Grading Assistant for Lecturers and Students

Milestone 3

Sean Vowles | 13065858 | CS3D66 | 2017

Supervisors

Dr. Janusz Kulon

Dr. Bertie Muller

Statement of Originality

This is to certify that, except where specific reference is made, the work described within this project is the result of the investigation carried out by myself, and that neither this project, nor any part of it, has been submitted in candidature for any other award other than this being presently studied.

Any material taken from published texts or computerized sources have been fully referenced, and I fully realize the consequences of plagiarizing any of these sources.

Student Name (Printed)	Sean Vowles
Student Signature	………………………………..
Registered Course of Study	BSc(Hons) Software Engineering
Date of Signing	……………………………….

Abstract

This project examines the marking and analysis provided to students who submit their assignments electronically. It will look at current and alternative software availability, what universities are currently using and how feedback is being provided. The report will explain web applications, what a rubric is, provide examples of rubrics and commercial alternatives. It will include an overall software design for the system, explain certain features and show its implementation.

Table of Contents

Contents

Introduction and Objectives

The readers of this document are expected to be lecturers at a University looking to improve the speed and accuracy of grading student's assignments.

The system described in this document is intended to be used to explain the process of analyzing and designing an object-oriented software solution. The system will be the automation of marking and grading of student's assignments using a web driven application which will be explained in the methodologies which will describe design techniques involved and be supplemented by Use Case specifications, site map and Class Diagrams. The aim of the project is to create a marking assistant for lecturers. The software should assist lecturers in grading coursework and providing feedback for students and their submissions. The lecturers should be able to create, rate and insert comments, create save and use rubrics which automatically rescale, total marks and convert the mark to percentages, weighted marks and grades. The feedback should be generated automatically based on the grading obtained for individual assignment tasks. The automation of feedback should be based on the rubrics reached, thus, showing the students exactly which criteria has been achieved, and display any short-comings. The significance of rubrics is a detailed marking grid or scheme with checkpoints that can be easily identified by lecturers and students. Lecturers will also be able to insert individual feedback that is more specific to the student.

Background

Applications

Web applications and desktop applications are both able to provide the end user with the desired data and tools to improve tasks that need to be completed. Therefore, when building an application, we need to ask ourselves, what is appropriate or suitable under the circumstances.

(Pcmag.com, 2017) defines a desktop application as "*an application that runs stand alone in a desktop or laptop computer*".

Under a windows environment, an application is typically installed and configured from an executable(.exe) file. The user will choose a directory for the installation and will launch the program by clicking an icon, either in the taskbar, start menu or desktop. The application can only be launched from the computer it has been installed and usually doesn't need an internet connection to use, however, this isn't always the case as adobe state in their system requirements "*Internet connection and registration are necessary for required software activation, validation of subscriptions, and access to online services.*" (Helpx.adobe.com, 2017). The user will need to manage the updates of the software manually; depending on the size of the update defines how long it will take to download and install. If a large team of people use the same software, you may have a member of an IT department that will manage software updates. This is a large task that will take a significant amount of time, be it an update or initial install.

A web application will have a development environment and a production environment, thus allowing developers to manage updates whilst the end user continues to operate the current version. When an update is deployed, the end user doesn't have to worry about installations and configurations. A user may need to clear the cache of that website which is a simple keyboard shortcut, (ctrl + F5) for Windows and Linux or (cmd + R) for MacOS users.

Native application must be developed specific to the environment in which it is going to be used. This can range from a windows computer to Apple's iPad. There are many devices in the twenty-first century that are available to consumers making it difficult for developers to target a single platform, therefore we need to cater for a market.

(Acunetix, 2016) states that a website application is a software solution that allows users to retrieve data and interact with content within pages within a website.

For a software solution to be considered a website application it must have three components that interact with each other; a client side browser and computer, a server side system and a database server.

The client will send a HTTP/S (secure) requests to the web server. The web server response will send HTML, CSS, data, JavaScript and images. A database server holds the database engine such as SQL Server, Oracle, MongoDB. This information is sent up to the web server, which is then passed to the client if request (See Appendix A).

The client will view a web application through a browser; this may be Internet Explorer, Microsoft Edge, Google Chrome, Safari, Opera, Mozilla Firefox, iOS Safari, Android browser or one of the many other browsers that are available today.

A web application may behave as desired in one browser, but may have debilitating issues when run in another browser which could be vital to the desired outcome of the application. There are many components involved with web applications that can produce different results such as CSS rendering, JavaScript, jQuery and Ajax requests. (Segue Technologies, 2016).

When developing a website application, the developer must consider the target audient. The software developer is either going to be building a solution that is going to be available on the public domain or the software is going to be available for an intranet (a local or restricted communications network created using World Wide Web Software) of the company purchasing / subscribing to the software. When developing specific for a company's intranet, you will have rigid specification and will typically be advised what web browser will need to be supported or used. This is important as large company's often use older operating systems and web browsers; especially in the public sector. When developing web applications that are targeted for the public domain, typically, application will need to be supported on a variety of client browsers. This is a time consuming and crucial part of the development and testing phase. Using W3schools we can identify the most popular browsers and make assumptions based on popularity which browsers need to be supported (See Appendix B).

The main purpose of the web server is to store the files for the website application which is then server back to the client as a response from the client request and to send data between the data server. A website server is assigned an IP address which is how the data is accessed from the client.

A data server is the midway point of persistent storage of any information relating to an application and can be stored in either relation or non-relational database. This is normally a design specification chosen before production. Relational and non-relational databases are not a substitute for one another rather chosen on the needs of the application being developed. SQL and relational databases are excellent for representing and working with sets of data such as dealing with user accounts of a banking system. SQL and relational databases have tight rules thus also ensuring data security. However, SQL and relational databases fall short when it comes to large scaling. The larger and more complicated a database becomes, the slower the database will iterate and query. Data servers have two types of scaling; vertical and horizontal. Vertical scaling is increasing the performance of an individual server whereas horizontal scaling is spreading the data across and adding multiple servers but at the cost of security for the application. Sometimes these are referred to as 'sharding'. (Upguard.com, 2016).

An example of a non-relational database is 'MongoDb' which is defined as a NoSQL database. NoSQL database use the concept of key-value pairs which are created and stored as BSON which is a binary form of representing data called objects or documents. A key-value store, or key-value database is a data storage paradigm designed for storing, retrieving and managing associative arrays which are commonly known as a hash. NoSQL databases avoid traditional table-based database structure, which is using tables and rows in favor of JSON-like documents with dynamic schemas.

For a complete web application solution, the developer requires a stack to run its services. A stack consists of a computer with and operating system, a web server to handle the requests and responses of the web files, persistent storage to store any information or data that is crucial to the application and a language that handles the logic of the application.

There is a variety of stacks available for web applications ranging from full JavaScript stacks such as Google's using node.js as the server and AngularJS as the logic, Microsoft's C-sharp's .Net framework with IIS server or LAMP stack.

The marking assistant for lecturers is going to be developed using a Linux / Unix based operating system, apache web server, MySQL database and PHP; this is commonly referred to as the LAMP stack. A LAMP stack is open source, requiring no fees to use, it is also robust and can be used locally, allowing the developer to have a production and development environment in place making it fitting for the given tasks.

What is Rubrics

"A rubric is a coherent set of criteria for student's work that includes descriptions of levels of performance quality on the criteria" (Brookhart, 2013).

Rubrics works by diving an assignment in parts or sections that are provided with a clear description of the tasks that need to be completed (Carnegie Mellon University, 2016).

The advantages of using a rubric is that the student is being graded to a specific criterion and that the lecturer also must mark to that criteria. This promotes all round fairness between students and prevents biased marking. Grading rubrics on large courses where there may potentially be more than one person marking an assignment allows for grading consistency.

On a smaller scale, rubrics can also help tutors and students get a clear picture of the student's strengths and weaknesses in a subject or area. On a larger scale, a lecturer will be able to identify an average mark across a set of students showing the lecturer and area of teaching that needs more attention.

A rubric is typically documented in table format. The table will have a list of criteria and relevant marking scheme which reads from left to right. Once a column on the left has been achieved the student can then move onto the next column, providing students with checkpoints. When developing a software solution, it can typically be assumed that if the grade on the left most has not been achieved, it’s fair to assume the student hasn’t met the minimum requirements to pass as rubrics work on incremental grading.

Examples using Rubrics

Table A: Developing the game "Tic-Tac-Toe"

Criteria	**Fail (<40)**	**Pass (40 -59)**	**Merit (60 – 69)**	**Distinction (70+)**
Use of arrays, variables and function names / 10 total marks	Missing or demonstrating little understanding of this technique.	Showing basic knowledge and understanding of this technique.	Shows moderate understanding of arrays and functions with meaningful names.	Clearly understands and well implanted array. Data types and functions have meaningful names.
Logic and structure / 10 total marks	Game doesn't work or follow any of the rules.	Basic functionality of the game has been implemented.	Game works fluidly and detects winners and losers.	Game works fluidly, saves scores and games won by each player.
Use of classes and functions / 55 total marks	No classes or functions have been attempted.	Little use of functions. Game works but lots of redundant code.	Good use of functions shows moderate understanding of classes.	Excellent understanding and implementation of using classes and functions and applying design patterns.

Report / 25 total marks	Poor report structure, hand drawn diagrams and no referencing.	Report structure unclear, diagrams could be clearer. Diagrams and documentation have inconsistencies. Spelling mistakes in report.	Report structure clearly reflects assessment criteria. Documentation is clear, diagrams and documentation have few inconsistencies.	Report well formulated. Diagrams are clear and nearly produced. Diagrams and documentation have very few or no inconsistencies.

Table B: Example rubric from Advanced Databases and Modelling Year 3 Assignment

Criteria	Fail (<40)	Pass (40 -59)	Merit (60 – 69(	Distinction (70+)
Primary / Foreign Keys / 20 total marks	Missing or demonstrating little understanding of this technique.	Identifying primary keys and foreign keys has been attempted, but a significant number are missing, or contain significant errors.	Most appropriate primary keys and foreign keys identified. Contains some errors.	All appropriate primary keys and foreign keys identified with no errors.
Local Data Structure / 35 total marks	Missing or demonstrating little understanding of this technique. Model fails to represent case study and / or requires extensive corrections,	Model loosely represents the case study. Reasonable use of notation but contains some major errors or several small errors. Adequate attempt at rationalizing M:M	Model represents the main aspects of the case study. Most entity types & relationships included. Good use of notation and contains only a few errors. Good attempt at rationalizing M:M	Model clearly represents the case study. All appropriate entity types & relationships included. Excellent use of notation and very few errors. Clear & correct rationalizing of M:M

		relationships (if needed).	relationships (if needed).	relationships (if needed).
Assumptions / 10 total marks	Missing or inappropriate assumptions.	Some reasonable assumptions made.	A good set of assumptions, mostly appropriate.	An excellent set of appropriate assumptions.
Normalisation / 25 total marks	Little / no demonstration or understanding normalisation. Missing or poor attempt containing many errors.	Normalisation process attempted. Contains errors, and / or documentation is incomplete.	Normalisation process attempted and documented. Normalisation attempted to 3NF, but contains errors.	Demonstrates excellent understanding of normalisation. Process clearly & accurately documented. All screens / forms fully normalised to 3NF.

(Boobyer, 2016)

Table C: Example rubric from Object Oriented Systems Year 3 Assignment

Criteria	**Fail (0/29)**	**Narrow Fail (30/39)**	**3rd Class / Pass (40/49)**	**Lower 2nd Class / Pass (50/59)**	**Upper 2nd Class / Merit (60/69(**	**1st Class / Distinction (70/100)**	**Mark**
The initial class diagram (15%)	The initial class diagram is missing or totally inadequate. Association details, multiplicities attributes and methods are missing. The assumptions made for the initial class diagram	The initial class diagram is reflecting the initial design poorly. Association details, multiplicities, attributes and methods were expected but only 1 of these is present adequately. The assumpti	The initial class diagram is reflecting the initial design requirement in some areas but is deficient in others. Associations details, multiplicities, attributes and methods were expected but only 2 of these	The initial class diagram is reflecting the initial design requirements in most areas. Association details, multiplicities, attributes and methods were expected but only 3 of these are present	The initial class diagram is reflecting the initial design requirements in all areas and has very few minor faults. The association details, multiplicities, attributes and methods are all present but slightly	The initial class diagram is reflecting the initial design requirements in all areas and is without faults. Association details, multiplicities, attributes and methods are all present and of excellent quality.	11

	are missing.	ons made for the initial class diagram are poorly develope d.	are present adequatel y. The assumptio ns made for the initial class diagram are rather superficial .	adequatel y. The assumptio n made for the initial class diagram are of good quality.	incomplet e or faulty. The assumptio ns made for the initial class diagram are very good quality.	The assumptio ns made for the initial class diagram are excellent showing full understan d of problem areas.	
Inform ation display proble m (15%)	Discussio n of problem and assumptio ns missing. A review of pattern choices was not done. No reference	Discussio n of problems and assumpti ons are merely repetition s of the task text. Pattern/p rocess choices were reviewed	Discussio n of problem and assumptio ns are minimal / obvious and stay on the surface. Pattern choices were reviewed	Discussio n of problems and assumptio ns are good. A reasonabl e attempt was made to explain what effect each of	Discussion of problem and assumptio ns are very good. A well argues case was presented that explains what effect	Discussion of problem and assumptio ns are excellent and of great depth. An excellently argued case was presented	10. 5

	or engagement or use of refencing in the text. No (or more than one) pattern was selected. Initial diagram not updated.	only in general (generically) and not with a view towards application to this problem. Very few references and little or no engagement or use of the references in the text. Diagram updated but almost completed incorrect.	mainly in general (generally) but is not explained what these mean for this application. Only a single reference was found and used.	the respective 3 patterns would have on the application. Two or three reference were found and discussed in some detail. A non-optimal pattern was selected. Initial diagram updated but with a few mistakes.	each of the respective 3 patterns would have on the application. Four or more reference were found and discussed in detail.	that explains in detail what effect each of the respective 3 patterns would have on the application. Four or more reference were found and discussed in great detail. The best pattern was selected. Initial Diagram	

						updated perfectly.	
Component editing problem (15%)	Discussion of problem and assumptions missing. A review of pattern/process choices was not done. No references or no engagement or use of references in the text. No (or both) pattern or	Discussion of problem and assumptions are merely repetitions of the task text. Pattern/process choices were reviewed only in general (generically) and not with a view towards application to this problem. Very few reference	Discussion of problem and assumptions are minimal / obvious and stay on the surface. Pattern/process choices were reviewed mainly in general (generically) but is not explained what these mean for this	Discussion of problem and assumptions are good. A reasonable attempt was made to explain what effect each of the respective 3 pattern/process would have on the application. Two or three	Discussion of problem and assumptions are very good. A well argued case was presented that explains what effect each of the respective 3 patterns/process would have on the application.	Discussion of problem and assumptions are excellent and of great depth. An excellently argued case was presented that explain in detail what effect each of the respective 3 patterns/process would have on	11

	process was selected. Diagram not updated.	s and little or no engagement or use of the references in the text. Diagram updated but almost completely incorrect.	application. Only a single reference was found and used.	reference were found and discussed in some detail. Diagram updated but with a few mistakes.	Four or more references were found and discussed in some detail.	the application. Four or more references were found and discussed in great detail. Diagram updated perfectly.	
Undo Problem (15%)	Discussion of problem and assumptions missing. A review of pattern choices was not done.	Discussion of problem and assumptions are merely repetitions of the task text. Pattern choices were	Discussion of problem and assumptions are minimal / obvious and stay on the surface. Pattern choices	Discussion of problem and assumptions are good. A reasonable attempt was made to explain what	Discussion of problem and assumptions are very good. A well argued case was presented that explains	Discussion of problem and assumptions are excellent and of great depth. An excellently argued	10.5

	No reference or no engagement or use of reference in the text. No (or both) pattern was selected. Diagram not updated to final version.	reviewed only in general (generically) and not with a view towards application to this problem. Very few reference and little or no engagement or use of the reference in the text. Diagram updated but almost completely incorrect.	were reviewed mainly in general (generically) but is not explained what these mean for this application. Only a single reference was found and used.	effect each of the respective 2 patterns would have on the application. Two reference were found and discussed in some detail. Diagram updated but with a few mistakes.	what effect each of the respective 2 patterns would have on the application. Three or more references were found and discussed in some detail.	case was presented that explains in detail what effect each of the 2 patterns would have on the application. Three or more references were found and discussed in great detail. Diagram updated perfectly to final diagram.	

Referencing (8%)	No reference.	Reference style in 'Reference' section incorrect. 1 to 3 or less reference. Reference style in main text incorrect.	References style in 'Reference' section occasionally correct. 4 or 5 references. Reference style in main text occasionally correct.	Reference style in 'Reference' section mostly correct. 6 or 7 references. Reference style in main text mostly correct.	Reference style in 'Reference' section good but with few mistakes or partially incomplete. 7 or 8 references /	Reference style in 'Reference' section is perfect (or close to it). 9 or more references. References style in main text fully correct.	8
Translation into code (20%)	No code or code does not work at all.	Very little code that does not work or is very rudimentary. Hardly any of the implemented code	A minimum amount of code that covers about half of the required functionality.	A minimum amount of code that covers about three quarters of the required	A minimum of code that covers almost all of the required functionality. All of the implement	A minimum amount of code that covers all of the required functionality. All of the implemented code	14.5

		works correctly.	About a third of the implementation code works correctly.	functionality. About three quarters o the implemented code works correctly.	ed code works correctly.	works correctly and has some additional features (e.g. input validation).	
Testing (12%)	Testing was not performed.	Only 1 or 2 of the challenges in Task 2 were tested. The test documentation is rudimentary.	All three challenges in Task 2 were tested but at a very superficial level. The test documentation is technically complete but makes it difficult to either follow the	All three challenges in Task 2 were tested at a good level. The test documentation is complete but has some omissions. Some attempts of	All three challenges in Task 2 were tested at a very good level. The test documentation is complete and correct but could benefit from more structure	All three challenges in Task 2 were tested exhaustively. The test documentation is complete, well structures and well laid out.	7.5

			procedure or interpret the result. No or very little structuring present.	structuring evident.	or better layout.		
Total (100%)							73 %

(Plassman, 2017)

The instructional rubric in Table A and Table B both have components of similarity. They both have criterion to meet and a marking scheme to follow. Using the criteria and marking scheme that is well written and documented allows students and lecturers to identify strengths and weaknesses in their work. Table C has more criteria's and targets to build up the outcome of an assignment and also providing more information with an emphasis on providing students with detailed information of weaknesses and short comings of a completed assignment.

Commercial Alternatives

The technique of using rubrics to describe a marking scheme of an assignment is becoming more popular in the twenty-first century of teaching. It is imperative that software houses keep up with the trend developing application that support academia and presenting diverse types of programs from word plugins to standalone applications. There is currently a handful of powerful rubric creation tools available to assist lecturers and teachers producing well-documented digital evaluation rubric for their students. However, software that assists the lecturers and teachers in marking is particularly scarce with only three reputable software platforms available: ‘Rubric-O-Matic’, ‘Moodle’ and ‘GradeMark’. Rubric-O-Matic is an extension for Microsoft Word, meaning the user needs a license and an operating system that supports this product (eMarking Assistant, 2016). GradeMark is a robust standalone application developed by Turnitin, available on a subscription basis developed for use on the web backed by persistent storage. Rubric-O-Matic supports penalties and bonuses, total marks, percentage conversion and assignment weight. Rubric-O-Matic is tied to singular word documents meaning it cannot weight a module across various assignments. Moodle’s rubric application is a plugin for their own web application which does not contain as many features as their competitor Turnitin. It supports a grid view much like the Rubric-O-Matic word plugin, a rubric can be clicked which will highlight an element in the grid and will add the scores once each criterion has a rubric mark assigned to it. A rubric has points associated to it where 5 is exceptional and 0 is poor. This makes the marking system rigid and doesn’t show the student or teacher how close the student was to achieving the next point up. Turnitin Refer to this type of marking as a ‘qualitive rubric’. (Docs.moodle.org, 2017).

Presently there is a shortage in commercial availability to choose from such a robust system allowing the company ‘turnitin’ to monopolize the market worldwide in which they state they are trusted by over fifteen thousand institutions and thirty million students (turnitin, 2017).

Turnitin products are offered to the consumer as a subscription at the institution level which is packaged as one of the largest enterprise level learning management system available as a web application. (turnitin, 2017).

Turnitin prices are not transparent and pricing guidelines are not available directly from the company's website. We can look at an article written in 2014 which is slightly outdated claiming that an employee from Glasgow University stated renewing its license with 'turnitin' would cost almost £25,000 a year. (Parr, 2014).

Turnitin have branded their rubrics marking system as 'GradeMark'. Turnitin state the instructor is able to edit and grade student papers online. The instructor can add comments within the body of the paper, point out grammatical and punctuative errors, evaluate assignment against qualitive or quantitative rubrics which as a feature can be automatically saved in 'GradeBook'.

A 'GradeMark' comment is equivalent to the notes that a teacher would typically write in the margins of a paper. Individual comments are limited to one thousand characters and can be edited or deleted at any time.

A 'GradeMark' rubric scorecard can be used to evaluate students work based on a defined criteria and scales. This is emblematic to a standard rubric marking system, however, 'GradeMark' supports custom rubrics and qualitive rubrics. A custom rubric allows any value to be entered directly into a cell and the sum of the rubric will be the highest value entered in each of the criteria rows. A qualitive rubric has the same outlines of a standard rubric but does not have numeric scoring. (Turnitin Instructor User Manual, 2017).

Design

Site Map

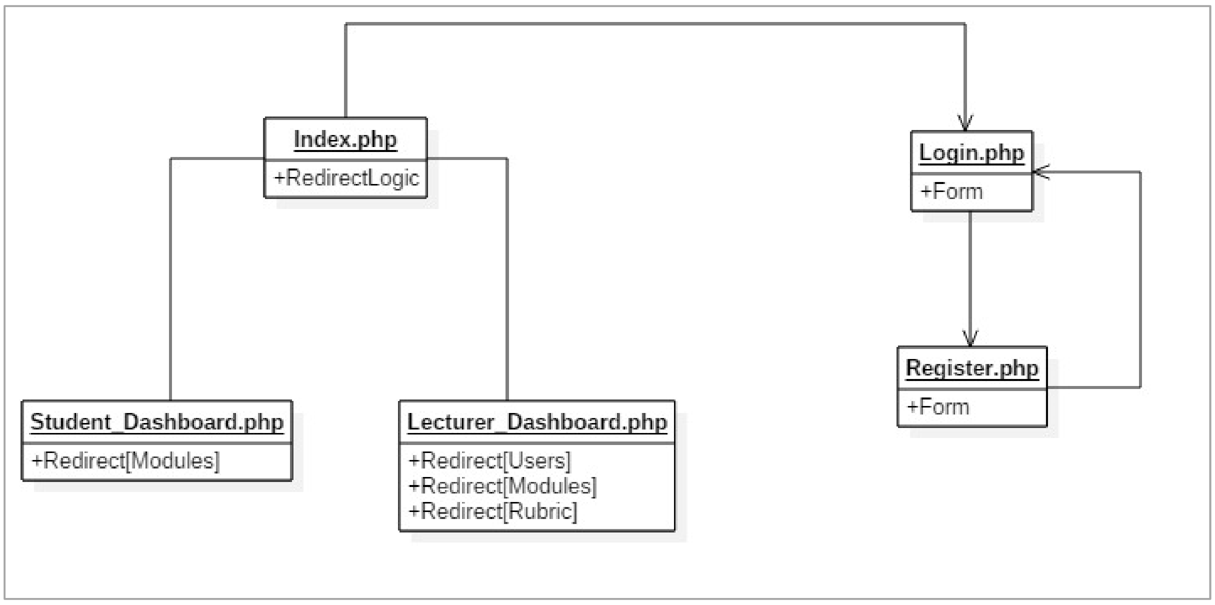


Figure 1 List of initial pages

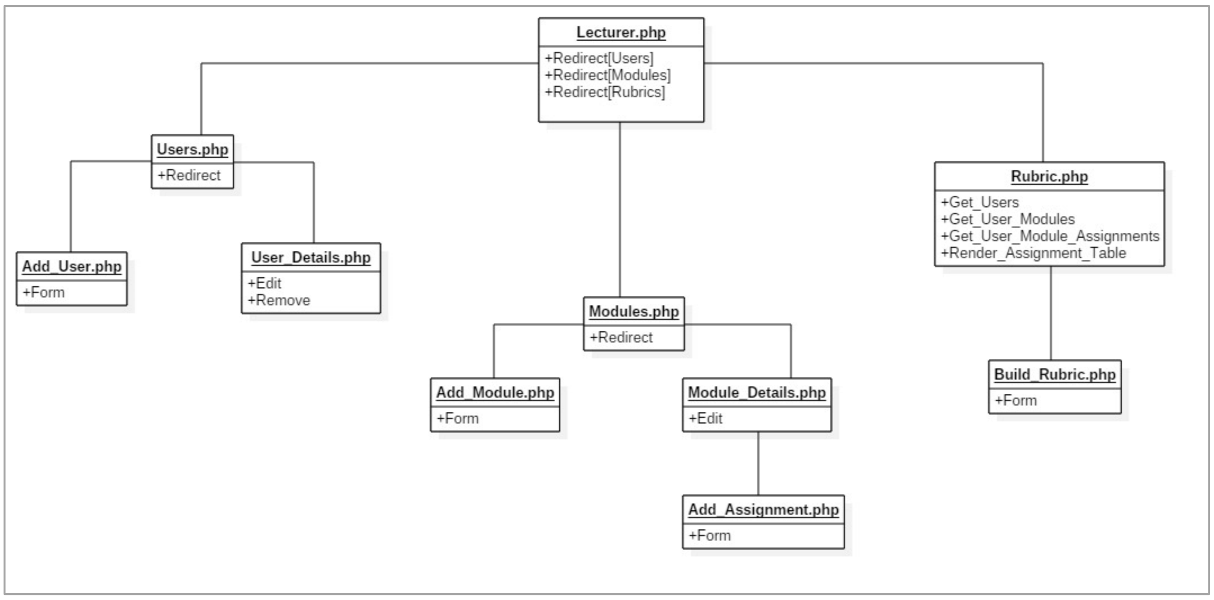


Figure 2 List of pages accessible to administrator users

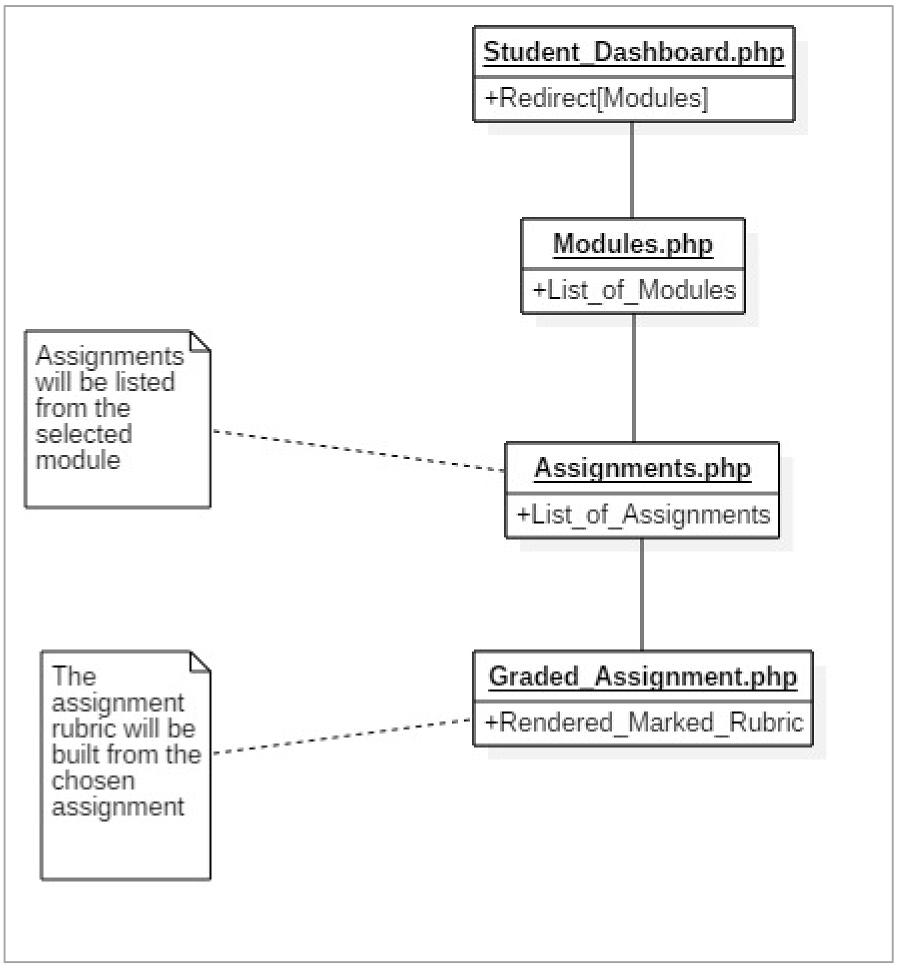


Figure 3 List of pages accessible to student users

When a user sends an initial web request to access the rubric web site they will be automatically directed to the index.php page which will perform a check to determine if the user is logged in. A redirect will then be performed based on the user's status (see Figure 1).

The Login.php page will contain the logic for a user to log in by taking a user name and password credentials. If the credentials match the database the user will be redirected to index.php The page will also have an option to be redirected to a register page if a user does not have an account (see Figure 1).

The Register.php page will contain several form fields required for a user to register an account to the database. (see Figure 1)

The Rubric.php will contain list of users, their associated modules and assignments, when an assignment is chosen, a table will be rendered to the page allowing marks to be assigned to the assignment (see Figure 2).

Build_Rubric.php will contain a list of available assignments from the database and build a rubric criterion for that assignment (see Figure 2).

Modules.php will contain a list of modules that are available. From here, the user will be able to add an additional mode or associate an assignment to the module (see Figure 2).

If the user is of type student they the page will contain a list of modules associated to the logged in user (see Figure 3).

Assignments.php will display a list of assignments associated to the module that the user selected (see Figure 3).

Graded_Assignment.php will render a table displaying the marks provided by the lecturer for the chosen assignment (see Figure 3).

Users.php will contain a list of users which can be edited or removed from the database. A redirect will be available to add a user to the database which will contain a form (see Figure 2).

Use Case Diagram

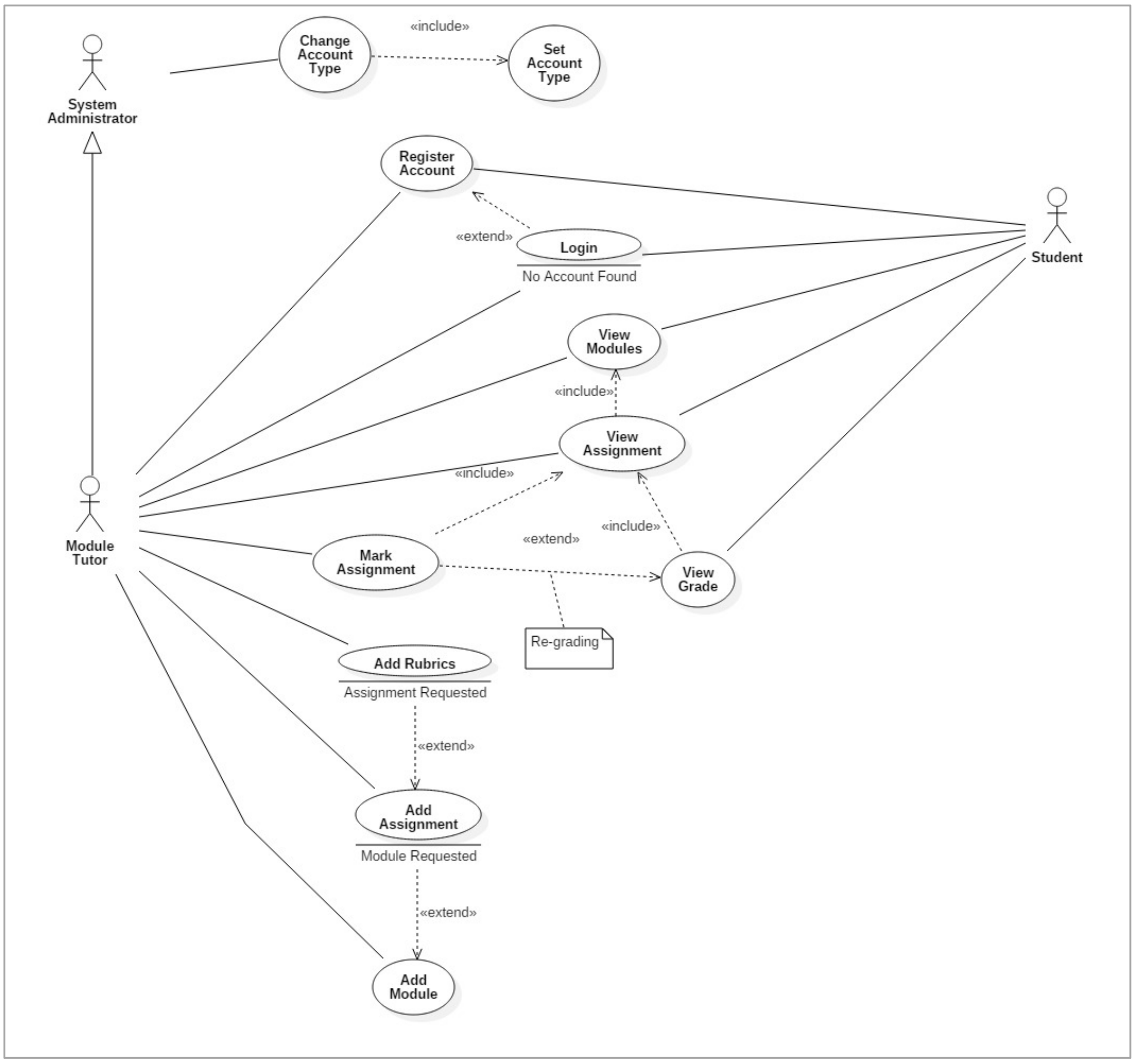


Figure 4 Use Case diagram of the system

Overview: Student

1. System prompts for user details
2. User inputs details
3. System checks if account exists
4. [Extension: Register Account]
5. System will check for incorrect login
6. System will show available modules
7. User will choose module
8. System will show available assignments
9. User will choose assignment
10. System will display the grade given

Extension Points:

Register Account: User must register an account

Use Case: Add Module

1. Tutor will request to add module
2. Tutor will input module details
3. Module details will be saved to the database

Use Case: Add Assignment

1. Tutor will request to add an assignment
2. [Extension: Add module]
3. Tutor will input assignment details
4. Assignment details will be saved to the database

Extension Points:

Add module: Module must exist

Use Case: Add Rubric

1. Tutor will request to add a rubric
2. [Extension: Add assignment]
3. Tutor will input rubric criteria
4. Tutor will input fail rubric
5. Tutor will input pass rubric
6. Tutor will input merit rubric
7. Tutor will input distinction rubric
8. Tutor will insert the overall weight of the individual rubric

Extension Points:

Add assignment: An assignment must exist for a rubric to be created

Use Case: Mark Assignment

1. Tutor will select a module
2. Tutor will select an assignment
3. Tutor will mark the rubrics of each criteria associated with the assignment

Use Case: Regrade Assignment

1. Tutor will select a module
2. Tutor will select an assignment
3. Tutor will mark the assignment

4. [Extension]: View grade

Extension Points:

View grade: A graded assignment must exist to be remarked

Use Case: Register Account

1. User will insert user name
2. User will insert email address
3. User will insert a password
4. System will check username doesn’t exist
5. System generated unique ID for user
6. System stores the new records

Class Diagram

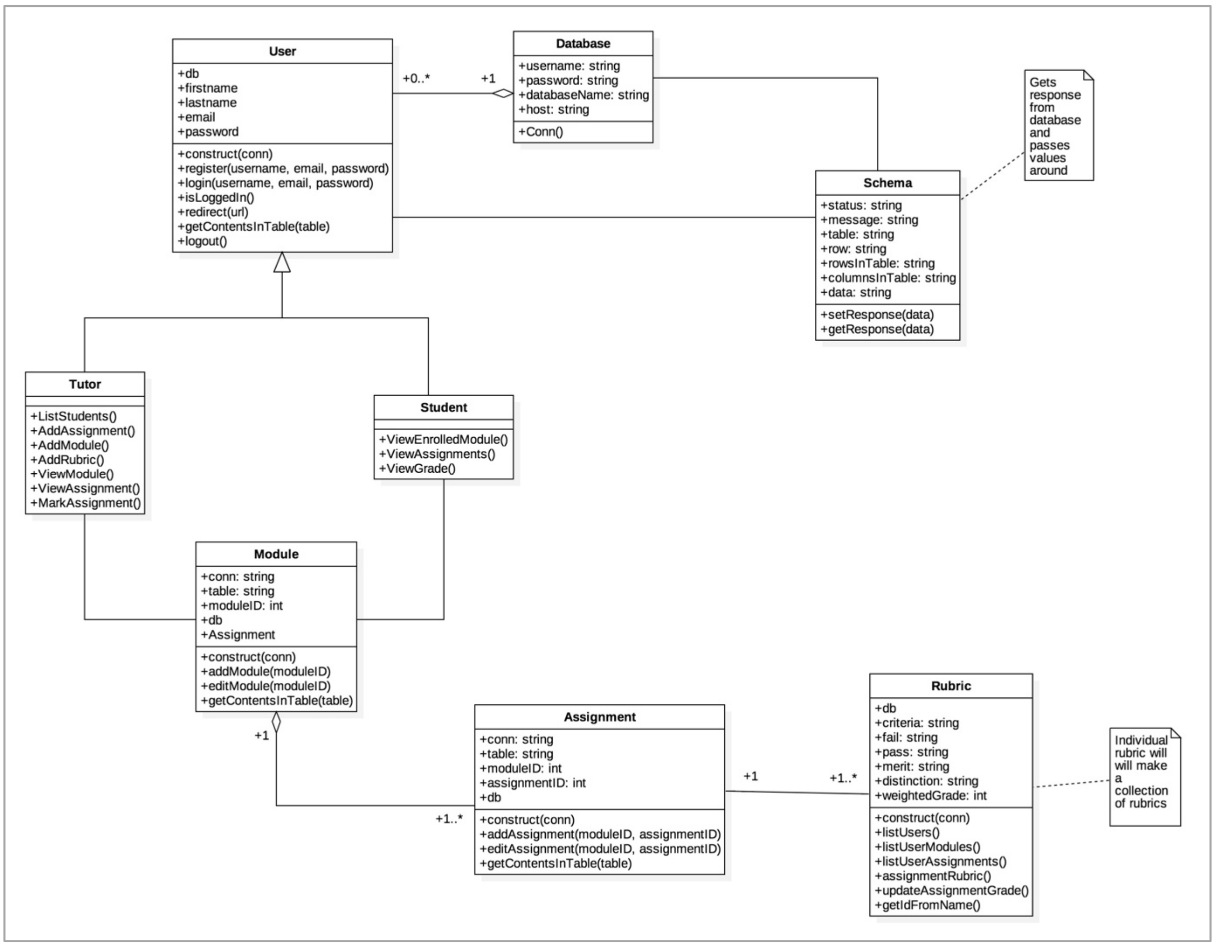


Figure 5 Class diagram representing the system

Database:

- Establishes a connection to the external database

User:

- Has an attribute of 'db' to reflect the aggregation relationship to the database
- Connection is passed to the constructor from the database
- Registers a user
- Logs in a user
- Logs out the user
- Checks the user's connection state
- Handles redirects based on connection

Tutor:

- Lists users
- Lists modules

- Lists assignments
- Marks assignments

Student:

- View their enrolled modules
- View their enrolled assignments
- View their assignment grades

Module:

- Has an attribute of assignment to represent the aggregation relationship to the assignment class
- Add module
- Edit module

Assignment:

- Add assignment
- Edit assignment

Rubric:

- Add rubric
- Mark rubric (Updates assignment grade)
- Gets list of students
- Gets list of modules
- Gets list of assignments

Schema

- Sets response
- Gets response from database

The schema class is associated to the entire model returning data from queries passed to the database.

Functions of classes that query a database will be returned as JSON data which can then be asynchronously called to prevent multiple page loads.

An assignment is going to be marked out of a total of 100 points. The rubric will hold the criteria grade weight and a calculation will be performed to generate a percentage. This will be rescaled and posted to the database.

Entity Relationship Diagram

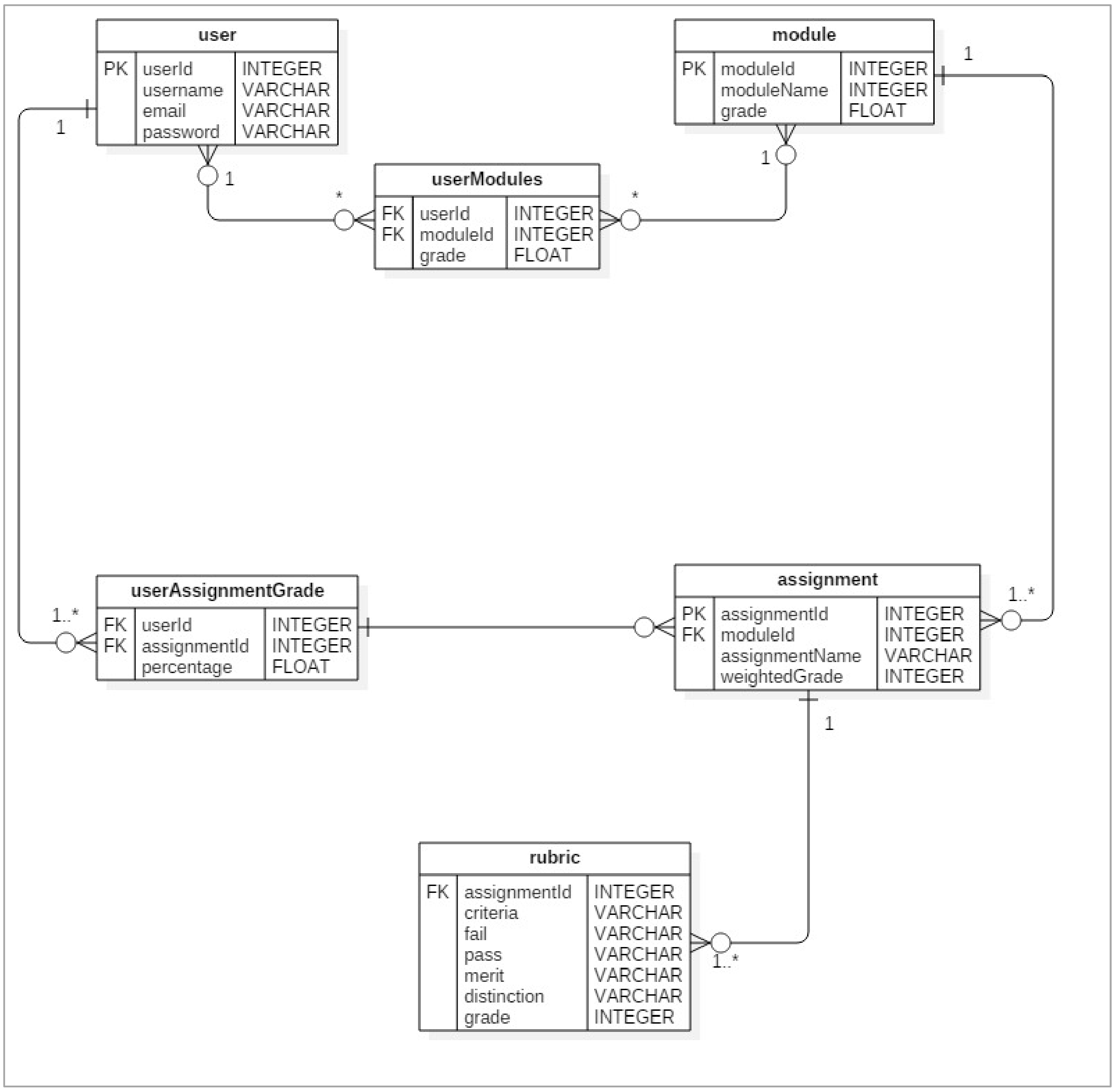
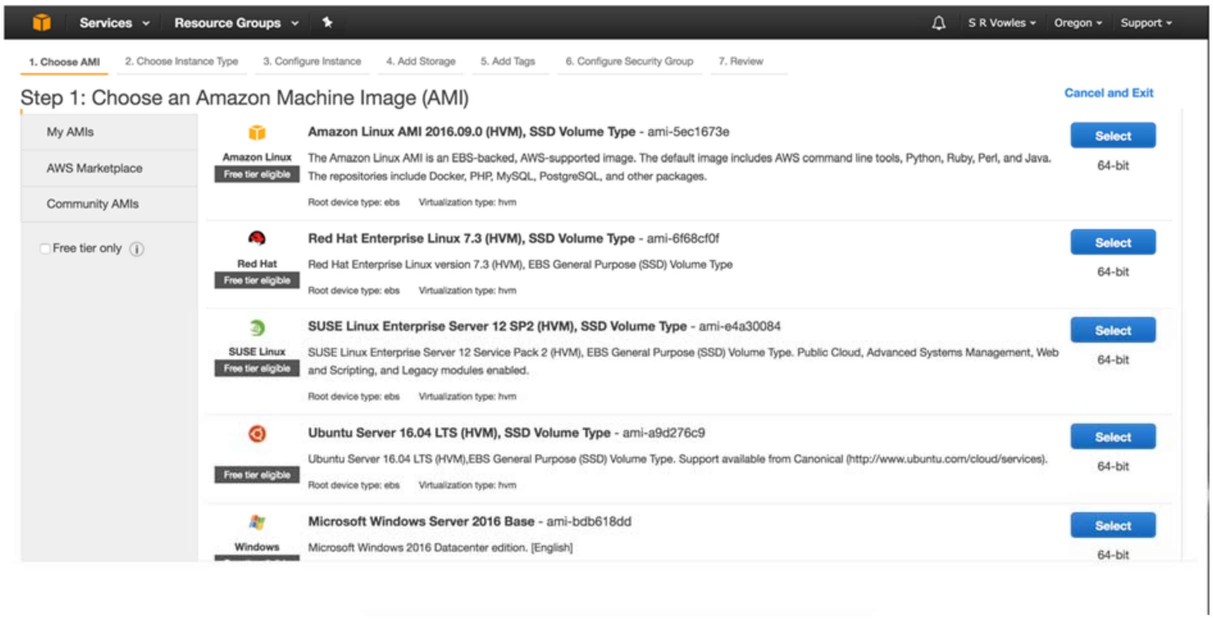


Figure 6 Entity Relation Diagram

Implementation

The marking assistant for lecturers is going to be hosted on a virtual machine in a cloud environment. The cloud environment of choice is going to be Amazon Web Services. Amazon Web Services offer secure platforms, database storage, content delivery and a free one year subscription of up to 750 computing hours per month.

Initially, a virtual machine will be launched using the Amazon Web Services control panel. Amazon provides a list of stable bootable operating systems that can be installed on their virtual machines using one click.



Step 1: Choose an Amazon Machine Image (AMI)

Amazon Linux AMI 2016.09.0 (HVM), SSD Volume Type - ami-5ec1673e
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm
64-bit

Red Hat Enterprise Linux 7.3 (HVM), SSD Volume Type - ami-6f68cf0f
Red Hat Enterprise Linux version 7.3 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm
64-bit

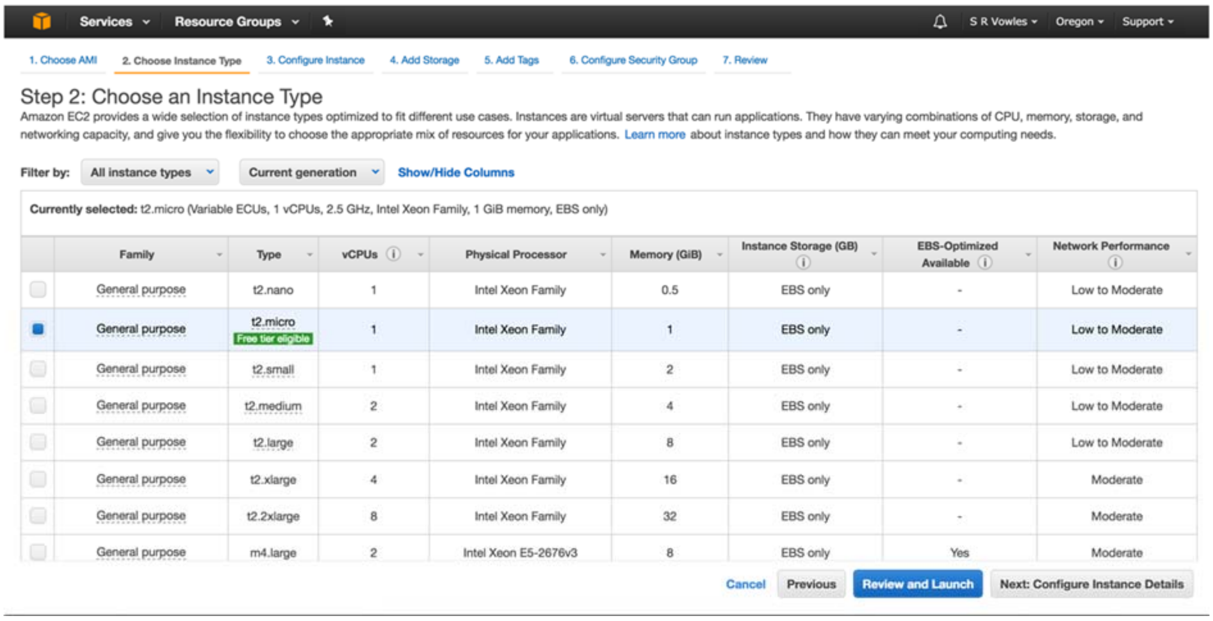
SUSE Linux Enterprise Server 12 SP2 (HVM), SSD Volume Type - ami-e4a30084
SUSE Linux Enterprise Server 12 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs Virtualization type: hvm
64-bit

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-a9d276c9
Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs Virtualization type: hvm
64-bit

Microsoft Windows Server 2016 Base - ami-bdb618dd
Microsoft Windows 2016 Datacenter edition. [English]
64-bit

Figure 7 Choosing an Amazon Machine Image

Once an operating system has been selected the user will be directed to the instance configurations. The user will be able to choose from a list of instances which will show how many CPU's the instance has, the processor type, physical memory, storage space and network performance. A user can choose a basic instance 't2.micro' for initial development and deployment and scale up the system to meet their requirements later.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

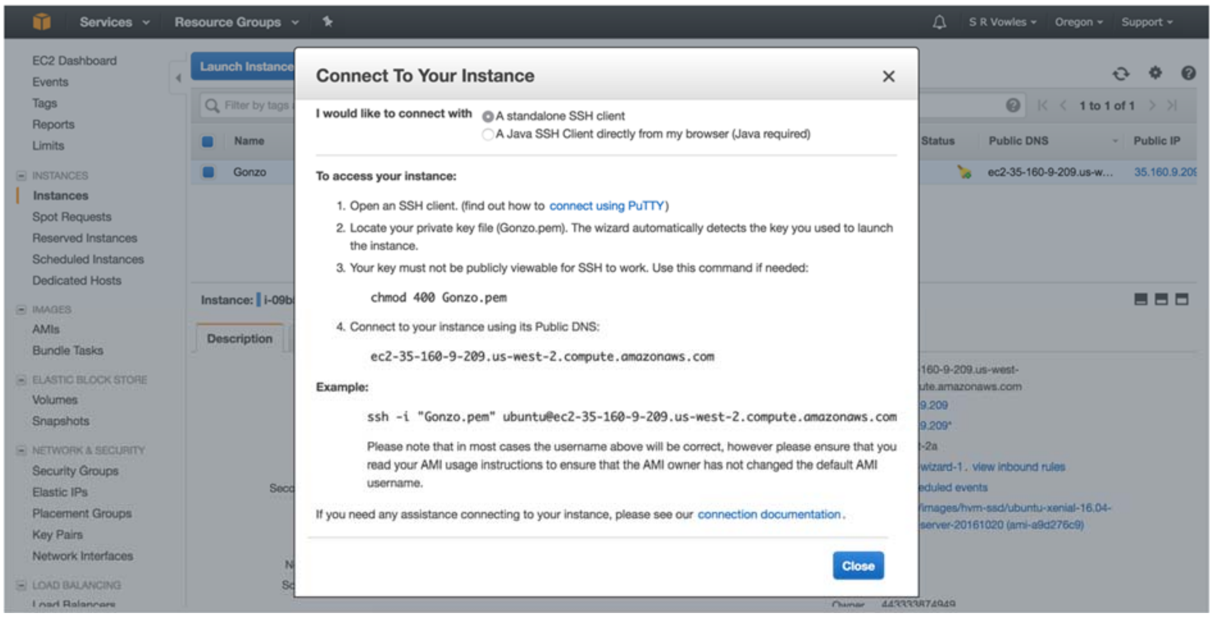
Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Physical Processor	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
	General purpose	t2.nano	1	Intel Xeon Family	0.5	EBS only	-	Low to Moderate
■	General purpose	t2.micro Free tier eligible	1	Intel Xeon Family	1	EBS only	-	Low to Moderate
	General purpose	t2.small	1	Intel Xeon Family	2	EBS only	-	Low to Moderate
	General purpose	t2.medium	2	Intel Xeon Family	4	EBS only	-	Low to Moderate
	General purpose	t2.large	2	Intel Xeon Family	8	EBS only	-	Low to Moderate
	General purpose	t2.xlarge	4	Intel Xeon Family	16	EBS only	-	Moderate
	General purpose	t2.2xlarge	8	Intel Xeon Family	32	EBS only	-	Moderate
	General purpose	m4.large	2	Intel Xeon E5-2676v3	8	EBS only	Yes	Moderate

Figure 8 Choosing an instance type

Once an instance is successfully set up it will be operating with a clean operating system installation. For the marking assistant, the operating system of choice is Ubuntu Server, which is a Linux based operating system with only a command line interface. For less experienced users, Ubuntu server can have a desktop environment installed on top and be accessed using remote desktop connection server.



Connect To Your Instance

I would like to connect with [✓ A standalone SSH client] A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (Gonzo.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 Gonzo.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-35-160-9-209.us-west-2.compute.amazonaws.com
```

Example:

```
ssh -i "Gonzo.pem" ubuntu@ec2-35-160-9-209.us-west-2.compute.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

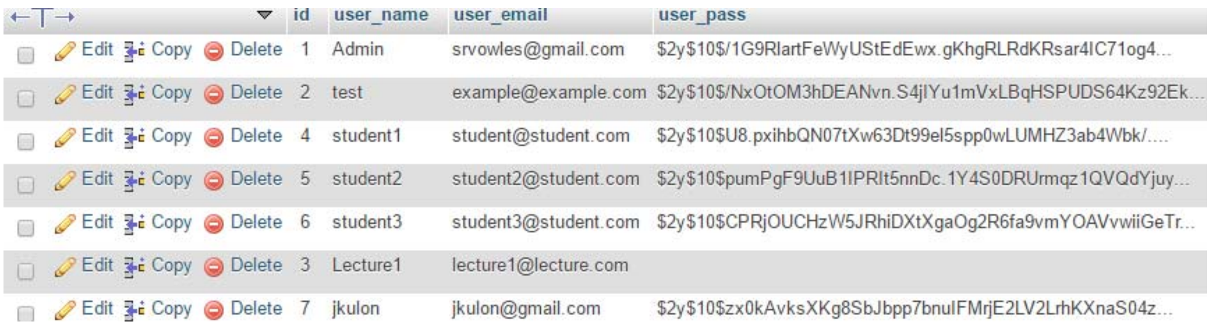
If you need any assistance connecting to your instance, please see our connection documentation.

Figure 9 Information regarding the instance

Upon completed the virtual private server the user is prompted with a dialogue box which provides the user with information on how to access the server using an SSH connection.

For this project, the operating system will be left at its default and only allowing SSH connections. For this, anyone who needs access to the server must request an SSH key from the system administrator.

For security purposes, upon creating an account, PHP's PDO statement has been used. This is where a parameter is passed to a PHP data object to build a query. The parameters are local to the user class register function and cannot be modified from the client and only takes specific values. The register function also instantiates a password hasher object, taking the user_password string as a parameter and hashes it. The hashed password is then stored in the database and requires PHP to decrypt the password when attempting to log in. If an attempt is made to log into an account with the hashed password without being decrypted this will not work.



←T→				id	user_name	user_email	user_pass
	Edit	Copy	Delete	1	Admin	srvowles@gmail.com	$2y$10$/1G9RlartFeWyUStEdEwx.gKhgRLRdKRsar4lC71og4...
	Edit	Copy	Delete	2	test	example@example.com	$2y$10$/NxOtOM3hDEANvn.S4jlYu1mVxLBqHSPUDS64Kz92Ek...
	Edit	Copy	Delete	4	student1	student@student.com	$2y$10$U8.pxihbQN07tXw63Dt99el5spp0wLUMHZ3ab4Wbk/....
	Edit	Copy	Delete	5	student2	student2@student.com	$2y$10$pumPgF9UuB1IPRIt5nnDc.1Y4S0DRUrmqz1QVQdYjuy...
	Edit	Copy	Delete	6	student3	student3@student.com	$2y$10$CPRjOUCHzW5JRhiDXtXgaOg2R6fa9vmYOAVvwiiGeTr...
	Edit	Copy	Delete	3	Lecture1	lecture1@lecture.com	
	Edit	Copy	Delete	7	jkulon	jkulon@gmail.com	$2y$10$zx0kAvksXKg8SbJbpp7bnuIFMrjE2LV2LrhKXnaS04z...

Figure 10 Example of hashed password in database

Login implementation:

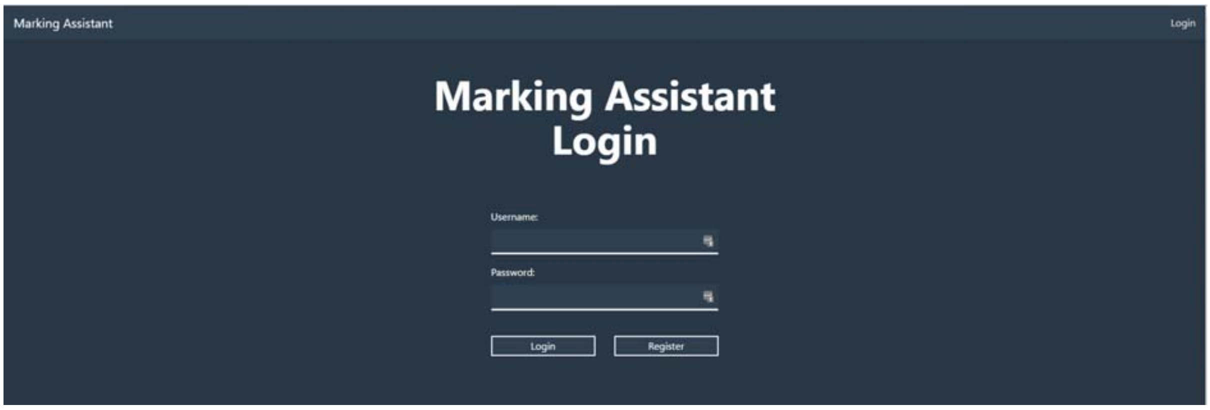


Figure 11 Login page

Initial page presents the users with a means to log in and a call to action to create an account if one does not exist for the users

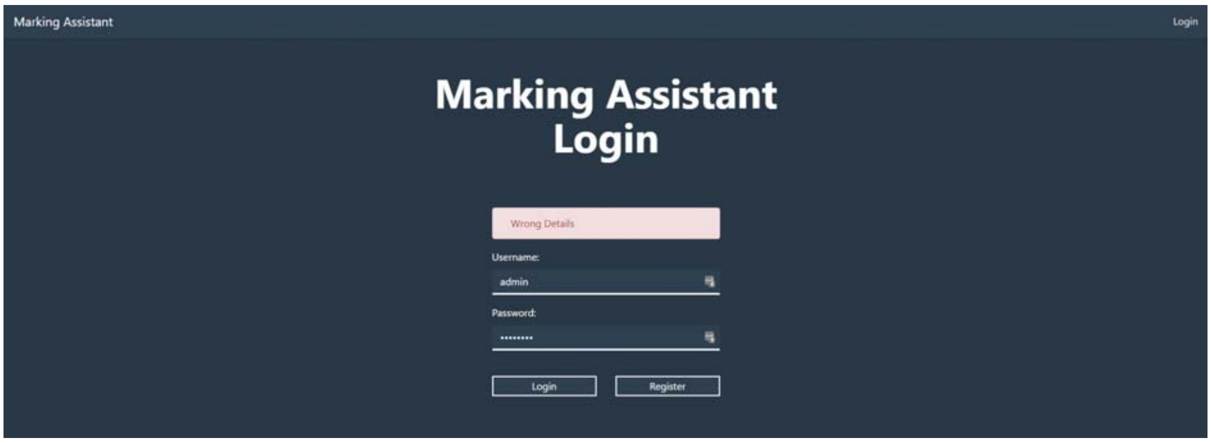


Figure 12 Incorrect details view

The user is notified that the details entered are incorrect

```
$stmt = $this->db->prepare(
                'SELECT * FROM user
                WHERE user_name=:user_name
                OR user_email=:user_email LIMIT 1');

            $stmt->execute(array(
                ':user_name' => $user_name,
                ':user_email' => $user_email));
```

The ':' before user_name or user_email is an example of building a prepared statement. The variable is then passed in the execute function.

```
if (password_verify($user_password, $user_row['user_password'])) {
                    $_SESSION['user_session'] = $user_row['user_id'];

                    return true;
                } else {
                    return false;
                }
```

The password is decrypted by PHP's password_verify method taking the password as a parameter. The user_row is an array of the user table and the password is identified by finding the row named 'user_password'.

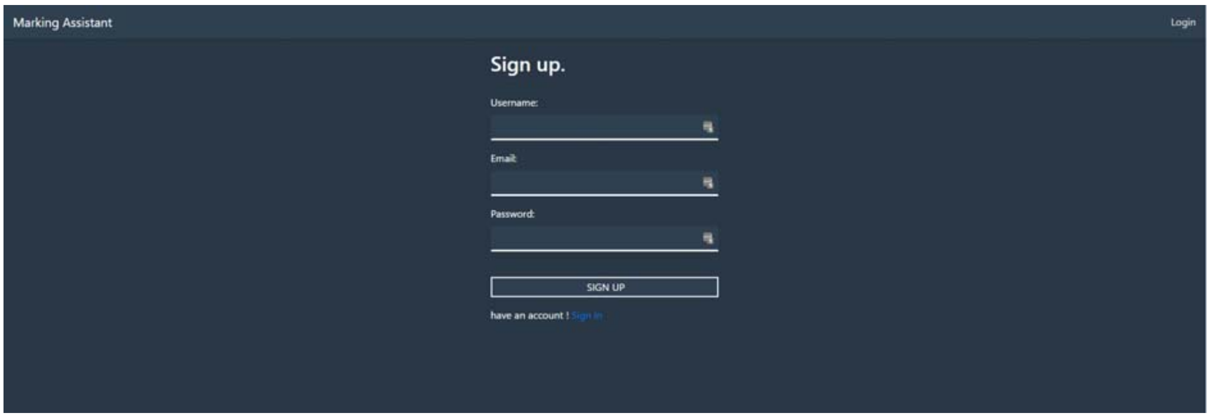
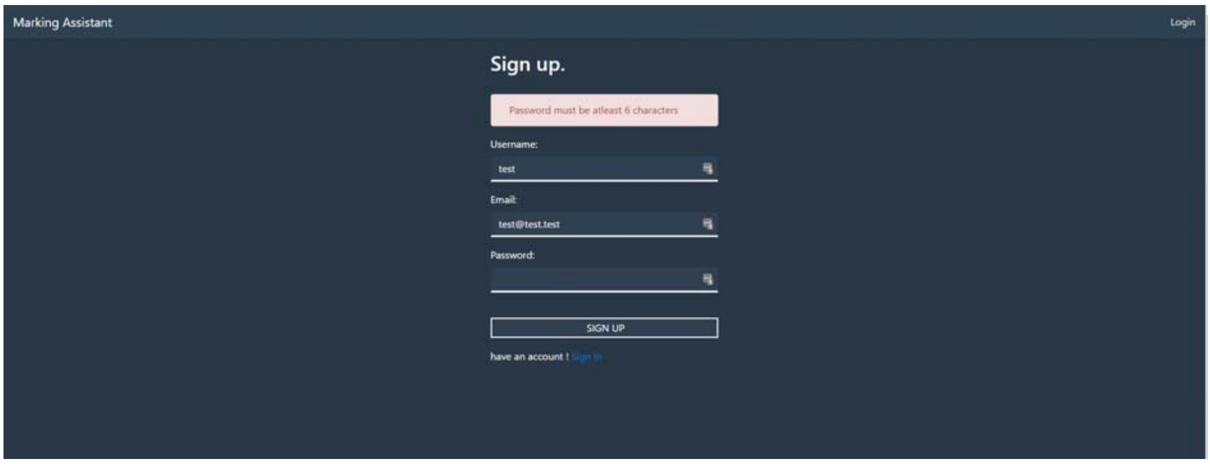


Figure 13 Registration page

The user is resented with a registration form to begin using the application. If the user has an account, they can redirect to the login page from a link at the top right of the navigation bar, which is common practice across multiple web applications. There is also an additional link at the bottom of the form.

```php
public function register($user_name, $user_email, $user_password)
    {
        try {
            $new_password = password_hash($user_password, PASSWORD_DEFAULT);

    // SQL statement to be held in variable
    $stmt = $this->db->prepare(
        'INSERT INTO user(user_name, user_email, user_password)
        VALUES(:user_name, :user_email, :user_password)');

            $stmt->bindparam(':user_name', $user_name);
            $stmt->bindparam(':user_email', $user_email);
            $stmt->bindparam(':user_password', $new_password);
            $stmt->execute();

            return $stmt;
        } catch (PDOException $e) {
            echo $e->getMessage();
        }
    }
```

The register function takes a user name, email and password as parameters. The function then calls PHP’s password hash function which takes the user password parameter. The hashed password is stored into a variable which is passed into a PHP data object.



Sign up.

Password must be atleast 6 characters

Username: test

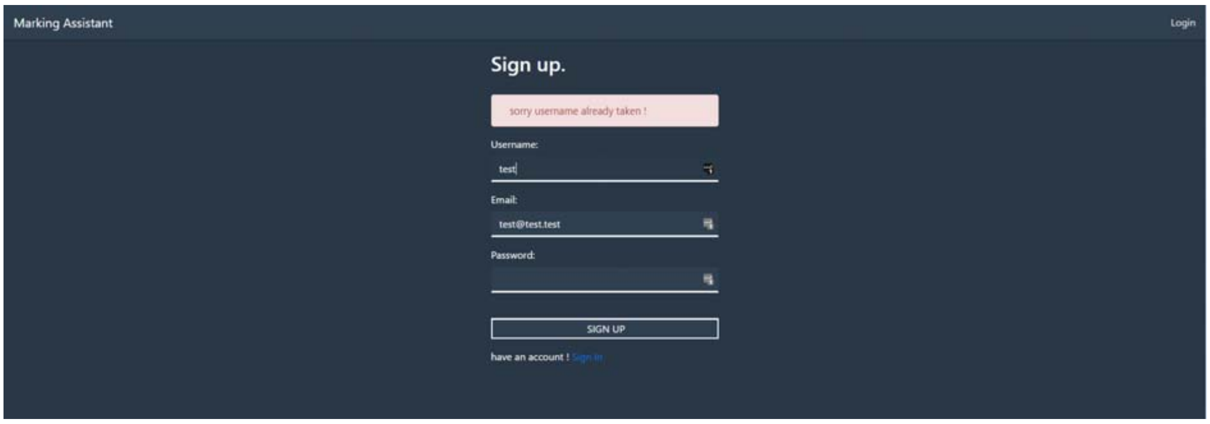
Email: test@test.test

Password:

have an account ! Sign In

Figure 14 Not enough characters

The user must have a password length of at least six characters for basic security measures.



Sign up.

sorry username already taken !

Username: test

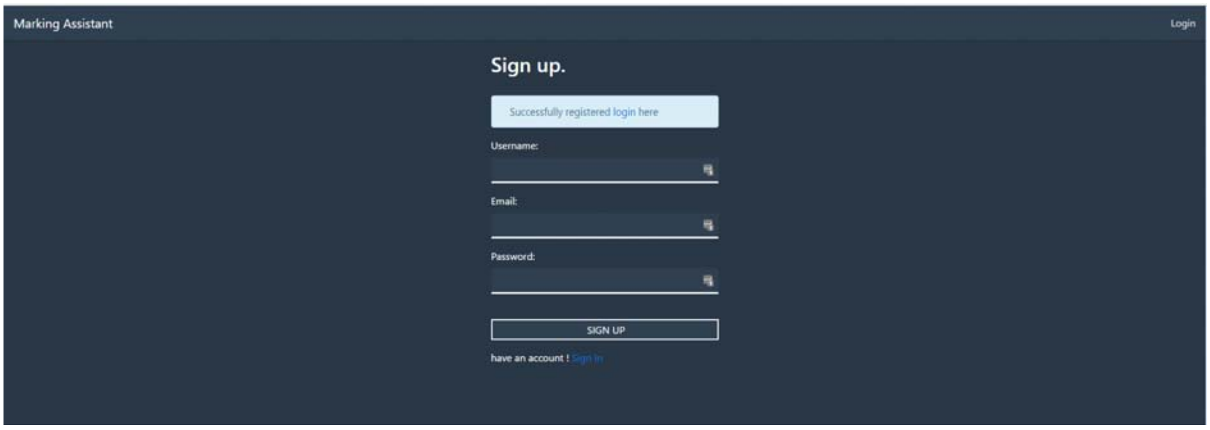
Email: test@test.test

Password:

have an account ! Sign In

Figure 15 Username is already taken

The system performs a check on the database to evaluate whether the user name already exists in the user table. If the user name exists the user is presented with an error message.



Sign up.

Successfully registered login here

Username:

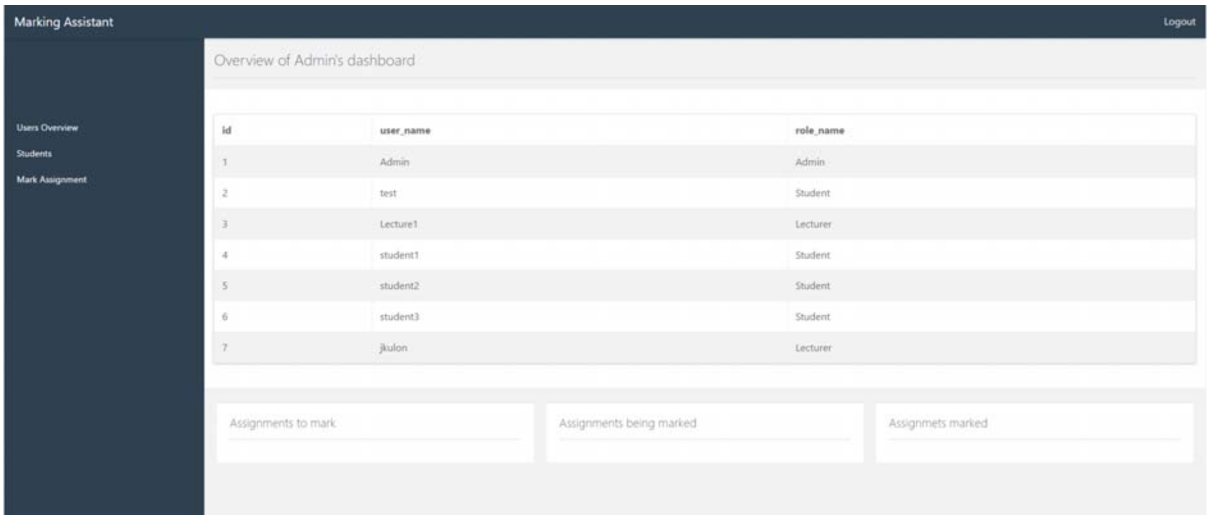
Email:

Password:

have an account ! Sign In

Figure 16 Registration is successful

User is notified the account has been created successfully and is given a third link within the notification box to redirect to the login page. The third link appears in the notification message, it’s no necessary but it’s good practice to include a call to action with a message.



Overview of Admin's dashboard

id	user_name	role_name
1	Admin	Admin
2	test	Student
3	Lecture1	Lecturer
4	student1	Student
5	student2	Student
6	student3	Student
7	jkulon	Lecturer

Assignments to mark

Assignments being marked

Assignmets marked

Figure 17 Admin dashboard

Basic implementation of the administration dashboard presenting the administrator with a list of users and their account level within a HTML table. The top right navigation bar link has changed text to logout. Here the user can see they are logged in and have a secure way to close the connection before they exit the application.

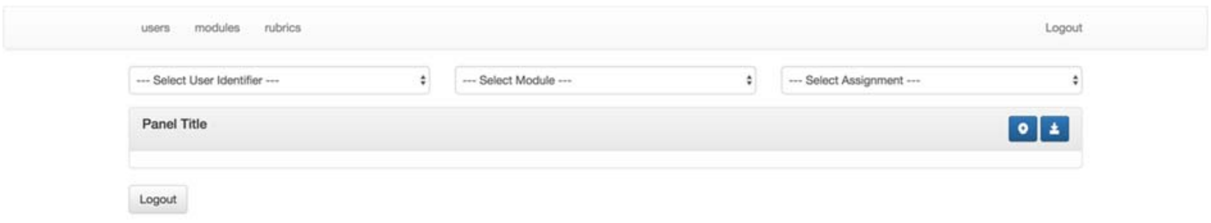


Figure 18 Initial rubric page for tutor

A tutor is initially provided with an empty page consisting of three HTML select boxes. The page is generated by a series of asynchronous calls to methods within the rubric class.

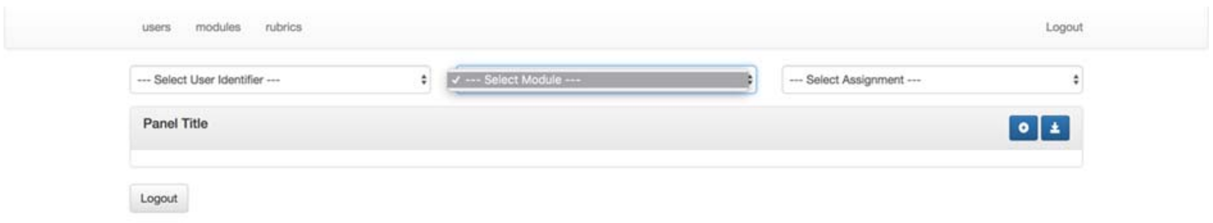


Figure 19 No modules or assignments

A tutor cannot select an assignment or module until they have chosen a user they wish to grade.

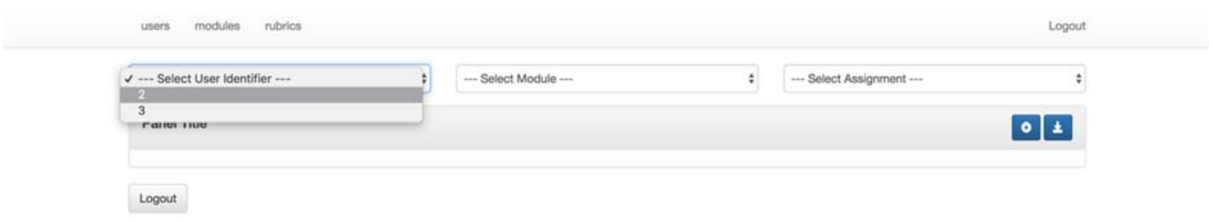


Figure 20 List of users in select box

A list of users is displayed in the first select box which must be selected before information becomes available in the module select box.

```php
public function list_all_users()
    {
        header('Content-Type: application/json');
        try {
            $stmt = $this->db->prepare(
                'SELECT user_id
                FROM user'
            );
            $stmt->execute();
            $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
            if ($stmt->rowCount() > 0) {
                return array('data' => $result);
            } else {
                return array(
                    'status' => 'ERROR',
                    'message' => 'No rows in table'
                );
            }
        } catch (PDOException $e) {
            echo $e->getMessage();
        }
    }
```

The list_all_users function sets its content as JSON (JavaScript Object Notation) and performs a simple query on the database to select all user_id's from the users table. When the page is initially loaded jQuery is used to call the list_all_users function using its shorthand ajax function call '$.getJSON'.

```
$.getJSON("http://localhost/rubric/json.rubric.php?action=list_all_users"
,function(result){
    $.each(result.data, function(index, value) {
        $.each(value, function(column, c) {
            $('#user_dropdown')
                .append($("<option></option>")
                .attr("value", c)
                .text(c));
        }) // $.each value
    }) // $.each data
}); // end $.getJSON
```

The ajax call iterates through the returned JSON array using a for each loop iterates across the X index with a nested for each loop iterating the Y index and storing the user id into the c variable. Using jQuery, the user select box HTML is built up of the returned user id's.

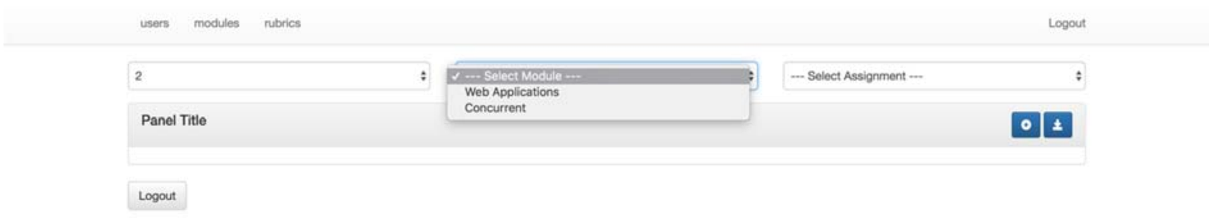


Figure 21 Module select box

Building the module select box from the selected user id.

```
$('#module_dropdown')
        .find('option')
        .remove()
        .end()
        .append($("<option></option>")
        .text("--- Select Module ---"));
```

```
var user_id = ($('#user_dropdown :selected').val());


$.getJSON("http://localhost/rubric/json.rubric.php?action=list_user_modules&user_i
d=" + user_id, function(result){
        $.each(result.data, function(index, value){
            $.each(value, function(column, c){
                $('#module_dropdown')
                    .append($("<option></option>")
                    .attr("value", c)
                    .text(c));
            }); // $.each value
        }) // $.each data
    }); // end $.getJSON
} // end function get_user_modules
```

when the user id is selected from the user select box the contents of the module select box is destroyed and rebuilt. The user id from the user select box is then stored in a variable which is passed to another asynchronous call which builds the module select box. A module select box will only contain modules associated to that user id.

```
public function list_user_modules($user_id)
    {
        header('Content-Type: application/json');
        try {
            $stmt = $this->db->prepare(
                "SELECT module.module_name
                FROM user_modules
                LEFT JOIN module ON user_modules.module_id = module.module_id
                WHERE user_modules.user_id = $user_id");

            $stmt->execute();
            $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
            if ($stmt->rowCount() > 0) {
                return array(
                    'rows_in_table' => $stmt->rowCount(),
                    'data' => $result
                );
            } else {
                return array(
                    'status' => 'ERROR',
                    'message' => 'No rows in table'
                );
            }
        } catch (PDOException $e) {
            $e->getMessage();
        }
    }
```

The query in the prepared statement displays the module name from the user module table by creating a join between the mode and user module table using the module id foreign key.

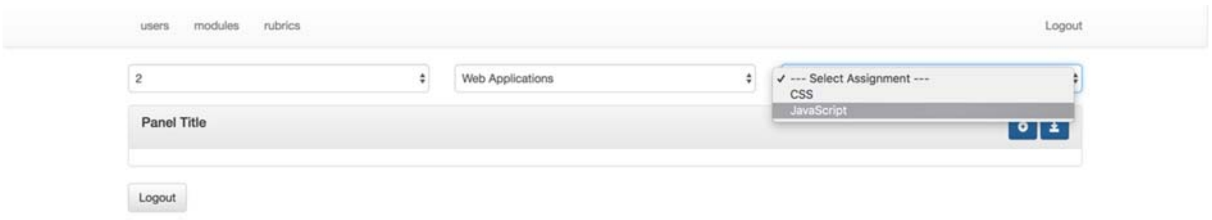


Figure 22 Building the assignment select box

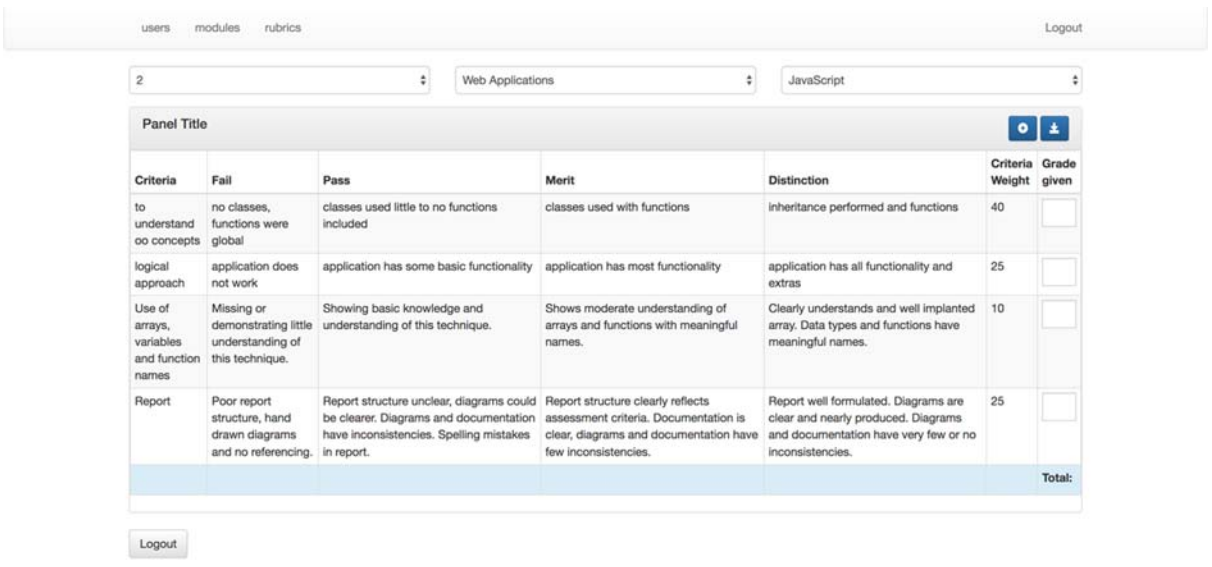
Building the assignment select box requires a query on the database by user id and module name.

```
public function list_user_assignments($user_id, $module_name)
    {
        header('Content-Type: application/json');
        try {
            $stmt = $this->db->prepare(
                "SELECT assignment.assignment_name
                FROM user_modules
                LEFT JOIN module ON user_modules.module_id = module.module_id
                LEFT JOIN assignment ON assignment.module_id = module.module_id
                WHERE (user_modules.user_id = $user_id and module.module_name =
\"$module_name\")");

            $stmt->execute();

            $result = $stmt->fetchAll(PDO::FETCH_ASSOC);

            if ($stmt->rowCount() > 0) {
                return array(
                    'rows_in_table' => $stmt->rowCount(),
                    'data' => $result
                );
            } else {
                return array(
                    'status' => 'ERROR',
                    'message' => 'No rows in table'
                );
            }
        } catch (PDOException $e) {
            $e->getMessage();
        }
    }
```

This query is slightly more complicated requiring a join on the module table and the assignment table. The parameters in the function are passed by building a URL string and being posted back via ajax.

```
function get_user_module_assignments()
{
    var user_id = ($('#user_dropdown :selected').val());
    var module_name = ($('#module_dropdown :selected').val());


$.getJSON("http://localhost/rubric/json.rubric.php?action=list_user_assignments&us
er_id=" + user_id + "&module_name=" + module_name, function(result){
        $.each(result.data, function(index, value){
            $.each(value, function(column, c){
                $('#assignment_dropdown')
                    .append($("<option></option>")
                    .attr("value", c)
                    .text(c));
            }); // $.each value
        }); // $.each data
    }); // end $.getJSON
} // end function get_user_module_assignments
```

The URL string is built by capturing the values in the select box and storing them in a variable. The variable is used to build a '$.getJSON' request which is passed back to the rubric class.



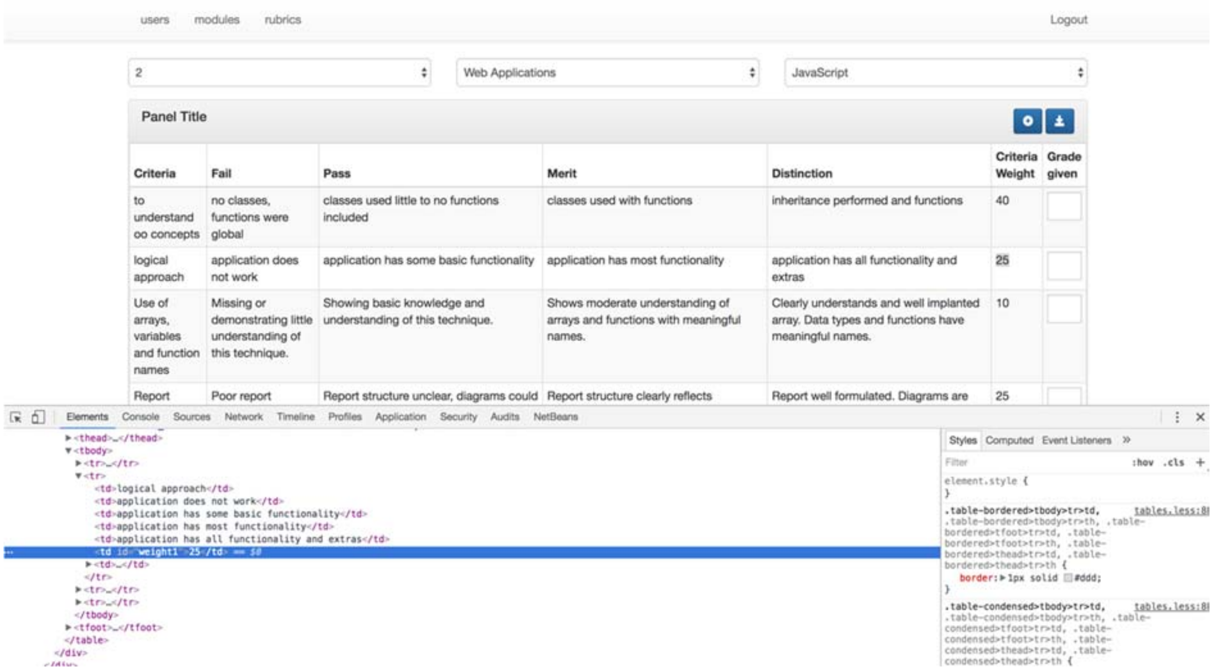
Criteria	Fail	Pass	Merit	Distinction	Criteria Weight	Grade given
to understand oo concepts	no classes, functions were global	classes used little to no functions included	classes used with functions	inheritance performed and functions	40	
logical approach	application does not work	application has some basic functionality	application has most functionality	application has all functionality and extras	25	
Use of arrays, variables and function names	Missing or demonstrating little understanding of this technique.	Showing basic knowledge and understanding of this technique.	Shows moderate understanding of arrays and functions with meaningful names.	Clearly understands and well implanted array. Data types and functions have meaningful names.	10	
Report	Poor report structure, hand drawn diagrams and no referencing.	Report structure unclear, diagrams could be clearer. Diagrams and documentation have inconsistencies. Spelling mistakes in report.	Report structure clearly reflects assessment criteria. Documentation is clear, diagrams and documentation have few inconsistencies.	Report well formulated. Diagrams are clear and nearly produced. Diagrams and documentation have very few or no inconsistencies.	25	
						Total:

Figure 23 Rendering the rubric table

The user is presented with a rubric table containing the information they have requested from the series of select boxes. The tutor can now insert the grades of an assignment, calculate them and store the overall grade to the database.

```
function build_rubric_table()
{
    // clear table contents before building
    delete_table();
    // capture values to pass to query
    var user_id = ($('#user_dropdown :selected').val());
    var module_name = ($('#module_dropdown :selected').val());
    var assignment_name = ($('#assignment_dropdown :selected').val());


$.getJSON("http://localhost/rubric/json.rubric.php?action=get_assignment_rubric&us
er_id=" + user_id + "&module_name=" + module_name + "&assignment_name=" +
assignment_name, function(result){

        html = '<table id="rubric_table" class="table table-bordered table-striped
table-condensed"> <thead> <tr>';
        html = html + '<th>Criteria</th>';
        html = html + '<th>Fail</th>';
        html = html + '<th>Pass</th>';
        html = html + '<th>Merit</th>';
        html = html + '<th>Distinction</th>';
        html = html + '<th>Criteria Weight</th>';
        html = html + '<th>Grade given</th>'
        html = html + '</tr></thead>';

        html = html + '<tbody>'

        // build the column array
        $.each(result.data, function(index, value){
            var row_number = index;

            html = html + '<tr>';

            $.each(value, function(column, c){
                if (column != "weight") {
                    html = html + '<td>' + c + '</td>'
                }
                else if (column == "weight") {
                    html = html + '<td id="weight' + row_number + '">'  + c +
'</td>';
                }
            }); // $.each column

            html = html + '<td>'+ '<input type="number" class="form-control"
id="score' + row_number +'" />' + '</td>';
            index_count.push(row_number + 1);
            html = html + '</tr>';
        }); // $.each data

        html = html + '</tbody>';
        html = html + '<tfoot><tr class="info">';
        html = html + '<td></td> <td></td> <td></td> <td></td> <td></td>
<td></td>';
        html = html + '<td> <label for="total_score"> Total: </label> <span
id="total_score"> </span> </td>';
        html = html + '</tr></tfoot>';
        html = html + '</table>'; // end the html builder
        $('.table_container').append(html);

    }); // end $.getJSON
} // end function build_rubric_table
```

The table is a large string stored into a variable called html. Data is added to the table by iterating over the returned JSON which is obtained from a function of the rubric PHP class. The JavaScript function iterates through each row and column and inserts the data from the database into each column.



Criteria	Fail	Pass	Merit	Distinction	Criteria Weight	Grade given
to understand oo concepts	no classes, functions were global	classes used little to no functions included	classes used with functions	inheritance performed and functions	40	
logical approach	application does not work	application has some basic functionality	application has most functionality	application has all functionality and extras	25	
Use of arrays, variables and function names	Missing or demonstrating little understanding of this technique.	Showing basic knowledge and understanding of this technique.	Shows moderate understanding of arrays and functions with meaningful names.	Clearly understands and well implanted array. Data types and functions have meaningful names.	10	
Report	Poor report	Report structure unclear, diagrams could	Report structure clearly reflects	Report well formulated. Diagrams are	25	

```
▶<thead>…</thead>
▼<tbody>
  ▶<tr>…</tr>
  ▼<tr>
     <td>logical approach</td>
     <td>application does not work</td>
     <td>application has some basic functionality</td>
     <td>application has most functionality</td>
     <td>application has all functionality and extras</td>
     <td id="weight1">25</td> == $0
    ▶<td>…</td>
   </tr>
  ▶<tr>…</tr>
  ▶<tr>…</tr>
 </tbody>
▶<tfoot>…</tfoot>
</table>
</div>
```

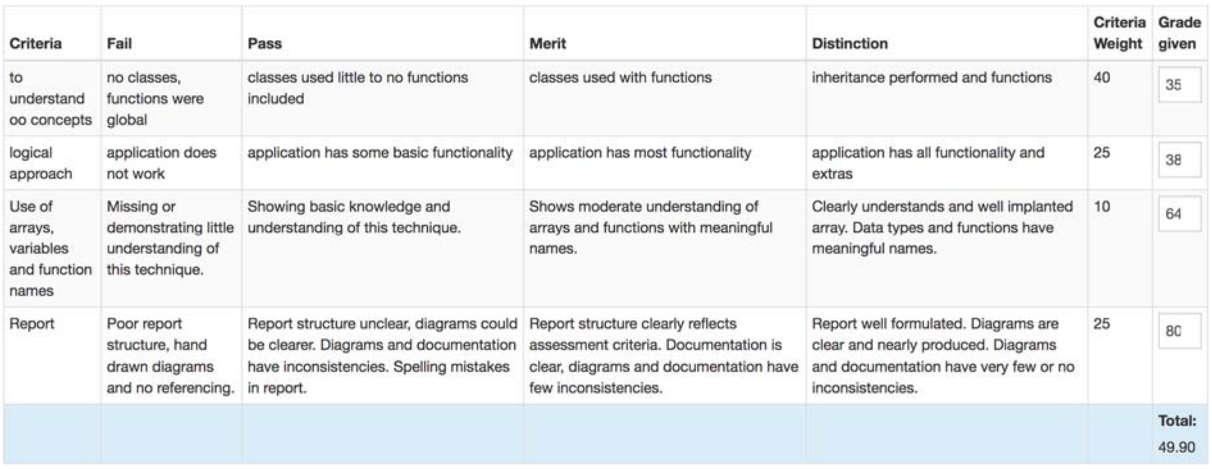
```
element.style {
}
.table-bordered>tbody>tr>td,        tables.less:8
.table-bordered>tbody>tr>th, .table-
bordered>tfoot>tr>td, .table-
bordered>tfoot>tr>th, .table-
bordered>thead>tr>td, .table-
bordered>thead>tr>th {
  border:▶1px solid #ddd;
}
.table-condensed>tbody>tr>td,       tables.less:8
.table-condensed>tbody>tr>th, .table-
condensed>tfoot>tr>td, .table-
condensed>tfoot>tr>th, .table-
condensed>thead>tr>td, .table-
condensed>thead>tr>th {
```

Figure 24showing the unique identifier on table cell

Once a column is found with the name 'weight' we apply a HTML id tag to the table cell and append the iterated index to the id. The same is performed for the grade given text box. Each weight and grade given column will have a unique identity that we can reference later when performing calculations (see Figure 24).

```
public function get_assignment_rubric($user_id, $module_name, $assignment_name)
    {
        header('Content-Type: application/json');
        try {
            $user_id = $_SESSION['user_session'];

            $stmt = $this->db->prepare(
                "SELECT rubric.criteria, rubric.fail, rubric.pass, rubric.merit, 
rubric.distinction, rubric.weight
                FROM user
                LEFT JOIN user_modules
                ON user_modules.user_id
                LEFT JOIN module ON user_modules.module_id = module.module_id
                LEFT JOIN assignment ON assignment.module_id = module.module_id
                LEFT JOIN rubric ON rubric.assignment_id = 
assignment.assignment_id
                WHERE (user.user_id = $user_id AND module.module_name = 
'$module_name' AND assignment.assignment_name = '$assignment_name')");

            $stmt->execute(array(':user_id' => $user_id));

            $result = $stmt->fetchAll(PDO::FETCH_ASSOC);

            if ($stmt->rowCount() > 0) {
                return array(
                        'rows_in_table' => $stmt->rowCount(),
                        'columns_in_table' => $stmt->columnCount(),
                        'data' => $result);
            } else {
                return array(
                        'status' => 'ERROR', 'message' => 'No data in table');
            }
        } catch (PDOException $e) {
            $e->getMessage();
        }
    }
```

To populate the entire rubric table, we need to query the database, selecting the columns we wish to display. A join is required on the user mode table with a foreign key on the module id, a join on the assignment table with a foreign key on the module id and a join on the rubric table with a foreign key on the assignment id.



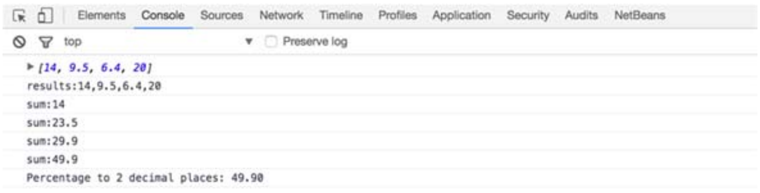
Criteria	Fail	Pass	Merit	Distinction	Criteria Weight	Grade given
to understand oo concepts	no classes, functions were global	classes used little to no functions included	classes used with functions	inheritance performed and functions	40	35
logical approach	application does not work	application has some basic functionality	application has most functionality	application has all functionality and extras	25	38
Use of arrays, variables and function names	Missing or demonstrating little understanding of this technique.	Showing basic knowledge and understanding of this technique.	Shows moderate understanding of arrays and functions with meaningful names.	Clearly understands and well implanted array. Data types and functions have meaningful names.	10	64
Report	Poor report structure, hand drawn diagrams and no referencing.	Report structure unclear, diagrams could be clearer. Diagrams and documentation have inconsistencies. Spelling mistakes in report.	Report structure clearly reflects assessment criteria. Documentation is clear, diagrams and documentation have few inconsistencies.	Report well formulated. Diagrams are clear and nearly produced. Diagrams and documentation have very few or no inconsistencies.	25	80
						Total: 49.90

Figure 25 Calculating score from rubrics

For each row in the table we divide that rows given grade by 100 and then multiple it by the criteria weight.

```
$('#calculate_score').click(function(){
    var user_id = $('#user_dropdown :selected').val();
    var assignment_name = $('#assignment_dropdown :selected').val();

    var results = [] // array to hold percentages
    var result = 0;
    var sum = 0; // calculations on array

    $.each(index_count, function(index, amount){
        var score = 0; // varaible for pushing array
        var weight = 0; // variable for pushing array
        var sum = 0;

        score = parseFloat($('#score' + index).val());
        weight = parseFloat($('#weight' + index).text());
        // percentage weight / score * 100
        result = (score / 100) * weight;
        results.push(result);
    });

    $.each(results, function(index, value) {
        sum += value;
    });

    percent_to_2dp = (sum);

    percent_to_2dp = parseFloat(percent_to_2dp).toFixed(2);

    $('#total_score').empty();
    $('#total_score').append(percent_to_2dp);


$.getJSON("http://localhost/rubric/json.rubric.php?action=get_assignment_id_from_n
ame&assignment_name=" + assignment_name, function(result) {
        $.each(result.data, function(index, value) {
            assignment_id = value.assignment_id;
        }); // $.each data
    }); // $.getJSON
}); // end calculate score click function
```

All elements on a HTML page will be returned as a string which means calculations cannot be performed easily. To overcome this problem, we can iterate through each row, capture the data we require, convert the string to the required datatype, perform the calculation and push the score of that specific criterion into an array. Here is where we use the unique identifier HTML tags we generated when building the table (see Figure 23 and 24).



```
▸ [14, 9.5, 6.4, 20]
results:14,9.5,6.4,20
sum:14
sum:23.5
sum:29.9
sum:49.9
Percentage to 2 decimal places: 49.90
```

Figure 26 Results array and calculations performed

The calculated results are stored in the results array. The results are then iterated over using a for each loop adding the values on each iteration. This is dynamic, therefore no matter the array size, a correct calculation will be performed. The results are also formatted to two decimal places.

```
public function update_user_assignment_grade($grade, $user_id, $assignment_id)
    {
        header('Content-Type: application/json');

        try {
            $stmt = $this->db->prepare(
                "UPDATE
                    user_assignment_grade
                SET
                    grade = $grade
                WHERE
                    user_id = $user_id AND assignment_id = $assignment_id");

            $stmt->execute();
            return $stmt;
        } catch (PDOException $e) {
            $e->getMessage();
        }
    }
```

```
$('#save_score').click(function() {
    user_id = $('#user_dropdown :selected').val();

    $.ajax({
        url:
"http://localhost/rubric/json.rubric.php?action=update_user_assignment_grade&grade
=" + percent_to_2dp + "&user_id=" + user_id + "&assignment_id=" + assignment_id
    }); // ajax save
}); // end save function
```

The final score calculation is captured and posted to the PHP rubric class and update user assignment grade function. This function contains an SQL statement which updates the user assignment grade table, sets the grade to the value passed from the parameter and checking the user id and assignment id which is also passed from the parameter. An insert statement shouldn't be required as the users and assignments will already be populated to the table.

Evaluation

The application is built on the principle that a rubric is uniquely associated to an assignment and an assignment can have many rubrics. A rubric criterion can have any weighted grade but the total rubrics assigned to an assignment must not exceed 100 points. A module has multiple assignments but an assignment can only belong to one module. A student is enrolled on multiple modules and a module can have multiple students.

The user interface of the application provides an easy to follow layout which guides the user through the steps to follow. When marking an assignment, an assignment cannot be chosen without a module and a module cannot be selected until the student they wish to grade has been selected.

Initially, the project was hosted on a private virtual server to simulate the application as a true commercial project. File contents were saved, created and edited directly on the server instance which became subject to an attack and was terminated by Amazon resulting in a complete loss of data. Upon rebuilding the application, the application was saved to a private GitHub repository and additional backups were created. This proved the importance of having multiple backups as the rebuild conflicted with other projects which would also truly apply to working in industry.

The largest problem was inserting automatically generated feedback. This is quite an abstract problem that can be easily solved by checking the grade achieved and inserted feedback based on the score, however, the challenge is producing feedback that offers more value and information that is currently being displayed.

The application follows the use case diagram and makes use of the classes from the class diagram. When building the application, small changes were made to the classes to separate logic as rubric class was handling functions and actions. The user class was also changed to follow the updated principles of the rubric class. The rubric and user class operated as the logical controllers, and an additional class was created called json.classname.php. The JSON class would call a function from its parent class depending on which action was targeted.

```
require_once('login_status.php');
require_once('resources/library/class.rubric.php');

// main program body    $rubric = new Rubric($conn);
$response = new Schema();

if (isset($_GET['action'])) {
    switch ($_GET['action']) {
        case 'get_tables':
            echo $response->get_response($rubric->get_tables());
            break;
        case 'get_table_contents':
            if (isset($_GET['table'])) {
                echo $response->get_response($rubric-
>get_table_contents($_GET['table']));
            }
            break;
        case 'list_all_users':
            echo $response->get_response($rubric->list_all_users());
            break;
        case 'list_user_modules':
            if (isset($_GET['user_id'])) {
                echo $response->get_response($rubric-
>list_user_modules($_GET['user_id']));
            }
            break;
        case 'list_user_assignments':
            if (isset($_GET['user_id']) and (isset($_GET['module_name']))) {
                echo $response->get_response($rubric-
>list_user_assignments($_GET['user_id'], $_GET['module_name']));
            }
            break;
        case 'get_assignment_rubric':
            if (isset($_GET['user_id']) and isset($_GET['module_name']) and
isset($_GET['assignment_name'])) {
                echo $response->get_response($rubric-
>get_assignment_rubric($_GET['user_id'], $_GET['module_name'],
$_GET['assignment_name']));
            }
            break;
        case 'update_user_assignment_grade':
            if (isset($_GET['grade']) and isset($_GET['user_id']) and
isset($_GET['assignment_id'])) {
                $rubric->update_user_assignment_grade($_GET['grade'],
$_GET['user_id'], $_GET['assignment_id']);
            }
            break;
        case 'get_assignment_id_from_name':
            if (isset($_GET['assignment_name'])) {
                echo $response->get_response($rubric-
>get_assignment_id_from_name($_GET['assignment_name']));
            }
            break;
        default:
            # code...
            break;
    }
}
```

Each of the case inside the switch statement return data from a database where the function performs the SQL query. Using JSON encoded data allowed for information returned to be legible to read and made handling data arrays and passing information robust.

While the application performs its tasks correctly, the project workspace soon became cluttered and individual files required lots of editing to overcome classes becoming too fat. To rectify this, it would be better to follow a design pattern such as the Model-View-Controller.

Model objects encapsulate the data specific to an application and define the logic and computation that manipulate and process that data. A model object can have a to-one and to-many relationship with other model objects.

A controller acts as an intermediary between one of more of the application's view and model objects. A controller object interprets the actions made in the view by the user and tells the model of changed data. The controller will also notify the view of any changes made by the model.

A view object is an object in an application that the client can see. A view object responds to user actions and knows how to render itself. The purpose of the view object is to display and edit data from the application model object.

By using a Model-View-Controller design pattern the application can easily be scaled to include new features and enhancements.

Future Enhancements and Recommendations

To improve the application a lecturer should have the option to not only select a module once the users have been loaded, but to select a user that is associated with module they wish to mark. As the data scales, it would become complicated to mark a specific assignment by needing to find the correct user identity.

To provide the user with valuable auto generated feedback, a feedback table could be created by a lecturer which is associated to an assignment much like how the rubric is associated to an assignment. The information will be displayed to the lecturer in the form of a HTML table. Each piece of feedback will have an associated checkbox that is added to a string if selected. The string can then be posted back to the database to the associated user and assignment. This allows the lecturer to be as specific as possible and provide the student with information that shows exactly what was achieved and where they can improve.

Another enhancement to the application would be to extend the rescaled marking feature. Currently the system grades, weights and rescales the percentage of a specific assignment. An additional calculation should be performed where the module grade is rescaled based on the grade achieved from the assignment and the overall weight of the assignment to the module.

Conclusion

The overall development of the project has successfully solved the problems and challenges that have been presented. Lecturers can create rubrics, save the rubrics and grade an assignment. A graded assignment will automatically rescale total marks and convert the marks to percentages, weighted marks and grades.

The application performs logic that prevents data from being duplicated and inserting information to the incorrect categories.

The project adheres to the challenges and follows object oriented principles by encapsulating tasks to specific classes which can be accessed and called when required.

The information that is presented to the user is obtained from and saved into persistent storage as required. The data entities have been normalized, unique with primary key's and one-to-many or many-to-many relationships have been associated with the use of foreign keys.

Using rubrics to grade assignments proves to be a positive method at guiding students on the tasks required. One student from the Utah State University stated rubrics are a good way rubrics are a good way for the student to understand not only what you are looking for when you are grading a paper, but also a good "gold standard" to compare our work to and therefore improve our writing skills in general.

Another student claimed they find the learning outcomes rubrics to be very helpful. Having three separate sections (Historical Knowledge, Historical Thinking, and Historical skills) as well as the subsections inside help identify where exactly the student is struggling.

However, not all students agree, one stating that although the rubric shows an area which needs to be worked on, they had difficulty determining where the areas they went astray. Mistakes such as pertinence of an argument are more difficult to identify in a returned essay. The student suggested numbering the rubric to make it easier to identify. (Student comments on rubrics: HIST 3750, 2017).

Bibliography

Acunetix. (2016). *What Are Web Applications?*. [online] Available at: http://www.acunetix.com/websitesecurity/web-applications/ [Accessed 2 Dec. 2016].

Boobyer, G. (2016). *IS3S602*. [online] Blackboard. Available at: http://www.blackboard.southwales.ac.uk [Accessed 29 Nov. 2016].

Brookhart, S. (2013). *How to create and use rubrics for formative assessment and grading*. 1st ed. Alexandria, VA: ASCD.

Data-archive.ac.uk. (2016). *UK Data Archive - ETHICAL / LEGAL*. [online] Available at: http://www.data-archive.ac.uk/create-manage/consent-ethics/legal [Accessed 1 Dec. 2016].

Docs.moodle.org. (2017). *Rubrics - MoodleDocs*. [online] Available at: https://docs.moodle.org/22/en/Rubrics [Accessed 15 Feb. 2017].

eMarking Assistant. (2016). *Rubric-O-Matic - automated rubric maker*. [online] Available at: http://emarkingassistant.com/products/rubric-o-matic/ [Accessed 21 Nov. 2016].

Go.turnitin.com. (2017). *Contact Sales*. [online] Available at: http://go.turnitin.com/en_gb/contact-sales?Product=Turnitin&Notification_Language=English&Lead_Origin=Website&source=Quote%20-%20Request&_ga=1.229940892.1638425842.1486059166 [Accessed 14 Jan. 2017].

Goodrich Andrade, H. (2016). Using rubrics to promote thinking and learning. *Educational Leadership*, 57(5), pp.13-18.

Helpx.adobe.com. (2017). *System requirements for Photoshop Lightroom for Mac and Windows OS*. [online] Available at: https://helpx.adobe.com/lightroom/system-requirements.html [Accessed 7 Mar. 2017].

Ico.org.uk. (2017). *Data protection principles*. [online] Available at: https://ico.org.uk/for-organisations/guide-to-data-protection/data-protection-principles/ [Accessed 3 Feb. 2017].

Its.uncg.edu. (2016). *Software Licensing, Information Technology Services, UNCG*. [online] Available at: https://its.uncg.edu/Software/Licensing/ [Accessed 1 Dec. 2016].

Parr, C. (2014). Some turn away from Turnitin over price and reliability. *The Times Higher Education*.

Pcmag.com. (2017). *desktop application Definition from PC Magazine Encyclopedia*. [online] Available at: http://www.pcmag.com/encyclopedia/term/41158/desktop-application [Accessed 8 Jan. 2017].

Plassman, P. (2017). *Design Patterns and Assignment*.

Protecting personal data in online services: learning from the mistakes of others. (2017). 1st ed. [ebook] pp.1-5. Available at: https://ico.org.uk/media/for-organisations/documents/1042221/protecting-personal-data-in-online-services-learning-from-the-mistakes-of-others.pdf [Accessed 2 Feb. 2017].

Reference. (2016). *What are examples of morals?*. [online] Available at: https://www.reference.com/world-view/examples-morals-5473b4217c1e38d [Accessed 2 Dec. 2016].

S, S. (2016). *Difference Between Morals and Ethics (with Comparison Chart) - Key Differences*. [online] Key Differences. Available at: http://keydifferences.com/difference-between-morals-and-ethics.html [Accessed 2 Dec. 2016].

Segue Technologies. (2016). *The Importance of Cross Browser Testing*. [online] Available at: http://www.seguetech.com/importance-cross-browser-testing/ [Accessed 2 Dec. 2016].

Student comments on rubrics: HIST 3750. (2017). 1st ed. [ebook] Utah state university, pp.1-3. Available at: https://history.usu.edu/files/uploads/student%20responses%20-%20upper%20division.pdf [Accessed 27 Mar. 2017].

Turnitin Instructor User Manual. (2017). 1st ed. [ebook] iParadigms, pp.87-119. Available at: https://turnitin.com/static/resources/documentation/turnitin/training/Instructor_GradeMark_Chapter_4.pdf [Accessed 21 Jan. 2017].

Turnitin. (2017). *Turnitin - Home*. [online] Available at: http://turnitin.com/ [Accessed 13 Jan. 2017].

University, C. (2016). *Rubrics-Teaching Excellence & Educational Innovation - Carnegie Mellon University*. [online] Cmu.edu. Available at: https://www.cmu.edu/teaching/designteach/teach/rubrics.html [Accessed 26 Nov. 2016].

University, S. (2016). *What is Ethics? - Ethical Decision Making - Ethics Resources - Markkula Center for Applied Ethics - Santa Clara University*. [online] Scu.edu. Available at: https://www.scu.edu/ethics/ethics-resources/ethical-decision-making/what-is-ethics/ [Accessed 1 Dec. 2016].

Upguard.com. (2016). *MySQL vs MongoDB*. [online] Available at: https://www.upguard.com/articles/mysql-vs-mongodb [Accessed 24 Nov. 2016].

W3schools.com. (2016). *Browser Statistics*. [online] Available at: http://www.w3schools.com/browsers/ [Accessed 2 Dec. 2016].

Webhostdesignpost.com. (2016). *Website Storyboarding | Examples, How To and Sitemap*. [online] Available at: http://www.webhostdesignpost.com/website/websitestoryboarding.html [Accessed 24 Nov. 2016].

Appendices

Appendix A:

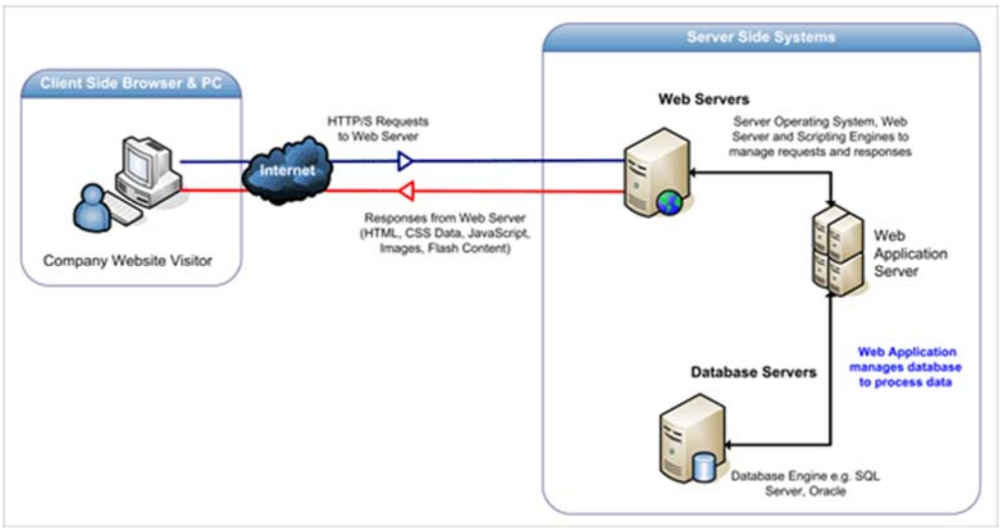

Client Side Browser & PC
Company Website Visitor
Internet
HTTP/S Requests to Web Server
Responses from Web Server (HTML, CSS Data, JavaScript, Images, Flash Content)
Server Side Systems
Web Servers
Server Operating System, Web Server and Scripting Engines to manage requests and responses
Web Application Server
Web Application manages database to process data
Database Servers
Database Engine e.g. SQL Server, Oracle

Figure 27 Acunetix, 2016

Appendix B:

2016	Chrome	IE	FireFox	Safari	Opera
October	73.0%	5.2%	15.7%	3.6%	1.1%
September	72.5%	5.3%	16.3%	3.5%	1.0%
August	72.4%	5.2%	16.8%	3.2%	1.1%
July	71.9%	5.2%	17.1%	3.2%	1.1%
June	71.7%	5.6%	17.0%	3.3%	1.1%
May	71.4%	5.7%	16.9%	3.6%	1.2%
April	70.4%	5.8%	17.5%	3.7%	1.3%
March	69.9%	6.1%	17.8%	3.6%	1.3%

February	69.0%	6.2%	18.6%	3.7%	1.3%
January	68.4%	6.2%	18.8%	3.7%	1.4%

(W3schools.com, 2016)

Agreed Marking Scheme Weightings

<u>Mark Category</u>	**<u>Proposed Weighting Ranges</u>**	**<u>Agreed Weighting</u>**	**<u>Mark</u>**
Project Management *(Only set by Supervisor 1)*	50 – 80		
Originality & Self-Direction	40 – 80		
Technical Complexity	20 – 80		
Solutions, Evaluation & Conclusions	80 – 120		
Final & Sub-Report Quality	50		
Prototype / System Demo Or Project Deliverable	50 – 100		
Sponsor Mark	00 – 60		
Sub-Total Marks	-------		
Sub-Total Percentage	-------	**50%**	%
<u>Milestone 1</u> Research Literature Review **<u>Milestone 2</u> Amended Research Research Applied to Design Deliverable Development**	-------	**10%** **10%**	% %
<u>LSEPI Appliance</u>	-------	**10%**	
<u>Final Presentation (including poster production)</u>	-------	**20%**	%
<u>TOTAL PERCENTAGE</u>	-------	**100%**	%

Legal Social Ethical and Professional Issues

A software licence is a legally binding agreement that specifies the terms of use of the application for the end user. All software must be legally licenced before it may be installed. (Its.uncg.edu, 2016)

As this is a project that could be available as a commercial platform it is important to look at the types of licensing available.

Propriety License:

Ownership of the software belongs to the software publisher. The software publisher grants a licence to **use** one or more copies of the software.

GNU Public License:

General agreements where many open source projects are licenced. Software licenced under GNU allow edits to the source code. However, a change in source code must also be made under the GNU licence, meaning that a piece of open source software cannot be used, changed and then sold. The software or changes to the software cannot be made distributed for a fee.

End User License Agreement:

The end user licence agreement indicates the terms in which an end user may use a piece of software. End user licence agreements can be typically seen with subscription based software.

Workstation License:

A type of licence that is only allowed to be installed on one machine. You can only install on multiple machines by purchasing multiple licences. Typically, under a workstation licence you can make a single backup of the product if it is used on the same machine.

Concurrent Use License:

Concurrent use licence allows the installation of a product on any number of machines, however, the software cannot be running on more machines that the amount of licences owned.

Site License:
Allows the use of a piece of software on any computer on a specified site. Unlimited site licences allow the installation of software on any number of computers on the site.

Perpetual License:
Most software installed on people's home machines fall under the perpetual licence category. A perpetual licence permits the use of software indefinitely as they do not have expiration dates.

Non-Perpetual License:
The user pays to use software over a period. If the user stops paying the licence fee they are required to remove the software.

License with Maintenance:
Software sold with a maintenance licence allows the user to upgrade and receive updates of the software until the maintenance agreement expires.

Data Protection
The project application has the potential to be storing and handling personal and sensitive information regarding information about its users and this needs to adhere to the data protection act.

The Data Protection Act 1998 is based around eight principles of 'good information handling'. These principles give specific rights to people in relation to their personal information and place certain obligations on those organisations whom are responsible for processing it.

(Ico.org.uk, 2017) The eight principles of data handling are:

“Personal data shall be processed fairly and lawfully and, in particular, shall not be processed unless – At least one of the conditions in Schedule 2 is met, and In the case of sensitive personal data, at least one of the conditions in schedule 3 is also met.”

“Personal data shall be obtained only for one or more specified and lawful purposes, and shall not be further processed in any manner incompatible with that purpose of those purposes.”

“Personal data shall be adequate, relevant and not excessive in relation to the purpose or purposes for which they are processed.”

“Personal data shall be accurate and, where necessary, kept up to date.

Personal data processed for any purpose or purposes shall not be kept for longer than is necessary for that purpose or those purposes.”

“Personal data shall be processed in accordance with the rights of data subjects under this Act. “

“Appropriate technical and organisational measures shall be taken against unauthorised or unlawful processing of personal data and against accidental loss or destruction of, or damage to, personal data.”

“Persona data shall not be transferred to a country or territory outside the European Economic Area unless that country or territory ensures an adequate level of protection for the rights and freedoms of data subjects in relation to the processing of personal data.”

Software updates, SQL injection, unnecessary services, decommissioning of software services, password storage, configurations of SSL and TLS, inappropriate locations for processing data and default credentials are eight areas of security in which the Information Commissioner’s Office has identified determined due to frequent appearance during investigations of data breaches. (Protecting personal data in online services: learning from the mistakes of others, 2017)

Ethics

As suggested by Santa Clara University ethics can be divided into two things; firstly, it refers to well-founded standards of right and wrong. Ethics are built up of rational examinations into people's moral beliefs and behaviours. Examples of ethics would be the obligations to refrain from rape, stealing, murder and assault. Ethics serve as a moral guide in everyday life.

Secondly is the study of individual's ethical standards. It is necessary to constantly examine the standards of individual's ethical beliefs to ensure they are reasonable and well-founded. The continuous effort of studying the beliefs of others allows us to live up to standards that are reasonable.

Applying Ethics to Computing and Software

Ethics in software and web driven application is primarily focused on the security of data and personal information. The UK Data Archive suggests that the publisher or developer of software has a duty of confidentiality towards information and participants; a duty to protect participants from harm, by not disclosing sensitive information. Individuals using software should have the right to choose whether their information may be used, processed or shared.

Software developers and professionals in the I.T. industry have a professional board to follow known as the British Computer Society. The BCS has the responsibility to set rules and standards to direct the behaviour of its members.

The release and sale / subscription of a product should have an equality of opportunity where prices for the product are fair between potential clients. It is not ethical to provide a company with better prices or discounts because they can order more products.

By accepting responsibility for your own work a product should be released to a high quality with as little to or no bugs. If bugs or issues are found with a product after it has been released they should be patched and available to the consumer at no cost.