

Digital Lab Marking System

Heriot-Watt University

Deliverable 1: Final Year Dissertation

MEng Software Engineering

Lewis Francis McNeill
supervised by Peter J King

April 9, 2017

Declaration

I, Lewis Francis McNeill, confirm that this work submitted for assessment is my own and is expressed in my own words. Any references, made within it, of the works of other authors in any way (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Lewis McNeill

Date: April 9, 2017

Abstract

The aim of this dissertation project is to replace the current system for the marking of computer labs with a new digital system. This will enable lecturers to create a marking scheme online. Lab helpers will select the student they are marking and the marking scheme will then be loaded, marks will be entered and then made immediately available to both student and lecturers to view. It will also provide useful statistics for both student and lecturers.

Contents

1	Introduction	1
2	Aims and Objectives	2
2.1	Aim	2
2.2	Objectives	2
2.3	Scope	2
3	Background Information	3
3.1	Web Applications	3
3.2	HTML	3
3.3	Cascading Style Sheets	4
3.4	JavaScript	5
3.5	PHP	5
3.6	Marking Systems	5
3.6.1	Lecturer Based	5
3.6.2	Peer Based	6
3.7	Digital Marking Systems	6
3.7.1	Reasons for Digital Marking	6
3.7.2	TurnItIn	8
3.7.3	BOSS System	8
3.7.4	Ceilidh	9
3.8	User Access Views	9
3.8.1	Social Media	9
3.8.2	Database Controlled Access	10
3.9	Custom Input Forms	10
3.9.1	SurveyMonkey	10
3.9.2	Customizing Forms In Electronic Mail Systems	11
3.10	Development Tools	11

3.10.1	jQuery	11
3.10.2	Ajax	11
3.10.3	Bootstrap	12
3.10.4	PHP Unit	13
3.10.5	D3	13
3.10.6	CakePHP	13
4	Requirements	15
4.1	Requirements Analysis	15
4.2	Requirements	15
4.2.1	MoSCow Prioritisation	15
4.2.2	Functional Requirements	16
4.2.3	Non-Functional Requirements	19
5	Design	21
5.1	User Interface Design	21
5.1.1	Lab Results	21
5.1.2	Lab Marking	23
5.1.3	Lab Creation	25
5.1.4	Lab Manager	27
5.2	Database Design	28
5.3	Functionality Design	30
6	Implementation	32
6.1	Database	32
6.2	PHP Class Structure	32
6.3	Background Functionality	33
6.3.1	Database Connection	33
6.3.2	Security	34
6.3.3	Dynamic Loading	36

6.3.4	Navigation Bar	36
6.4	Lab Creator	39
6.4.1	Design	39
6.4.2	Generating Question Tiles	39
6.4.3	Create Lab	42
6.4.4	Insert Lab Name	46
6.4.5	Insert Question	47
6.4.6	Error Checking	48
6.5	Marking Labs	48
6.5.1	Design	48
6.5.2	Retrieving Courses, Labs and Students	50
6.5.3	Searching Students	50
6.5.4	Loading Student Marks	52
6.5.5	Processing Submitted Mark	52
6.5.6	Submitting Marks	53
6.6	Results Display	53
6.6.1	Different Views	53
6.6.2	Retrieving Marks	55
6.6.3	Expanding Table	57
6.7	Lab Management	58
6.7.1	Making Labs Mark-able	59
6.7.2	Export Lab Results	60
6.7.3	Edit Labs	61
6.7.4	Delete Lab	62
6.8	Admin Panel	64
7	Testing	65
7.1	Development Testing	65
7.1.1	Security Class Tests	65

7.2	Final Testing	66
8	Evaluation	67
8.1	Usability Case Study	67
8.2	Feedback	67
8.2.1	Quantitative Questions	68
8.2.2	Q3.8 Creation Alerts	71
8.2.3	Qualitative Questions	71
8.3	Implementation Of Feedback	71
8.4	Evaluate Requirements	72
8.5	Aim and Objective	75
8.6	Overall Evaluation	75
9	Discussion	76
9.1	Development	76
9.1.1	Issues	76
9.2	Limitations	76
9.2.1	Only Three Question Types	77
9.2.2	Student Only Lab Helpers	77
9.3	Future Improvements	77
9.3.1	Peer Marking	77
9.3.2	Comparing Students	77
9.3.3	Track Who Marked What Student	77
9.4	Conclusion	78
	References	79

List of Figures

1	Example Form (Patent [8])	11
2	Results Page: Desktop Designs	22
3	Results Page: Mobile Designs	23
4	Marking Page: Desktop Design	23
5	Marking Page: Mobile Designs	25
6	Lab Creation Page: Desktop Design	26
7	Lab Creation Page: Mobile Design	26
8	Lab Manager Page	27
9	Database Schema	28
10	Database Sections	29
11	Implemented Database Schema	32
12	Navigation Bar	36
13	Navigation Bar	36
14	Lab Creation Page	39
15	Lab Marking Page	48
16	Selecting Student Page	49
17	Marking Student Page	52
18	HTML Structure Of Questions	52
19	Results Page: Student View	54
20	Results Page: Lecturer View	55
21	Lab Management Page	58
22	Editing A Lab	61
23	Bar Graph of Question Results	69
24	Q2.3: Know what the different student colours mean?	70
25	Q3.6 Know what visible / not visible means?	70
26	Q3.8: Did an alert appear	71

List of Tables

1	Functional User Requirements	16
2	Non-Function Requirements	19
3	Usability Case Study: Scale Question Results	68
4	Functional Requirements	72

Listings

1	Database Connection Class - PHP	33
2	Convert Access Name To Accesslevel - PHP	34
3	Has Access Required - PHP	35
4	Has Great Access Than - PHP	35
5	Navigation Bar Button Adder - PHP	37
6	Load Navigation - JavaScript	38
7	Add Question - JavaScript	40
8	Call Create Question - PHP	40
9	Create Question Function - PHP	41
10	Scale Question - PHP	41
11	Lab Submit - JavaScript	43
12	lab_creator script - PHP	43
13	LabCreator Constructor - PHP	43
14	createLab Function - PHP	44
15	insertLabName Function - PHP	46
16	insertQuestion Function - PHP	47
17	studentButtonFilter Function- PHP	50
18	Simplified getStudents Function- PHP	51
19	Results Layout Selector - PHP	53
20	Expanding Row Size - JavaScript	56

21	Expanding Row Size - JavaScript	57
22	Lab Mark-able - JavaScript	59
23	IO Class Export Function - PHP	61
24	Load Editable Lab - JavaScript	62
25	Load Editable Lab - JavaScript	63
26	hasAccessLevel Function Test	65

1 Introduction

The current system for marking of computing science labs is to use multiple lab helpers, each given a list of students and the marking scheme for them. Generally marking schemes consist of a selection of tasks which students must have completed and lab helpers tick them off when this has been achieved. The biggest problem with this part of the marking is the length of time it takes lab helpers to locate the students on the list. This causes frustration with increased waiting times for students. Multiple other issues can also arise from this: students can be marked by two helpers and obtain different grades from both; the lab helper omits to tick off a completed task; they assign marks for the wrong student on the sheet or simply they misplace the actual marking sheet.

After the lab helpers have completed their marking, the sheets are provided to the lecturer who collates them into one spreadsheet to calculate the marks. After that it is entered it into vision. This too can cause its own set of problems-the chances of transcription errors are increased as it is possible for the lecturer to misread marks when they are transferring them across. The lecturer may not enter the marks immediately into the spreadsheet increasing the chance that a marking sheet goes missing, and finally this system means that students are having to wait even longer to receive their results.

The objective is to develop a system that will reduce and hopefully eliminate the problems of the current system. Along with this, it should hopefully reduce the amount of time taken to mark students work and therefore speed up labs in general. It should also enable students to see their grades immediately, allow lecturers to see the result of the assignments as they are being marked and make marking quicker for lab helpers.

2 Aims and Objectives

2.1 Aim

The aim of this dissertation is to design and implement a system for the digital marking and analysis of computer labs and to help improve the speed at which they are marked. The system will also provide useful statistics for both lecturers and students.

2.2 Objectives

- Simplify the way that labs marks are currently processed.
- Allow lecturers to create marking schemes on-line that lab helpers can access.
- Lab helpers can mark students in labs using marking schemes.
- Lab helpers able to mark labs using an on-line application.
- Allow students to see the mark they achieved from the lab instantly.
- Provide useful statistics and graphs for lecturers and students.
- Provide different views for student, lab helpers and lecturer.

2.3 Scope

3 Background Information

This section contains summaries of literature relating to the topic and should help to create a context for the development of a digital marking system. It will cover which marking systems are currently used, also which current digital marking systems actually exist and why they are an improvement. Along with this it will also cover how to control what users are allowed to see, as well as explaining systems for creation of custom website forms, and finally it will cover the graphical displaying of statistics.

3.1 Web Applications

The digital lab marking system will be developed as a web application. To help understand how web applications are built this sub section covers the core technologies required to develop web applications.

3.2 HTML

HTML is short for Hypertext Markup Language and is a markup language which in fact is less a programming language and more a set of instructions [brooks'introduction'2007]. It creates the structure of a website through the use of elements which are represented by: example tags include: <html>, <href>, <a>.

An individual element is made up of an open tag (<a>) and a closing tag which is the same as the opening tag but includes a back slash at the start (<\a>). Elements can exist inside other elements creating a hierarchical structure.

Web browsers do not display the elements and tags directly [w3'introduction'nodate] but instead they process these elements to decide how the web page should look and then renders it in the browser's screen for the user to interact with. The downside to web browsers deciding how to render elements is that different browsers can choose to render the same element two different ways. For instance if a web page was loaded using Google Chrome it may render a table where the column width is not defined with each column getting equal space, but if the same table was loaded

using Fire-fox it may decide that each column only gets the minimum size required and therefore the table looks narrower. This can have a really major effect if the the website was designed with accurate measurements in one browser and another calculates sizes differently resulting in the website displaying wrongly.

3.3 Cascading Style Sheets

Cascading Style sheets (CSS)[21] are used to change the properties of elements created using HTML; through the changing of these properties developers can change the look and response of a website. Developers can also create classes and ids, both of which are given names and then element properties are set inside. This means for instance a class called "bigText", which makes the text large and bold- when a element uses this class all the specified properties are changed. HTML elements can have multiple classes but can only have one id which creates a priority system.

The priority system [6] is where the cascading part of the style comes into effect, elements inherit the property of any class assigned to them, but if another class is added it cascades through the current set properties changing any that the class also declares. This means that classes declared later have a higher priority for example class=" bigText smallText", in this the element has a big and small text class. Since the smallText class is declared after bigText the elements properties will set to it. The only thing that would not have changed are properties declared by the id, since an element can only have one id it has a higher priority than any class and so will override properties with its own values.

This priority and cascading system also works with multiple files. If two css files are being used by an HTML document and both declare a class called "bigText", then the "bigText" class from the css file that is declared last, will be the one that is used. This will be very useful for this project as it allows for the use of frameworks which will provide their own css files, but adding my own file afterwards will allow me to customise the initial css file.

As part of this project I will be using CSS 3 which at time of writing (2017) is the currently agreed standard [7].

3.4 JavaScript

JavaScript is a high-level interpreted language and loosely typed, meaning that you do not need to declare the type of variable. JavaScript programs are called scripts and can interact with the user, their web browser, and is capable of editing HTML content that is displayed inside the user's browser [flanagan`javascript:`2006]. JavaScript is generally embedded into HTML web pages and is usually called client-side JavaScript as the scripts are run on the user's computer not the web page's server.

3.5 PHP

PHP stands for Hypertext Preprocessor and is an open source scripting language first developed in 1994 [24] which can be embedded into HTML. Unlike HTML which is sent to the user and processed by their computer, PHP scripts are instead processed by the website server. Once it has been processed HTML is generated and sent to the user.

PHP currently has two major versions which are version 5 and 7. For this project I will be using version 5.6.25 as version 5 is still the most common version of PHP [noauthor`usage`nodate] with 95% of websites that use PHP using version 5. It also allows for easy installation of the lab marking system into the university's server.

3.6 Marking Systems

3.6.1 Lecturer Based

The way lecturer based marking works is that students complete their assignment, the lecturer or tutor marks it and provides results in a timely manner with useful feedback which can be majorly important in helping students improve their skills [28].

The advantage of this style of marking is that students can obtain useful feedback from their lecturer that can help improve their learning. An article study [15] found that of the students, when surveyed 82% agreed to the question "I pay close attention to the comments I get" in response to assignment feedback.

A downside to this style of marking is that as the number of students increases on courses the amount of time required to mark assignments consequently takes longer and in some cases this can actually cause marked assessments to be scrapped completely due to the amount of time taken to give feedback to students [4].

3.6.2 Peer Based

To cope with increasing class sizes some courses are beginning to move towards peer marking. Peer marking system works by having students assess each other and in some cases the students produced their own marking criteria [23].

This style allows students to gain experience in evaluating other people's work, which some graduates feel is a necessary skill to possess. [20]. Peer marking also deals with increasing amounts of students very well- this is because as the number of students increases the number of markers also increases!

Peer marking however has its own set of problems - for example, "Students may have a less well developed sense of the criteria compared to the lecturer which could lead to a lack of reliability of student marking." [23].

3.7 Digital Marking Systems

3.7.1 Reasons for Digital Marking

Digital marking systems are designed to mirror the current paper based marking systems but with the advantage of the electronic environment [14]. These systems help to reduce the increasing workload caused by more and more students taking courses. Along with this it also allows administrative tasks associated with coursework to be automated, thus enabling more time for other tasks.[17].

For students digital marking is great as it allows for quick feedback as the assessor is able provide students with feedback immediately after they have written it up, instead of having to wait for a class to receive it. In one study[9] they found that 78% of students would like get their feedback electronically .

According to the highlighted article [10] plagiarism is on the rise amongst student. This is where digital marking can help to reduce plagiarism as the programme can do what a human marker cannot. They can compare a submission with thousands of documents and judge if a person has plagiarised. They can also help to show patterns in assignments and marks that normally might go unnoticed.

3.7.2 TurnItIn

There currently exists an on-line electronic plagiarism system called TurnItIn, [30] currently being used by many universities around the world. It allows students to upload their essay assignments online. It then checks for plagiarism in the document by searching the internet and using a large database of documents. After it processes the document it assigns a plagiarism percentage and highlights any areas that were plagiarised. Lecturers can then login and view all the submitted documents and mark them .

Current research highlighted [9] conducted a questionnaire and found that students felt that the system was easy to use and more convenient than having to provide paper copies. It also found that 50% of students strongly agreed and 33.3% just agreed that they preferred to have their grade shown online rather than have a cover sheet.

3.7.3 BOSS System

The BOSS system was developed at the University of Warwick to help deal with their problem of having too many students for the number of staff and yet wanting students to have accurate and quickly available feedback [16]. It is an electronic submission and assessment system created to allow computing science students to submit their programming assignments and have them tested and marked online [17]. The system is not designed to remove human markers completely, instead simply "assist the instructor in achieving a quicker, more accurate and more consistent assessment of programming assignment"[16].

When a file is first submitted it is run through a plagiarism check to make sure that the submission is actually the students own work. It also checks that the submission passes pre-set tests to make sure it works. After this it goes into the evaluations stage, since evaluation attributes of code can be very subjective what the second step does is generate metrics about the submitted program. Some of these metrics are a number of comments and percentage of methods declared [17], which will help human markers evaluate the submission quicker.

3.7.4 Ceilidh

The Ceilidh System was developed at Nottingham University, it is described as "a courseware assessment and management system" [benford'ceilidh:nodate]. It was designed to support teaching and to do so it was required to manage courses, process the assessment of students, support learning and present information to students. The requirement that is of most interest is that of the processing of student assignments.

Ceilidh was to "perform automatic marking of students work in various forms" [benford'ceilidh'1995] these forms included: Computer programs, multiple different types of questionnaires and exercised and finally reports.

According to the highlighted article [benford'ceilidh'1995] at the time of Ceilidh's development Nottingham University computing classes contained roughly 160 students and required multiple staff and post graduate students to mark all the programming submissions. It would often result in a long time between submission and results being handed out, and the marking standard would vary quite significantly.

This all changed with the creation of Ceilidh, students could submit their code and get a response in few seconds, and with it being a machine it was completely impartial- meaning that there was a uniform marking standard across all the students. As Ceilidh was further developed, lecturers gained more control over which statistics were immediately available to students. The digital system also allowed for easy storage of student submissions, so they could be easily referenced in case of plagiarism, also the centralised storage of student results meant that lecturers saved time by not having to gather up all the marked results from other people.

3.8 User Access Views

3.8.1 Social Media

Controlling the view that users have, based on their access level, is common practice. Social media websites for instance allow users to limit what others can see, through the use of a privacy setting [29]. This means that another user's view is determined by the access level they are given: for example, a user that is a friend will have a higher access and be allowed to see their whole feed,

while another user's access may only allow them see the profile name.

3.8.2 Database Controlled Access

The patent highlighted [27], describes a system of limiting user web page access through the use of relation databases. The system would work by using two databases; one would hold a list of all the url's and associated access level, while the second database would hold all the user id's along with their assigned access level. When a user requests a webpage, the access level for that webpage and the user are looked up. If the users do not have the appropriate access they are denied permission to load the page and depending on implementation may be redirected to another webpage. The design of this system is well suited for scalability since no matter how large the two datasets are only one piece of data is need from each database to confirm whether a user is allowed access.

3.9 Custom Input Forms

3.9.1 SurveyMonkey

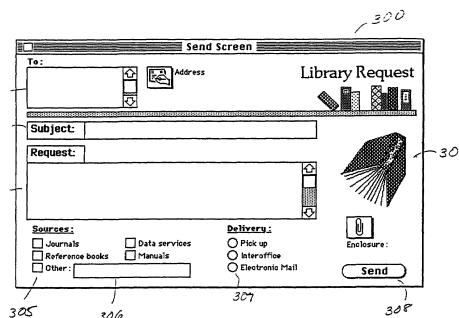
Survey Monkey [12] is an example of custom web forms being created by users. Founded by Ryan Finley in 1999 Survey Monkey enables users to create their own surveys and easily distribute them. It builds the surveys by letting the user select the contents of the question and what the response type will be: The user can also decide if the responses are completely anonymous by default and the participants ip address is stored when they complete the survey. The users can continue to add as many questions as they would like, even after the survey is initially created. After designing the survey the user chooses how they would like to have their survey distributed. The available options that can be selected are a web link, social media, email or embeddable on a website [31].

When participants complete the survey their results are immediately stored and the results of the survey are visible to the user by logging into their account on surveymonkey. They can choose to look at the responses individually or look at metrics about how participants responded.

3.9.2 Customizing Forms In Electronic Mail Systems

A patent [8] describes a process for user-customisable forms in an e-mail system where the administrator selects custom field types and behaviours. For example current e-mail forms have a field for address, subject and one for the the actual message to be sent. While an example of what the patent is suggesting can be seen in figure 1, it shows the inclusion of additional fields allowing for wide variety of form to be created and not limit the users to the few forms that already are designed.

Figure 1: Example Form (Patent [8])



The figure shows a 'Send Screen' window with various fields and options. The title bar is labeled 'Send Screen'. The main area contains a 'To:' field with a dropdown arrow, an 'Address' field with a magnifying glass icon, a 'Subject:' field, and a 'Request:' field. To the right of the 'Address' field is a 'Library Request' section with a bar chart icon. Below the 'Request:' field is a 'Services:' section with checkboxes for 'Journals', 'Reference books', 'Data services', and 'Manuals'. To the right of this is a 'Delivery:' section with radio buttons for 'Print up', 'Interoffice', and 'Electronic Mail'. At the bottom right is an 'Enclosure:' section with a paper icon and a 'Send' button. The entire form is enclosed in a box labeled 300. Various parts are labeled with reference numerals: 305 for the 'To:' field, 306 for the 'Address' field, 307 for the 'Request:' field, and 308 for the 'Send' button.

This increased flexibility in email forms would allow for easier interpretation of messages, making responding or providing information via email a lot simpler and quicker.

3.10 Development Tools

As part of this developing the lab marking system there will be a wide variety of different development tools used. To help understand how the system is developed these different tools and how they function are explained here.

3.10.1 jQuery

jQuery is an open source JavaScript library, that is designed to improve the default Javascript used for web development [19]. It makes it easier to perform many actions such as: event handling, Ajax and animation. jQuery also functions on mobile devices meaning that developers do not have to worry that scripts developed for desktop computers will not function correctly on mobile devices.

3.10.2 Ajax

The way that normal websites function is that the user sends a request to the server, the server then processes the user's requests and then creates and sends a response. Once the the user receives this response the web browser interprets it and displays the web page[26]. This process model causes

long periods of delay for users as they have to wait for the server to process their request and then wait for the browser to process the response. Meaning users have very disjointed experiences as they have to wait for the whole page to reload every time they try to navigate through it. This is where Ajax comes in as it provides an improved process model that enhances users' experience.

Ajax stands for asynchronous JavaScript and XML [18]. It enables browsers to communicate with servers without needing a web page to reload. Ajax's requests are triggered using javascript, though as stated in the previous section (3.10.1) jQuery made the process of triggering Ajax's requests a great deal easier.

Ajax is made up of a collection of different technologies but each individual technology can exist on their own [13]. The technologies that make up Ajax are:

- HTML and CSS
- Dynamic display and interaction
- Asynchronous retrieval of data
- Javascript

3.10.3 Bootstrap

Bootstrap is a front-end framework [2] designed to help developers create dynamical responsive websites quickly and easily. It provides cascading style sheets and java-script files which are designed to work on all devices.

The way which bootstrap helps create a dynamically responsive website is through the creation of a grid pattern. It breaks the screen up vertically with 12 equally sized sections calculated by percentage. When elements are created the developer declares how many sections wide the element will be. This means that if the screen is scaled down it will still maintain that percentage of the screen size, so the element will scale larger or smaller. Developers can also set an element to use a different number of sections depending on the size of the screen, which means that if a table is initially given four sections then when the screen squashes the table too much then it can be given more sections and therefore more room to expand.

Bootstrap also provides pre-designed navigation bars, side bars, easily customisable button designs and form designs. Altogether this will help speed up the development of the user interface.

The reason that I am using Bootstrap for this project is that is specifically designed for projects that use mobile devices. This will make is easier to build a desktop and mobile version of the lab marking system meaning more time can be spent of the development application's actual functionality.

3.10.4 PHP Unit

PHP Unit is a testing framework for PHP. It is designed to allow the easy creation of automated tests in the form of unit tests [1].

The most recent stable version is PHPUnit 6.0, this version is only compatible with PHP version 7 and was released on the 8th of February 2017. Since I am using PHP 5.6.25, as stated earlier in Section 3.5, I cannot use this most recent version and will instead be using an older stable version 5.7 which supports PHP version 5.6, 7.0 and 7.1. That version was released on 2nd December 2016.

3.10.5 D3

D3 [3] is a javascript library, which was designed for the creation of interactive visualisations of data and was first developed in 2011. D3 uses precreated Javascript functions to create scalable vector graphics (SVG's) which are embedded into the HTML of websites."SVG is a language for describing two-dimensional graphics in XML"[11] and can have displays changed using Cascading Style Sheet (CSS).

Data-sets can also be bound to an SVG allowing for a visual way to interpret the data-set, and as the data-set changes the SVG will be changed allowing for a dynamic display.

3.10.6 CakePHP

CakePHP is an open source framework for php. It is developed to help the rapid development of web applications and make them simpler,faster and less complex to build [5]. It allows the

development of well structured and robust web applications [25], and its latest version is version 3.3 called "Red Velvet" which can be downloaded from (<https://cakephp.org/>).

The usefulness of the framework comes in the form of its predefined functions, which allows for a more secure code, since you cannot accidentally forget to escape character from input as the functions automatically do this. Cakephp helps to improve the maintainability of systems as it is easy to understand with plenty of comments.

A downside to cakephp is its testability, compared to other web frameworks it lacks in testing tools and does not contain a testing environment [25].

4 Requirements

4.1 Requirements Analysis

To help discover the requirements of the lab marking system I created a requirements analysis questionnaire(Full questionnaire in Appendix ??), the questionnaire was given both to lab helpers and to lecturers to help understand their individual requirements from the system.

The results of the requirements analysis were used to develop and refine the list of requirements displayed- functional requirements can be found in section 4.2.2 and non-functional requirements in section 4.2.3.

4.2 Requirements

Requirements for the system are each given an id depending on the type of requirement: FR for functional requirements and NFR for non-functional requirements.

Along with this, each requirement has a description stating what the requirement is and a priority is assigned using the MoSCoW prioritization method discussed in the next section (4.2.1).

4.2.1 MoSCoW Prioritisation

MoSCoW Prioritisation [22] was designed to be used in projects that utilise agile development and have a fixed time limit in which to be completed. Its method is that all the requirements of the project are prioritised into four categories, these are:

- **Must:** This category of requirements makes up the "Minimum Usable SubseT" of requirements that the project must implement to be successfully delivered.
- **Should:** This category contains requirements that are not vital to the delivery of the project, but their absence can affect the project overall.
- **Could:** This category contains requirements that are wanted in the system but are less important than the "should" priority. Reasons that a requirement "would" become a "could"

instead of a "should" are if the amount of time taken to implement the requirement and how much of an affect its absence would have on the project.

- **Won't:** This category contains all the requirements that will not be implemented in the delivered project, but could be implemented in later development.

For this project I will be attempting to implement all 'the must priority functional and nonfunctional requirements'. I will also try and implement all of the "should" requirements. Finally I will implement as many of the "could" priority requirements that I can, starting with ones which best improve the system.

4.2.2 Functional Requirements

Functional requirements also include an access column which defines what users should be able to use. Some requirements are restricted to lecturers and lab-helpers and not able to be accessed by students. The table is sorted first by access levels- starting with requirements accessible to all users, then sorted in access order 1 - 4. Secondly it is sorted by priority.

The access levels are: 1-Admin, 2-Lecturers, 3-Lab Helpers and 4-Students

Table 1: Functional User Requirements

ID	Requirement	Access	Priority
FR-01	Login to view system	1,2,3,4	Must
FR-02	Accounts created for them	1,2,3,4	Must
FR-03	Change password	1,2,3,4	Must
FR-04	Logout	1,2,3,4	Must
FR-05	Login using university ID	1,2,3,4	Could
FR-06	Remove students from courses	1, 2	Must

FR-07	Update student accounts	1,3	Could
FR-08	Look up students in lab	2,3	Must
FR-09	Select students from lab list	2,3	Must
FR-10	Leave comments about students	2,3	Must
SR-11	Save marks	2,3	Must
SR-12	Update marks	2,3	Must
SR-13	Delete marks	2,3	Must
FR-14	Search for student by name	2,3	Should
FR-15	Mark student even if they are not in the system	2,3	Could
FR-16	Assign students to courses	1	Should
FR-17	Assign lectures to courses	1	Should
FR-18	Create marking schemes	2	Must
FR-19	Display generated stats	2	Must
FR-20	See submitted marks	2	Must
FR-21	Generate end of year spread sheets	2	Should
FR-22	Editing of students in class	2	Should
FR-23	Create peer marking scheme	2	Could
FR-24	Look at students stats	2	Should
FR-25	Set what parts of the marking scheme students can see	2	Could
FR-26	Update marking scheme	2	Should

FR-27	Delete marking schemes	2	Should
FR-28	Able to assign students to set labs	2	Could
FR-29	Set penalties for late marking	2	Could
FR-30	Able to export to vision	2	Could
FR-31	Access Marking Scheme	3	Must
FR-32	Enter selected student's mark	3	Must
FR-33	Submit student's mark	3	Must
FR-34	Select the lab they are helping in	3	Must
FR-35	See current mark	4	Must
FR-36	Show different displays depending on access level		Must
FR-37	Load student's current lab mark scheme		Must
FR-38	Apply penalty for late lab completion		Could
FR-39	Create a set of useful stats based on lab		Must
FR-40	Store what class student belong too		Must
FR-41	List of all students in class		Must

4.2.3 Non-Functional Requirements

Table 2 lists all the non-functional requirements for the development of the system which are ranked in order of priority.

Table 2: Non-Function Requirements

ID	Requirement	Priority
NFR-01	All person data encrypted	Must
NFR-02	Update stats as marks are entered	Must
NFR-03	Take less than 2 seconds to generate stats	Must
NFR-04	PHP Should use prepared statements	Must
NFR-05	Website dynamically designed	Must
NFR-06	Should make sure inputs are valid	Must
NFR-07	Should prevent SQL Injection	Must
NFR-08	HTML, CSS and Javascript should be validated	Should
NFR-09	Function on a wide variety of smart phones and tablets	Should
NFR-10	Handle a large number of users without any faults	Should
NFR-11	Check passwords contain alphanumerics and have a minimum and maximum length	Should
NFR-12	Take less than 2 second to load student marking scheme	Should
NFR-13	Auto save marks as they are entered	Could
NFR-14	Auto record what lab help marked what student	Could
NFR-15	List all students who did not attend the lab	Could

NFR-16	Track how long it takes to mark a student	Could
NFR-17	Disability options (Increase text size, colour layout)	Could
NFR-18	Readable by screen readers	Could
NFR-19	Group marked people	Could
NFR-20	Backup database regularly	Could
NFR-21	Retrieve student images from university system	Won't

5 Design

This section is to explain all the design decisions taken in developing the marking system using sketches, models and diagrams and thus assisting in the understanding of how the system works as a whole. The design aspects that are covered are: the user interface, database design and the main functionality required to make the system work.

5.1 User Interface Design

For the system to be useful the user interface must be easy to use and understand, along with this it will have to be functional on mobile devices as well. To achieve this I have developed mock-ups for main web pages required to make the system work, including a design for both desktop and mobile. For both the desktop and mobile designs there will be explanations of why specific design decisions were made and how they are meant to function.

5.1.1 Lab Results

To make the new digital lab marking system useful for students and lecturers both need to be able to see the results of a lab. For this part of the application there needs to be two different views depending on the type of person signed in. This is because students need to see only their own results, while lecturers need to be able to select students and see their marks. To help develop the two different displays I have created a design for how lab results should be displayed for students and another one for lecturers.

Desktop Design

The desktop views for both students and lecturers are designed in similar ways (figure 2), with the result of labs being shown in a table structure at the centre of screen.

The student view (figure 2a) shows all the courses that the specific student is on, along with each of the labs that those courses have. The labs are then broken down into their individual marks making it easier for students to understand. The information contained in the break down is: what the question was, what the submitted answer was and how many marks they gained for

the question. This makes it easy for students to find where they gained or lost marks and should help them understand how to improve for future labs.

The lecturer view (figure 2b) has the results shown in a similar way. Though due to the fact that for each lab they will have x number of students it would be useless to display all of them at once as the lecturer would have to scroll far down to find specific labs and students. Instead the lecturer selects the course / lab for which they want the result from the side bar. Then all the students on that courses are shown with their marks underneath them. The lecturer can use the search bar at the top to find specific students.

Figure 2: Results Page: Desktop Designs

Course A			
Lab A			
Question Number	Question	Answer	Mark
1	Question 1	Answer	1/1
2	Question 2	Answer	1/1
3	Question 3	Answer	1/1
Lab B			
Question Number	Question	Answer	Mark
1	Question 1	Answer	1/1
2	Question 2	Answer	1/1
3	Question 3	Answer	1/1
Course B			
Lab A			
Question Number	Question	Answer	Mark
1	Question 1	Answer	1/1
2	Question 2	Answer	1/1
3	Question 3	Answer	1/1
Lab B			
Question Number	Question	Answer	Mark
1	Question 1	Answer	1/1
2	Question 2	Answer	1/1
3	Question 3	Answer	1/1

(a) Student Results Screen

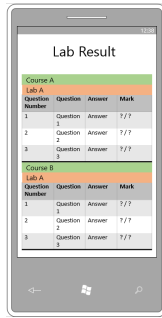
Course A Lab B			
Question Number	Question	Answer	Mark
1	Question 1	Answer	1/1
2	Question 2	Answer	1/1
3	Question 3	Answer	1/1
Course B Lab A			
Question Number	Question	Answer	Mark
1	Question 1	Answer	1/1
2	Question 2	Answer	1/1
3	Question 3	Answer	1/1
Course B Lab B			
Question Number	Question	Answer	Mark
1	Question 1	Answer	1/1
2	Question 2	Answer	1/1
3	Question 3	Answer	1/1

(b) Lecturer Result Screen

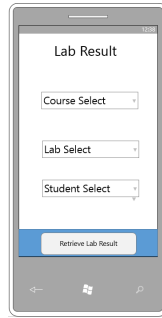
Mobile Design

The mobile design (figures 3) of the results page uses the same table display method for the displaying of the results and in the student version is identical. While for lecturers when they load the results page (figure 3b) they select the course, lab and student they wish to view. This is to make it easier to navigate for the lecturer as they do not need to scroll through all the students. In addition when the results load there is a back button to return and select a different student.

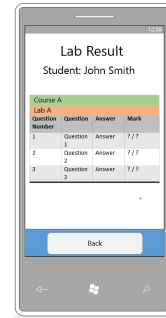
Figure 3: Results Page: Mobile Designs



(a) Student: Result Screen



(b) Lecturer: Result Selector



(c) Lecturer: Result Screen

5.1.2 Lab Marking

For the lab marking system to be useful lecturers and lab-helpers will have to be able to mark students. The process will also have to be simple as every student in a lab will need to be marked and this project is trying to make the process of marking easier and not harder.

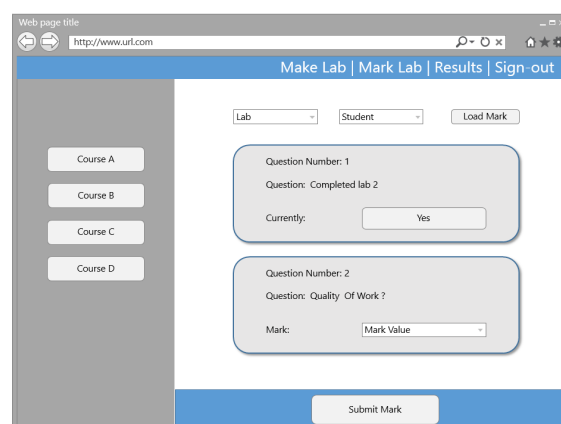
To facilitate this the "Lab Marking Page" is designed to make quick and easy to select which course and lab they would like to mark, and then be provided with a list of students for the lab- thus making it simple to select the student they which to mark.

Desktop Design

The desktop design (figure 4) uses the side bar to let lecturers / lab-helpers select the course they are wanting to mark. Once they have selected the course they wish to mark two drop down menus appear= one to select the lab they wish to mark, and the other for the student. Once both are selected they click the load mark button which displayed the marking scheme for the lab. It will also be auto filled with the students previous marks, if they have already been marked.

The markers then input their marks using a combination of differ-

Figure 4: Marking Page: Desktop Design

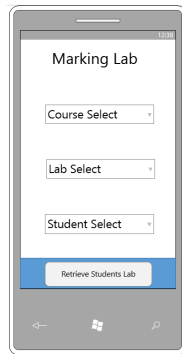


ent input types (depending on question type) and once they are happy with the mark they then click 'submit mark' and the student's mark is saved and is visible to the lecturer and student on the results page.

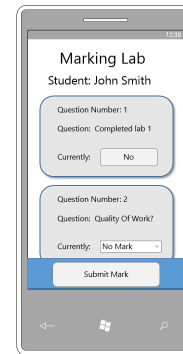
Mobile Design

I would expect that a majority of markers will use the mobile devices to mark, since it easier carry a mobile device than to have to login to a computer every time they mark a student. The mobile design figure(5) is designed with this in mind, the marking page is broken into two different pages.

Figure 5: Marking Page: Mobile Designs



(a) Selector Screen



(b) Marking Screen

The first page (figure 5a) is the selector screen and replaces the course selection side bar buttons and the drop down menu from the desktop version. Instead the marker is provided with three drop down menus which are: course, lab and student. Once they are filled out, they click the retrieve button at the bottom which will take them to the second page.

The second page (figure 5b) is where the lab marking scheme will be loaded. The marking of labs is the same as it was for the desktop design, when the 'submit mark' button is pressed the marker is taken back to the selector screen. This time the course and lab drop down menu are filled so the marker simply has to select a new student making the process quick to repeat.

5.1.3 Lab Creation

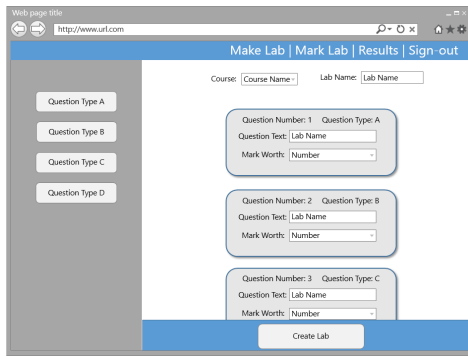
The "Lab Creation" page will be very important for lecturers as they will have to use it to create all their labs. Given that courses tend to have a large number of labs it is likely lecturers will have to use this part of the system fairly often, therefore the process for creating a lab should be as simple as possible. These ideas are reflected in the designs shown in figures(6 & 7).

Desktop Design

The desktop design shown in figure(6) shows a lab in the process of being created. When the lecturer first loads the "Lab Maker Page" the main section will be empty, except for the drop down menu to select the course and lab name input. Lecturers will be able to select question types from the side bar by clicking the button representing the question type; this will add a question tile to the main section.

The question tiles are designed with simplicity in mind; they state at the top what question type and number it is. Lecturers will be able to input what the question will be, they will also, depending on the type of question, select how many marks it is worth from the drop down menu. For instance if the question is a scale they can select the minimum mark and maximum mark.

Figure 6: Lab Creation Page: Desktop Design



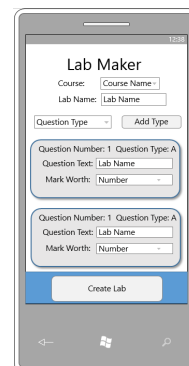
while if the question is a yes / no question they select marks that will be received if yes.

Once the lecturer has added all the questions that they would like to the lab, they then click the 'create lab' button. If something is missing in the lab and an error will appear and highlight where the issue actually occurred or else the lecturer will be take to the lab management screen.

Mobile Design

The mobile design (figure 7) is the same as the desktop design but the side bar has been removed and replaced with a drop down selection box underneath the course and lab input. This drop down menu functions the same way as the buttons did, but now lecturers select the question type they are want to add then click the

Figure 7: Lab Creation Page: Mobile Design

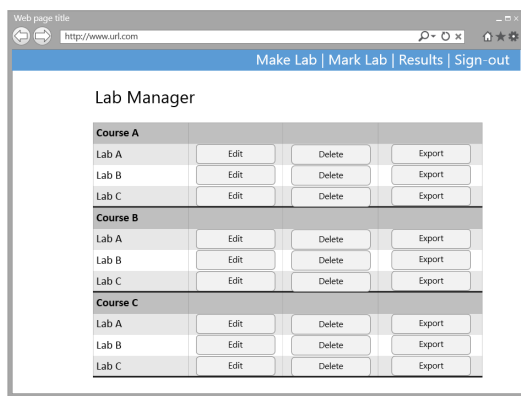


'add Type' button, and this will add the question tile to the main screen.

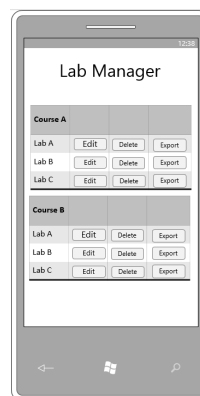
5.1.4 Lab Manager

The "Lab Manager" page is what lecturers will use to control labs which they have already created. Functionality available from here will include: the ability to delete, edit and be able to export all the results for a selected lab. The desktop and mobile designs can be seen in the figure(8).

Figure 8: Lab Manager Page



(a) Desktop Design



(b) Mobile Design

Desktop Design

The lab management page is designed to be easy to understand and allow lecturers to perform actions on labs without the need for complex menus. To allow this simplicity the desktop design for the lab management is laid out in a table structure with labs being broken down into their respective courses. This makes it easy for lecturers to find the lab they are wanting to manage. Then for each of the lab there are four buttons: 'edit' which will let the lecturer edit the lab, 'delete' which will delete the lab and finally 'export' which will export all the results of the lab as a spread sheet that can be uploaded to vision.

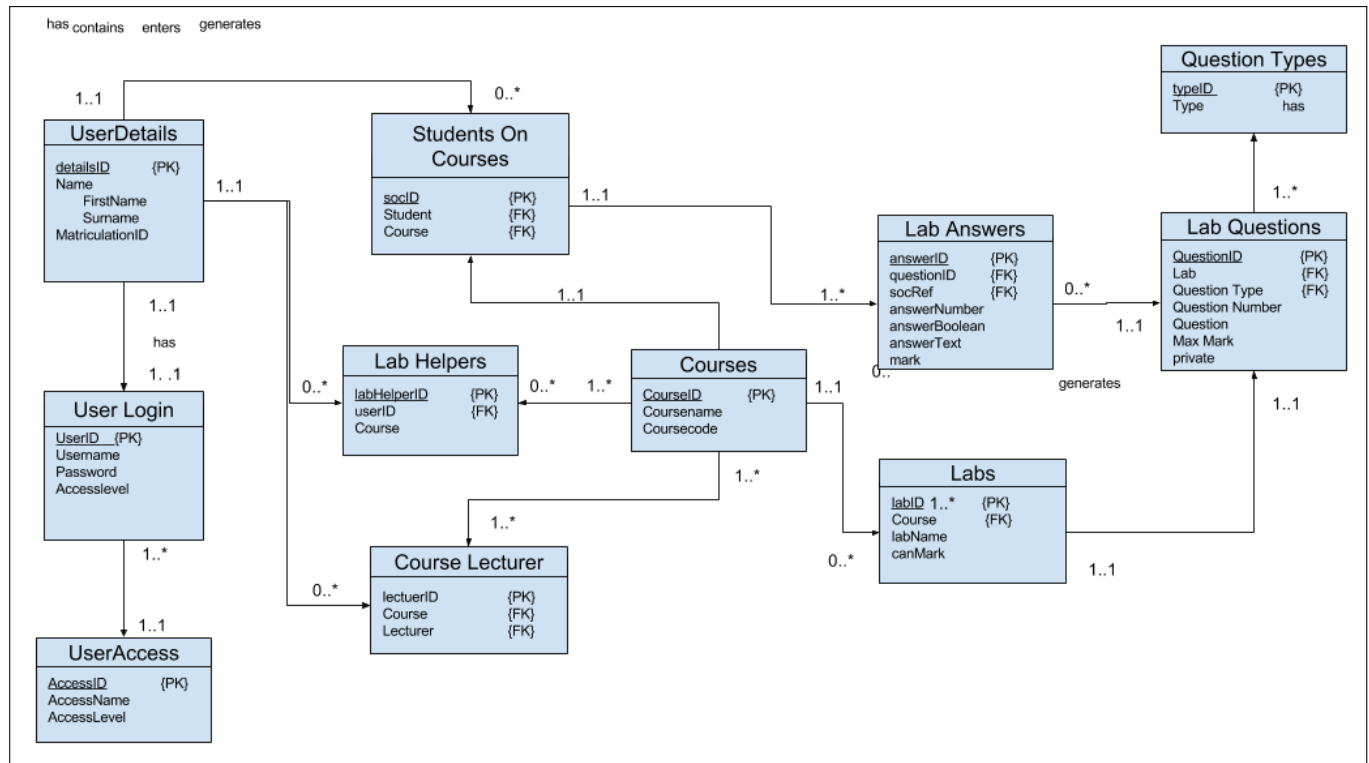
Mobile Design

The mobile design for the lab management page is exactly same as that of the desktop design.

5.2 Database Design

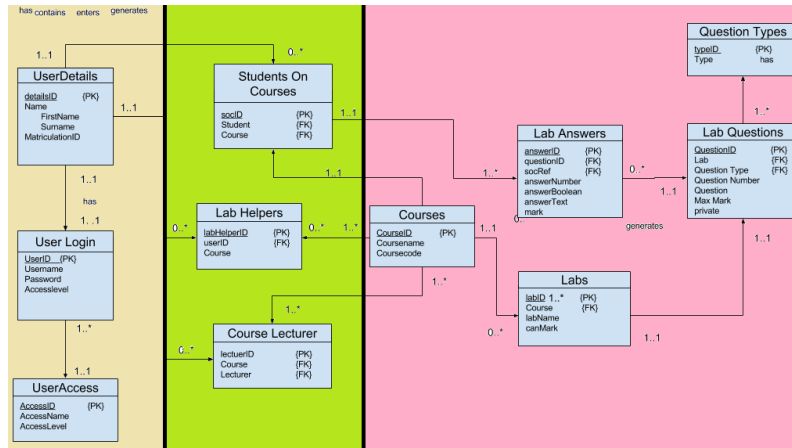
For the system to work there will need to be database that stores students, labs, and marks received. The database schema I will be using for this system is shown in figure(9) below.

Figure 9: Database Schema



The database can be broken down into three sections as shown in figure(10). The first section ([lfm]) is designed for storing all the users of the database, it is designed to be easily changed to allow for easy integration into already existing databases. As long as users have unique ID's the rest of the database will be able to function with no changes being required.

Figure 10: Database Sections



Section two ([lfm]) is comprised of the tables: Students On Courses, Lab Helpers, Course Lecturers and Courses and is used to know what users are on what specific course.

- **Course Table** Stores all the university courses, each course is stored with its own unique ID, the course name and the course code.
- **Students on Courses Table** Stores what students are on what course each record represents a student and one course they are on. The structure of the record is that it has its own unique ID, contains the user ID of a student and the course ID for the course they are on.
- **Lab Helpers Table** Stores the lab-helpers user ID and the course ID of the course they are a lab-helper for.
- **Course Lecturer** tables stores lecturers user ID and the course ID for which they are a lecturer for.

The third section ([lfm]) is the most important part of the database design as it is to do entirely with the storage of information relating labs and marks, it contains the tables: Courses, Labs, Lab Questions, Question Types and Lab Answers.

- **Labs Table:** This table stores all the different labs and the details about them. Each record has the course ID for the course the lab belongs to, it also contains a name for the lab and whether the lab can currently be marked by lab-helpers.

- **Lab Question Table:** This table stores all lab questions, each question consists of a Question ID to identify the question, the ID of the lab it belongs to, the ID of what type it is, its question number, what the actual question is, what the maximum mark that can be given to the question is and finally whether the question results can be seen by students or not.
- **Question Type Table:** This table stores the different possible questions types. each record represents a question type and is made up of a ID number and the name of the question type.
- **Lab Question Answers:** This table is responsible for storing all the answers and marks given to students, each record contains the mark for one question for one specific student, meaning multiple records are needed to get the result of one lab for a student. The structure of a the record is firstly it has its on unique ID, then the ID of the question it is the answer for, followed by the ID of the student. Then because there can be different types of answers submitted there is a attribute for number answers, text answers and Boolean answers, usually only one of these will be used by the question. Finally it stores the mark that was given to the answer.

5.3 Functionality Design

To help build the system quickly it is important to figure out the key functionality that will be needed to implemented. To that end I created a UML diagram (figure [lfm]) that shows the major functions of the lab marking system and additionally shows what users will be able to preform what functionality.

Students

Students are the simplest users of the system the only key functionality that they need to have access to is the ability to view there own lab results.

Lab-Helper

Since Lab-Helpers are also student they have the same functionality to view their results as them, but in addition to this Lab-Helpers also have the ability to select and mark students on courses that they have been assigned too.

Lecturers

The majority of the functionality for the lab marking system is used by lecturers. They use the same functionality as lab-helpers (lab marking & view result) but for view results they do not view their own but instead can select a student on their course and view their mark. Lecturers also have the ability to create labs, as well as edit or delete already created labs, they will also be able to export labs as spread sheets that can be uploaded to vision.

Admin

Admins have the same functionality as lecturers, but unlike lecturers are able to view all courses for both results and marking. Additional functions they will be able to access over lecturers are management of the database and assigning lecturers to courses.

6 Implementation

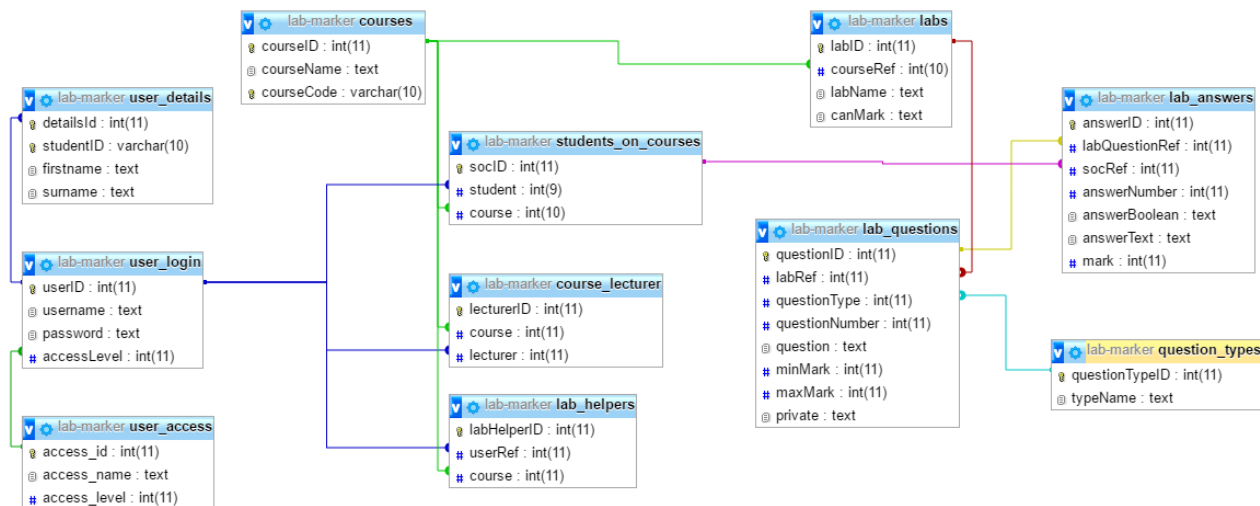
This section documents and explains how the lab marking system was developed and the key functionality works.

This is done through the use of screen shots showing the functionality, code snippets with explanations on what it does and how.

6.1 Database

The database schema discuss in the design section (5.2) was converted into a relational database using mysql. Figure 11 displayed below shows the database structure and the lines between tables shows their relations. There a no major changes from the original design, the only difference that occurred during implementation was that some of the attribute names were changed different.

Figure 11: Implemented Database Schema



6.2 PHP Class Structure

All the PHP developed as part of this project was built on a class structure, each class contains the functionality for a part of the system. Many of the classes inherit functions from other classes and the relationship diagram can be seen in figure([lfm]).

When it came to the creation of classes and their inheritance there are limitation created by

PHP. In PHP class are only able to extend one other class, the only way to inherit multiple classes is to create an instance of it instead. This is usually done in the construct so that the class may be accessed through a private variable.

In my classes I do a combination of constructor declared instances, and instances declared in functions

6.3 Background Functionality

This section is for the classes and functionality that are not dedicated to one part of the system, but are used by multiple different classes to provided key functionality for them to work.

6.3.1 Database Connection

The ConnectDB class is designed to create a connection to the database and allow other classes to retrieve information through the connection. The class also has a secondary function of checking if a session has been started and if not starting one, this is to allow access to sessions that may be needed for querying the databse. The full code listing can be seen below (1).

Listing 1: Database Connection Class - PHP

```
1 class ConnectDB
2 {
3     public $link;
4
5     function __construct()
6     {
7         $this->link = mysqli_connect('host', 'username', 'password');
8         if(!$this->link) {
9             die('Could not connect to MySQL: ' . mysqli_connect_error());
10        }
11
12        mysqli_select_db($this->link, "lab-marker"); //Selects the Database
13
14        if (session_status() == PHP_SESSION_NONE)
15            session_start();
16    }
17 }
```

The way that the class works is, it contains one public variable called \$link; when the class is initialised the constructor function called. The constructor makes a mysqli connection using the provided host, username and password details, it then sets the \$link variable to the result of the

connection. After this it check if the connection was successful, if it is not then a die command is thrown and an error message is shown. If it is a successful connect then is selects the "lab-maker" database. Checks if a session is already run, and starts one if it is not.

6.3.2 Security

The Security PHP class was developed to handle checking users access rights. This is to make sure that only users with the required accesslevel are able to run functions. The security class is quite long so I have shortened it to only the key functions required to preform this task.

The `getAccessValue` function shown below converts a passed in access name to what its access level is. It does this by first making a connection to the database and run a prepared mysqli statement that selects the `access_level` for the provided access name. Once the result is receive the connection to the database is closed, the result is then checked to see that it exists if it does the access level is return. Otherwise -1 will be return to inform the check that access name does not exist and to not grant access.

Listing 2: Convert Access Name To Accesslevel - PHP

```
1 //Returns the access value of an access name
2 public function getAccessValue($access_name)
3 {
4     $con = new ConnectDB();
5
6     $get_access_level = mysqli_stmt_init($con->link);
7     mysqli_stmt_prepare($get_access_level, "SELECT access_level FROM
8         user_access WHERE access_name= ?");
9     mysqli_stmt_bind_param($get_access_level, 's', $access_name);
10    mysqli_stmt_execute($get_access_level);
11    $result = mysqli_stmt_get_result($get_access_level);
12
13    mysqli_close($con->link);
14    ($result->num_rows === 0) ? $value = -1 : $value = $result->
15        fetch_row()[0];
16    return $value;
17 }
```

The security class has two different check for a users access right. The first check is to see if the user has the required access. This is done by by calling the `hasAccessLevel` function and passing in the required access in the form of the access name. The function then uses the `getAccessValue` function describe earlier to retrieve its access level. The users access level that is stored in a session

is then compared to retrieved access level, if the users is equal to or greater than the received one then true is returned allowing the user access to whatever the security check protected. How ever if the users access level is lower or retrieved result is less than zero (invalid access name) then false is returned preventing the user from preforming the protected action.

Listing 3: Has Access Required - PHP

```
1 //Returns true if user has at least the required access level
2 public function hasAccessLevel($access_name)
3 {
4     if (session_status() == PHP_SESSION_NONE)
5         session_start();
6     $required_access = $this->getAccessValue($access_name); //Gets
        access value for required accessname
7     return ($_SESSION["accesslevel"] >= $required_access &&
        $required_access >= 0); //Returns True if
        useraccess is greater or equal to required accesslevel
8 }
```

The second check works the exact same way as the first but instead of checking the user has at least the access requested, they must have a greater access level. This check is done using the hasGreaterAccess function and its code listing can be seen below.

Listing 4: Has Great Access Than - PHP

```
1 //Returns true if user has a greater access level than passed in
2 public function hasGreaterAccessThan($access_name)
3 {
4     if (session_status() == PHP_SESSION_NONE)
5         session_start();
6
7     $required_access = $this->getAccessValue($access_name); //Gets
        access value for required accessname
8     return ($_SESSION["accesslevel"] > $required_access &&
        $required_access >= 0); //Returns True if useraccess is greater
        than required accesslevel
9 }
10 }
```

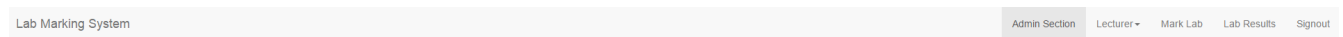
These three functions make it easy to check if a user has required access through the use of if statements and the returned results from the checks. Examples of this in practise are visible in future sections when explaining other functionality.

6.3.3 Dynamic Loading

Dynamic loading of the different selection screens and the marking scheme are important. Firstly it means that every click does not require the complete loading of a new page and therefor slow down how fast users can mark students.

6.3.4 Navigation Bar

Figure 12: Navigation Bar



The main navigation bar (figure 12) at the top of each page is designed to change depending on what user is logged in. This controls what functionality each of the different access levels has, for instance students only have the Lab Results tab, while Lab-Helpers also have Marking tab. All the different navigation bar can be seen in figure(13), the navigation bars from top are: admins, lecturers, lab-helpers and at the the students navigation bar.

Figure 13: Navigation Bar



In addition the navigation bar is not hard coded in any of the lab managers HTML files. It is instead in its own file which is loaded every time a page is loaded. This is so that one change to the navigation bar occurs to all of them and time does not have to be spent going through each file update the navigation bar.

Changing Navigation Bar

The navigation bars layout is done using a PHP script (figure 5) which is included in default structure of the navigation bar. The script decided what buttons should be included in the navigation bar and now how the navigation bar should look. This is done by using the Security class described previously ([lfm]) to check the users access level and add the appropriate buttons.

Listing 5: Navigation Bar Button Adder - PHP

```
1 require_once(dirname(__FILE__) . '/../core/classes/Security.php');
2 $secure = new Security();
3
4 $nav_bar_content = '<li id="results-nav"><a href="labresults.php">Lab
    Results</a></li>
5                     <li id="signoutbtn"><a href="../../php/core/signout.php
                        ">Signout</a></li>';
6
7 if($secure->hasAccessLevel("lab helper")) {
8     $nav_bar_content = '<li id="marking-nav"><a href="marking.php"> Mark
        Lab </a></li>' . $nav_bar_content;
9
10 if ($secure->hasAccessLevel("lecturer")) {
11
12 $nav_bar_content = '<li id="labs-nav" class="dropdown">
13                     <a href="#" class="dropdown-toggle" data-toggle="
                        dropdown" role="button" aria-haspopup="true" aria-
                        expanded="false">Lecturer<span class="caret"></span
                        ></a>
14                     <ul class="dropdown-menu">
15                         <li><a href="labresults.php">Lab Results</a></li>
16                         <li role="separator" class="divider"></li>
17                         <li id="course-nav"><a href="coursemanager.php"> Course
                            Manager </a></li>
18                         <li><a href="labmanager.php">Lab Manager</a></li>
19                         <li role="separator" class="divider"></li>
20                         <li><a href="labmaker.php">Make Lab</a></li>
21                         </ul>
22                     </li>' . $nav_bar_content;
23
24 if ($secure->hasAccessLevel("admin")) {
25     $nav_bar_content = '<li id="admin-nav"><a href="admin.php"> Admin
        Section </a></li>' . $nav_bar_content;
26 }}}
27 echo $nav_bar_content;
```

The script creates a variable that contains default buttons (results and sign-out) it then uses a nested if statement to check how high an access level the user has by using the `hasAccessLevel` function from the Security Class. If the user has the required access level then the buttons related to that level are added to the variable, once all the access levels have been checked the variable is echoed adding its content to the navigation bar.

Loading Navigation Bar

```
1 function include_navbar (section) {  
2  
3     $(document).ready(function(){  
4         $("#navbar_area").load("../components/navbar.php #navbar_code",  
5             function () {  
6                 $("#" + section + "-nav").addClass("active");  
7             });  
8     });  
9 }
```

Listing 6: Load Navigation - JavaScript

The navigation bar is loaded by including the "navbar.js" file on any of the pages where the navigation bar is wanted. The function `include_navbar` is then called passing in the name of the page that the user is currently one. The `include_navbar` function can be seen in the code listing above (figure 6), the way that it works is that it uses jQuery to wait until the web documents has been loaded "`$(document).ready`". Once the web page is loaded the jQuery load function is used, load allows JavaScript to load html from another file into a specified area on the current page. In this case it loads the "navbar.php" file into an element with the id "`#navbar_area`", once it is loaded using the passed in section name jQuery highlights its respective section in the navigation bar so that users can easily tell what section they are in.

6.4 Lab Creator

6.4.1 Design

The design of the lab creation page can be seen in figure(14) is almost identical to what its initial design showed. There is only one difference and that is to the question tiles, there was the inclusion of a button that can be clicked to assign whether the questions results will be shown to students or not.

There are three key pieces of functionality required to make the creation of labs possible each are discussed in more detail later. The first is being able to click the question types and have the question tile be added to the page. The second is the most important and it is how the created lab is processed and stored in the database and finally being able to detect errors in the submitted lab before it is processed and uploaded to the database.

Figure 14: Lab Creation Page

The screenshot shows the 'Lab Marking System' interface. On the left is a sidebar with buttons for 'Scale Question', 'Boolean Question', 'Value Question', and 'Text Question'. The main content area has a 'Course' dropdown set to 'Games Programming' and a 'Lab Title' input field with 'Lab 1'. Below these are three question tiles. Each tile has a 'Question Number' and a 'Type'. The first tile is 'Question Number: 1' with 'Type: Scale'. It includes a 'Question' input field, 'Select Questions Minimum Mark (select one):' and 'Select Questions Maximum Mark (select one):' dropdowns, and a 'Result is:' section with a green 'Visible' button and 'to students' text. The second tile is 'Question Number: 2' with 'Type: Boolean', featuring a 'Select Question Value (select one):' dropdown and a similar 'Visible to students' section. The third tile is 'Question Number: 3' with 'Type: Text'. At the bottom of the main area is an orange 'Create Lab' button.

6.4.2 Generating Question Tiles

Question tiles are generated using jQuery, Ajax to call a PHP class that creates the question and returns its. The function used to call the PHP is shown in listing(7), this function is run when one of the question type buttons is clicked. The button passes the type of the question to the Ajax request which posts then posts it, along with the question number and a unique Id number. All of which is used by the PHP class to create the new question tile. Once the PHP class has

successfully process the input the result is then appended to the screen, if an error occurred an alert is shown to inform the user of an issue.

```
1 function include_navbar (section) {
2   var qCount = 0;
3   var maxQ = 1;
4
5   function add_question(type)
6   {
7     $.ajax({
8       type: 'POST',
9       url: "../php/labs/add_lab_question.php",
10      dataType: 'json',
11      data: {type:type, id: qCount, qnum: maxQ},
12      cache: false,
13      success: function(result){
14        qCount++;
15        maxQ ++;
16        $.when($("#form-area").append(result.question)).then(
17          scroll_bottom());
18      },
19      error: function(xhr, status, error) {
20        alert(xhr);
21      }
22    });
23  }
```

Listing 7: Add Question - JavaScript

The PHP file called by the Ajax request (list 8) checks that the three variables (type, question number & id) were posted, it then creates an instance of the LabMaker() class. Finally it checks that the id and question number are both numbers and calls the createQuestion Function passing all three variables into it. The result of this function is then echoed therefor providing Ajax with its result data.

```
1 function include_navbar (section) {
2   require_once "classes/LabMaker.php";
3   if(isset($_POST["type"]) && isset($_POST["id"]) && isset($_POST["qnum"]))
4   {
5     $maker = new LabMaker();
6     $type = $_POST["type"];
7     $id = $_POST["id"];
8     $qnum = $_POST["qnum"];
9
10    if(is_numeric($id) && is_numeric($qnum))
11      echo($maker->createQuestion($type,$id,$qnum));
12  }
13 }
```

Listing 8: Call Create Question - PHP

The LabMaker class is quite large, to better understand how questions are generated I have broken it down in to the main functions needed to create a new question. The first function is the createQuestion function which was called by the previous PHP script, this function takes the type of question then calls related function for making it through the use of a case statement. This is because different question types result in the question needing a different layout.

```
1 public function createQuestion($type,$id,$question_num)
2 {
3     switch ($type)
4     {
5         case "boolean":
6             return $this->booleanQuestion($id, $question_num);
7             break;
8         case "scale":
9             return $this->scaleQuestion($id, $question_num);
10            break;
11         case "text":
12             return $this->textQuestion($id, $question_num);
13            break;
14         default:
15             return $this->unknownQuestion($id,$question_num);
16     }
17 }
```

Listing 9: Create Question Function - PHP

To show how a question is created we will use the scaleQuestion function (listing 10). The functions called by it are the same as for the other question types the only difference is that two input are provided one for max mark and the other for the minimum.

```
1 private function scaleQuestion($id, $question_num)
2 {
3     $scale = $this->startQuestion($id);
4     $scale.= $this->title("Scale", $question_num);
5     $scale.= $this->questionType("scale");
6     $scale.= $this->textInput("question[]");
7     $scale.= $this->scaleInput("Select Questions Minimum Mark","min-value[]");
8     $scale.= $this->scaleInput("Select Questions Maximum Mark","max-value[]");
9     $scale.= $this->visablityButton($id);
10    $scale.= "</div>";
11
12    return json_encode(array('question'=>$scale));
13 }
```

Listing 10: Scale Question - PHP

To make it easy to add in new question types the structure of a question is made up of calling different functions to add html tags and content to a variable which once all run creates a completed question. The first function "startQuestion" added the opening div tags and the closes button, the next function "title" add the tags for the provided tile and the question number. After this the "questionType" function adds a hidden input that stores what the type of the question is. Then the "textInput" adds a text box for users to enter the question text into. It is at this point that different functions are called depending on the type of function, in the case the scale type question two scale inputs are created. The question is then finished of by adding the visability button and the closing div tag the result is then returned as a json object which Ajax can decode and display in the browser.

6.4.3 Create Lab

The lab is submitted using a form and all the inputs are stored in arrays rather than individual variables, this allows the form to be expanded infinitely without having to make a custom PHP script for each combinations. Arrays are created by adding a pair of square brackets ([]) on to the end of a input name, this means that each inputs that uses that variable will store its value in the array rather than override the variable. For questions there are three name arrays that are used and forth one is used for scale type question these names are:

- **question[]**: This name array is used to store the individual questions that are inputted into the question input box.
- **type[]**: This is used to store each of the questions different types, it gets its contents from a hidden variable which is set to the questions type when the question was created
- **max-value[]**: This is used to store the maximum mark that each question can have.
- **min-value[]**: This name array is only used by the scale type question and is used to set what the minimum mark for a question can be.

When the submit button is clicked the JavaScript function "submit_new_lab()" is called which first checks that the lab is valid by calling the "valid_lab()" function which will be discussed later

(section [lfn]). Once the validation checks are done the form is submitted using jQuery.

```
1 function submit_new_lab()
2 {
3     if (valid_lab())
4         $('#form-area').get(0).submit();
5 }
```

Listing 11: Lab Submit - JavaScript

The PHP script called by form is "lab_creator.php" this script is used to call the right function from the "LabCreator" class. The script first creates an instance of the LabCreator class, it then checks that the types, lab name and course name were posted. If they are posted then the "createLab" function is called, otherwise it checks if the form posted that it was updating a lab in which case it runs the "updateLab" function.

```
1 require_once "classes/LabCreator.php";
2 $create = new LabCreator();
3
4 if (isset($_POST["type"]) && isset($_POST['lab-name']) && isset($_POST['
    course-name'])) {
5     $create->createLab();
6 }
7 elseif(isset($_POST["update"])) {
8     $create->updateLab($_POST["update"]);
9 }
```

Listing 12: lab_creator script - PHP

The "LabCreator" class is very large and as only the key functionality will be shown and an explanation on how it all works together will be provided.

Constructor

```
1 function __construct(){
2     $this->courses = new Courses();
3
4     $this->questions = isset($_POST["question"]) ? $_POST["question"] :
        null;
5     $this->max_marks = isset($_POST["max-value"]) ? $_POST["max-value"] :
        null;
6     $this->min_marks = isset($_POST["min-value"]) ? $_POST["min-value"] :
        null;
7     $this->types = isset($_POST["type"]) ? $_POST["type"] : null;
8     $this->visibility = isset($_POST["visibility"]) ? $_POST["visibility"]
        : null;
}
```

```

9
10     $this->course = $this->courses->getCourseId($_POST['course-name']);
11     $this->lab_name = $_POST['lab-name']; //Variable containing lab title
12 }

```

Listing 13: LabCreator Constructor - PHP

When the LabCreator class is initialised the constructor function shown in listing(13). The constructor creates an instance of the "Courses" class which provides functions related to courses, it retrieves the posted information from the lab creation form and assigns them to class variables. Finally it retrieves the course Id of the retrieves course name using the Courses class function "getCourseId".

Create Lab

The function "createLab" is shown in listing(14), the function is fairly large and to make it shorter sections of repeated code have been removed and replaced with (:) to make it easy to tell where code was removed.

The first thing that happens in the function is that a connection is made to the database using the ConnectDB class. It then declares the variables for storing the question number, the current position in the minimum and maximum mark array, after this it retrieves the type ID for each of the different question types storing them each in their own variable. This section was shorted as it was the same command repeated for each of the types.

The function then check that the provided inputs are valid using the validInput function which check that provided inputs were not empty. If the input is not valid then the page is redirected back to the lab creation page unfortunately due to the use of the php script the inputs are erased, this is why a javascript is used to check the inputs before submission. On the other hand if the inputs were valid it sets up a transaction by using the "mysqli_autocommit" function which disables the auto saving of the database meaning if an error occurs during the insertion the lab the database can be roled back and all insertions are undone.

```

1 public function createLab(){
2     $successful = false;
3     $con = new ConnectDB();
4     $this->link = $con->link;

```

```

5
6 $qNum = 1;           //Question Number
7 $minPos = 0;        //Array position of minMarks
8 $maxPos = 0;        //Array position of maxMarks
9
10 $booleanTypeID = $this->getTypeID("boolean"); //ID value of boolean type
11 :
12 $textTypeID = $this->getTypeID("text"); //ID value of text type
13
14 if ($this->validInput()) { //Checks that input is valid before attempting
    to insert
15     mysqli_autocommit($con->link, FALSE); //Sets up transaction for
        database insertion
16
17     $labID = $this->insertLabName($this->course, $this->lab_name); //
        Inserts new lab in to labs table and gets the new ID it creates
18
19     if ($labID !== false) { //Checks that lab was successfully inserted
20
21         foreach ($this->types as $t) { //Loops through each question by its
            type
22             switch ($t) { //Case statement checking what type each
                question is
23                 case "boolean": //Inserts boolean type questions
24                     $successful = $this->insertQuestion($labID,
                        $booleanTypeID, $qNum, $this->questions[$qNum - 1],
                        NULL, $this->max_marks[$maxPos], $this->visibility[
                            $qNum - 1]);
25                     $maxPos++;
26                     break;
27                     :
28                     :
29                 default:
30                     echo "default"; //Default if type doesn't exist
31                     mysqli_rollback($con->link); //Undoes all inserts into
                        the database during the transaction
32                     $successful = false; //Sets successful to false
33             }
34
35             if (!$successful) //Checks if insertion was successful
36                 break;
37             $qNum++; //Increments the question number
38         }
39     }
40 }
41
42 mysqli_commit($con->link);
43 mysqli_close($con->link);
44
45 ($successful) ? $redirect = "../html/pages/labmanager.php" : $redirect
    = "../html/pages/labmaker.php";
46 header("Location: " . $redirect); //Redirects to webpage
47 }

```

Listing 14: createLab Function - PHP

After this the "insertLabName" function which inserts the lab name into the database and returns the unique ID that it was given. This Id is then checked to make sure that insertion was successful, if it not the database is rolled back and user redirected to the lab creation screen. While if it was successful the function begins to loop through the types of question that were provided, and using a switch statement to run the "insertQuestion" function with the correct variables. How this function works is explained later on (section 6.4.5) but what parameters mean starting from the first is: the Id of the lab, the Id of the question type, what its question number is, the actual question, the minimum mark, the maximum mark and the visibility of the question.

Once all the question have been inserted successfully they are committed using the "mysqli_commit" function which means that database is updated, then the user is redirected to the lab management page were they can see there new lab in the management panel.

6.4.4 Insert Lab Name

The way the "insertLabName" function (listing 15) described earlier works is that it provided the course Id and lab name for the new lab. It creates a prepared mysqli statement that inserts the course Id, lab name and the lab is not markable yet. This statement is then executed, if it failed to insets the database is rolled back and false is returned so that the "createLab" function knows an error occurred. Otherwise the Id for the new lab is returned thanks to the "mysqli_insert_id" function which obtains the newly created Id.

```

1 //Inserts the name of the lab and course into the database and returns its
  ID number or false if it failed
2 private function insertLabName($course, $name)
3 {
4     $state = "false";
5     $insertLab = mysqli_stmt_init($this->link);
6
7     //Initialises prepared statement
8     mysqli_stmt_prepare($insertLab, "INSERT INTO labs (courseRef, labName,
  canMark) VALUES (?, ?, ?)"); //Prepares the statement
9     mysqli_stmt_bind_param($insertLab, 'iss', $course, $name, $state);
10    //Binds parameter
11
12    if (!mysqli_stmt_execute($insertLab)) { //Executes statement and
13        check if it failed
14        mysqli_rollback($this->link); //Undoes all inserts
15        into the database
16        echo "Error Inserting Lab Name";

```



```

12         return false;                                //Returns false to show
           insert failed
13     }
14     return mysqli_insert_id($this->link);              //Returns the ID
           number created by inserting
15 }

```

Listing 15: insertLabName Function - PHP

6.4.5 Insert Question

The storing of questions is done using the same prepared statement in the "insertQuestion" function regardless of what the question type is. The function takes in 7 variables: lab Id, question type, question number, question text, minimum mark, maximum mark and its visibility to students. This means that any question type can be submitted as long as it matches these parameters. The parameters are then plugged into to a prepared statement that inserts them into the lab_questions table, when the statement is executed it is check that the insertion was successful. If it failed to insert the database is rolled back and false is returned so that the "createLab" knows an error occurred and stops the insertion of the lab. While if it was successful true is returned to so that the lab insertion can continue.

```

1 //Inserts question into the lab_questions table
2 private function insertQuestion($labID, $type, $number, $question,
   $minValue, $maxValue, $visible){
3     $insertQuestionQuery = 'INSERT INTO lab_questions (labRef, questionType
   , questionNumber, question, minMark, maxMark, private) VALUES (?, ?,
   ?,?, ?, ?, ?)';
4     $insertQuestion = mysqli_stmt_init($this->link);
5     mysqli_stmt_prepare($insertQuestion, $insertQuestionQuery);
6     mysqli_stmt_bind_param($insertQuestion, 'iisiiis', $labID, $type,
   $number, $question, $minValue, $maxValue, $visible);
7
8     if (!mysqli_stmt_execute($insertQuestion)) { //Runs the insertion and
   checks if it failed
9         mysqli_rollback($this->link);    //Undoes all the inserts all ready
   done to the database
10        echo "Error Inserting";
11        return false;    //Returns false to show insert failed
12    }
13    return true;}

```

Listing 16: insertQuestion Function - PHP

6.4.6 Error Checking

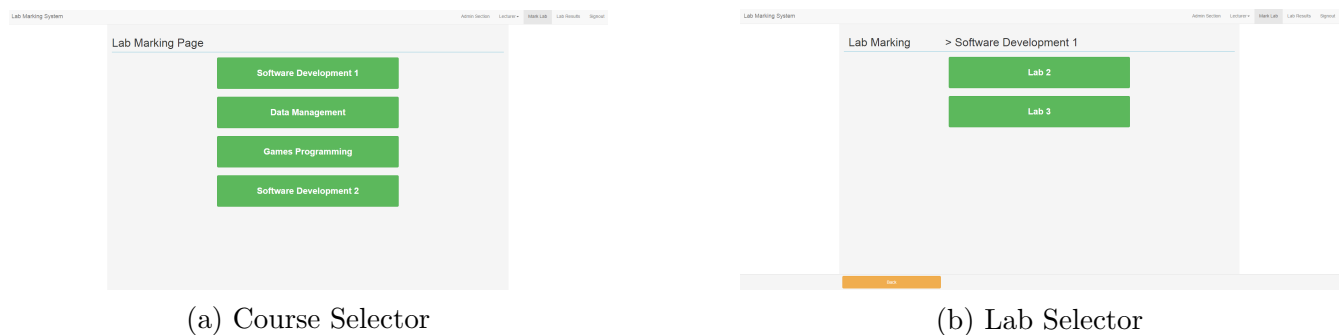
In addition to the error checking done by the PHP script there is additional error checking run before the lab is submitted. Using jQuery all the different inputs are checked to make certain they are not empty and in the case of drop down menus that a value has been selected. If any errors are found the submission of the lab is cancelled, the errors are then highlighted in red using CSS and an alert stating what type of errors occurred is posted.

6.5 Marking Labs

6.5.1 Design

The lab marking page changed majorly from its initial designs. The drop-down boxes for selecting courses, labs and students have been replaced with buttons. When the marker first loads the marking page they are showing buttons for each of courses they can mark (15a), by clicking the button they select the course. The labs for the selected class are then loaded and displayed (figure 15b) dynamical using Ajax so the users does not have to wait for the whole page to load. Additionally a back button is added to the screen to allow markers to easily change the course they are marking. Once the marker has selected the lab they wish to mark by clicking on it the student list for that lab is loaded and displayed again using Ajax.

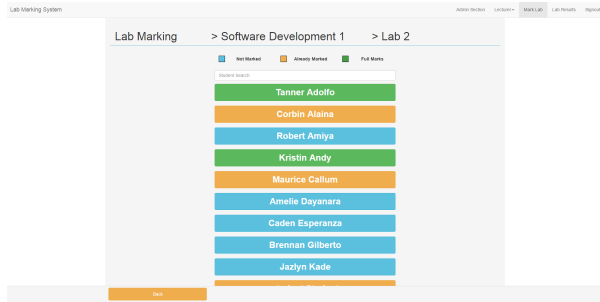
Figure 15: Lab Marking Page



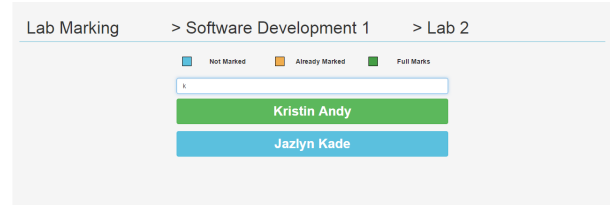
The student selection screen is designed differently from the course and lab selection screens. The student buttons are colour coded: blue means they have not been marked, orange means the student has been marked and green means the student was marked and got obtained full marks.

There is a legend at the top of the page to make it easy to find out what the different colours means. Also included on this page is a search bar (figure 16b), this is to make it easier for markers to find students, especially in labs that have large numbers of students.

Figure 16: Selecting Student Page



(a) Student Select



(b) Student Search

Once the marker has selected a student they would like to mark the marking scheme for the lab is loaded (figure 17a) and if the student has already been marked their current marks are loaded in using jQuery (figure 17b). A submit button is also added to the page which when clicked will save the marks set and take the marker back to the student selection page to make it easy to select the next student to mark.

Each of the different question types is displayed slightly differently. Question one in figure (17a) is a scale question and provides the marker with a drop-down menu to select the mark. Question two is a Boolean type question and is a simple button that can be clicked to switch between yes and no. Then question three is a text question which contains a text-box to allow markers to enter text which will be saved, additionally a drop-down menu can be included to allow for the selection of a mark for the question.

The key functionality behind the lab marking page is firstly dynamic loading which was discussed previously (section [lfn]), after that is being able to retrieve the course, labs and students. Similar to retrieving students being able to search for specific ones is vital to the usability of the system and the last pieces of functionality need are loading students previous marks and the most important function is the ability to process submitted mark.

6.5.2 Retrieving Courses, Labs and Students

The retrieval and displaying of the course, labs and students is done using JavaScript Ajax request. Each of the different sections runs a different Ajax function, when a courses button is clicked the "display_labs_for" function is called and the course Id is passed to a PHP function. This function retrieves all the labs for the course and then places HTML tags around so they can be displayed by the Ajax request, the course name is also stored in a session variable so it can be easily accessed.

The same is done when a lab is clicked but instead of the "display_labs_for" function the "display_students_for" function is called. The PHP function called by its Ajax request uses the course name stored in the session variable to retrieve all the students on that course. As it places the student into the HTML tags to display them it

6.5.3 Searching Students

The searching of students functionality is also done using Ajax that calls the "studentButtonsFilter" function in the "LabStudent" class, additionally it passes the the current text in the search box to act as the filter string.

The "studentButtonsFilter" functions (listing 17) uses the course name that is stored as a session variable along with the filter to call the "getStudents" function which will return an array of students that match the filter. Each of the students is then added into layout of a button so it can be displayed and then returned as a Json object which the Ajax request then displays on screen. Thus showing all the filtered students.

```
1 public function studentButtonsFilter($filter){
2     $con = new ConnectDB();
3     $courseName = $_SESSION["MARKING_COURSE"];
4     $labID =$this->getQuestionID($this->get_lab_id($courseName, $this->lab)
5         , 1);
6     $result = $this->get_students($courseName, $filter);
7     $buttons = "";
8     while ($student = $result->fetch_row()) {
9         $buttonType = $this->buttonStyle($con->link, $student[2], $labID,
10             $this->lab, $courseName);
11         $buttons .= "<div class='col-md-6 col-md-offset-3 col-sm-12
12             clickable-btn'>
13             <button class='" . $buttonType . " btn-text-wrap btn-
14                 student' onclick='display_schema_for(\"" . $student[2]
```

```

        . "\"')>" . $student[0] . " " . $student[1] . "</
        button>
11     </div>";
12 }
13 return json_encode(array('successful' => true, 'buttons' => $buttons));
14 }

```

Listing 17: studentButtonFilter Function- PHP

The "getStudents" function in the Lab class is designed to return all the the students on a course, it can also be passed a filter that limits what students are returned. The listing below (??) show a simplified version of the "getStudents" function only showing the security check and the prepared statement used to retrieve students.

```

1 public function getStudents($course, $filter="")
2 {
3     if ($this->can_mark_course($course)) {
4         $con = new ConnectDB();
5
6         $filter = "%$filter%";
7         mysqli_stmt_prepare($get_students, "SELECT d.firstname, d.surname, d.
            studentID FROM students_on_courses AS soc
8             JOIN user_details AS d ON soc.student = d.
                detailsId
9             JOIN courses AS c ON soc.course = c.courseID
10            WHERE c.courseName = ? AND (CONCAT(d.firstname
                , ' ', d.surname) LIKE ?)
11            ORDER BY d.surname, d.firstname");
12         mysqli_stmt_bind_param($get_students, 'ss', $course, $filter);
13         mysqli_stmt_execute($get_students);
14 }

```

Listing 18: Simplified getStudents Function- PHP

The security check just makes certain that the user trying to mark the course actually has the permission to mark the lab. While the prepared statement gets all the students on the course name provided, but added a like check that means that the resulting first-name or surname should look like the provided filter. This is what is used to filter the student list.

6.5.4 Loading Student Marks

Figure 17: Marking Student Page

Lab Streaming System

Admin Tools Labview Lab Tools Help

Lab Marking

> Software Development 1

> Lab 2

Question Number 1

Quality of code

0

Question Number 2

Compare the LIL

No

Question Number 3

Additional Comments

New Saved

(a) Not Marked Yet

Lab Marking System

Admin Dashboard
Lab Marking
Lab Results
Logout

Lab Marking
> Software Development 1
> Lab 2

Question Number
1

Quality of code

Question Number
2

Completed the lab

Question Number
3

Additional Comments

Back

Submit

(b) Already Marked

6.5.5 Processing Submitted Mark

Question Structure

A huge part of how marks can be saved is down to how each of the question tiles are structured. This structure allows for multiple different question types to be processed in one form rather than requiring a separate form for each of them. The HTML structure for both a scale question and a Boolean question can be seen in figures 18a and 18b respectively.

Figure 18: HTML Structure Of Questions

```
<div class="col-sm-6 col-md-offset-3 col-md-8 col-md-offset-2 title" id="question-1">
  <div class="col-md-5 col-md-offset-1">
    <label for="sell">
      "Question-Number-5" 1
      <input id="question-number-1">1</div>
    </label>
  </div>
  <div class="form-group row">
    :before
    <label for="question-label-input" class="col-md-12 col-md-offset-1 col-form-label">Quality of code</label>
    :after
  </div>
  <div class="form-group col-md-4 col-md-offset-4">
    <input type="hidden" class="question-type" name="type[]" value="scale"> 2
    <label for="sell">Select Mark (select one):</label>
    <select class="form-control mark-input lab-input" name="mark[]" id="scale-input"> 3
      <option value="0">0</option>
      <option value="1">1</option>
      <option value="2">2</option>
      <option value="3">3</option>
      <option value="4">4</option>
    </select>
  </div>
</div>
```

(a) Scale Question

[illegible]

(b) Boolean

It can be seen that both types of questions share a similar structure. What allows saving of the different types is shown in the highlighted areas 2 & 3 in both structures. Normally form information is passed to PHP scripts by naming the input for instance "`<input name="title"/>`", where the name of the input is title. If this

6.5.6 Submitting Marks

The saving of marks is done through a form submission, the submission is done by the javascript in listing([lfn])

The PHP script used to save marks is very large

6.6 Results Display

The implementation of the results page required that it only show students their own marks. While lecturers were able to select a student and see his mark for a lab. The way that I went about doing this was by designing two similar web pages one for students and one with more functionality. Through the use of the security class discussed earlier (section 6.3.2) when the user access the results page, their access level is check and then using PHP the respective layout is shown to them. Details on the ways that the displays are different and how they both function are discussed in the next two sections.

6.6.1 Different Views

The different views are done using a PHP script. when the results page is loaded a PHP file called "result_page_layout.php" (listing 19 is run. It uses the Security class to check if the user is a lecturer, if they are it include the file that provides the lecturer layout along with adding the JavaScript files that provide functionality. If the user is not a lecturer then the file providing the student layout is loaded along with the JavaScript file for student functionality.

```
1 require_once(dirname(__FILE__)."../../core/classes/ConnectDB.php");
2 require_once(dirname(__FILE__)."../../core/classes/Security.php");
3
4 $sec = new Security();
5
6 if($sec->hasAccessLevel("lecturer"))
7 {
8     echo "<script type='text/javascript' src='../..js/lecturer/
9         lecturer_lab_results.js'></script>";
10     include(dirname(__FILE__) . "../../lecturer/lecturer_lab_results.php");
11 }
12 else {
13     echo "<script type='text/javascript' src='../..js/student/
14         student_lab_results.js'></script>";
```

```

13     include(dirname(__FILE__) . "../students/display_lab_marks.php");
14 }

```

Listing 19: Results Layout Selector - PHP

Student View

The design of the students results page remained similar to the initial design, however there was one major design change. Which was that in the initial design the results for each lab was shown straight way in there own table. This was replaced with lab summary area that when clicked expand to show the individual marks for the questions. This was done to help make the area nicer and make it easier to distinguish when one lab ended another one began. The students view of the results page can be seen in figure(19).

Figure 19: Results Page: Student View

Lab Marking System	Lab Results	Signout
Lab Results		
Software Development 1		
▼	Lab Name: Lab 2	Mark: 5 / 6 Percentage: 83.33%
Question	Answer Submitted	Mark
Quality of code	3	3 / 4
Completed the lab	Yes	2 / 2
Additional Comments	Could do with additional commenting	0 / 0
Lab Name: Lab 3		
Mark: Lab Not Marked Yet		
Percentage: Lab Not Marked Yet		
Games Programming		
Lab Name: Lab 1		
Mark: Lab Not Marked Yet		
Percentage: Lab Not Marked Yet		

Lecturers View

The design of the results page for lecturers was changed quite significantly form the initial design (figure 20), it has been made to look similar to the students results page with the removal of the side bar. The new design allows the lecturer to see all the course he teaches along with all the labs each course has. The statistics for each lab are easy to see, providing the lecturer with knowledge of how many students have been marked and the average mark percentage obtained.

When the lecturer clicks on a specific lab the tab expand to show a list of students in the lab and a display area. When the lecturer selects a student there stats are shown in the display area along with a break down of each of the questions. This makes it quick and easy to find students results while also not overwhelming the user with too much information at any one time.

Figure 20: Results Page: Lecturer View

The screenshot displays the 'Lab Results' page for a lecturer in the 'Lab Marking System'. The interface includes a top navigation bar with links for 'Admin Section', 'Lecturer', 'Mark Lab', 'Lab Results', and 'Signout'. The main content area is titled 'Lab Results' and features three lab sections: 'Software Development 1', 'Data Management', and 'Games Programming'. The 'Software Development 1' section is expanded, showing 'Lab Name: Lab 2', 'Students Marked: 5 / 11', 'Marked Average: 76.67%', and 'Overall Average: 34.85%'. A 'Student select list' dropdown is open, showing a list of students with 'Tanner Adolfo' selected. To the right of the dropdown, a table shows the results for 'Tanner Adolfo' with a mark of 5 / 6 and a percentage of 83.33%. The table has columns for 'Question', 'Answer Submitted', and 'Mark'. The 'Data Management' section shows 'No Lab Exists For Course'. The 'Games Programming' section shows 'Lab Name: Lab 1', 'Students Marked: 0 / 9', 'Marked Average: 0.00%', and 'Overall Average: 0.00%'.

6.6.2 Retrieving Marks

The retrieve of lab marks works different for students and lecturers. Students have all there results loaded as the pages loads, while lecturers can select students and then there results are loaded using Ajax. But both types use the same function to retrieve student marks this function is called "studentLabAnswers" and is shown in listing (20).

The function takes in four variables: the first is the name of the course and the second the lab these two together enables the function retrieve the answers to the right lab. The third variable is the username of the student that the results are wanted for and finally the last variable is to whether or not hidden questions should be shown. This is because students should not be able to see them but lecturers should, the inclusion of this variable allows for one function for both rather than needing two separate functions.

```

1 public function studentLabAnswers($course, $lab, $username, $visibility) {
2     $con = new ConnectDB();
3     $studentAnswers = mysqli_stmt_init($con->link);
4
5     if($visibility === "true"){
6         mysqli_stmt_prepare($studentAnswers, "SELECT lq.question, t.typeName
7             , la.answerNumber , la.answerBoolean, la.answerText, la.mark, lq
8             .maxMark
9             FROM lab_answers as la
10            JOIN lab_questions as lq ON la.labQuestionRef = lq.questionID
11            JOIN question_types AS t ON lq.questionType = t.questionTypeID
12            JOIN labs as l ON lq.labRef = l.labID JOIN courses as c ON l.
13            courseRef = c.courseID
14            JOIN students_on_courses as soc ON la.socRef = soc.socID
15            JOIN user_details AS ud ON soc.student = ud.detailsId
16            WHERE c.courseName = ? AND l.labName = ? AND ud.studentID = ?")
17        ;
18        mysqli_stmt_bind_param($studentAnswers, 'sss', $course, $lab,
19            $username);
20    }
21    else {
22        mysqli_stmt_prepare($studentAnswers, "SELECT lq.question, t.
23            typeName, la.answerNumber , la.answerBoolean, la.answerText, la.
24            mark, lq.maxMark
25            FROM lab_answers AS la
26            JOIN lab_questions AS lq ON la.labQuestionRef = lq.questionID
27            JOIN question_types AS t ON lq.questionType = t.questionTypeID
28            JOIN labs AS l ON lq.labRef = l.labID JOIN courses AS c ON l.
29            courseRef = c.courseID
30            JOIN students_on_courses AS soc ON la.socRef = soc.socID
31            JOIN user_details AS ud ON soc.student = ud.detailsId
32            WHERE c.courseName = ? AND l.labName = ? AND ud.studentID = ?
33            AND lq.private = ? ");
34        mysqli_stmt_bind_param($studentAnswers, 'ssss', $course, $lab,
35            $username, $visibility);
36    }
37
38    mysqli_stmt_execute($studentAnswers);
39    $result= mysqli_stmt_get_result($studentAnswers);
40
41    $outputArray = [];
42    while($output = $result->fetch_row())
43        array_push($outputArray, $output);
44
45    mysqli_close($con->link);
46    return $outputArray;
47 }

```

Listing 20: Expanding Row Size - JavaScript

In the function depending on whether the visibility variable is set to true or false can results in two different prepared statements being run to retrieve the lab results. Lets go through prepared statement that is run is visibility is set to true, this prepared statement is designed to return all the

answers a student had for a lab regardless of visibility of the individual questions. This prepared statement is used by lecturers to view all the results for a student. The information that this function returns is: the question text, type, question number, the three possible answer types, the answers mark and finally the questions max mark.

The second function works the same way as the first but with an addition limitation on what answers can be returned. This limitation is that the visibility of the question if false (should not be shown), this prepared statement is used for when students are looking at there results. The statement returns the same information as the previous one just wont show know visible questions.

The information returned by both prepared statements is then used by other other functions to generate the table designs for students and to insert the result in the view section for lecturers..

6.6.3 Expanding Table

The expandable tables are created using a combinations of JavaScript and CSS. Initially when the results pages are loaded all the content that can be seen when the rows are expended are loaded in, but thanks to css anything that expands outside of the row is shown. So what the JavaScript 21 does is change the height of the row using jQuery so that it is tall enough to show all its contents. Also using jQuery made it so that rather than the size of the row suddenly changing it animates the change making it look nicer.

```
1 function change_div_size(divID)
2 {
3     var selected = $(divID);
4     var count = 0;
5
6     $(".results-lab-row").each(function () {
7         if($(this).css("height") != "80px") {
8             $(this).animate({height: '80px'}, 500);
9             $("#result-row-arrow-"+count).toggleClass("glyphicon-triangle-
              right glyphicon-triangle-bottom");
10        }
11        count ++;
12    });
13
14    if(selected.css("height") == "80px") {
15        var curHeight = selected.height();
16        selected.css("height", "auto");
17        var newHeight = selected.height();
18        selected.height(curHeight).animate({height: newHeight}, 500);
```

```

19         selected.find("div[id^='result-row-arrow-']").toggleClass("
20             glyphicon-triangle-right glyphicon-triangle-bottom");
21     }

```

Listing 21: Expanding Row Size - JavaScript

When ever a row is clicked on the "change_div_div" function is run passing the row numbers into the function. The first thing the function does is to go through all the rows and too close them so that only one row is expanded at a time. The script then uses jQuery to find the height the row needs to expands to and uses it as part of the animation function, along with this it also turns the arrow so as to know that the row is expanded.

6.7 Lab Management

The "Lab Management" page (figure 21) remained very similar to its initial designed it did have a few changes done to it. The most noticeable is the addition of the create new lab button at the top of the page which when clicked will take the lecturer to "Lab Creation" page, this was done as to bring all the management of labs into one screen. Along with this each lab also show the maximum available mark, and a key function I forgot to include in the initial design of being able to set if a lab can be marked has been added in the form a radio button.

Figure 21: Lab Management Page

Lab Marking System

Admin Section

Lecturer ▾

Mark Lab

Lab Results

Signout

Lab Management Table

Create New Lab

Software Development 1

Lab 2

Max Mark: 6

☒ Markable

Export Results

Edit

Delete

Lab 3

Max Mark: 7

☐ Markable

Export Results

Edit

Delete

Data Management

No Labs Exist

Games Programming

Lab 1

Max Mark: 7

☐ Markable

Export Results

Edit

Delete

Software Development 2

No Labs Exist

6.7.1 Making Labs Mark-able

The ability to set if a lab can be marked is done through the use of JavaScript and PHP, the tick box contained the ID for the lab it relates to and the state it should change to (true/false). When the box is clicked it runs a function call `lab_markable` (Listing 22) and pass the two values to it.

The function then uses Ajax to run a PHP file and posts the id and state to it. Then depending on the result it does one of two things. If it was successful function on the tick box is updated with the new state so that it can be clicked immediately if a mistake was make. While if it was not successful the tick box is set back to the state it was in previously and an error message is shown to the user.

```
1 function lab_markable(id,state)
2 {
3 $.ajax({
4   type: 'POST',
5   url: ".../php/labs/change_lab_markable.php",
6   dataType: 'json',
7   data: {labID: id, newState: state},
8   cache: false,
9   success: function(result) {
10     if(result.success)
11     {
12       (state === "true") ? state = "false" : state= "true";
13       $("#check-"+id).attr("onclick",'lab_markable('+id+', '"+state+'
14       "));
15     }
16     else
17     {
18       if (state === "true")
19         $('#myCheckbox').attr('checked', false);
20       else
21         $('#myCheckbox').attr('checked', true);
22       alert("Failed to update please refresh and try again");
23     }
24   },
25   error: function(xhr, status, error) {
26     alert("Error Occurred Trying To Update If Lab Can Be Marked" + xhr)
27     ; //Displays an alert if error occurred
28   } }); }
```

Listing 22: Lab Mark-able - JavaScript

6.7.2 Export Lab Results

The export function could not be run using Ajax as it would cause an issue with the exporting of the file, so instead on the Lab Management page there is a hidden form. When the export button is clicked the value in the form is updated to that of the lab and posts its along with that it is the action type "export" to a PHP file called lab_manager. The PHP script checks that the action and the lab ID were posted, creates an Instance of the LabManager() class then through an if statement runs the exportLabResults function.

Lab Manger

The exportLabResults function is a long function but can be easily broken down and explained in small parts.

1. All the question texts are retrieved from the database and stored in an array.
2. The matriculation ID of all students on the course along with their mark for each question along with their total mark this stored as an array inside a separate array.
3. All students who have not been marked have their marks set to zero.
4. Finally the lab name and the two arrays are passes to the export function in the IO class.

IO Class

The IO Class is designed to control all functions that input or output files from the system, at the moment it only contains two functions which both out put .csv files which can be open and read as spreadsheets. The first being exportUsers which exports all the users in the database this is for testing and transfer between systems. However the second function is called export and is whats used to export the data created in the Lab Manager Class.

The export function (listing 23) takes in a file name, an array of column titles and an array of data. All of this is provided by the Lab Manager class, were the array of questions is the column titles and the array containing student results is the data. The export function then creates a new file with the file name, it creates a new file write and adds the titles using the "fputcsv" which

converts the array to a spreadsheet format. It then loops through all the students adding each one and their results to the file, and once this is complete the file writer is closed and the file is exported to the user.

```

1 public function export($file_name, $titles, $data)
2 {
3     ob_clean();
4     header('Pragma: public');
5     header('Expires: 0');
6     header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
7     header('Cache-Control: private', false);
8     header('Content-Type: text/csv');
9     header('Content-Disposition: attachment;filename=' . $file_name . '.csv');
10
11     $output = fopen('php://output', 'w');
12     fputcsv($output, $titles);
13     foreach ($data AS $row) {
14         fputcsv($output, $row);
15     }
16     fclose($output);
17 }
18 }

```

Listing 23: IO Class Export Function - PHP

6.7.3 Edit Labs

The edit labs button provides away for lecturers to change labs they have already created. The editing of labs is limited they can only change questions that already exists, new ones cannot be added and the type of currently existing question cannot be change either. This is to prevent lecturers from changing labs that have already been marked and causing issues to arise with the currently stored data. If a lecturer wishes to do these actions they will have to create a new lab. The way that the editing functionality works is that when the edit button is clicked the "editLab" JavaScript (listing 24) function is called, the button passes the name of the lab and its id to the function. The "editLab" function uses Ajax to the "editable_lab" PHP file which returns the lab in the same style used to in the Lab Creation page providing a away to change them. The script then replaces the content of the

Figure 22: Editing A Lab

The screenshot shows a web interface for 'Lab Management Table'. At the top, there are navigation links: 'Admin Home', 'Lab Management', 'Mark Lab', 'Lab Results', and 'Logout'. Below the navigation bar, there's a 'Create New Lab' button. The main content area is titled 'Software Development 1: Lab 2' and contains a 'Edit' button. Below the 'Edit' button, there are two question forms. The first form is for 'Question Number 1' and the second is for 'Question Number 2'. Each form has a 'Question' field, a 'Type' dropdown menu, and a 'Result' field. The 'Result' field has a 'Mark' button and a 'No students' label.

screen with the received layout(22) and adds

a submit button to confirm the edit and a back

button easily go back to the lab management table.

The displayed question tiles have the same editable ability as the lab creation page. This means that lecturers can change the questions text, marking range and whether or not its results can be seen by students. Questions are updated in the same way as they were inserted into the database but instead of insertion and update query is run that changes the currently stored questions.

```
1 function editLab(lab_name, labID)
2 {
3     $.ajax({
4         type: 'POST',
5         url: "../php/labs/editable_lab.php",
6         dataType: 'json',
7         data: {labID: labID},
8         cache: false,
9         success: function(result) {
10             $("#main-text-area").html("<legend>"+lab_name+"</legend>");
11             $("#main-text-area").append("<button class='col-md-4 col-md-6 col-sm-6 col-sm-offset-3 col-xs-12 btn btn-warning' id='back' onclick='display_table()'>Back</button>");
12             $("#main-text-area").append(result.questions);
13             $("#main-text-area").append("<button class='col-md-4 col-md-6 col-sm-6 col-sm-offset-3 col-xs-12 btn btn-success' onclick='submitEdit()'>Submit</button>");
14             $(".remove-btn").remove();
15             fillLab(labID);
16         },
17         error: function(xhr, status, error) {
18             alert("Error Occurred Trying To Retrieve Lab" + xhr); //
19             // Displays an alert if error occurred
20         }
21     });
22 }
```

Listing 24: Load Editable Lab - JavaScript

6.7.4 Delete Lab

The delete function allows lecturers to remove labs that they have created, and thanks to the structure of the database all the marks related to the lab are also deleted. To prevent a lab accidentally being deleted when the delete button is clicked a popup window appears informing the lecturer of what will happen and asked them to confirm that they wish to delete a lab. Upon clicking the delete button a Javascript function is called that uses Ajax to run the "lab_manager"

PHP file describe previously in the export functionality section ([lfn]). This time script calls the "deleteLab" function from the LabManager this function is shown in listing(25).

```
1 function editLab(lab_name, labID)
2     public function deleteLab()
3     {
4         $deletion = false;
5         if ($this->hasAccessLevel("lecturer")) {
6             if ($this->labID !== null) {
7                 $con = new ConnectDB();
8                 $course = $this->courseFromLabID($this->labID);
9
10                if ($this->isLecturerOfCourse($course)) //Checks if user
11                has access to delete the course
12                {
13                    $delete_lab = mysqli_stmt_init($con->link);
14
15                    //Init
16                    Prepared Statment
17                    mysqli_stmt_prepare($delete_lab, "DELETE FROM labs
18                    WHERE labID = ?");
19                    //Query deletes
20                    labs that match the labID
21                    mysqli_stmt_bind_param($delete_lab, "i", $this->labID);
22                    //Bind labID to
23                    query
24                    mysqli_stmt_execute($delete_lab);
25
26                    //
27                    Execute prepared statement
28
29                    $deletion = true;
30                }
31                mysqli_close($con->link);
32
33                //Closes DB connection
34            }
35        }
36        return json_encode(array("success" => $deletion));
37    }
```

Listing 25: Load Editable Lab - JavaScript

The "deleteLab" function first checks that the user is a lecturer using the Security Class functions, this prevents unauthorised people from preforming the delete function. It then checks that the user is actually a lecturer of the course for which the lab belongs too, therefor stopping lecturers deleting other courses labs. It then deletes the lab that matches the labID that was passed by the JavaScript function. Finally it returns whether the deletion was successful so that the JavaScript knows to display an error or not.

6.8 Admin Panel

The Admin Panel did not have an initial design. This was because I was not certain what functionality it would have and therefor could not think of a design for it, instead I designed it as I went along knowing that I wanted it to be simple to perform any of the admin functionality. As development went on I decided that I should break up the functionality into two sections, the first being to manage users currently in lab marking system. The second section contained the functions that related to the database.

When the admin first loaded the admin page (figure [lfm]) the buttons for the "Manage User" and "Manage Database" section are shown. Then depending on which one the admin clicks the functionality buttons and a back button are loaded (figures [lfm]) using aJax to provide a nice user experience.

7 Testing

This section states how the lab marking system was tested to assure the quality of the system and discover any unexpected bugs.

Testing of the system comprised of two stages, the first was the continues testing being done during development and the second being tests that were run when the system was completed to find any remaining bugs.

7.1 Development Testing

To help test functionality unit tests were created using the PHPUnit testing framework. Unit tests were developed for classes that provide key functionality to the lab marking system.

7.1.1 Security Class Tests

The security class does all the checks on whether a user has the required access being requested.

To test the security class but the `hasAccessLevel` and `hasGreaterAccessThen` functions were tested. Since the `hasAccessLevel` function checks if a users has equal to or greater access than that being requesting, all possible combinations of user access and requested access were tested and the expected outcomes compared to the actual result. The code for this can be seen in listing(26)

```
1 public function testHasAccessLevel($user_access, $required_access, $expected
2 )
3 {
4     if(session_status() == PHP_SESSION_NONE)
5         session_start();
6     $_SESSION["accesslevel"] = $this->security->getAccessValue($user_access
7 );
8     $result = $this->security->hasAccessLevel($required_access);
9     $this->assertEquals($expected, $result, $user_access . " and " .
10         $required_access . ": Failed");
11 }
```

Listing 26: `hasAccessLevel` Function Test

7.2 Final Testing

The final testing of the system came in two parts. The first part was the running of all the unit tests that were created to make sure all the different parts of the marking systems worked correctly, and could interact without causing any issues.

The second part of the final testing was to get users to test the system, and any bug found by users be recorded so that the source of the problem could be found. This testing was done as part of the usability case study

8 Evaluation

This section shows how I evaluated the lab marking system and help determine how successful I was at implementing the system, and what future improvements could be done to make the system more useful.

8.1 Usability Case Study

The way the marking system was evaluated was through the use of a usability case study (UCS), participants for the study were students, lab helpers and lecturers. The case study provided a range of quantities and qualitative questions to help get the maximum amount of feedback about the lab marking system as possible.

The usability case study consisted of four sections: student, lab helpers, lecturers and Mobile. Each section has specific tasks that participants were required to preform, they then evaluate how challenging the task was and provided comments about the design and any additional improvements that could be made.

In the first three sections the participant signed into the system as that particular type of user (student, lab helper and lecture)

(Appendix ??)

8.2 Feedback

Due to the amount of time the case study required to be completed and the time at which I was trying to run it at, only twelve people took part in the in it. The breakdown of participants was: four lecturers, four lab-helpers and four students, thus providing an even sample for the types of users of the lab marking system.

The feedback that was obtain from the case study will be very useful in showing how successful the project was at creating a digital lab marking system, the feedback is broken up into two sections. The first sections is for the quantitative questions and the results that can be obtained from them. While the second question section will deal with all the qualitative questions and what

can be gleaned from them.

8.2.1 Quantitative Questions

This section is dealing with all the question in the usability case study that quantitative. In the case study there were two types of quantitative questions: scale questions and the other being yes and no questions.

Scale questions were used to allow participants to evaluate how difficult a specific task was and ranged from one to five. One being very easy, three being challenging and five meaning the task was impossible to complete. The table below (table 3) contains the mean, median, mode and overall difficulty of the task for each of the scale question in the case study. The difficulty is decided by rounding the mean value up or down.

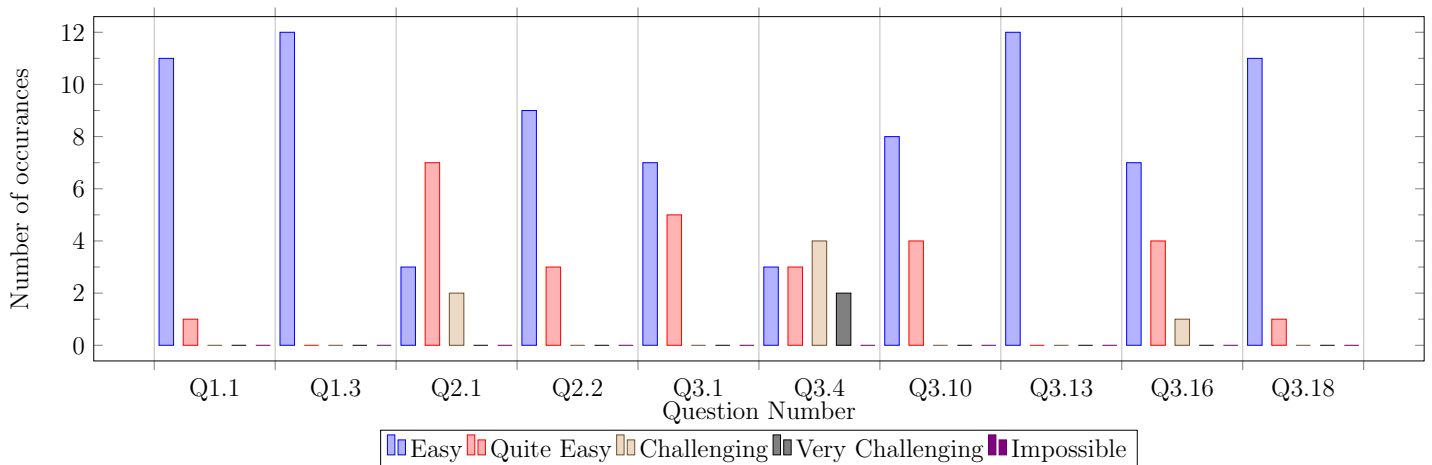
Table 3: Usability Case Study: Scale Question Results

Number	Question	Mean	Median	Mode	Difficulty
Q1.1	Difficulty to find the students mark	1.08	1	1	Easy
Q1.3	Difficulty to sign out	1	1	1	Easy
Q2.1	Difficulty to mark a student	1.92	2	2	Quite Easy
Q2.2	Difficulty to change student mark	1.25	1	1	Easy
Q3.1	Difficulty to view student marks as lecturer	1.42	1	1	Easy
Q3.4	Difficulty to make a new lab	2.41	2.5	3	Quite Easy
Q3.10	Difficulty to export lab	1.33	1	1	Easy
Q3.13	Difficulty to delete lab	1	1	1	Easy
Q3.16	Difficulty to remove and add students	1.5	1	1	Easy

Q3.18	Difficulty to remove and add lab helpers	1.08	1	1	Easy
-------	--	------	---	---	------

The graph below (figure 23) displays how challenging all the participant found each of the questions, The y-axis represents the number of participants that thought at specific task was that difficulty. The x-axis represents each of the questions and are broken up into five bars each representing a difficulty. The left most bar represents easy and as the bars go to the right the difficulty increases until final one which represents impossible.

Figure 23: Bar Graph of Question Results



The first thing that can be noticed is that overall the participants found the lab marking system to be fairly easy to use and understand. The two task that users found the hardest to do were mark students and create labs, which are the most complex part of the lab marking system . It would be expected that first time users of the system would find these are to understand at first. Tho it can be seen with the change student mark task that once the participant understood how to mark one student they could easy update their mark.

It can be seen thanks to graph shown in figure(23) that many users found the creating of new labs to be a difficult task to complete. While watching participants I noticed that many of them were confused on how questions were added to labs.

Question 1.1 tasked participants to sign in as a student and see what mark they were given. As the results of both the table(3) and the graph(23) all participants found this to be easy to do. This is great to see, as the lab marking system is meant to make it easier for students to find out their marks and this task has shown that process for a student finding there marks is very straight forward.

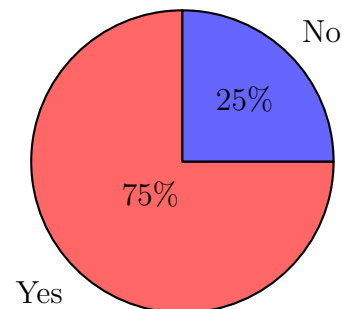
Yes/No Question

Three of the questions in the usability study were yes and no questions. These questions were designed to see if users understood key parts of the lab marking system.

Q2.3: Student Button Colours

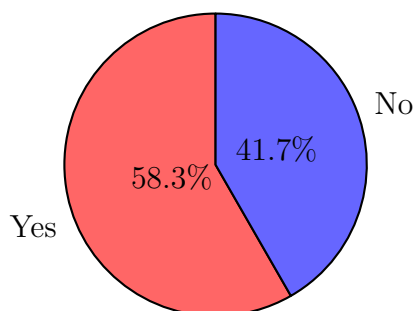
The first question was to check if participants understood what the different coloured buttons for students represented and 75% did understand what they represented. The fact that 25% of the participants did not understand does that the legend that shows the what colours correlate too should be improved.

Figure 24: Q2.3: Know what the different student colours mean?



Q3.6: Visibility Button

Figure 25: Q3.6 Know what visible / not visible means?



The second yes/no question was to do with the creation of new labs and the understand of what the visible / not visible button for individual questions. Only 58.3% of participants understood what this button meant. Since this is a key piece of functionality and allows lecturers to control what questions students can see in

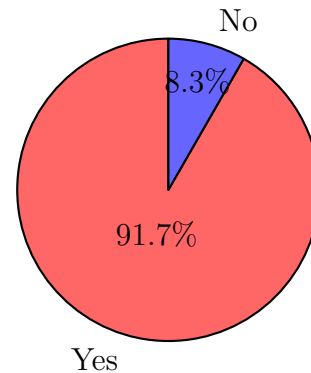
their result, it must be improved. Many participants suggested the inclusion of tool tips to help

understand what the button was meant to do.

8.2.2 Q3.8 Creation Alerts

The final yes/no question dealt with the error handling of the lab creation page. Participants were asked to create a lab that was not complete and attempt to submit it. They were then asked to note if an alert appeared informing of an incomplete lab. It was expected that this question would have a 100% yes response, however one participant managed to find a combination of inputs that was not complete but did submit. The participant in question provided how he got this outcome and I fixed the error detection to prevent it from occurring again.

Figure 26: Q3.8: Did an alert appear



8.2.3 Qualitative Questions

As part of each of the task participants were asked to provide feedback about the task and any improvements that could be done to each of the different parts of the lab marking system. These qualitative questions allowed participants to provide useful feedback and improvements that could be done. The summarised feedback will be broken down into the three different sections: student, lab-helper and lecturer.

Student

Lab-Helper

Lecturer

8.3 Implementation Of Feedback

Using the feedback obtained through the usability case study I have implemented a few changes to the design of the lab marking system. This is to help improve the overall usefulness

of the system and make certain parts of the system easier to understand and interact with. The improvements done to the system are listed below with description and comparison showing the lab marking system before and after the changes.

8.4 Evaluate Requirements

In addition to the usability case study, to evaluate the lab marking system I must check how successful the project was at implementing the requirements of a the digital lab marking system.

To evaluate if the project matches the requirements of for the digital lab marking system, the functional requirements originally show in section 4.2 have been repeated here. Thought this time each requirement has been highlighted according to its implementations. The colour code is: Green means the requirement was implemented in the project, Yellow means that the requirement was not implemented but not guaranteed to be and finally Red is for requirements that were promised but did not get implemented.

Table 4: Functional Requirements

ID	Requirement	Access	Priority
FR-01	Login to view system	1,2,3,4	Must
FR-02	Accounts created for them	1,2,3,4	Must
FR-03	Change password	1,2,3,4	Must
FR-04	Logout	1,2,3,4	Must
FR-05	Login using university ID	1,2,3,4	Could
FR-06	Remove students from courses	1, 2	Must
FR-07	Update student accounts	1,3	Could
FR-08	Look up students in lab	2,3	Must
FR-09	Select students from lab list	2,3	Must
FR-10	Leave comments about students	2,3	Must
SR-11	Save marks	2,3	Must
SR-12	Update marks	2,3	Must

SR-13	Delete marks	2,3	Must
FR-14	Search for student by name	2,3	Should
FR-15	Mark student even if they are not in the system	2,3	Could
FR-16	Assign students to courses	1	Should
FR-17	Assign lectures to courses	1	Should
FR-18	Create marking schemes	2	Must
FR-19	Display generated stats	2	Must
FR-20	See submitted marks	2	Must
FR-21	Generate end of year spread sheets	2	Should
FR-22	Editing of students in class	2	Should
FR-23	Look at students stats	2	Should
FR-24	Update marking scheme	2	Should
FR-25	Delete marking schemes	2	Should
FR-26	Able to assign students to set labs	2	Could
FR-27	Set penalties for late marking	2	Could
FR-28	Able to export to vision	2	Could
FR-29	Set what parts of the marking scheme students can see	2	Could
FR-30	Create peer marking scheme	2	Could
FR-31	Access Marking Scheme	3	Must
FR-32	Enter selected students mark	3	Must
FR-33	Submit student mark	3	Must
FR-34	Select the lab they are helping in	3	Must
FR-35	See current mark	4	Must
FR-36	Show different displays depending on access level		Must
FR-37	Load students current lab mark scheme		Must
FR-38	Apply penalty for late lab completion		Could
FR-39	Create a set of useful stats based on lab		Must

FR-40	Store what class student belong too		Must
FR-41	List of all students in class		Must

The table of requirements makes it easier to tell how many of the requirements the project meets. This projects covers 75% of the requirements having implemented 31 of the 41 possible requirements for a lab marking system. This leaves only ten non-implemented requirements and of those ten only two of them were must requirements, one of them was a should and the remaining seven were coulds.

Miniumum Requirements

The fact that only twenty-two out of the twenty-four (91%) of the must requirements were implemented shows that the developed lab marking system does meet the minimum requirements of the system. The two requirements that the system did not meet are the ability for users to change their password and the ability to delete marks.

As this project was meant to provide a prototype version of digital lab marking system whose focus was on being able to create a mark labs. The inability of users to update their passwords is not a major issue, firstly because when accounts are created random passwords are generated meaning there is some security for accounts and secondly if the marking system was to be integrated with existing university accounts users would already have a way to change their password.

The inability to delete marks is a more major issue, though during development I did not see any reason on why a mark would need to be deleted. Currently the only way to delete a mark complete for the database is to remove a student form a course and then add them back on to it. This has its on issue in that it is time consuming to do and deletes all the lab marks for the student not just a specific one. A mark can also be updated to its default which would be the equivilant of the mark being deleted.

Could and Should

8.5 Aim and Objective

In addition to evaluating the project on how successful it was at meeting the requirements for a lab marking system, it can also be evaluated on how successful it was at reaching its aim and its multiple objectives.

Firstly let's start with the objective of this project which was to "implement a system for the digital marking and analysis of computer labs and to help improve the speed at which they are marked".

8.6 Overall Evaluation

Overall the evaluation of the system shows it to be a good implementation of a digital marking system.

It needs some improvements, especially the lab creation section which is shown to be a little complex

9 Discussion

This section is for discussion of the digital lab marking system as a whole. It includes discussion about the overall development of the system, limitations that currently exist in the lab marking system, improvements that can be implemented later and a conclusion cover all the different parts of the project.

9.1 Development

9.1.1 Issues

Originally I had be planning to develop the lab marking system using cakePHP, as I did more research I quickly found out that it would take more time to setup my computer at home and at the university than the framework would help me save. I therefore decided to not to use it and just no framework when developing the system.

In addition to not using cakePHP I did not manage to use d3 to provide graphical representations of statistics. This was because when I started trying to implement d3 I discovered that it was naturally dynamically responsive, this meant that any change in the screen size would result in elements moving expectantly and cause the website to look bad. There is a way to make d3 graphs responsive but given the amount graphs would add to the lab marking system I decided instead to focus my time on more important functionality.

9.2 Limitations

The aim of this project that would enable lecturers to mark students using a digital marking system. This project was successful in creating this marking system, but it has limitations on what it is able to do. These limitations are all discussed here, long with how to solve them or mitigate the effect they have on the system as a whole.

9.2.1 Only Three Question Types

The current version of the lab marking system only allows lecturers to create lab questions of three different types, these being : scale questions, Boolean questions and text questions. This lack of question variety can be seen as a limitation of the system, though I have designed the system so that it is easy to create new types of questions.

9.2.2 Student Only Lab Helpers

9.3 Future Improvements

The digital lab marking system has many improvements that can be implemented; some of these are requirements that I was not able to implement in the time available. Other improvements for the system came from feedback obtained from the usability case study.

With a bit more time d3 could be converted to be dynamically responsive and make it possible to generate graphs for the statistics being generated.

9.3.1 Peer Marking

Adding to the ability to do peer marking using the lab marking system would make it more useful.

9.3.2 Comparing Students

It was suggested by a few of the participants of the usability case study that lecturers should be able to select multiple students and see their marks.

9.3.3 Track Who Marked What Student

One of the original requirements which was then brought up again during the usability case study, was the ability to track who lab-helpers were marking. This is to make it easy to find out who marked a student if an issue arises, and helps lecturers analyse the quality of a lab-helper's marking. I was unable to implement this functionality into the lab marking system due to time constraints and the need to redesign part of the database to store the markers details. I feel that it would provide a very useful tool for lecturers and should be implemented in the next version.

9.4 Conclusion

In conclusion this project to develop a digital lab marking system has provided a variety of challenges

Overall I would state that this project was successful in developing a digital system for the marking of labs. It has a few flaws at the moment but nothing that could not be sorted out in the next version of the system.

References

- [1] S. Bergmann. “PHPUnit Manual”. In: *????? ?????: http://www.phpunit.de/pocket_guide/3.2/en/index.html*, *????* (2005). bibtex: bergmann2005phpunit bibtex: bergmann_phpunit_2005.
- [2] “Bootstrap The world’s most popular mobile-first and responsive front-end framework.” In: (). bibtex: _bootstrap_???? bibtex: noauthor_bootstrap_nodate bibtex: noauthor_bootstrap_nodate. URL: <http://getbootstrap.com/> (visited on 03/28/2017).
- [3] M. Bostock. *D3.js - Data-Driven Documents*. bibtex: bostock_d3.js_???? bibtex: bostock_d3.js_nodate bibtex: bostock_d3.js_nodate. URL: <https://d3js.org/> (visited on 11/20/2016).
- [4] S. A. Brown and A. Glasner, eds. *Assessment matters in higher education: choosing and using diverse approaches*. bibtex: brown_assessment_1999 bibtex: brown_assessment_1999 bibtex: brown_assessment_1999. Buckingham [England] ; Philadelphia, PA: Society for Research into Higher Education & Open University Press, 1999. ISBN: 978-0-335-20243-0 978-0-335-20242-3. URL: https://books.google.co.uk/books?id=KVblAAAAQBAJ&pg=PA1&lr=&source=gbs_toc_r&cad=4#v=onepage&q&f=false.
- [5] *CakePHP at a Glance*. bibtex: _cakephp_???? bibtex: noauthor_cakephp_nodate bibtex: noauthor_cakephp_nodate. URL: <http://book.cakephp.org/3.0/en/intro.html> (visited on 11/24/2016).
- [6] “Cascading Style Sheets”. In: (). bibtex: noauthor_cascading_nodate bibtex: noauthor_cascading_nodate. URL: <https://www.w3.org/Style/CSS/> (visited on 04/03/2017).
- [7] “CSS Current Status - W3C”. In: (). bibtex: noauthor_css_nodate bibtex: noauthor_css_nodate. URL: https://www.w3.org/standards/techs/css#w3c_all (visited on 04/03/2017).
- [8] “Customizing forms in an electronic mail system utilizing custom field behaviors and user defined operations”. Pat. bibtex: holt_customizing_2006 bibtex[holder=Holt, Nick and Thomas, Steve] bibtex: noauthor_customizing_2006 bibtex: noauthor_customizing_2006. May 2006. URL: <http://www.google.com/patents/US7051273> (visited on 11/19/2016).

- [9] S. Dahl. “Turnitin The student perspective on using plagiarism detection software”. In: *Active Learning in Higher Education* 8.2 (July 2007). bibtex: dahl_turnitin_2007 bibtex[langid=english] bibtex: dahl_turnitin_2007 bibtex: dahl_turnitin_2007, pp. 173–191. ISSN: 1469-7874, 1741-2625. DOI: 10.1177/1469787407074110. URL: <http://alh.sagepub.com/content/8/2/173> (visited on 11/21/2016).
- [10] B. Derby. “Duplication and plagiarism increasing among students”. In: *Nature* 452.7183 (Mar. 2008). bibtex: derby_duplication_2008 bibtex[rights= 2008 Nature Publishing Group;langid=english] bibtex: derby_duplication_2008 bibtex: derby_duplication_2008, pp. 29–29. ISSN: 0028-0836. DOI: 10.1038/452029c. URL: <http://www.nature.com/nature/journal/v452/n7183/full/452029c.html> (visited on 11/24/2016).
- [11] J. Ferraiolo, F. Jun, and D. Jackson. *Scalable vector graphics (SVG) 1.0 specification*. bibtex: ferraiolo_scalable_2000 bibtex: ferraiolo_scalable_2000 bibtex: ferraiolo_scalable_2000. iuniverse, 2000. URL: <https://www.w3.org/TR/2001/REC-SVG-20010904/REC-SVG-20010904.pdf> (visited on 11/21/2016).
- [12] R. Finley. *SurveyMonkey*. bibtex: finley_surveymonkey_1999 bibtex: finley_surveymonkey_1999 bibtex: finley_surveymonkey_1999. 1999. URL: <https://www.surveymonkey.com/home/> (visited on 11/04/2016).
- [13] J. J. Garrett and others. “Ajax: A new approach to web applications”. In: (2005). bibtex: garrett2005ajax.
- [14] E. Heinrich and Y. Wang. “Online marking of essay-type assignments”. In: *In*. bibtex: heinrich_online_2003 bibtex: heinrich_online_2003 bibtex: heinrich_online_2003. 2003, p. 768772. URL: <http://www-ist.massey.ac.nz/MarkTool/Publications/EdMedia2003Onscreen.pdf>.
- [15] R. Higgins, P. Hartley, and A. Skelton. “The Conscientious Consumer: Reconsidering the role of assessment feedback in student learning”. In: *Studies in Higher Education* 27.1 (Feb. 2002). bibtex: higgins_conscientious_2002 bibtex[langid=english] bibtex: higgins_conscientious_2002 bibtex: higgins_conscientious_2002, pp. 53–64. ISSN: 0307-5079, 1470-174X. DOI: 10.1080/

03075070120099368. URL: <http://www.tandfonline.com/doi/abs/10.1080/03075070120099368> (visited on 11/23/2016).
- [16] M. Joy, N. Griffiths, and R. Boyatt. “The Boss Online Submission and Assessment System”. In: *J. Educ. Resour. Comput.* 5.3 (Sept. 2005). bibtex: joy_boss_2005 bibtex: joy_boss_2005 bibtex: joy_boss_2005. ISSN: 1531-4278. DOI: 10.1145/1163405.1163407. URL: <http://doi.acm.org/10.1145/1163405.1163407> (visited on 11/23/2016).
- [17] M. Joy and M. Luck. “Effective electronic marking for on-line assessment”. In: *ACM SIGCSE Bulletin* 30.3 (Sept. 1998). bibtex: joy_effective_1998 bibtex[langid=english] bibtex: joy_effective_1998 bibtex: joy_effective_1998, pp. 134–138. ISSN: 00978418. DOI: 10.1145/290320.283096. URL: <http://portal.acm.org/citation.cfm?doid=290320.283096> (visited on 11/15/2016).
- [18] j. F. jquery.org. *Ajax | jQuery Learning Center*. URL: <https://learn.jquery.com/ajax/> (visited on 04/04/2017).
- [19] j. F. jquery.org. *jQuery*. URL: <https://jquery.com/> (visited on 04/04/2017).
- [20] A Langan and C Wheeler. *Some Insights into Peer Assessment \textbackslashtextbackslash LTiA Issue 4 \textbackslashtextbackslash CeLT \textbackslashtextbackslash MMU*. bibtex: langan_insights_???? bibtex: langan_insights_nodate bibtex: langan_insights_nodate. URL: <http://www.celt.mmu.ac.uk/ltia/issue4/langanwheater.shtml> (visited on 11/21/2016).
- [21] H. W. Lie and B. Bos. “Cascading style sheets”. In: *Faculty of Mathematics and Natural Sciences* (2005). bibtex: lie2005cascading bibtex: lie_cascading_2005 bibtex: lie_cascading_2005.
- [22] “MoSCoW Prioritisation”. In: *Agile Business Consortium* (Nov. 2015). bibtex: noauthor_moscow_2015 bibtex: noauthor_moscow_2015. URL: <https://www.agilebusiness.org/content/moscow-prioritisation> (visited on 04/02/2017).
- [23] P. Orsmond, S. Merry, and K. Reiling. “The Use of Student Derived Marking Criteria in Peer and Self-assessment”. In: *Assessment & Evaluation in Higher Education* 25.1 (Mar. 2000). bibtex: orsmond_use_2000 bibtex: orsmond_use_2000 bibtex: orsmond_use_2000, pp. 23–38. ISSN: 0260-2938. DOI: 10.1080/02602930050025006. URL: <http://dx.doi.org/10.1080/02602930050025006> (visited on 11/16/2016).

- [24] “PHP: History of PHP - Manual”. In: (). bibtex: noauthor_php:_nodate bibtex: noauthor_php:_nodate. URL: <http://php.net/manual/en/history.php.php> (visited on 04/03/2017).
- [25] J. Plekhanova. “Evaluating web development frameworks: Django, Ruby on Rails and CakePHP”. In: *Institute for Business and Information Technology* (2009). bibtex: plekhanova_evaluating_2009 bibtex: plekhanova_evaluating_2009 bibtex: plekhanova_evaluating_2009. URL: <https://ibit.temple.edu/wp-content/uploads/2011/03/IBITWebframeworks.pdf> (visited on 11/23/2016).
- [26] K. Smith. “Simplifying Ajax-Style Web Development”. en. In: *Computer* 39.5 (May 2006), pp. 98–101. ISSN: 0018-9162. DOI: 10.1109/MC.2006.177. URL: <http://ieeexplore.ieee.org/document/1631955/> (visited on 04/04/2017).
- [27] “System and method for restricting user access rights on the internet based on rating information stored in a relational database”. Pat. bibtex: baker_system_1997 bibtex[holder=Baker, Brenda Sue and Grosse, Eric] bibtex: noauthor_system_1997 bibtex: noauthor_system_1997. Oct. 1997. URL: <http://www.google.com/patents/US5678041> (visited on 11/22/2016).
- [28] J. Tang and C. Harrison. “Investigating university tutor perceptions of assessment feedback: three types of tutor beliefs”. In: *Assessment & Evaluation in Higher Education* 36.5 (Aug. 2011). bibtex: tang_investigating_2011 bibtex: tang_investigating_2011 bibtex: tang_investigating_2011, pp. 583–604. ISSN: 0260-2938. DOI: 10.1080/02602931003632340. URL: <http://dx.doi.org/10.1080/02602931003632340> (visited on 11/23/2016).
- [29] Z. Tufekci. “Can You See Me Now? Audience and Disclosure Regulation in Online Social Network Sites”. In: *Bulletin of Science, Technology & Society* 28.1 (Feb. 2008). bibtex: tufekci_can_2008 bibtex[langid=english] bibtex: tufekci_can_2008 bibtex: tufekci_can_2008, pp. 20–36. ISSN: 0270-4676, 1552-4183. DOI: 10.1177/0270467607311484. URL: <http://bst.sagepub.com/content/28/1/20> (visited on 11/24/2016).
- [30] *Turnitin - Home*. bibtex: _turnitin_???? bibtex: noauthor_turnitin_nodate bibtex: noauthor_turnitin_noda URL: http://turnitin.com/en_us/home (visited on 11/21/2016).

- [31] E. Waclawski. “How I Use It: Survey Monkey”. In: *Occupational Medicine* 62.6 (Sept. 2012).
bibtex: waclawski_how_2012 bibtex[langid=english] bibtex: waclawski_how_2012 bibtex: wa-
clawski_how_2012, pp. 477–477. ISSN: 0962-7480, 1471-8405. DOI: 10.1093/occmed/kqs075.
URL: <http://occmed.oxfordjournals.org/content/62/6/477> (visited on 11/22/2016).