lecturers and departmental staff with valuable statistical data.
Fit criteria: QuizMaster will return meaningful data to lecturers that either could not be extracted from setting paper based tests or would be very timely to do in a paper based testing environment.

Learning Curve

Learning Curve Requirement: Lecturers and students will be able to use the Quiz Master system without any formal training.
Fit criteria: 80% of all first time users should be able to create a quiz and three questions within 10 minutes.

Accessibility

Accessibility Requirement: The system will be accessible by the whole student body, allowing those with disabilities to use the system
Fit criteria: If the system is to be used instead of paper-based tests, it must be accessible to all students within an institution.

Performance

Performance Requirement: The response time of the system should be of an acceptable time, in that users are not left frustrated.
Fit criteria: No response of the system will take longer than 3 seconds.

Availability

Availability Requirement: The Quiz Master system shall be available a large majority of the time.
Fit criteria: The system will have an uptime of at least 95%

Capacity

Capacity Requirement: The system will be able to deal with all University departments online at the same time.
Fit criteria: The system will be able to cater to 100 user requests simultaneously.

Operational

Operational Requirement: Students and lecturers shall use the Quiz Master system at home and on campus.

Operational Requirement: The system will function on all of the 5 most popular web browsers (Chrome, Firefox, IE, Safari and Opera).

Maintainability

Maintainability Requirement: The database will be easily updated and maintained by an administrator in times of change or need for recovery.

Supportability

Supportability Requirement: The system code will be clearly and concisely commented as to ensure new members of the development team understand the existing code.

Security

Security Requirement: All user accounts shall be password protected and passwords.

Security Requirement: Lecturer functionality such as quiz creation will only be an available action to users that are logged in as a lecturer.

Security Requirement: User information should be kept secure and passwords stored as a 160-bit hash of a salt concatenated to the password.

Cultural

Cultural Requirement: The Quiz Master system will not cause offence to religious and cultural groups

Legal

Legal Requirement: Information stored about users shall comply with the Data Protection Act.

Scalability

Scalability Requirement: The system must be able to handle increases in user load, as the reach of the system among institutions increases.

Specification/Data-Structures/ Design/Algorithms

Software Architecture

This section of the report justifies the architectural choices with use of supporting diagrams. All high-level design diagrams throughout this document have been produced in Dia, a UML diagram editor.

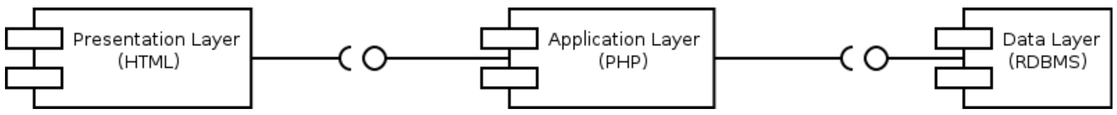
The Quiz Master system is primary built using PHP and MySQL technologies, whilst also making use of a number of other elements.

The web technologies that were implemented in the question and answer web application are the following:

- PHP
- HTML
- MySQL
- JavaScript
- AJAX

The Quiz Master system is built upon the general purpose server side scripting language PHP that runs the application logic and generates the web pages HTML code. The system is connected to a repository in the form of a MySQL relational database management system, implementing SQL statements to manipulate and edit the database.
Throughout the implementation of the system JavaScript has also been used, located on the client side, communicating asynchronously altering the webpages content that is displayed. Ajax has also been used, a client side technology creating an asynchronous web application with real time responsive features to create an intuitive user experience.

The component diagram below depicts how components are wired together to form the software system.



The Presentation layer: built using HTML provides the user interface, translating tasks and results to something the user can understand as useful information.

The Application layer: built in PHP coordinates the application processes, makes decisions and performs calculations as well as moving data between the two surrounding layers.

The Data layer: is where information is stored and retrieved from the database, here is where all quiz and user data is stored for retrieval and updating upon request of the user. This tier consists of the MySQL database management system.

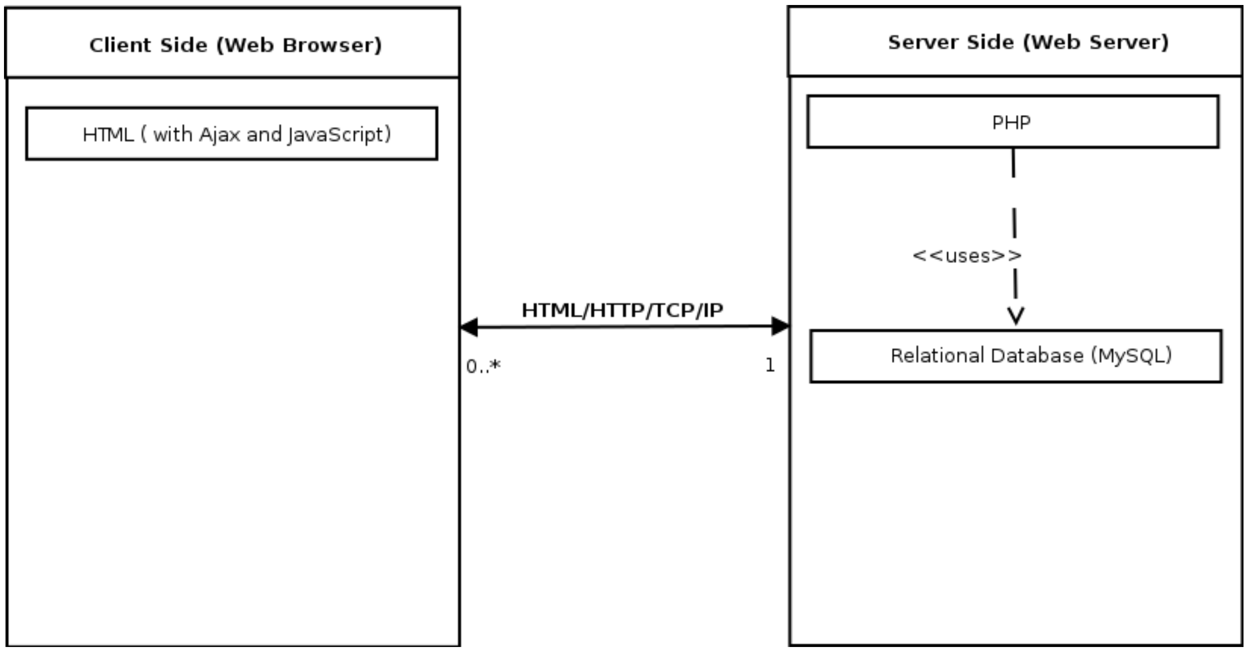

Client Side (Web Browser)
HTML (with Ajax and JavaScript)
Server Side (Web Server)
PHP
<<uses>>
Relational Database (MySQL)
HTML/HTTP/TCP/IP
0..*
1

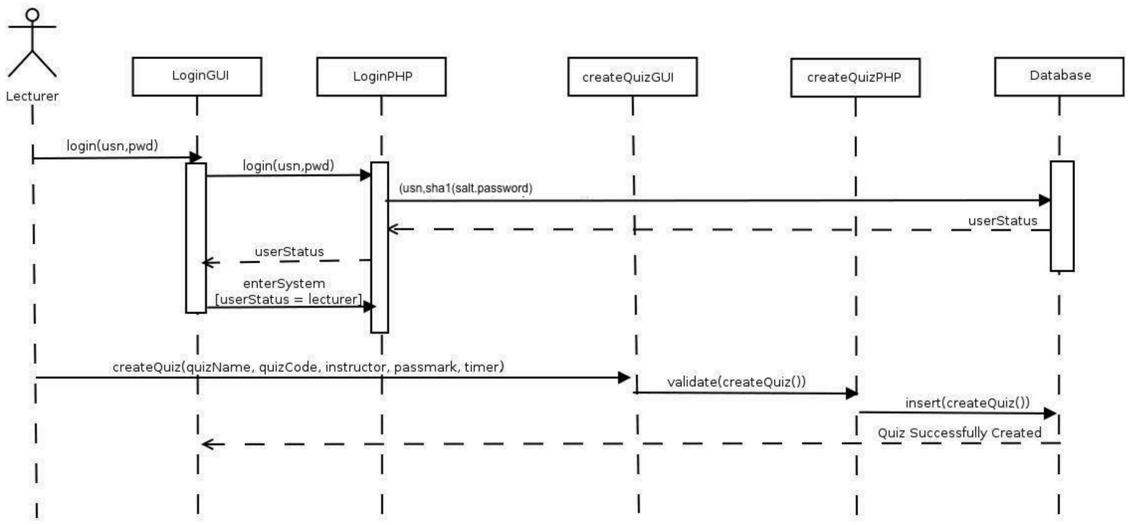
Diagram above adapted from [24]

PHP was chosen because it is a server side scripting language, meaning the application logic is done on the server side making for an efficiently thin client when numerous users are on the system.

The system has been design so that only users of modality ‘lecturer’ will ever be accessing pages within the core system folder. When a lecturer logs in then will be admitted entry into the core folder and all of their functionalities lay here, however when a student logs in, all of their actions take place on a page outside the core system folder.

Analysis Sequence Diagram

The diagram below shows the object interactions of the system arranged in time sequence, depicting the objects and classes involved in the scenario and the sequence of messages exchanged between the objects. The scenario below displays the sequence of events for a user to create a quiz.



As mentioned in the above sequence diagram, user passwords are stored as a hash in the database. The password is turned into a 160-bit hash by using the following method of salt and the sha1 PHP hash function.

```
$password_hash = sha1($salt.$password);
```

A password hash is created for added security as even if the intruder were to know the stored hash value they would not be able to “decrypt” it to retrieve the plaintext password. Sha1 was used over other hash functions because it is currently the mostly widely used out of them all.

GUI Design

In this section we shall talk about the overall GUI design for the QuizMaster system, the design has been kept simple with clear navigation where the time taken to learn the workings of the system are very short for a new user. Clear and concise dictionary English has been user to ensure users understand what the system is asking of them.

Font sizes and page colour schemes were thoughtfully chosen to be easily readable by the mass, all pages follow the same structure and the home button is always in the same position meaning users always have a button to escape out of a task and return to the home screen.

The front-end framework Bootstrap has been used to implement a clean user interface, I chose to use Bootstrap due to the high volume of extensions and support available. Use of the framework has encouraged consistency across the system, creating a friendly user environment for students and lecturers.
Bootstrap also comes equipped with JavaScript libraries that go above and beyond basic structural and styling as well as adapting to any change in platforms with super speed and efficiency.

The Bootstrap CSS file is used as a style sheet on most of the system pages, to which I have edited how and what stylings are displayed.
```
<link href="css/bootstrap.min.css" rel="stylesheet">
```

Database Design

Below is the database design diagram, displaying the relationship among tables within the database.
Normalization has been adopted, the process by which data is organised efficiently to achieve the following goals:

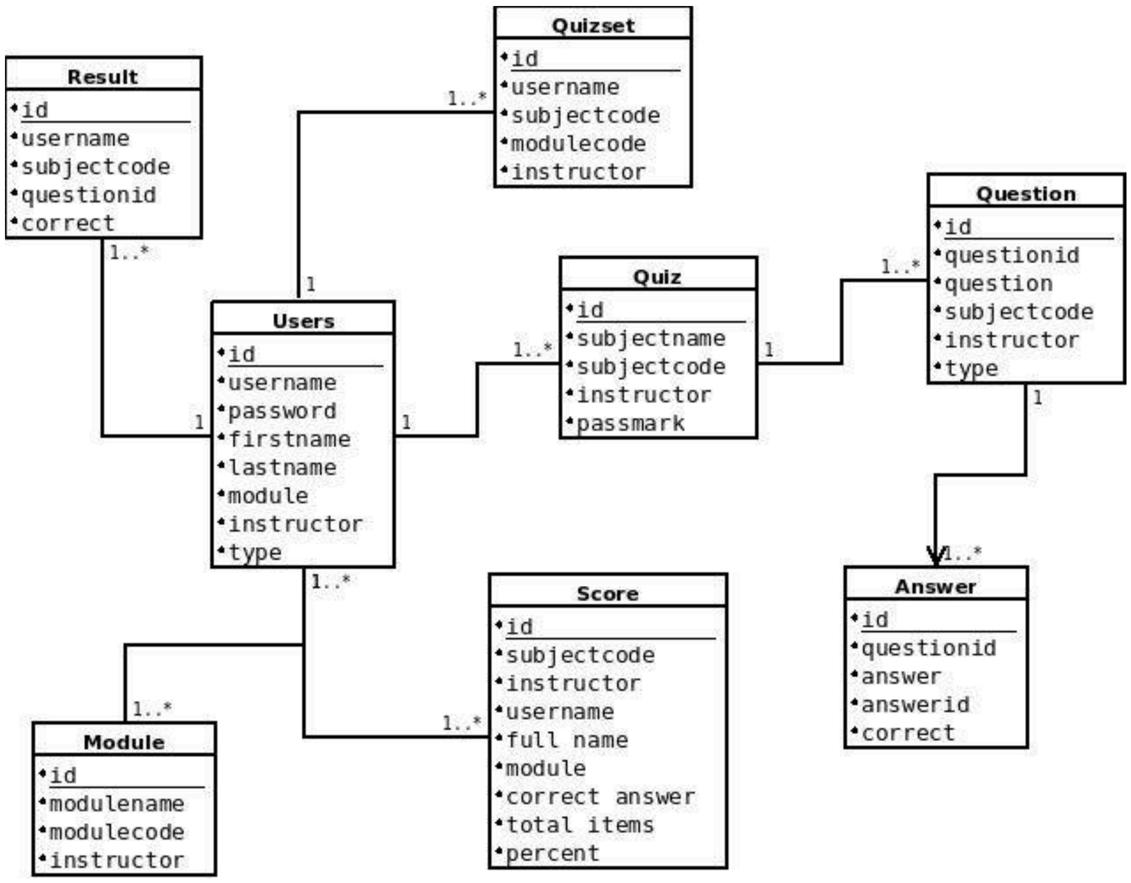
- Elimination of redundancy.
- Ensuring data is stored in the correct table and data dependencies make sense.
- Eliminating the need for restructuring the database when data is added.

The database design follows third normal form, in that functional dependency on non-key fields has been eliminated and all non-key fields are dependent only on the key.

The database has been designed so that data is logically split up and inserted into its appropriate table, for example question data and answer data have been separated. Implementing third normal form means that data duplication is reduced and therefore the database becomes smaller. Another advantage of normalisation to this degree is data integrity. When duplicated data changes, there's a big risk of updating only some of the data, especially if it's spread out in a number of different places in the database.

Data insertion has been controlled, only lecturers have the ability to directly input data. Student data is only entered into the database derived from their actions e.g. when they submit a quiz their score is entered but they have no way of manipulating this data.

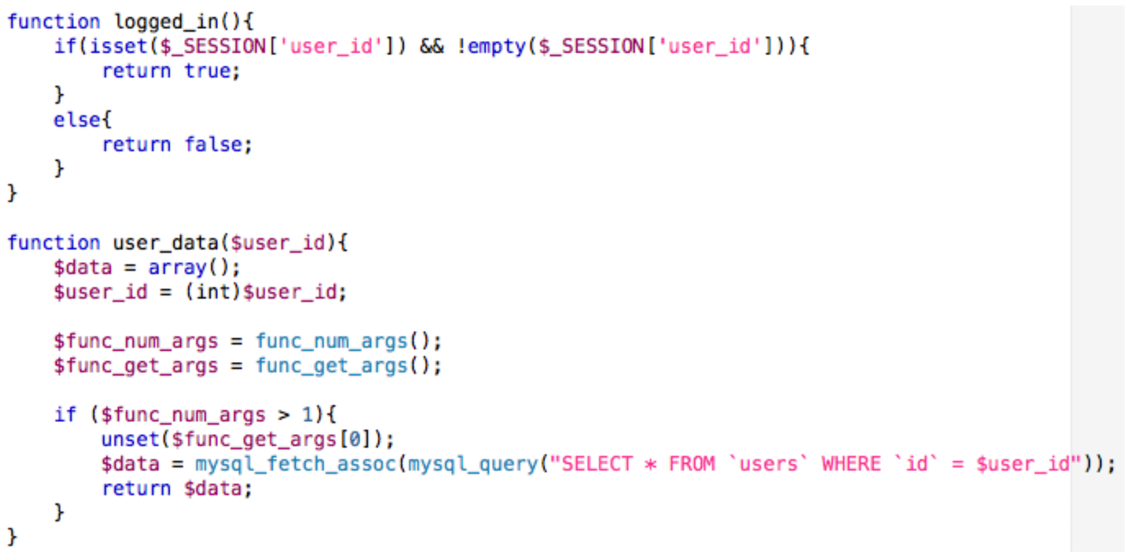
The database was designed to store atomic data and for specific data to be accessed with ease and use a minimal number of SQL queries. Table rows can be uniquely identifiable by their id and data gathered accordingly. A lecturer can create many quizzes, each quiz having many questions, which in turn each question can have many possible answers although only one of these answers per question is correct.



To ensure only logged in users can access the system past the login screen I have included the following code which is containing the functional code on every page within the system, if not logged in the user is returned to the log in page. As shown below, if a user is successfully logged in then the system stores a collection of session variables on the logged in user. This means that whenever a user performs an action their credentials are easily at hand.

```
if(logged_in()){

    $session_user_id = $_SESSION['user_id'];
    //User data for the session of logged in user
    $user_data = user_data($session_user_id, 'id','username','firstname','lastname','type');
    $instructor = $user_data['username'];
    }
```

The relating code below is kept in ‘core.php’ that is then included at the top each of the system file.



```php
function logged_in(){
    if(isset($_SESSION['user_id']) && !empty($_SESSION['user_id'])){
        return true;
    }
    else{
        return false;
    }
}

function user_data($user_id){
    $data = array();
    $user_id = (int)$user_id;

    $func_num_args = func_num_args();
    $func_get_args = func_get_args();

    if ($func_num_args > 1){
        unset($func_get_args[0]);
        $data = mysql_fetch_assoc(mysql_query("SELECT * FROM `users` WHERE `id` = $user_id"));
        return $data;
    }
}
```

Implementation and Testing

In this section we will discuss the methods of implementation concerning certain aspects of the software system as the reasoning behind these decisions.

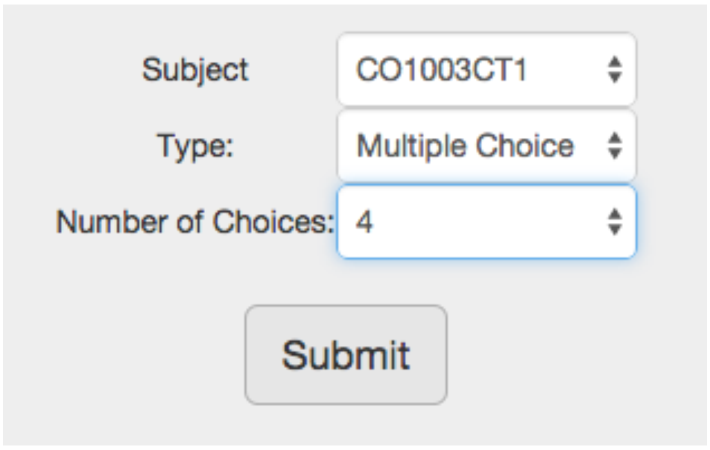
In its simplest form, the purpose of the system is for lecturers to create quizzes that they can populate with questions and then students can take the quiz.
When a lecturer submits their question the parsed data is split up, the question is stored in the question table and then the answers are stored in the answer table with the question id it relates to.

Create Question

Quiz Master offers the ability to create a number of different question types within a quiz; these are multiple choice questions, blank answer questions and true and false questions.

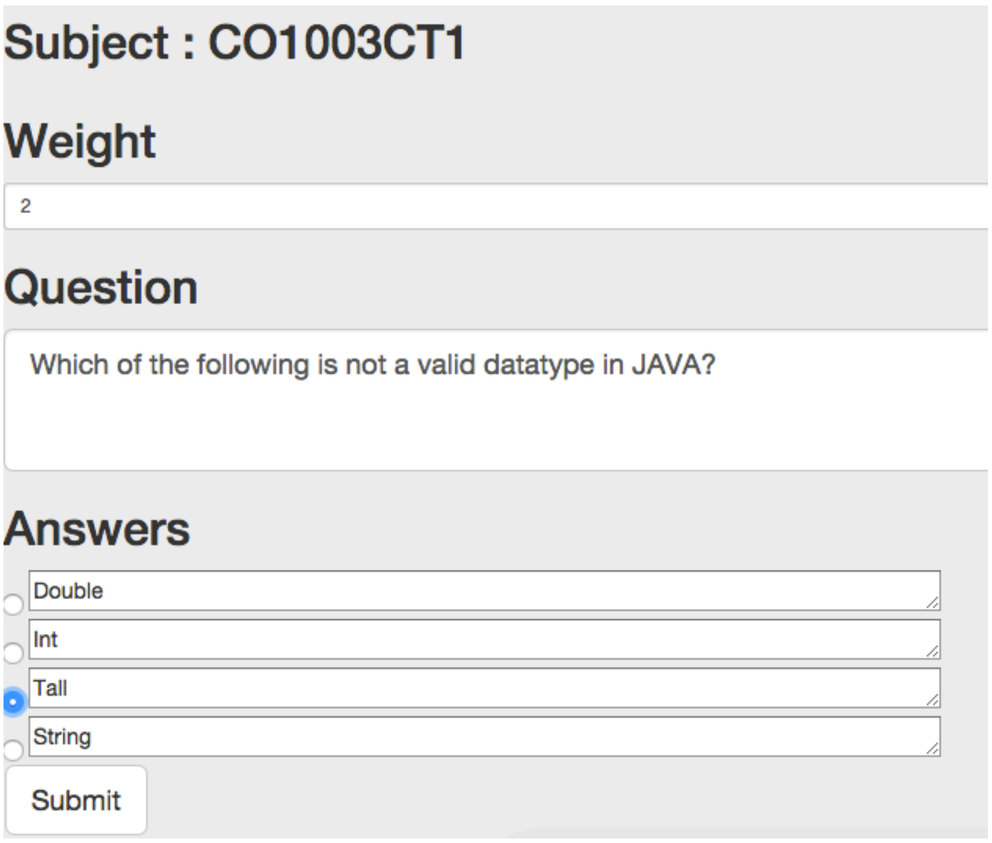
Multiple Choice

We will first talk about how the different question type formats are dealt with and fed through the tiers in the system. When a lecturer chooses to create a question they must firstly select the subject they wish to create the question for from the drop down menu, this is populated with existing subjects in the database that the logged in user is the lecturer for. The user is then prompted to select a question type, if they choose to create a multiple choice question another drop down menu appears populated with the numbers 2 to 5 corresponding to the number of possible answer choice boxes will appear for that question.



Subject	CO1003CT1
Type:	Multiple Choice
Number of Choices:	4

Once the user has submitted these details the appropriate text fields are displayed, one for the question data to be entered into as well as however many choices the user selected for the answer fields each accompanied by a radio button. All fields must contain data and one radio button selected corresponding to the correct answer before the question is successfully submitted.



Subject : CO1003CT1

Weight

2

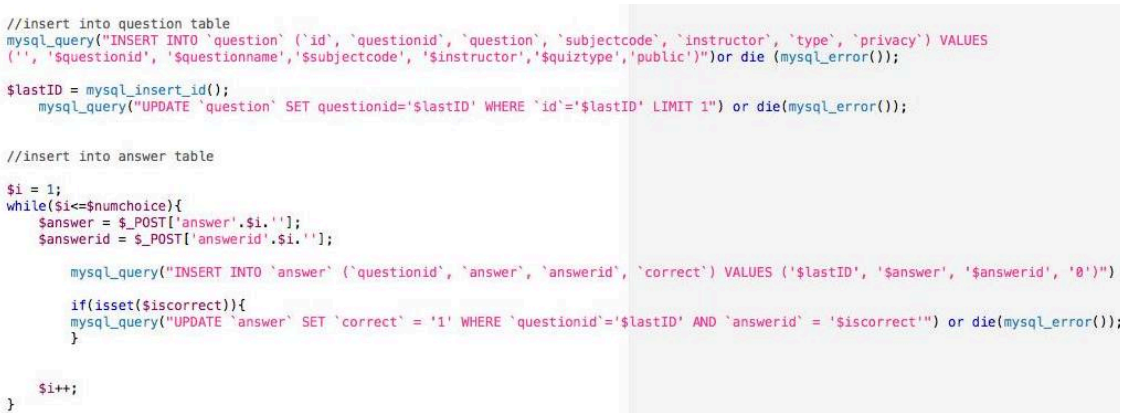
Question

Which of the following is not a valid datatype in JAVA?

Answers

- [ ] Double
- [ ] Int
- [✓] Tall
- [ ] String

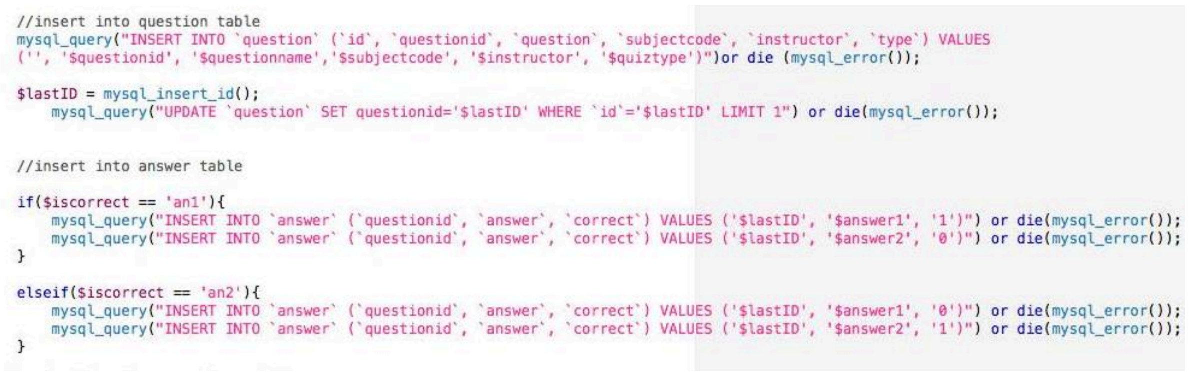
Upon submission the SQL query is ran and the data is sent, question data being stored in the question table and the answers being stored in the answer table looping through database insertion for however many answer choices was submitted. The correct answer is then updated and represented by entering a ‘1’ in the ‘correct’ column in the answer table for the answer whereby the user selected the corresponding radio button.



```
//insert into question table
mysql_query("INSERT INTO `question` (`id`, `questionid`, `question`, `subjectcode`, `instructor`, `type`, `privacy`) VALUES
('', '$questionid', '$questionname','$subjectcode', '$instructor','$quiztype','public')")or die (mysql_error());

$lastID = mysql_insert_id();
    mysql_query("UPDATE `question` SET questionid='$lastID' WHERE `id`='$lastID' LIMIT 1") or die(mysql_error());


//insert into answer table

$i = 1;
while($i<=$numchoice){
    $answer = $_POST['answer'.$i.''];
    $answerid = $_POST['answerid'.$i.''];

        mysql_query("INSERT INTO `answer` (`questionid`, `answer`, `answerid`, `correct`) VALUES ('$lastID', '$answer', '$answerid', '0')")

        if(isset($iscorrect)){
        mysql_query("UPDATE `answer` SET `correct` = '1' WHERE `questionid`='$lastID' AND `answerid` = '$iscorrect'") or die(mysql_error());
        }


    $i++;
}
```

True False

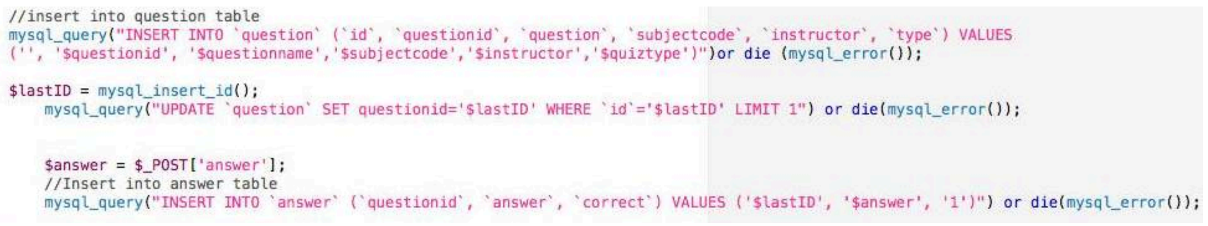
The process is similar for a true and false question, however the user just needs to enter question data and select a radio button as the two answers of true and false and set. When the question data is submitted an else if statement executes data insertion into the database depending on whether the user selected true or false as being the correct answer.



```php
//insert into question table
mysql_query("INSERT INTO `question` (`id`, `questionid`, `question`, `subjectcode`, `instructor`, `type`) VALUES
('', '$questionid', '$questionname','$subjectcode', '$instructor', '$quiztype')")or die (mysql_error());

$lastID = mysql_insert_id();
    mysql_query("UPDATE `question` SET questionid='$lastID' WHERE `id`='$lastID' LIMIT 1") or die(mysql_error());


//insert into answer table

if($iscorrect == 'an1'){
    mysql_query("INSERT INTO `answer` (`questionid`, `answer`, `correct`) VALUES ('$lastID', '$answer1', '1')") or die(mysql_error());
    mysql_query("INSERT INTO `answer` (`questionid`, `answer`, `correct`) VALUES ('$lastID', '$answer2', '0')") or die(mysql_error());
}

elseif($iscorrect == 'an2'){
    mysql_query("INSERT INTO `answer` (`questionid`, `answer`, `correct`) VALUES ('$lastID', '$answer1', '0')") or die(mysql_error());
    mysql_query("INSERT INTO `answer` (`questionid`, `answer`, `correct`) VALUES ('$lastID', '$answer2', '1')") or die(mysql_error());
}
```

Blank Answer

The third possible question format is blank answer, when selected two text fields are displayed prompting the user to enter question data and then the answer as text, as before this data is then sent to the database and split up to be stored in the relevant tables.

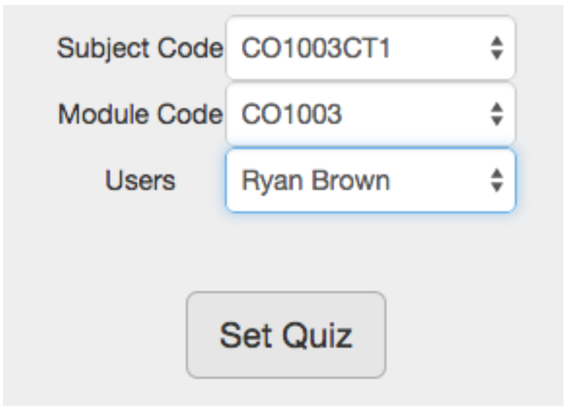


```php
//insert into question table
mysql_query("INSERT INTO `question` (`id`, `questionid`, `question`, `subjectcode`, `instructor`, `type`) VALUES
('', '$questionid', '$questionname','$subjectcode','$instructor','$quiztype')")or die (mysql_error());

$lastID = mysql_insert_id();
    mysql_query("UPDATE `question` SET questionid='$lastID' WHERE `id`='$lastID' LIMIT 1") or die(mysql_error());


    $answer = $_POST['answer'];
    //Insert into answer table
    mysql_query("INSERT INTO `answer` (`questionid`, `answer`, `correct`) VALUES ('$lastID', '$answer', '1')") or die(mysql_error());
```

Set Quiz

Lecturers also have the ability to set a quiz they have created for all of the students or just particular students of a module they are supervising to complete (e.g. a lecturer could set the students of a module a quiz to complete whereby the mark achieved on this quiz is counted as part of the coursework component for the module).
A supervisor need just navigate to the set quiz page whereby they are prompted to select from dropdown lists the appropriate subject code for the quiz they wish to select and the module code for the students they wish to set the quiz for.

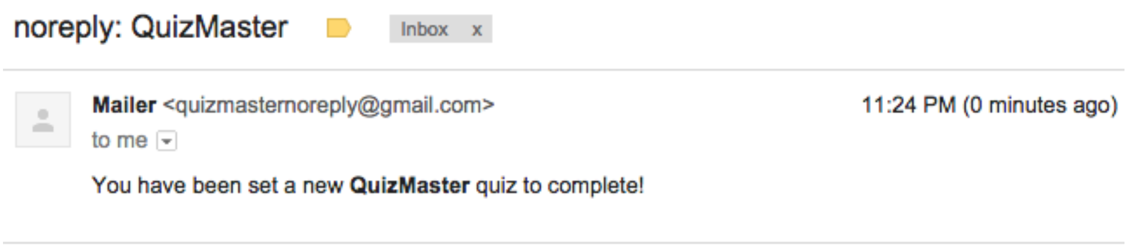
Once these parameters have been selected, the getUser.php script is called through an AJAX XMLHttpRequest, populating the users drop down menu with all the users part of that module. The lecturer can then select 'All' to set the quiz for all students enrolled on that module or for just a particular student (for example in the event that they may have had to retake the quiz).



Subject Code	CO1003CT1
Module Code	CO1003
Users	Ryan Brown

If 'All students' is selected, the query inserts the quiz into the 'Quizset' table for all students enrolled on that module through use of a while loop. If just one individual student is selected, the quiz data is just entered for that single student.

PHPMailer, an external library has been used to send out a notification in the form of email to students when they have been set a quiz to complete.



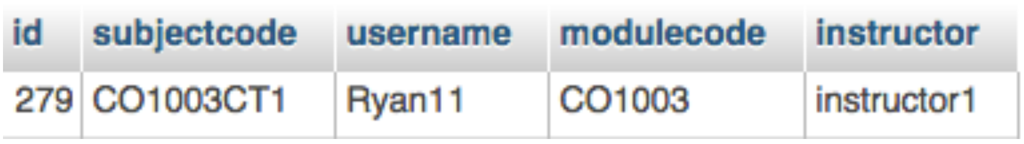
noreply: QuizMaster

Mailer <quizmasternoreply@gmail.com> 11:24 PM (0 minutes ago)

to me

You have been set a new **QuizMaster** quiz to complete!

All quizzes that have been set are stored in the 'quizset' table, this table is then read and any set quizzes for the logged in user are retrieved. An example row of this table can be seen below.



id	subjectcode	username	modulecode	instructor
279	CO1003CT1	Ryan11	CO1003	instructor1

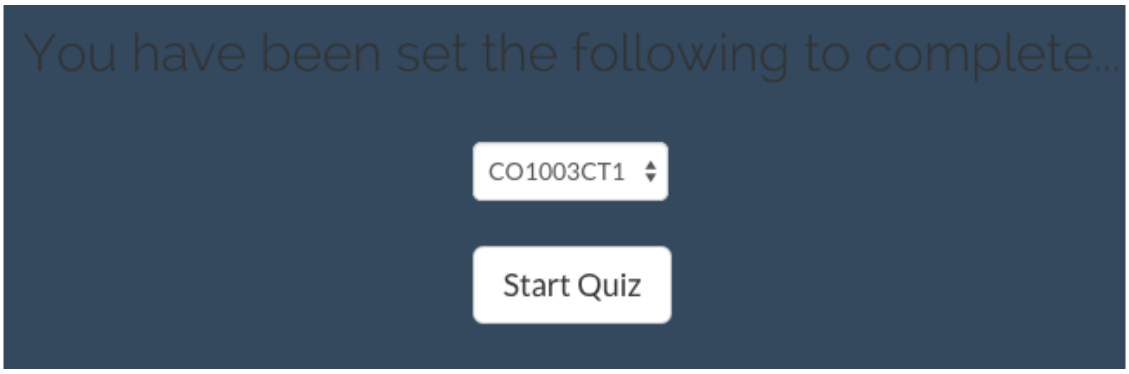
Play Quiz

We will now go on to discuss another of the core functionalities, this being the ability for a student to play through a quiz that they have been set to complete by their lecturer.

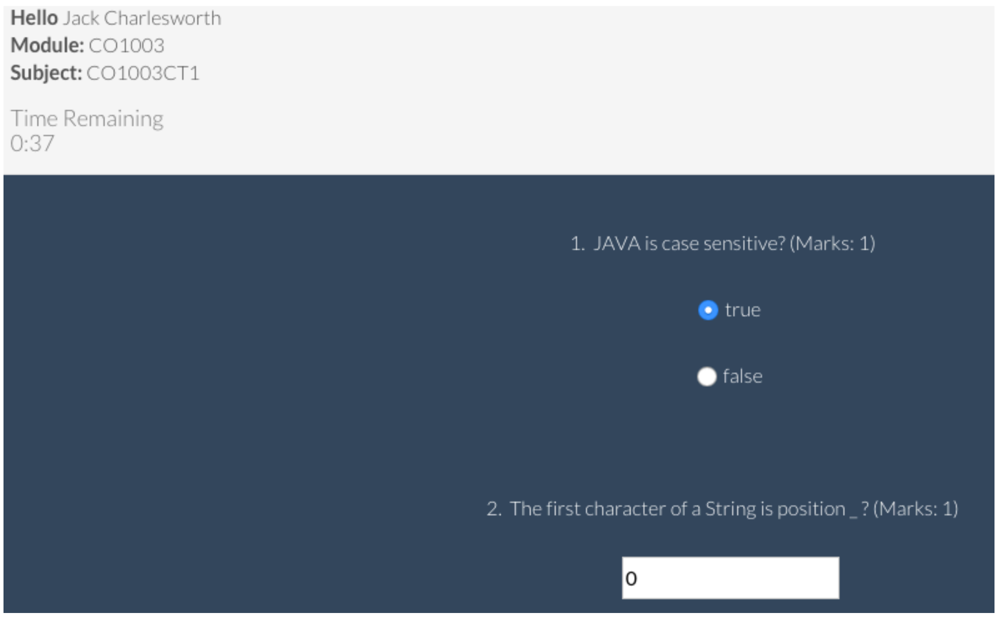
When a student logs into Quiz Master if they have no currently set quizzes they will not be able to play a quiz, this is because a student only has access to a view and play a quiz if they have been specifically set to do so. If the student does have outstanding quizzes to complete then this data will be retrieved from the database and displayed in a drop down menu for the user to select which quiz to play through. Quizzes that have been set for students to complete are stored in the 'quizset' table in the database.

```
$username = $user_data['username'];

$query="SELECT `subjectcode` FROM `quizset` WHERE `modulecode`='$module' AND `username`='$username'";
$result = mysql_query($query);
while($row = mysql_fetch_assoc($result)){
    $subjectcode = $row['subjectcode'];

    echo '<option value="'.$subjectcode.'">'.$subjectcode.'</option>';

}
```

Subject code data populated from the 'Quizset' table where 'username' = username of the currently logged in user. Clicking 'Start Quiz' will start the quiz for that subject currently displayed in the drop down menu.



The screenshot below is of the screen that appears when 'Start Quiz' is clicked, the user is playing the quiz. Quiz components are retrieved from the database and displayed accordingly. The timer begins to countdown, the user is prompted to enter their answers and submit the quiz. The timer component is fixed to the header, meaning when the user is scrolled down to the e.g. the seventh question they can still view the remaining time at the top of their screen at all times throughout the quiz.



Hello Jack Charlesworth
Module: CO1003
Subject: CO1003CT1

Time Remaining
0:37

1. JAVA is case sensitive? (Marks: 1)

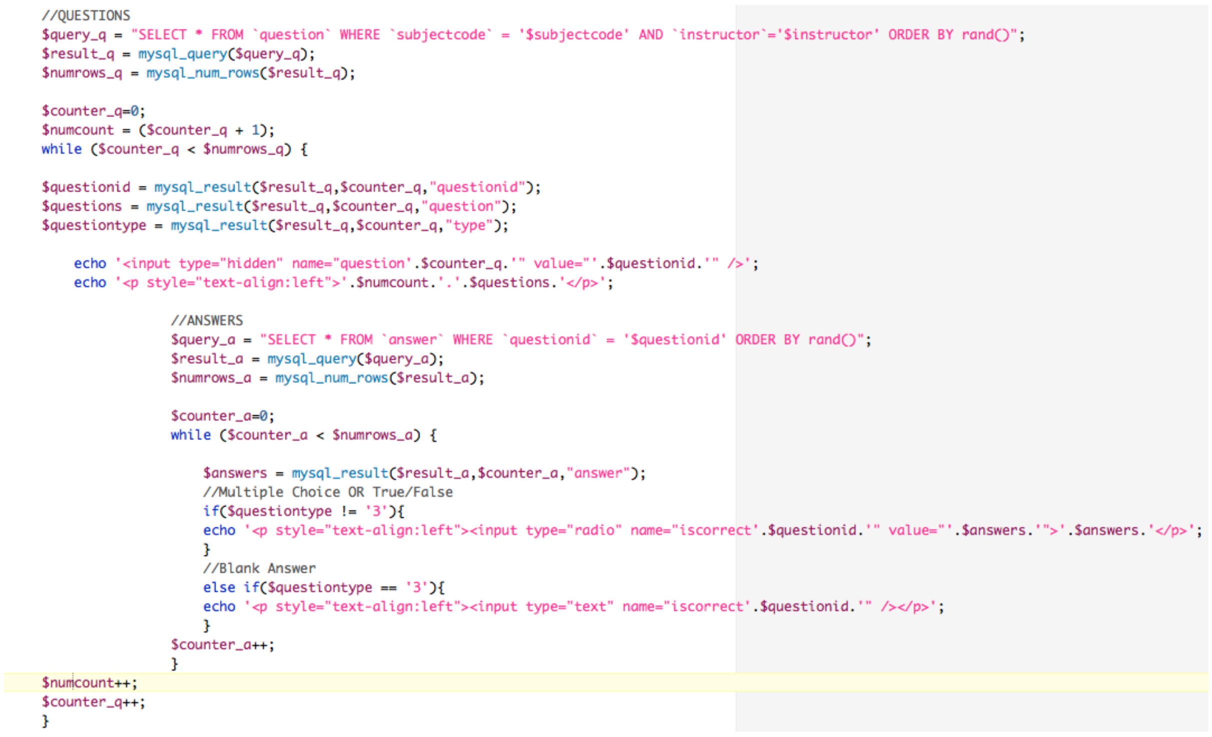
[✓ true]

false

2. The first character of a String is position _ ? (Marks: 1)

0

The code snippet below shows the code executed once the Start Quiz button is selected, looping through and returning on screen for each question the appropriate answer fields depending on the type of question.

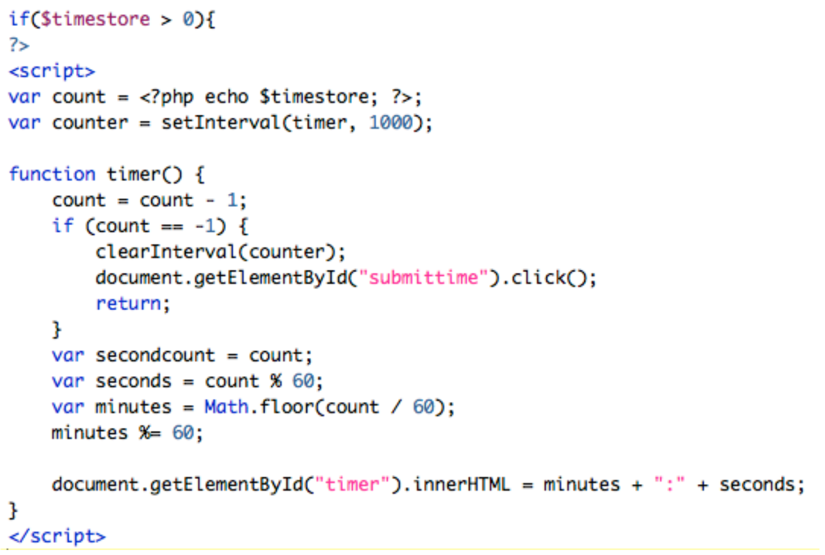


```
//QUESTIONS
$query_q = "SELECT * FROM `question` WHERE `subjectcode` = '$subjectcode' AND `instructor`='$instructor' ORDER BY rand()";
$result_q = mysql_query($query_q);
$numrows_q = mysql_num_rows($result_q);

$counter_q=0;
$numcount = ($counter_q + 1);
while ($counter_q < $numrows_q) {

$questionid = mysql_result($result_q,$counter_q,"questionid");
$questions = mysql_result($result_q,$counter_q,"question");
$questiontype = mysql_result($result_q,$counter_q,"type");

    echo '<input type="hidden" name="question'.$counter_q.'" value="'.$questionid.'" />';
    echo '<p style="text-align:left">'.$numcount.'.'.$questions.'</p>';

                //ANSWERS
                $query_a = "SELECT * FROM `answer` WHERE `questionid` = '$questionid' ORDER BY rand()";
                $result_a = mysql_query($query_a);
                $numrows_a = mysql_num_rows($result_a);

                $counter_a=0;
                while ($counter_a < $numrows_a) {

                    $answers = mysql_result($result_a,$counter_a,"answer");
                    //Multiple Choice OR True/False
                    if($questiontype != '3'){
                    echo '<p style="text-align:left"><input type="radio" name="iscorrect'.$questionid.'" value="'.$answers.'">'.$answers.'</p>';
                    }
                    //Blank Answer
                    else if($questiontype == '3'){
                    echo '<p style="text-align:left"><input type="text" name="iscorrect'.$questionid.'" /></p>';
                    }
                $counter_a++;
                }
$numcount++;
$counter_q++;
}
```

Once the student successfully submits their answers to the quiz, and the system displays to them their results another query is run deleting the row from the ‘quizset’ table for that student and quiz combination meaning they no longer have that quiz outstanding to complete.

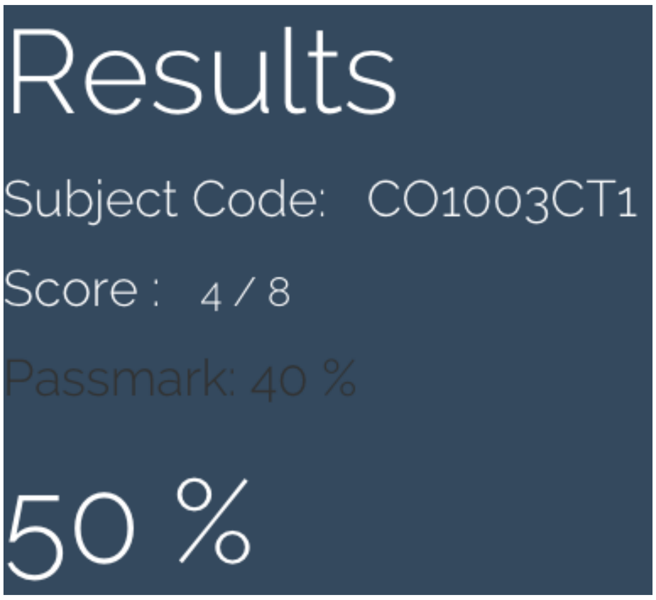
Quiz Timer

When a lecturer creates a quiz they are prompted to set a timer, this value specifies the amount of time allowed to complete this quiz, starting from when the quiz is started to being submitted. This timer data is also retrieved from the database and displayed on screen and begins counting as soon as the page is loaded, counting down from that value in seconds with use of a JavaScript timer. The time store is entered and stored in the database as seconds, when the timer is called this value in seconds is formatted and displayed as minutes and seconds for readability.



```
if($timestore > 0){
?>
<script>
var count = <?php echo $timestore; ?>;
var counter = setInterval(timer, 1000);

function timer() {
    count = count - 1;
    if (count == -1) {
        clearInterval(counter);
        document.getElementById("submittime").click();
        return;
    }
    var secondcount = count;
    var seconds = count % 60;
    var minutes = Math.floor(count / 60);
    minutes %= 60;

    document.getElementById("timer").innerHTML = minutes + ":" + seconds;
}
</script>
```

When the student submits their finished quiz **OR** the timer reaches 0 the following code is executed, comparing the users posted answer with the correct answer for that question which is retrieved from the database. If the quiz is submitted because the timer reaches 0 the students answer form is submitted in its current state, where questions have been answered they will be marked. If questions are left unanswered the user shall receive zero marks for those questions. The result table stores every question submitted along with how the student faired for that question, storing a '1' in the correct column for a correct answer and a '0' if else.


Results

Subject Code: CO1003CT1

Score : 4 / 8

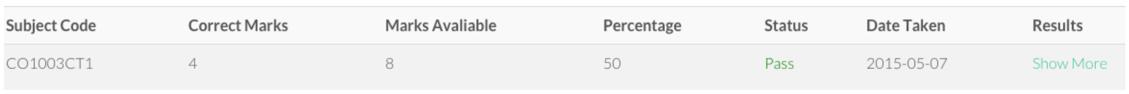
Passmark: 40 %

50 %

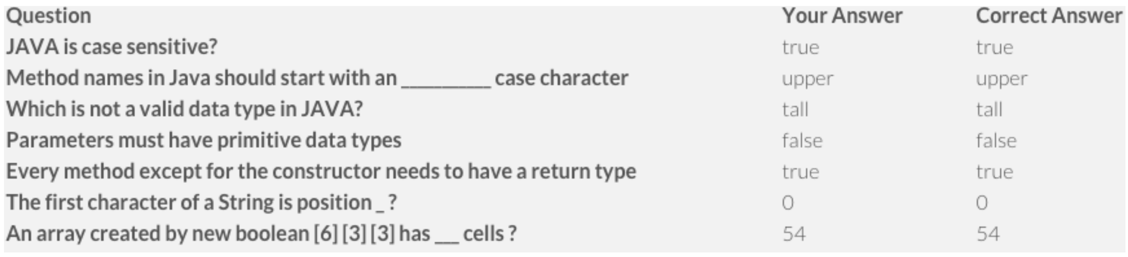
To gather statistical data to give immediate feedback to the student the number of correct answers is gathered from the data that was just entered into the result table, then adding up the weight of each question answered correctly to attain a score.
This score is then displayed over the total number of marks that were available on the quiz. A percentage is also calculated which is compared to the pass mark set by the lecturer for the quiz to indicate whether the student has passed or failed.
These score details are then sent and stored in the 'score' table in the database where they can be accessed and viewed by both the student to whom the score belongs as well as the lecturer to whom created the quiz.

Student View of Results

Once a student has successfully completed at least one quiz, they have access to a history of their performance in past quizzes they have taken. The Quiz results table shown below allows students to view their past scores and details. Should they wish to view more about their performance in this quiz, including their posted answers against the correct answers this can be viewed by clicking "show more".



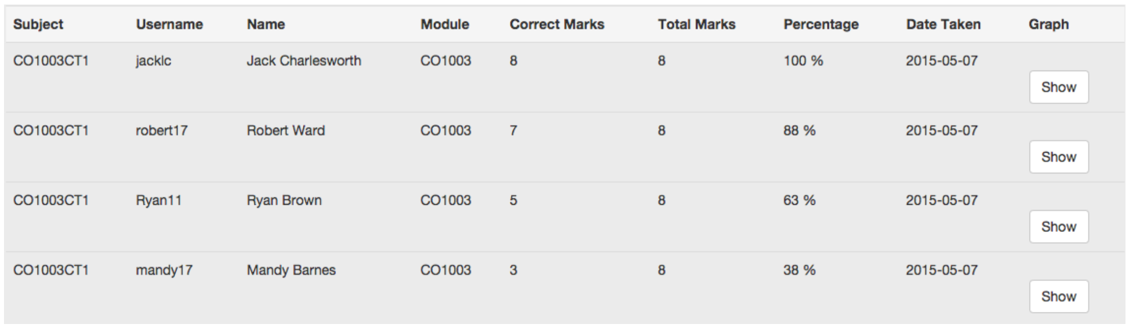
Subject Code	Correct Marks	Marks Avaliable	Percentage	Status	Date Taken	Results
CO1003CT1	4	8	50	Pass	2015-05-07	Show More



Question	Your Answer	Correct Answer
JAVA is case sensitive?	true	true
Method names in Java should start with an ________ case character	upper	upper
Which is not a valid data type in JAVA?	tall	tall
Parameters must have primitive data types	false	false
Every method except for the constructor needs to have a return type	true	true
The first character of a String is position _ ?	0	0
An array created by new boolean [6] [3] [3] has __ cells ?	54	54

Lecturer View of Results

When students successfully complete a quiz set, their results and score are then fed back to the lecturer of that module to view. The lecturer has access to information on the students including their percentage, whether they passed of failed, how many marks they achieved and the date they sat the quiz.

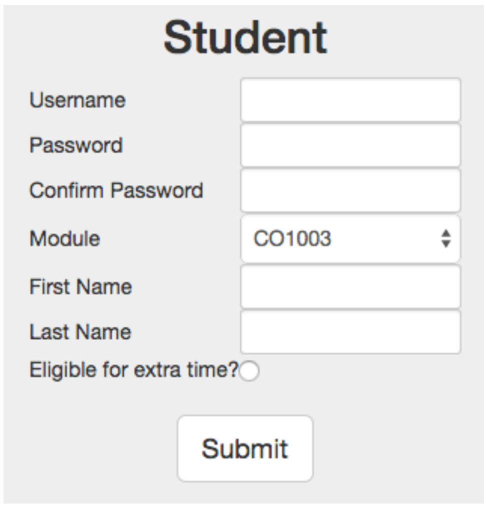


Subject	Username	Name	Module	Correct Marks	Total Marks	Percentage	Date Taken	Graph
CO1003CT1	jacklc	Jack Charlesworth	CO1003	8	8	100 %	2015-05-07	Show
CO1003CT1	robert17	Robert Ward	CO1003	7	8	88 %	2015-05-07	Show
CO1003CT1	Ryan11	Ryan Brown	CO1003	5	8	63 %	2015-05-07	Show
CO1003CT1	mandy17	Mandy Barnes	CO1003	3	8	38 %	2015-05-07	Show

The lecturer has the ability to view a graph of the results from the students within a particular module given a particular quiz. This graph is created making use of the pChart library discussed later on in this document.

Create Student

A lecturer has the ability to create new students in the QuizMaster system using the below form. The student will be created and can be assigned to a module that the preexisting lecturer is the convener, taking one of the previously mentioned accessibility requirements in mind it can be noted if this student is eligible for extra time during quizzes.



Student

Username	
Password	
Confirm Password	
Module	CO1003
First Name	
Last Name	
Eligible for extra time?	○

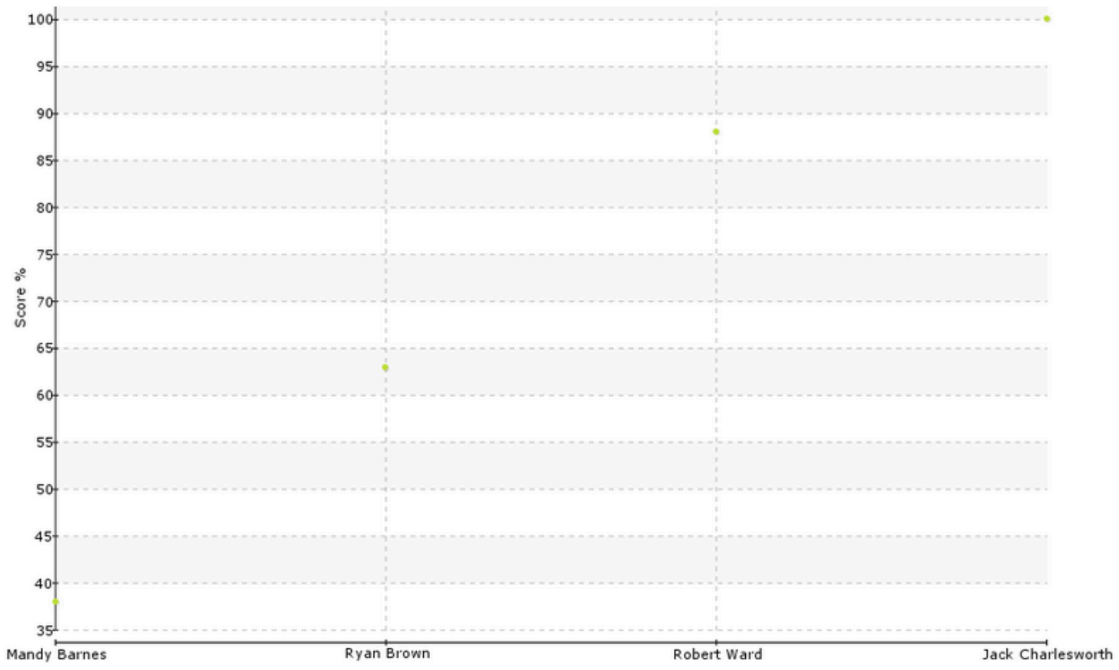
If selected, for all of the quizzes the student attempts they will receive an extra 20% of time permitted for that exam as outlined in the University exam regulations.

Note: If this software were to be commercially released, students would be automatically added onto the QuizMaster system by incorporating the institutions existing database of students. Allowing for seamless integration with the institutions library of currently enrolled student data.

pChart Framework

QuizMaster makes use of the pChart PHP framework to draw up statistical graphs from collected database data on the scores attained by students. I made use of the pChart library to create a plot chart of student scores for a particular quiz that the lecturer may wish to view. By viewing the graph the lecturer has the ability to graphically see any trend in the class scores.

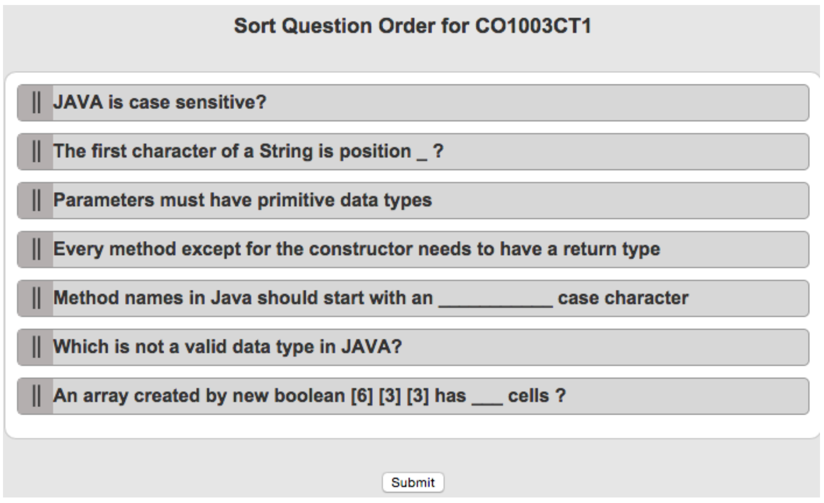
I made the implementation decision to use pChart over other libraries due to the documentation made available by the creators and the ability to plot data through MySQL queries. Unlike other frameworks, pChart draws the graph data into an image that can be viewed on screen but also saved, by allowing lecturers to save graphs they can file this data away for reference when it comes to viewing average scores over the years.



Dynamic Drag and Drop Sorting of Items

Within the QuizMaster I made use of some free to use code from author Ali Aboussebaba that makes use of jQuery (Found in bibliography [7] the files in my project that make use of this code are set_order.php, sortquiz.php and config.php). This existing code was incorporated and then edited by myself to apply it for the functions that I needed it for in this project, ordering question items within a quiz. This allows lecturers the ability to set the order for which they wish the questions to appear to students taking their quiz.

This feature separates QuizMaster from similar question and answer web applications, as they do not offer this interactive sorting functionality.
Once the drag and drop sorting form is submitted, the 'item_order' column in the question database is updated, giving each question a number value according to the position in which each question element was submitted by the user thus representing their position in the quiz. The screenshot below shows how the code was used for ordering questions, each question can be dragged and dropped to its new position in the quiz. The question positioned at the top being question number one, then the questions go down in ascending order.


Sort Question Order for CO1003CT1

|| JAVA is case sensitive?

|| The first character of a String is position _ ?

|| Parameters must have primitive data types

|| Every method except for the constructor needs to have a return type

|| Method names in Java should start with an __________ case character

|| Which is not a valid data type in JAVA?

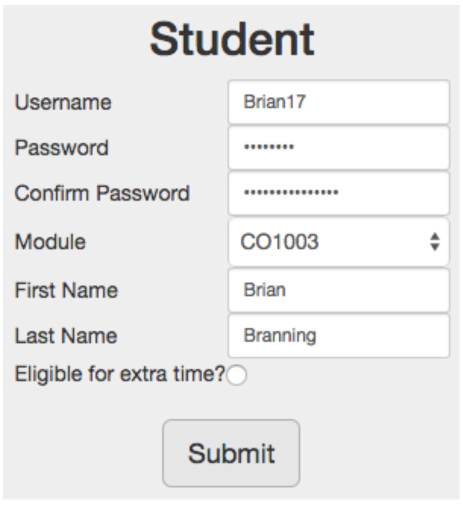
|| An array created by new boolean [6] [3] [3] has ___ cells ?

Testing

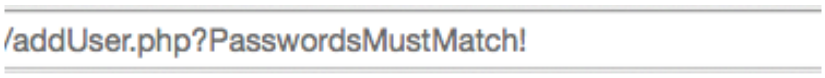
Testing of the QuizMaster system was carried out during the implementation phases along the way, the data users are permitted to supply the system with has been limited by use of drop down boxes populated with safe data.

Form validation has been implemented in an effort to prevent empty or non-valid strings of data to be entered into forms and fed through the system. This means the data the user supplies to the system is controlled, no meaningless blank entries can be sent and stored in the database. It also means the chance for user error can be reduced, for example when creating a new student they are prompted to confirm the password meaning the system can check they match preventing mistypes.

User form validation messages are unobtrusive however do the job of the cutting off the flow of data, an example of this can be seen below.



Student	
Username	Brian17
Password	••••••••
Confirm Password	•••••••••••••••
Module	CO1003
First Name	Brian
Last Name	Branning
Eligible for extra time?	○

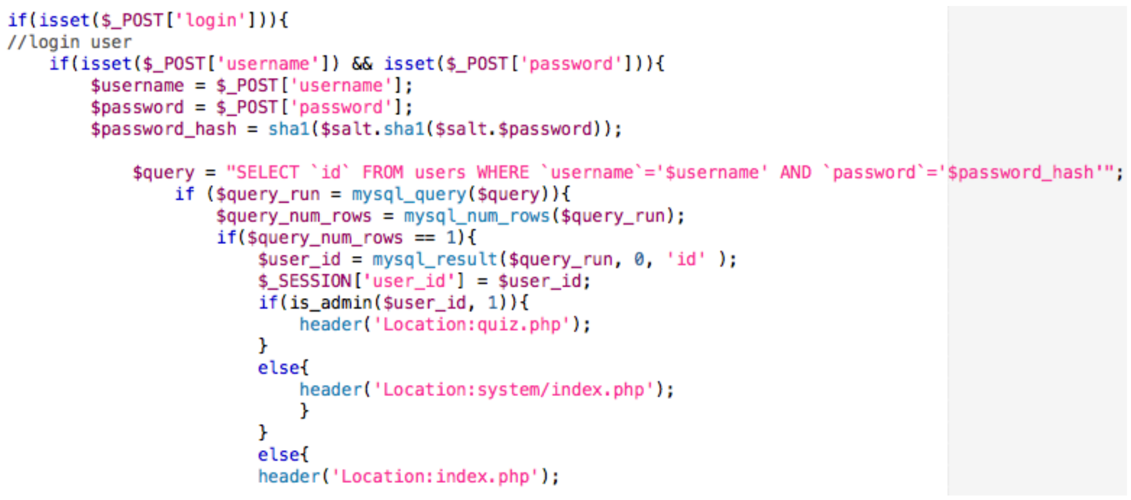


/addUser.php?PasswordsMustMatch!

Log In

Good Input: Password that exists for the username in the database	Bad Input: Password that does not match with the username attempting to log in
- System creates a hash using sha1 and salt of the plaintext password - Checks to see if the hash created matches the hash stored in the database for the username that is entered - When a match is found the user is granted their log in request and allowed entrance into the system	- System creates a hash using sha1 and salt of the plaintext password - Checks to see if the hash created matches the hash stored in the database for the username that is entered - When no match is found, user is denied their log in request and returned to the log in page

The code that executes the log in function can be found below.



```
if(isset($_POST['login'])){
//login user
    if(isset($_POST['username']) && isset($_POST['password'])){
        $username = $_POST['username'];
        $password = $_POST['password'];
        $password_hash = sha1($salt.sha1($salt.$password));

            $query = "SELECT `id` FROM users WHERE `username`='$username' AND `password`='$password_hash'";
                if ($query_run = mysql_query($query)){
                    $query_num_rows = mysql_num_rows($query_run);
                    if($query_num_rows == 1){
                        $user_id = mysql_result($query_run, 0, 'id' );
                        $_SESSION['user_id'] = $user_id;
                        if(is_admin($user_id, 1)){
                            header('Location:quiz.php');
                        }
                        else{
                            header('Location:system/index.php');
                            }
                        }
                        else{
                        header('Location:index.php');
```

Create Question

We will specifically be testing the implementation of creating a multiple choice question.

Good Input: All question data fields are submitted containing valid data. Question and answer data is entered, one answer has been set as being the correct answer.	Bad Input: Question data is submitted however fields are missing data, in this test no answer has been declared as being the correct answer.
- User selects which quiz to add the question to, multiple choice question type and 4 answer choices - The system creates a form accordingly, data is input and the user submits - The system enters the question data into the database and loops through the number of answer choices and enters this data into the database - User is notified the question has been successfully added	- User selects which quiz to add the question to, multiple choice question type and 4 answer choices - The system creates a form accordingly, data is input by the user however no answer is declared as correct and the user submits - There is missing compulsory data - User is notified they are missing compulsory question elements and are prompted to try again - Question data that is not complete is not sent through to the database

Peer Testing

Peer reviews were conducted at several phases during the implementation phase, trials were conducted where a focus group of users interacted with the system documenting their feelings in real time and giving critical evaluations of the system.
According to the Capability Maturity Model, the purpose of conducting peer reviews is to provide an engineering practice for detecting and correcting defects.

At the end of peer reviews, notes were taken documenting the results from users. This information was used to improve upon functionality as well as the graphical user interfaces throughout the system.

Peer testing was done within a focus group of 5 people after construction one and again after the second construction phase. They were asked to complete a number of tasks and told to think aloud as they were using the system, this information gained was vital to the progression of development allowing me an insight into the user experience at a number of stages.

Good PHP Programming Practices

PHP can be highly inconsistent and sometimes buggy, I have worked throughout the implementation using the following good programming practices.

There is debate online as to the use of single quotes being more optimal for performance over the use of double quotes. Single quoted strings are not parsed whereas double quoted strings are parsed. The theory is that using single-quoted strings will improve performance because PHP will not evaluate every single string. However all this for an application of this size does not really matter, although single quotes has been used with future expansion and user load kept in mind.

In addition to this, as previously mentioned in a previous section passwords are never stored or even sent as plaintext. The password data received is immediately turned into a 160-bit hash making use of SHA1 and salt techniques.
Built in PHP functions have been used where appropriate, the 'stripslashes' function has been used on user input before it is sent through to the SQL statements to return a string with the backslashes removed.
Another function used is mysql_real_escape_string() which escapes special characters in a string ready for use in an SQL statement.
These are both examples of SQL injection prevention, which refers to the act of someone inserting a MySQL statement to be maliciously run on the database by exploiting user input forms.

Throughout the project I have used the 'DRY approach', which stands for Do Not Repeat Yourself in the effort to not waste time writing redundant code. An example of this that can be seen throughout the code is when connecting to the database, making use of code from 'core.php' and then writing an include statement at the top of every page with database access.

Critical Appraisal

There are of course in reflection a few things I may of done differently in hindsight, during the initial project phases I would have looked into using a PHP framework such as CodeIgniter and the benefits it could have given me during the development of software, for example speed of prototyping. At the beginning of the project I thought I had correctly allocated time when constructing Gantt charts, however there was of course unexpected events and delays that at times caused deviation.
If I were to do this project again I would have allowed for some delay when drawing up plans. I would have also further researched how I could implement and incorporate more existing API's and libraries to further the technical capabilities of the system.

In reflection, during the project once I was comfortable manipulating the database through use of while loops and if statements it became a case of not branching out and looking for other ways of performing actions. This meant in some cases that once I had implemented one function, to implement a similar function was a case of repeating and altering although this meant a good level of efficient reuse.
Although the GUI design implemented for QuizMaster allows for cross platform support including on mobile devices, given further development time I would perhaps of built a standalone mobile application allowing lecturers to configure and tweak their content. Given further development time I would have opted to build the GUI, as opposed to using an external library and template although these did speed up prototype deliveries.

The QuizMaster software system as a product was produced bearing in mind a number of factors such as the system being readily available for commercial purposes within teaching institutions. Technologies that are widely used and freely available as of the current date were utilised, this means should a higher education institution wish to purchase a licence for QuizMaster they will have minimal in the way of start up costs.

The QuizMaster system can be used on the institutions already existing server and accessed via a web browser on institution terminals, as mentioned in previous sections PHP is a server side scripting language with the system able to be run on low specification terminals as they only need to render the HTML pages.

If the system were to be further developed it could be personalised for the institution buying the licence, the system could also be refined to incorporate the institutions student and staff alumni and module data which would retire the need for manually adding users and modules.
Reflecting back to one of the original aims "to build a web application that supports the creation and maintenance of question and answer tests aiding module lecturers and conveners within teaching institutions". I believe the system has enough of a commercial value to be used by higher education institutions across the country, giving a far superior user experience than similar software available.

PHP Data Objects would have been research and implemented if I were to do this project again, PDO is as an abstraction layer used for accessing databases. PDO provides both security and consistency when connecting to databases, with raw MySQL functions we needed to sanitise the input manually using mysql_real_escape_string(). With PDO this is all taken care for us and means we no longer have to worry about SQL injection. In reflection this is something I have taken away with me about the development of web applications and I hope lessons I can take forward into my future work.

Given more development time I would look to implement a time machine type feature whereby lecturers can go to the time capsule to search through previous set quizzes for the module from previous years and offer them the ability to drag questions from the question pool from previous years and drop them into the current quiz. This feature could be further extended to include a rating out of 5 awarded to each question that could be calculated by data on how past students scored on that question.

Having read over methods of form security in the later stages of the project, if given more time or if especially the system were to be rolled out commercially I would invest more in form security methods. Taking actions to prevent malicious activities such as SQL injections and XSS attacks.

In regards to personal development, there were a number of lessons learnt during this project that I will take with me into my career. I gained the ability to independently learn new technologies, in the field of web development that was of a particular interest to me. I refined my ability to plan timescales and be a good judge of time constraints, with the patience and perseverance to complete tasks to the best of my ability attempting to iron out all errors and problems I encountered on the way.

Survey of Information Sources

The initial stages of the project involved a fair amount of time consisting of research into how to best implement a solution to the problem. Once I had chosen PHP as the core language I went on to familiarise myself again with the language as I had not used it since first year. I spent a number of hours reading over resources from the official PHP.net manual and third party recourses such as codeacademy interactive tutorials to relearn aspects of the language, in reflection this was time very well spent.
The Internet was a valuable resource throughout this project, both when it came to making design decisions and to solving construction problems.
When it came to making architectural design choices, there was a lot of debate online about which was the most suitable to a web development application e.g. three tier, MVC or client server. These discussions online led to me making informed decisions about what was best for my needs and situation, allowing me to understand the pros and cons of each without having to encounter the problems discussed online myself in my own deliverable.

Conclusion

In order to evaluate the work during this project the evaluation will focus on how well each of the aims, objectives and requirements have been met that were outlined in previous sections.

Requirements were implemented in order according to their MoSCoW prioritisation, all of the 'Must' requirements and some of the 'Should' were delivered during the first phase of construction.
This meant that the second phase of construction could lead onto refinement of these core features as well as development of some of the features that were not deemed to be core requirements.
The 'Could' level requirements were those that would add interesting and unique functionality to the system however due to time constraints a few of these were not implemented.
One of these 'Could' features I would hope to implement if given more time would be to 'Implement a time machine type backlog of past quizzes set for a module over the years'. I feel this feature would add a reusability factor that would make QuizMaster a system with longevity in its work environment.

Perhaps the reason there were a few low priority features I was not able to implement was due to the development methodology choice I made, agile methodologies. Although I feel working along side this methodology was beneficial for the project as it allowed me to time box tasks, it meant the project could easily get taken off track if the client requirements are not concisely identified then the final outcome is not clearly defined.

The project objectives drawn up at the start were as follows:

- One of the first objectives during the initial phases was to research and collect data on the most suitable technologies that could be utilized for implementing such a project. ***At the initial planning phase I spent a number of hours researching the most suitable technologies, in relation to my existing knowledge and how it was fit for the solution.***
- To research current software solutions in the marketplace, critically evaluating them to find areas where they lack and areas where they perform badly. ***During the initial stages of planning and design I spent a lot of time gathering and evaluating existing solutions. Critically evaluating them in order to best design the QuizMaster system around the gap in the market where these systems did not perform.***

- To make informed design decisions on how the QuizMaster system will be implemented, designing the top-level structure and how to allocate and prioritise time as well as how to best modularise the system into tiers. ***I researched, architectural styles, database designs and external libraries that could be put to use.***

- System functionality will be split into appropriate classes and directories to allow for accessible maintenance whilst creating seamless integration between technologies, components and external libraries used. ***The system was built and split into logical classes and files, the student never gains access to the 'core' system file. The only two pages they ever operate in are index.php and quiz.php, this was done for a number of reasons including security and a better user experience not having to navigate pages.***

- To evaluate the delivered quiz application as a tool within a learning environment both quantitatively and qualitatively to understand its value and limitations. ***The extent to which the system has been implemented has been testing through quantitatively testing functionality through rational, statement coverage and peer testing.***

In conclusion the aims and objectives were met, fulfilling the problem area with an effective solution that has scope for scale and further development. The delivered system allows lecturers the creative freedom to build quizzes that can be easily maintained for the lifetime of a module, offering them a real alternative to paper based quizzes with numerous advantages should they adapt.
By achieving all of the objectives set out at the beginning of this project, including the conducting of thorough background research to explore the problem area I was able to design and construct a solution to satisfy the project aims.
Through background research of existing quiz creation applications such as QuizGlobal and TestMoz it is clear to see they lack areas of functionality, of which I have built into QuizMaster for example the ability to order questions and receive graphical data from quiz results.

Bibliography and Citations

[1] "PHP Manual"
Accessed on April 2015.
http://php.net/manual/en/index.php

[2] "W3Schools PHP"
Accessed on April 2015.
http://www.w3schools.com/php/

[3] "W3Schools JavaScript"
Accessed on April 2015.
http://www.w3schools.com/js/default.asp

[4] Boronat, A. "Requirements and Use Cases"
Accessed on April 2015.
CO2006 Software Engineering and System Development
https://campus.cs.le.ac.uk/teaching/resources/CO2006/slides/L08.pdf

[5] "W3Schools SQL"
Accessed on April 2015.
http://www.w3schools.com/sql/default.asp

[6] "Bootstrap"
Accessed on April 2015.
http://getbootstrap.com/examples/theme/

[7] Aboussebaba, A. "Dynamic Drag and Drop with jQuery and PHP"
Accessed on April 2015.
http://www.bewebdeveloper.com/tutorial-about-dynamic-drag-and-drop-with-jquery-and-php

[8] "PHP Sessions, why use them?"
Accessed on April 2015.
http://www.tizag.com/phpT/phpsessions.php

[9] "Countdown timer in JavaScript"
Accessed on April 2015.
http://stackoverflow.com/questions/29338202/countdown-timer-in-javascript-for-each-row-of-table-in-html

[10] "JavaScript getElementByID"
Accessed on April 2015.
http://www.tizag.com/javascriptT/javascript-getelementbyid.php

[11] "PHP GET and POST Methods"
Accessed on April 2015.

http://www.tutorialspoint.com/php/php_get_post.htm

[12] "PHP Syntax Overview"
Accessed on April 2015.
http://www.tutorialspoint.com/php/php_syntax_overview.htm

[13] "pChart Online Documentation"
Accessed on April 2015.
http://wiki.pchart.net/

[14] "AJAX Documentation and Examples"
Accessed on April 2015.
http://www.w3schools.com/ajax/

[15] "Salted Password Hashing"
Accessed on April 2015.
https://crackstation.net/hashing-security.htm#salt

[16] "Database Design and Normalization"
Accessed on April 2015.
http://www.databasedev.co.uk/database_normalization_process.html

[17] "What is Agile model?"
Accessed on April 2015.
http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/

[18] "PHP Best Practices"
Accessed on April 2015.
https://phpbestpractices.org/

[19] "11 Best Practices for PHP Development"
Accessed on April 2015.
http://www.phpbuilder.com/articles/application-architecture/optimization/explore-the-top-11-php-best-practices.html

[20] "Volere Template"
Accessed on April 2015.
http://www11.informatik.uni-erlangen.de/Lehre/WS1415/PR-SWE-WINF/Material/volere-template.pdf

[21] "6 Disadvantages of Traditional Paper-based Course Evaluations"
Accessed on April 2015.
http://www.explorance.com/blog/2013/05/6-disadvantages-of-traditional-paper-based-course-evaluations/

[22] "PHPMailer"
Accessed on April 2015.
https://github.com/PHPMailer/PHPMailer

[22] "Pratt Bootstrap Theme"
Accessed on April 2015.
http://www.blacktie.co/2013/10/pratt-app-landing-page/

[23] "User session data"
Accessed on April 2015.
http://stackoverflow.com/questions/20439686/session-returns-0-for-all-users

[24] "Client side AJAX"
Accessed on April 2015.
http://www.openajax.org/images/ClientSideAjax.gif