Digital Lab Marking System

Heriot-Watt University

Final Year Dissertation

MEng Software Engineering

Lewis Francis McNeill

supervised by Peter J King

April 14, 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 14, 2017

Abstract

The aim of this 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 on-line. 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

List of Figures

List of Tables

Listings

Chapter 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 allotted tasks which lab helpers tick off when completed. The biggest problem here 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 or simply they misplace the actual marking sheet.

After the lab helpers have completed their marking, the sheets are given 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 marks can be misread when being transferred. The lecturer may not enter the marks immediately into the spreadsheet increasing the chance that a marking sheet goes missing, and the final problem is the long wait for students before they 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 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.

1.1 Aims and Objectives

1.2 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.

1.3 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.

1.4 Scope

For this project the lab marking system will be limited to labs designed for the computing science course. Additionally the types of questions that can be used for a lab will be limited

Chapter 2

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, plus current digital marking systems and why they are an improvement on conventional marking. In addition it will 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 include the graphical displaying of statistics.

2.1 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 them.

2.1.1 HTML

HTML is short for Hypertext Markup Language but in fact this is less a programming language and more a set of instructions [**?**]. 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 forward slash at the start (). Elements can exist inside other elements creating a hierarchical structure.

Web browsers do not display the elements and tags directly [**?**] but instead process these elements to decide how the web page appear and then render 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 Google Chrome may render a table where the column width is not defined with each column getting equal space, but Fire-fox 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.

2.1.2 Cascading Style Sheets

Cascading Style sheets (CSS)[**?**] 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" will make the text large and bold. When this class is used all the properties specified by the class are changed. HTML elements can have multiple classes but can only have one id which creates a priority system.

The priority system [**?**] 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 [**?**].

2.1.3 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 are capable of editing HTML content that is displayed inside the user's browser [**?**]. JavaScript is generally embedded into HTML web pages and is normally called client-side JavaScript as the scripts are run on the user's computer not the web page's server.

2.1.4 PHP

PHP stands for Hypertext Preprocessor and is an open source scripting language first developed in 1994 [**?**] 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 [**?**] 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.

2.2 Marking Systems

2.2.1 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 [**?**].

The advantage of this style of marking is that students can obtain useful feedback from their lecturer which can help improve their learning. An article [**?**] found that in a survey 82% of students agreed with the statement "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 [**?**].

2.2.2 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 [**?**].

This allows students to gain experience in evaluating other people's work, which some graduates feel is a necessary skill to possess [**?**]. Peer marking also deals with increasing amounts of students very well-

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." [**?**].

2.3 Digital Marking Systems

2.3.1 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 [**?**]. These systems help to reduce the increasing workload caused by more and more students taking courses. Along with this they allow administrative tasks associated with coursework to be automated, thus enabling more time for other tasks.[**?**].

For students digital marking is very useful 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 [**?**] they found that 78% of students would like get their feedback electronically.

Plagiarism is becoming more common [**?**] and is on the rise amongst students. Digital marking can help to reduce plagiarism as the program is able do what human markers 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.

2.3.2 TurnItIn

The on-line electronic plagiarism system, TurnItIn is [**?**], currently being used by many universities around the world. It allows students to upload their essay assignments on-line. It checks for plagiarism in the document by searching the Internet- using a large database of documents. It then assigns a plagiarism percentage and highlights any affected areas. Lecturers can then log-in and view all the submitted documents and mark them.

One recent study [**?**] 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.

2.3.3 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 [**?**]. 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 on-line [**?**]. 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"[**?**].

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 the program works. Next is the evaluation stage, since evaluation attributes of code can be very subjective, this second step does generates metrics about the submitted program. Some of these metrics are a number of comments and percentage of methods declared [**?**], which will help human markers evaluate the submission quicker.

2.3.4 Ceilidh

The Ceilidh System was developed at Nottingham University, which is described as "a courseware assessment and management system" [**?**]. It was designed to support teaching and 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" [**?**] these forms included: Computer programs, multiple different types of questionnaires and exercises and finally reports.

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 declared, 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.

2.4 User Access Views

2.4.1 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 [**?**]. 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.

2.4.2 Database Controlled Access

A patent "System and method for restricting user access rights on the internet based on rating in-formation stored in a relational database" [**?**], 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 required from each database to confirm whether a user is allowed access.

2.5 Custom Input Forms

2.5.1 SurveyMonkey

Survey Monkey [**?**] is an example of custom web forms being created by users. Users can 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

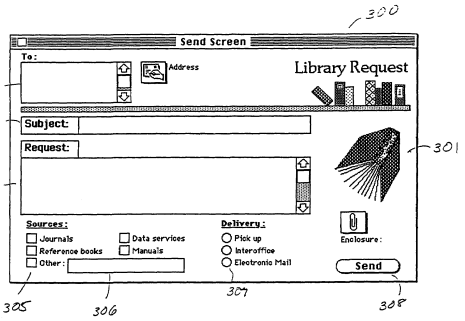
designing the survey the user chooses how they would like to have their survey distributed. The available options that can be selected are a web link, social media, email or embeddable on a website [?].

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.

2.5.2 Customizing Forms In Electronic Mail Systems

Google have patented [?] 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 2.1, it shows the inclusion of additional fields allowing for a wide variety of form to be created without limiting the users to the few forms that already are designed.

Figure 2.1: Example Form [?]



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.

2.6 Development Tools

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

2.6.1 jQuery

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

2.6.2 Ajax

The way that normal websites function is that the user sends a request to the server who then processes the user's requests and creates and sends a response. Once the the user receives this response the web browser interprets it and displays the web page[**?**]. This process model causes long periods of delay for users as they have to wait for the server to process their request plus waiting for the browser to process the response. Users consequently have very disjointed experiences since 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 [**?**]. 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 (2.6.1) jQuery made the process of triggering Ajax's requests much easier.

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

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

2.6.3 Bootstrap

Bootstrap is a front-end framework [**?**] designed to help developers create dynamically 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 it 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.

2.6.4 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 [**?**].

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 2.1.4, I cannot use this most recent version and will instead be 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.

2.6.5 D3

D3 [**?**] 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"[**?**] 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.

2.6.6 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 [**?**]. It allows the development of well structured and robust web applications [**?**], 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 provides 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 testing tools and does not contain a testing environment [**?**].

Chapter 3

Requirements

3.1 Requirements Analysis

To help discover the requirements of the lab marking system I created a requirements analysis questionnaire(Full questionnaire in Appendix **??**), which 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 3.2.2 and non-functional requirements in section 3.2.3.

3.2 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 (3.2.1).

3.2.1 MoSCow Prioritisation

MoSCow Prioritisation [**?**] 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 effect 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.

3.2.2 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 accessible to 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 3.1: 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

3.2.3 Non-Functional Requirements

Table 3.2 lists all the non-functional requirements for the development of the system which are ranked in order of priority.

Table 3.2: 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 are minimum 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

Chapter 4

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.

4.1 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 should function.

4.1.1 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 displayed for students and another one for lecturers.

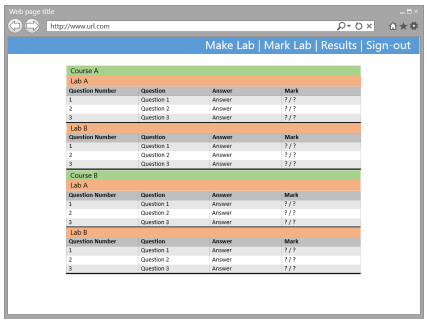
Desktop Design

The desktop views for both students and lecturers are designed in similar ways (figure 4.1), with the result of labs being shown in a table structure at the centre of screen.

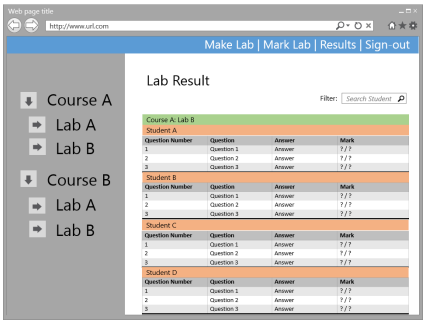
The student view (figure 4.1a) shows all the courses that the specific student is undertaking, along with each of the required labs. The labs are then broken down into their individual marks making it easier for students to understand. The information contained in the breakdown is: what the question 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 4.1b) 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 4.1: Results Page: Desktop Designs



(a) Student Results Screen

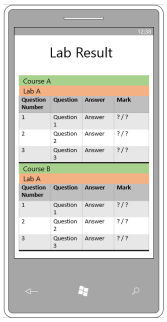


(b) Lecturer Result Screen

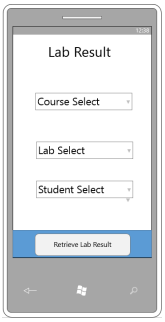
Mobile Design

The mobile design (figures 4.2) 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 4.2b) 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.

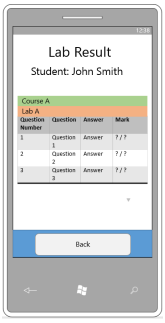
Figure 4.2: Results Page: Mobile Designs



(a) Student: Result Screen



(b) Lecturer: Result Selector



(c) Lecturer: Result Screen

4.1.2 Lab Marking

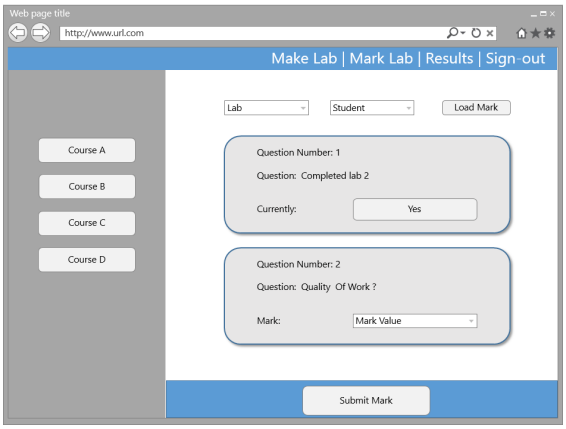
For the lab marking system to be useful lecturers and lab-helpers will have to be able to mark students. The process will also have to be simple as every student in a lab will need to be marked and this project is trying to make the process of marking easier and not harder.

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 4.3) uses the side bar to enable lecturers / lab-helpers to select the course to be marked. 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.

Figure 4.3: Marking Page: Desktop Design



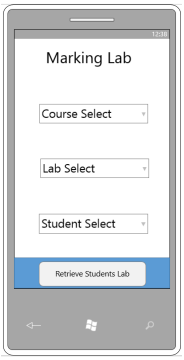
The markers then input their marks using a combination of different input types (depending on question type) and once they are happy with the mark they then click 'submit mark' and the student's mark is saved and is visible to the lecturer and student on the

results page.

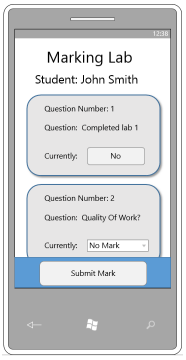
Mobile Design

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(4.4) is designed with this in mind, the marking page is broken into two different pages.

Figure 4.4: Marking Page: Mobile Designs



(a) Selector Screen



(b) Marking Screen

The first page (figure 4.4a) 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 4.4b) is where the lab marking scheme will be loaded. The marking of labs is the same as the desktop design. When the 'submit mark' button is pressed the marker is taken 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.

4.1.3 Lab Creation

The "Lab Creation" page will be very important for lecturers as they will have to use it to create all their labs. Courses contain a large number of labs which necessarily entails frequent use of this part of the system by lecturers, therefore the process for creating a lab should be as simple as possible. These ideas are reflected in the designs shown in figures(4.5a & 4.5b).

Figure 4.5: Lab Creator Page

(a) Desktop Design

(b) Mobile Design

Desktop Design

The desktop design shown in figure(4.5a) 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 which will add a question tile to the main section.

The question tiles are designed with simplicity in mind; they state at the top the question type and number. Lecturers will be able to input the question and its value, depending on the type of question, from the drop down menu. For instance if the question is a scale they can select the minimum mark and maximum mark. 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 want to the lab, they will click the 'create lab' button. If something is missing in the lab an error will appear and highlight where the issue actually occurred or the lecturer will be taken to the lab management screen.

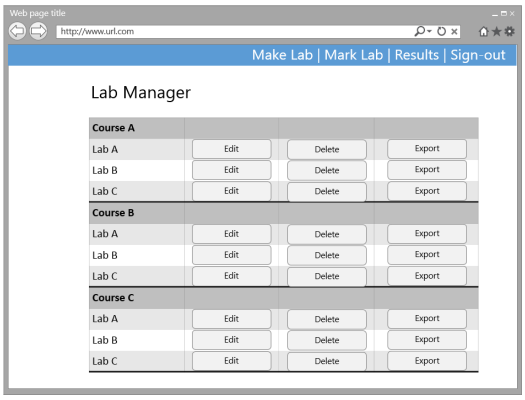
Mobile Design

The mobile design (figure 4.5b) 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 in the same way as the buttons did, but now lecturers select the question type they want to add click the 'add Type' button, and this will add the question tile to the main screen.

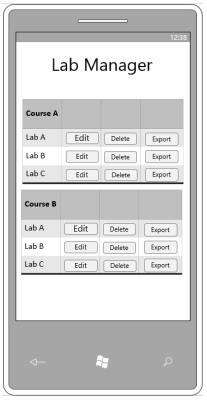
4.1.4 Lab Manager

The "Lab Manager" page is used by lecturers 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 are visible in the figure(4.6).

Figure 4.6: Lab Manager Page



(a) Desktop Design



(b) Mobile Design

Desktop Design

The lab management page is designed for ease of understanding 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 arranged under their respective courses. This makes it easy for lecturers to find the lab they are wanting to manage. Then for each of the labs there are three 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.

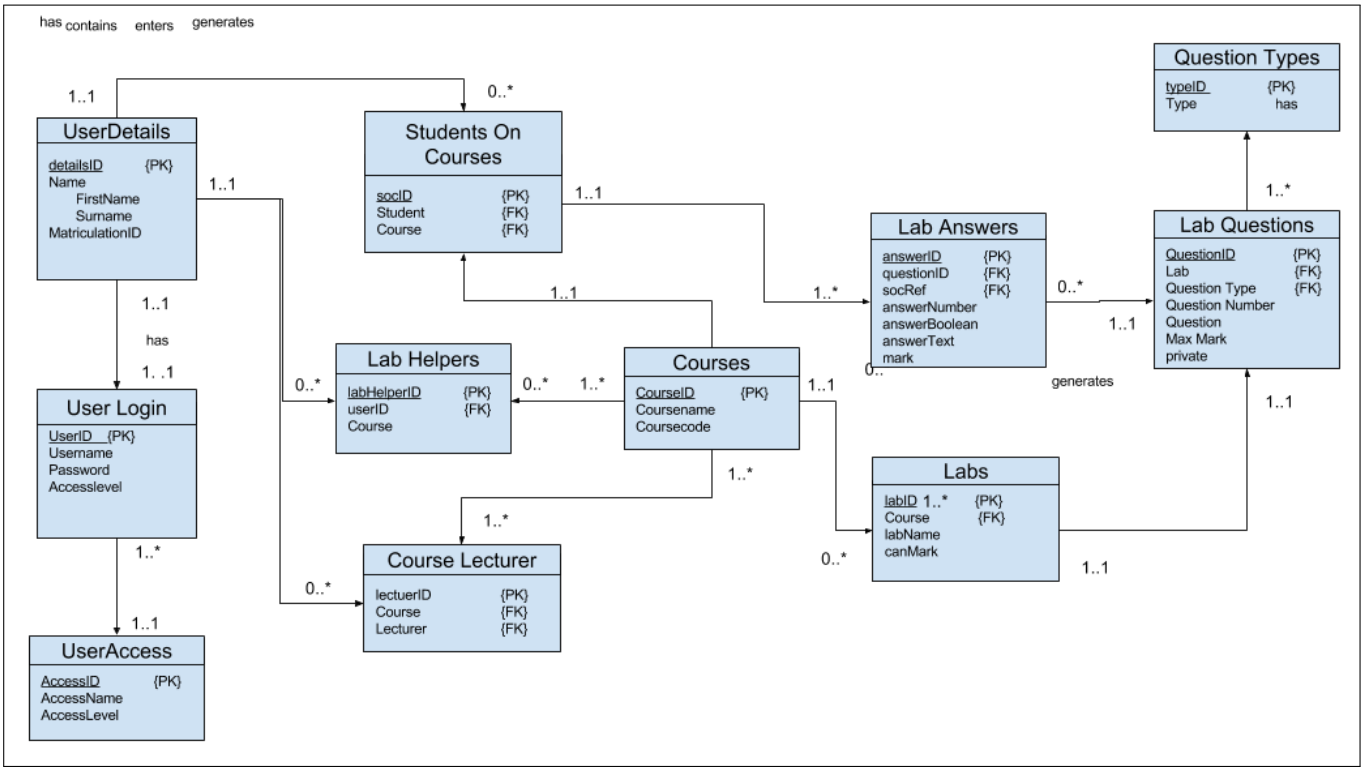
Mobile Design

The mobile design for the lab management page is identical to the desktop design.

4.2 Database Design

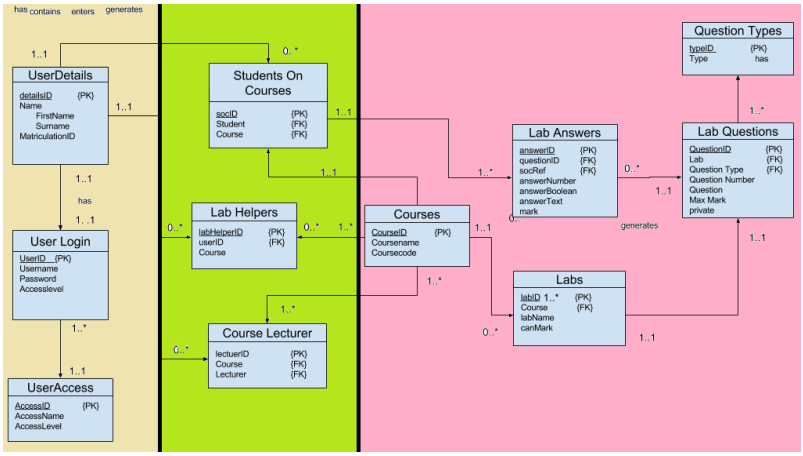
For the system to work there will need to be a database that stores students, labs, and marks received. The database schema I will be using for this system is shown in figure(4.7) below.

Figure 4.7: Database Schema



The database can be broken down into three section as shown in figure(4.8). 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 4.8: Database Sections



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

- **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, which course each record represents, a student and one course they are undertaking. 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 their course.
- **Lab Helpers Table** Stores the lab-helpers user ID, and the course ID, and the course for which they are a lab-helper.
- **Course Lecturer** tables stores lecturers user ID and the course ID on which they are lecturing.

The third section ([lfm]) is the most important part of the database design as it deals 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 their details. Each record has the course ID for the course the lab belongs to, it also contains a name for the lab and whether it can currently be marked by lab-helpers.
- **Lab Question Table:** This table stores all lab questions, each question consists of a Question ID for identification, the ID of its lab it, the ID of its type, its question number, the actual question, its maximum mark 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 an 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 obtain the result of one lab for a student. The structure of the record is firstly it has its own unique ID, then the ID of the specific question, followed by the student's ID. Then because there can be different types of answers submitted there is an attribute for the number of 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 for the answer.

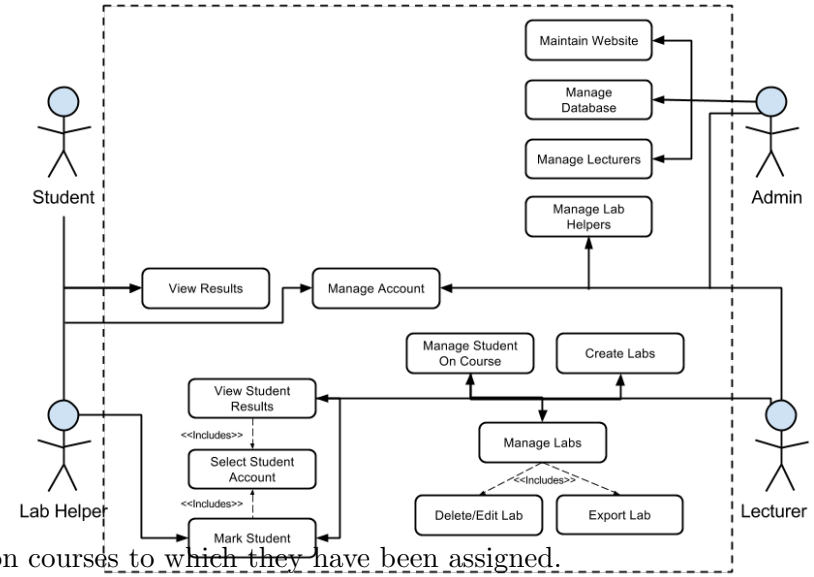
4.3 Functionality Design

To help build the system quickly it is important to understand the key functionality that will be needed to implemented. To that end I created a UML diagram (figure 4.9) that shows the major functions of the lab marking system and additionally shows what users will be able to perform what functionality.

Students

The only key functionality that students require is access to view their own lab results and manage their own account. consequently this is the most basic level.

Figure 4.9: UML Diagram



Lab-Helper

Since Lab-Helpers are also students they have the same functionality to view their results and manage their account, but in addition to this Lab-Helpers also have the ability to select and mark students on courses to which they have been assigned.

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 to 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 functionality that they will be able to access over lecturers are management of the database and assigning of lecturers to courses.

Chapter 5

Implementation

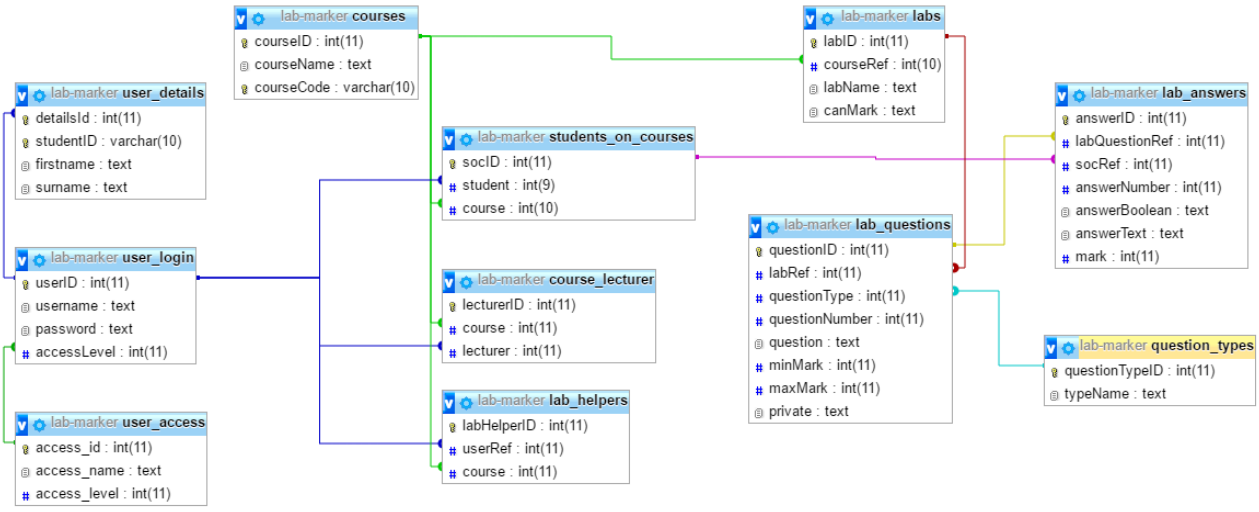
This section documents and explains how the lab marking system was developed and the key functionality works.

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

5.1 Database

The database schema discussed in the design section (4.2) was converted into a relational database using mysql. Figure (5.1) shows the database structure and the lines between tables shows their relations. There are no major changes from the original design, the only difference that occurred during implementation was that some of the attribute names were changed.

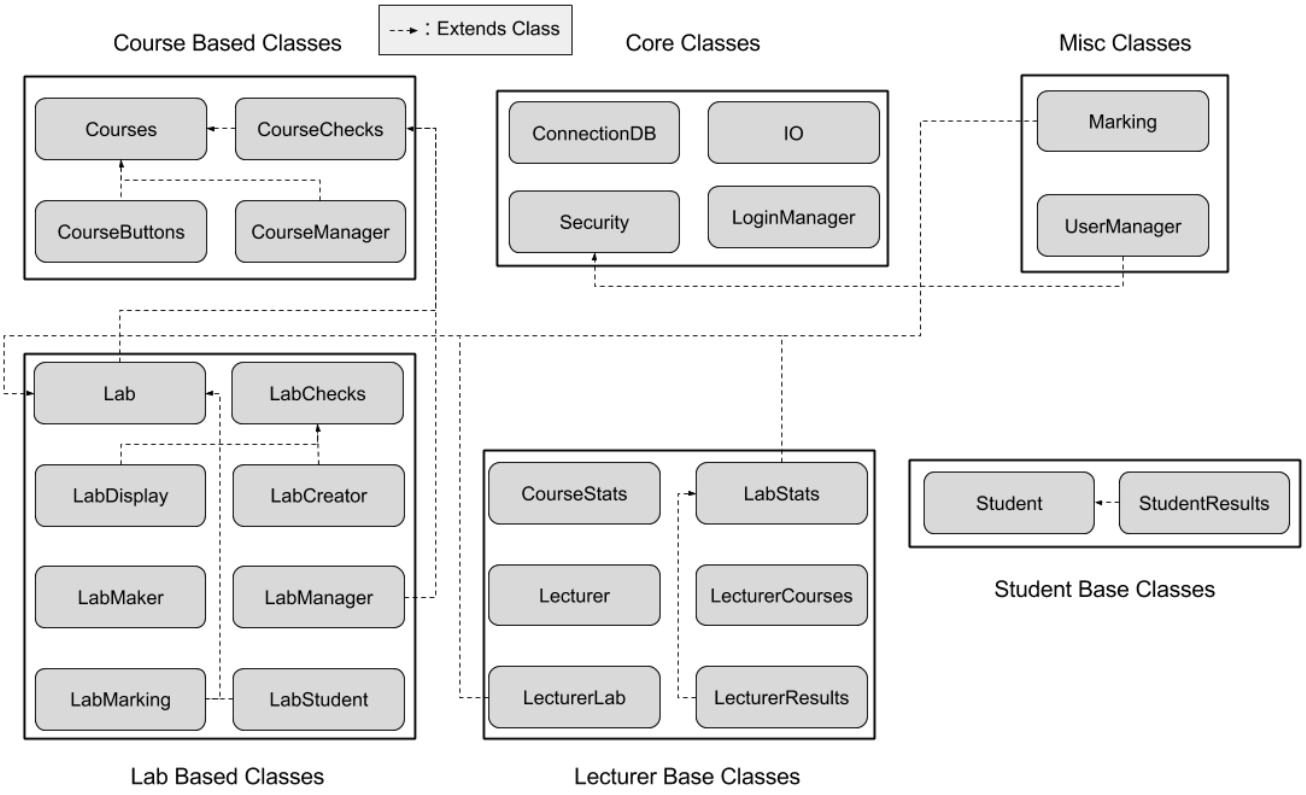
Figure 5.1: Implemented Database Schema



5.2 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(5.2).

Figure 5.2: Class Relationship Diagram



When it came to the creation of classes and their inheritance there are limitations created by PHP. In PHP classes are only able to extend from one other class; the only way to inherit multiple classes is to create an instance of it instead. This is usually done in the constructor so that the class may be accessed through a private variable.

5.3 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.

5.3.1 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 database.

The way that the class works is, it contains one public variable called $link; when the class is initialised the constructor function is 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 checks 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.

5.3.2 Security

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

The "getAccessValue" function shown below converts the access name that is passed in, to what its access level is. It does this by first making a connection to the database and runs a prepared mysqli statement that selects the access_level for the provided access name. Once the result is received 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 it will be returned to inform the check that access name does not exist and not to grant access.

Listing 5.1: 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($get_access_level, "SELECT access_level FROM user_access WHERE access_name= ?");
8           mysqli_stmt_bind_param($get_access_level, 's', $access_name);
9           mysqli_stmt_execute($get_access_level);
10          $result = mysqli_stmt_get_result($get_access_level);
11
12          mysqli_close($con->link);
13          ($result->num_rows === 0) ? $value = -1 : $value = $result->fetch_row()[0];
14          return $value;
15      }
```

The security class has two different check for a user's access right. The first check is to see if the user has the required access. This is done 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 described earlier to

retrieve its access level. The user's access level that is stored in a session is then compared to retrieved access level; if the user's is equal to or greater than the received one then true is returned allowing the user access to whatever the security check protected. However 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 performing the protected action.

Listing 5.2: 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 in exactly the 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.

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 practice are visible in future sections when explaining other functionality.

5.3.3 Navigation Bar

The main navigation bar 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 bars can be seen in figure(5.3), the navigation bars from the top are: admins, lecturers, lab-helpers and finally the students navigation bar.

Figure 5.3: 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 updating the navigation

bar.

Changing Navigation Bar

The navigation bars layout uses a PHP script (figure 5.3) 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 uses the Security class described previously (section 5.3.2) to check the users access level and add the appropriate buttons.

Listing 5.3: Navigation Bar Button Adder - PHP

```php
require_once(dirname(__FILE__) . "/../core/classes/Security.php");
$secure = new Security();

$nav_bar_content = '<li id="results-nav"><a href="labresults.php">Lab Results</a></li>
                    <li id="signoutbtn"><a href="../../php/core/signout.php">Signout</a></li>';

if($secure->hasAccessLevel("lab helper")) {
    $nav_bar_content = '<li id="marking-nav"><a href="marking.php"> Mark Lab </a></li>' . $nav_bar_content;

if ($secure->hasAccessLevel("lecturer")) {

$nav_bar_content = '<li id="labs-nav" class="dropdown">
                    <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false
                        ">Lecturer<span class="caret"></span></a>
                    <ul class="dropdown-menu">
                    <li><a href="labresults.php">Lab Results</a></li>
                    <li role="separator" class="divider"></li>
                    <li id="course-nav"><a href="coursemanager.php"> Course Manager </a></li>
                    <li><a href="labmanager.php">Lab Manager</a></li>
                    <li role="separator" class="divider"></li>
                    <li><a href="labmaker.php">Make Lab</a></li>
                    </ul>
                </li>' . $nav_bar_content;

if ($secure->hasAccessLevel("admin")) {
    $nav_bar_content = '<li id="admin-nav"><a href="admin.php"> Admin Section </a></li>' . $nav_bar_content;
}}}
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

```javascript
function include_navbar (section) {
    $(document).ready(function(){
        $("#navbar_area").load("../components/navbar.php #navbar_code", function () {
            $("#" + section +"-nav").addClass("active");
        });
    });
}
```

Listing 5.4: 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 on. The “include_navbar” function which can be seen in the code listing above (figure 5.4). It works is that it uses jQuery to wait until the web documents have 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.

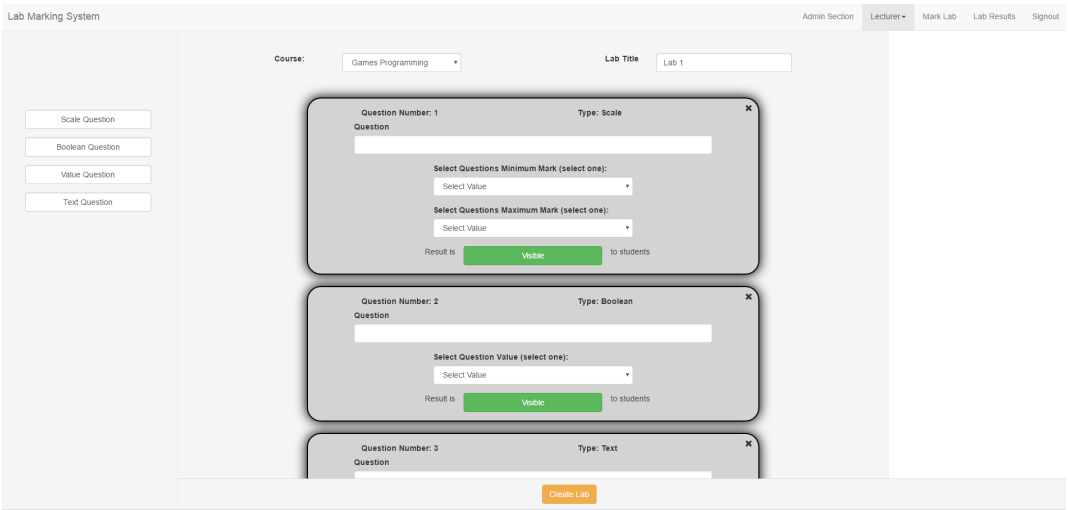
5.4 Lab Creator

5.4.1 Design

The design of the lab creation page can be seen in figure(5.4) 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 5.4: Lab Creation Page



5.4.2 Generating Question Tiles

Question tiles are generated using jQuery, Ajax to call a PHP class that creates the question and returns it. The function used to call the PHP is shown in listing(5.5), 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 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 processed 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(scroll_bottom());
17         },
18         error: function(xhr, status, error) {
19             alert(xhr);
20         }
21     });
22 }
```

Listing 5.5: Add Question - JavaScript

The PHP file called by the Ajax request (list 5.6) 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, therefore 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 }
```

Listing 5.6: Call Create Question - PHP

The LabMaker class is quite large, to enhance understanding of how questions are generated I have broken it down into 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 require different layouts.

```php
public function createQuestion($type,$id,$question_num)
{
    switch ($type)
    {
        case "boolean":
            return $this->booleanQuestion($id, $question_num);
            break;
        case "scale":
            return $this->scaleQuestion($id, $question_num);
            break;
        case "text":
            return $this->textQuestion($id, $question_num);
            break;
        default:
            return $this->unknownQuestion($id,$question_num);
    }
}
```

Listing 5.7: Create Question Function - PHP

To show how a question is created we will use the scaleQuestion function (listing 5.8). The functions called by it are the same as for the other question types, the only difference is that two inputs are provided one for max mark and the other for the minimum.

```php
private function scaleQuestion($id, $question_num)
{
    $scale = $this->startQuestion($id);
    $scale.= $this->title("Scale", $question_num);
    $scale.= $this->questionType("scale");
    $scale.= $this->textInput("question[]");
    $scale.= $this->scaleInput("Select Questions Minimum Mark","min-value[]");
    $scale.= $this->scaleInput("Select Questions Maximum Mark","max-value[]");
    $scale.= $this->visablityButton($id);
    $scale.= "</div>";

    return json_encode(array('question'=>$scale));
}
```

Listing 5.8: Scale Question - PHP

To make it easy to add 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 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 visibility button and the closing div tag, the result is then returned as a json object which Ajax can decode and display in the browser.

5.4.3 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. 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 5.4.6). Once the validation checks are done the form is submitted using jQuery.

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.

```php
require_once "classes/LabCreator.php";
$create = new LabCreator();

if (isset($_POST["type"]) && isset($_POST['lab-name']) && isset($_POST['course-name'])){
    $create->createLab();
}
elseif(isset($_POST["update"])){
    $create->updateLab($_POST["update"]);
}
```

Listing 5.9: 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

```php
function __construct(){
    $this->courses = new Courses();

    $this->questions = isset($_POST["question"]) ? $_POST["question"] : null;
    $this->max_marks = isset($_POST["max-value"]) ? $_POST["max-value"] : null;
    $this->min_marks = isset($_POST["min-value"]) ? $_POST["min-value"] : null;
    $this->types = isset($_POST["type"]) ? $_POST["type"] : null;
    $this->visibility = isset($_POST["visibility"]) ? $_POST["visibility"] : null;
    $this->course = $this->courses->getCourseId($_POST['course-name']);
```

```
10      $this->lab_name = $_POST['lab-name'];  //Variable containing lab title
11  }
```

Listing 5.10: LabCreator Constructor - PHP

When the LabCreator class is initialised the constructor function is shown in listing(5.10). 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 using the Courses class function "getCourseId" which takes in a course name and queries the database returning its id.

createLab Function

The function "createLab" large and a shorter version of it can be found in appendix(A.1), where repeated codes 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 shortened as it was the same command repeated for each of the types.

The function then checks that the provided inputs are valid using the validInput function which checks that the provided inputs were not empty. If the input is not valid then the page is redirected back to the lab creation page but 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 rolled back and all insertions are undone.

After this the "insertLabName" function which inserts the lab name into the database and returns the unique ID that it was given is utilised. This Id is then checked to make sure that insertion was successful, if it not the database is rolled back and the 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 (section 5.4.5) but what the 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 questions have been inserted successfully they are commited using the “mysqli_commit” function which means that database is updated, then the user is redirected to the lab management page where they can see their new lab in the management panel.

5.4.4 Insert Lab Name

The way the “insertLabName” function (listing 5.11), described earlier, works is that it provides 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 insert the database is rolled back and false is returned so that the “createLab” function knows an error has 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);                                                    //Initialises
           prepared statement
6      mysqli_stmt_prepare($insertLab, "INSERT INTO labs (courseRef, labName, canMark) VALUES (?, ?, ?)");     //Prepares the statement
7      mysqli_stmt_bind_param($insertLab, 'iss', $course, $name, $state);                               //Binds parameter
8
9      if (!mysqli_stmt_execute($insertLab)) {    //Executes statement and check if it failed
10         mysqli_rollback($this->link);               //Undoes all inserts into the database
11         echo "Error Inserting Lab Name";
12         return false;                          //Returns false to show insert failed
13     }
14     return mysqli_insert_id($this->link);           //Returns the ID number created by inserting
15 }
```

Listing 5.11: insertLabName Function - PHP

5.4.5 Insert Question

The storing of questions uses 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 at it matches these parameters. The parameters are then plugged into a prepared statement that inserts them into the lab_questions table. When the statement is executed it checks 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, 'iiisiis', $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 5.12: insertQuestion Function - PHP

5.4.6 Error Checking

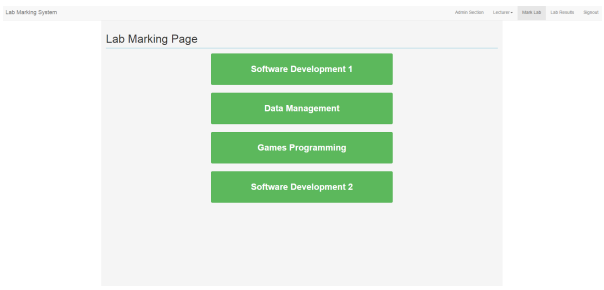
In addition to the error checking done by the PHP script there is additional error checking run before the lab is submitted. Using jQuery all the different inputs are checked to make certain they are not empty and in the case of drop down menus that a value has been selected. If any errors are found 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.

5.5 Marking Labs

5.5.1 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 shown buttons for each of the courses they can mark (5.5a), by clicking the button they select the course. The labs for the selected class are then loaded and displayed (figure 5.5b) dynamically using Ajax so the users do 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 5.5: Lab Marking Page

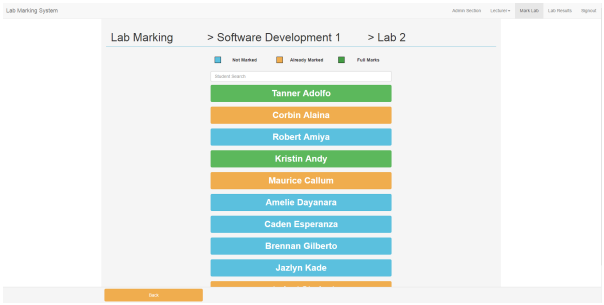


(a) Course Selector

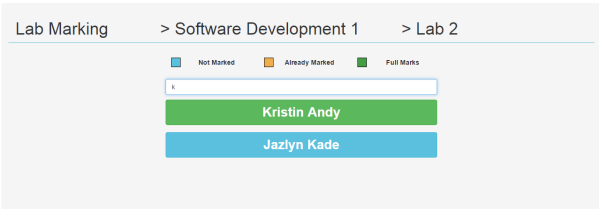
(b) Lab Selector

The student selection screen is designed differently from the course and lab selection screens. The student buttons are colour coded: blue-not marked, orange-marked and green-full marks. A legend at the top of the page offers an easy explanation of the colours. Also included on this page is a search bar(figure 5.6b), this is to make it easier for markers to find students, especially in labs that have large numbers of students.

Figure 5.6: Selecting Student Page



(a) Student Select



(b) Student Search

Once the marker has selected a student to mark, the marking scheme for the lab is loaded (figure 5.7a) and if the student has already been marked their current marks are loaded using jQuery (figure 5.7b). A submit button is also added to the page which when clicked with 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 question type is displayed slightly differently. Question one in figure (5.7a) 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.

[lfm] The key functionality behind the lab marking page is firstly dynamic loading, 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 are loading students' previous marks and the most important function is the ability to proceess submitted marks.

5.5.2 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 thing happens 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 that are used to display them to the user are added.

5.5.3 Searching Students

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

The "studentButtonsFilter" functions (listing 5.13) 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), 1);
5      $result = $this->get_students($courseName, $filter);
6      $buttons = "";
7      while ($student = $result->fetch_row()) {
8          $buttonType = $this->buttonStyle($con->link, $student[2], $labID, $this->lab, $courseName);
9          $buttons .= "<div class='col-md-6 col-md-offset-3 col-sm-12 clickable-btn'>
10                  <button class='" . $buttonType . " btn-text-wrap btn-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 5.13: studentButtonFilter Function- PHP

The "getStudents" function in the Lab class is designed to return all the students on a course, it can also be passed a filter that limits which students are returned. The listing below (5.14) shows 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 5.14: 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 returns a shortened list of students that match the search that the user is looking for.

5.5.4 Loading Student Marks

To enable lab-helpers to view students' current marks once the question tiles have been loaded a JavaScript function is run that uses Ajax to call the "getStudentAnswers" function in the "Marking" class which queries the database using the stored Student name, course name and lab name to retrieve their current results. If no results are found the page is just left with the default values, while if results are returned then jQuery is used to fill out the inputs with the students current value. The figure(5.7a) shows the loaded marking page when the student has not already been marked, while figure(5.7b) shows the marking page with the students current marks already loaded in.

Figure 5.7: Marking Student Page

(a) Not Marked Yet

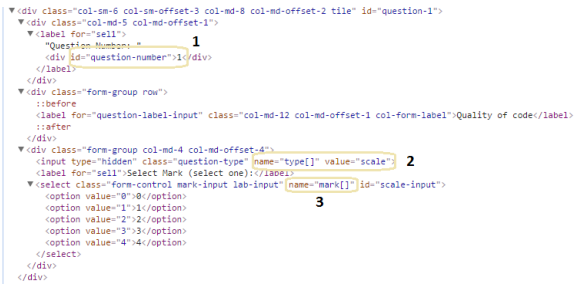
(b) Already Marked

5.5.5 Processing Submitted Mark

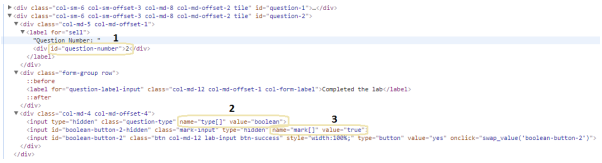
Question Structure

A huge part of how marks can be saved depends on how each of the question tiles is 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 5.8a and 5.8b respectively.

Figure 5.8: HTML Structure Of Questions



(a) Scale Question



(b) Boolean Question

It can be seen that both types of questions share a similar structure (figure 5.8). This structure allows saving of the different types and is shown in the highlighted areas 2 & 3 in both structures. The question tiles are based on the same structure as the question tiles on the lab creation page, so all the inputs are stored in an array which is passed to the "Marking" class where multiple functions are used to insert the inputs into the database.

Marking Class

The "Marking" class contains all the functions relating to the insertion / updating of student marks. When the lab marks are submitted the function "submitMark" is called. The first thing this function does is retrieve the posted variables and stores them in its own local variables, additionally it obtains the user Id for the student and the labId for the lab that is being marked. Then using the array of question types the function loops through all the questions running the "processAnswer" function, which depending on question type provides different variables.

```
1   private function processAnswer($link, $already_present, $questionID, $studentID, $ansNum, $ansBool, $ansText, $mark){
2       if($mark <= $this->get_avalible_marks($link, $questionID)) {
3           if (!$already_present) {
4               $successful = $this->insertAnswer($link, $questionID, $studentID, $ansNum, $ansBool, $ansText, $mark);
5           } else {
6               $successful = $this->updateAnswer($link, $questionID, $studentID, $ansNum, $ansBool, $ansText, $mark);
7           }
8       }
9       else
10          $successful = false;
11      if (!$successful) {     //checks if insert or update failed
12          mysqli_rollback($link); //Undoes all the inserts all ready done to the database
13          return false;   //Returns false to show insert failed
14      }
15      return true; }//Returns true to show insert was successful
```

Listing 5.15: processAnswer Function- PHP

The "processAnswer" function (listing 5.15) takes in a link to the database, whether the student has already been marked so that it can decide to perform an update or inserting to the database. Then it takes in the Id for the question in the lab so that the answer references the correct lab, this is followed

by the student Id which is used to note to which student the answer belongs. Finally it takes in the three different answer types (number, boolean, text) along with the mark for the question.

The first thing the function does is check that the submitted mark is valid by checking it is less than or equal to the max mark set for the question. This stops users attempting to inject higher marks for questions than is allowed. Once it has passed this check it then uses the provided parameters to check if the user has already been been marked.

```
1  private function insertAnswer($link, $questionID, $studentID, $ansNum, $ansBool, $ansText, $mark){
2      $insertAnswerQuery = 'INSERT INTO lab_answers (labQuestionRef, socRef, answerNumber, answerBoolean, answerText, mark) VALUES (?,
           ?, ?, ?, ?, ?)';
3      $insertAnswer = mysqli_stmt_init($link);
4      mysqli_stmt_prepare($insertAnswer, $insertAnswerQuery);
5      mysqli_stmt_bind_param($insertAnswer, 'iiissi', $questionID, $studentID, $ansNum, $ansBool, $ansText, $mark);
6
7      return mysqli_stmt_execute($insertAnswer);
8  }
9  private function getSocID($link, $course, $matric)
10 {
11     $socID = mysqli_stmt_init($link);
12 mysqli_stmt_prepare($socID, "SELECT soc.socID FROM students_on_courses AS soc
13                          JOIN courses AS c ON soc.course = c.courseID
14                          JOIN user_details AS ud ON soc.student = ud.detailsId
15                          WHERE c.courseName = ? AND ud.studentID =?");
16     mysqli_stmt_bind_param($socID, "ss", $course, $matric);
17     mysqli_stmt_execute($socID);
18     return mysqli_stmt_get_result($socID)->fetch_row()[0];
19 }
```

Listing 5.16: Insertion/Update Functions - PHP

If the student has not been marked then the question Id, student Id, answer types and mark are passed to the "insertAnswer" function (listing 5.16) which uses a mysqli prepared statement to insert the answer into the database. If the student has been marked then the "updateAnswer" function (listing 5.16) which takes in the same parameters. The function runs a mysqli prepared statement that updates the student's current answer in the database with the provided data.

Both these functions then return true or false depending on whether the answer was successfully inserted/updated, if false is returned the database is rolled back to undo all insertions and false is returned to stop the submission of the mark.

5.6 Results Display

The design of the results page requires that it only shows students their own marks. While lecturers were able to select a student and see his mark for a lab. Two similar pages were designed- one for students and one with more functionality for lecturers. Using the security class discussed earlier (section 5.3.2) when the user accesses the results page, their access level is checked and the appropriate 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.

5.6.1 Different Views

The different views are achieved using a PHP script. when the results page is loaded a PHP file called "result_page_layout.php" (listing 5.17 is run. It uses the Security class to check if the user is a lecturer, if they are it includes 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.

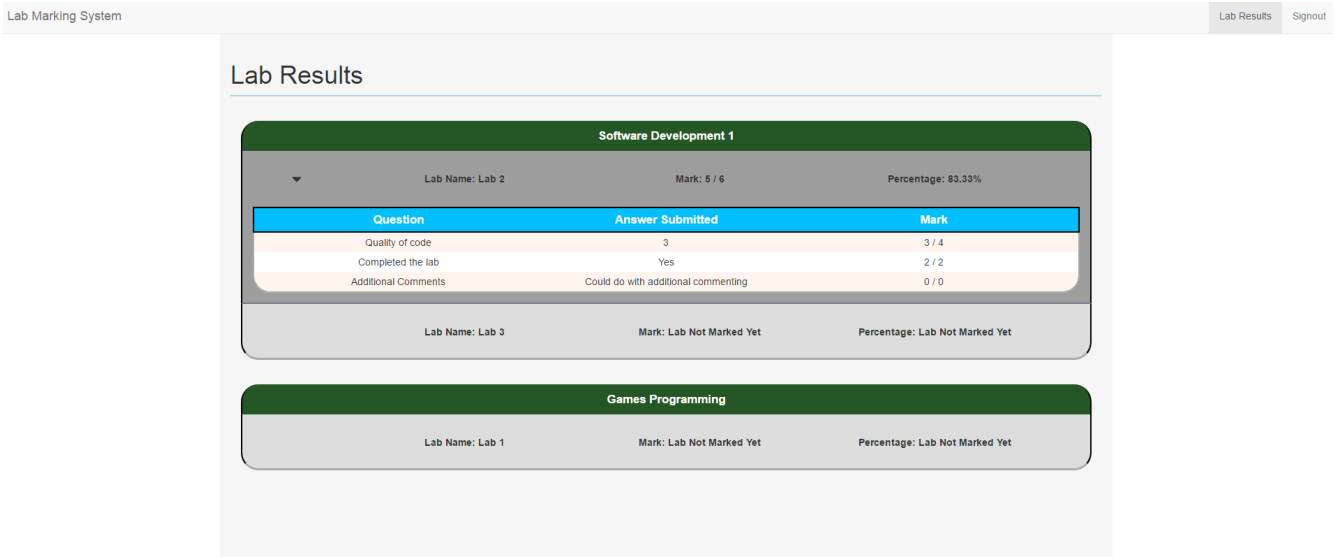
```php
require_once(dirname(__FILE__)."/../core/classes/ConnectDB.php");
require_once(dirname(__FILE__)."/../core/classes/Security.php");

$sec = new Security();

if($sec->hasAccessLevel("lecturer"))
{
    echo "<script type='text/javascript' src='../../js/lecturer/lecturer_lab_results.js'></script>";
    include(dirname(__FILE__) . "/../lecturer/lecturer_lab_results.php");
}
else {
    echo "<script type='text/javascript' src='../../js/student/student_lab_results.js'></script>";
    include(dirname(__FILE__) . "/../students/display_lab_marks.php");
}
```

Listing 5.17: 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. In the initial design the results for each lab were shown straight away in their own table. This was replaced with a lab summary area that when clicked expands to show the individual marks for the questions. This was to help improve the appearance of the area and make it easier to distinguish where one lab ended and another began. The students' view of the results page can be seen in figure(5.9).

Figure 5.9: Results Page: Student View

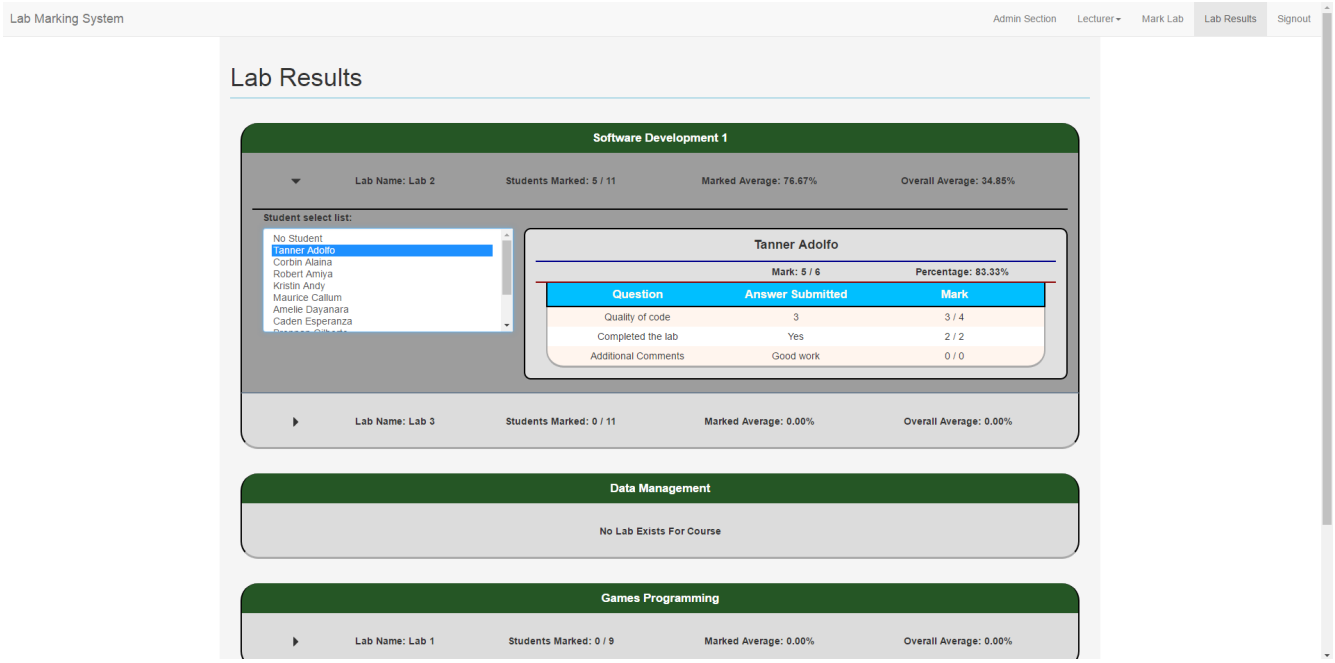


Lecturers View

The design of the results page for lecturers was changed quite significantly from the initial design (figure 5.10), 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 view all the courses 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 expands to show a list of students in the lab and a display area. When the lecturer selects a student their 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 5.10: Results Page: Lecturer View



5.6.2 Retrieving Marks

The retrieve of lab marks works differently for students and lecturers. Students have all their results loaded as the pages loads, while lecturers can select students and then their results are loaded using Ajax. But both types use the same function to retrieve student marks this function is called "studentLabAnswers" and can be found in appendix(A.2).

The function takes in four variables: the first is the name of the course and the second the lab these two together enable the function to retrieve the answers for the right lab. The third variable is the

username of the selected student and finally the last variable is 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.

In the function depending on whether the visibility variable is set to true or false, can result in two different prepared statements being run to retrieve the lab results. If we proceed through the prepared statement that is run if 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 viewing their results. The statement returns the same information as the previous one but will not show known visible questions.[MAT]

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.

5.6.3 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 expanded are loaded in, but thanks to css anything that expands outside of the row is shown. So what the JavaScript 5.18 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("glyphicon-triangle-right glyphicon-triangle-bottom");
20     }
21 }
```

Listing 5.18: Expanding Row Size - JavaScript

Whenever a row is clicked on the "change_div_div" function is run passing the row numbers into the function. The first function goes through all the rows and closes them so that only one row is expanded at a time.[MaT] The script then uses jQuery to find the height the row needs to expand 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.

5.7 Lab Management

The "Lab Management" page (figure 5.11) remained very similar to its initial design with only a few changes. 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 to bring all the management of labs into one screen. Along with this each lab also shows 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 5.11: Lab Management Page

5.7.1 Making Labs Mark-able

The ability to see if a lab can be marked is 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 5.19) 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.

```
function lab_markable(id,state)
{
$.ajax({
    type: 'POST',
    url: "../../php/labs/change_lab_markable.php",
    dataType: 'json',
    data: {labID: id, newState: state},
    cache: false,
    success: function(result) {
        if(result.success)
        {
            (state === "true") ? state = "false" : state= "true";
            $("#check-"+id).attr("onclick",'lab_markable('+id+',"'+state+'")');
        }
        else
        {
            if (state === "true")
                $('#myCheckbox').attr('checked', false);
            else
                $('#myCheckbox').attr('checked', true);
            alert("Failed to update please refresh and try again");
        }
    },
    error: function(xhr, status, error) {
        alert("Error Occurred Trying To Update If Lab Can Be Marked" + xhr);    //Displays an alert if error occurred
    } }); }
```

Listing 5.19: Lab Mark-able - JavaScript

5.7.2 Export Lab Results

The export function could not be run using Ajax as it would cause an issue with the writing of the output 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 the lab Id that is to be exported. It then posts this variable along with the variable storing 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 Manager

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 is 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 output .csv files which can be opened 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 what is used to export the data created in the Lab Manager Class.

The export function (listing 5.20) 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, where 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 5.20: IO Class Export Function - PHP

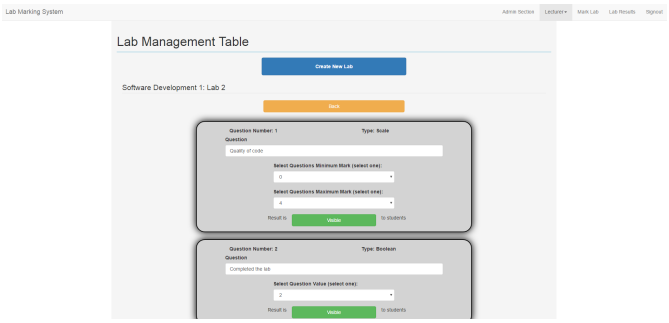
5.7.3 Edit Labs

The edit labs button provides a way for lecturers to change labs they have already created. The editing of labs is limited, they can only change questions that already exist, new ones cannot be added and the type of currently existing question cannot be changed 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 5.21) 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 way to change them. The script then replaces the content of the screen with the received layout(5.12) and adds a submit button to confirm the edit and a back button easily go back to the lab management table.

Figure 5.12: Editing A Lab



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.

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

Listing 5.21: Load Editable Lab - JavaScript

5.7.4 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 accidently being deleted when the delete button is clicked a popup window appears informing the lecturer of what will happen and asks them to confirm that they wish to delete a lab. When clicked the delete button runs a Javascript function that uses Ajax to run the “lab_manager” PHP file described previously in the export functionality (section 5.7.2). This time script calls the “deleteLab” function from the LabManager this function is shown in listing(5.22).

```
function editLab(lab_name, labID)
    public function deleteLab()
    {
        $deletion = false;
        if ($this->hasAccessLevel("lecturer")) {
            if ($this->labID !== null) {
                $con = new ConnectDB();
                $course = $this->courseFromLabID($this->labID);

                if ($this->isLecturerOfCourse($course)) //Checks if user has access to delete the course
                {
                    $delete_lab = mysqli_stmt_init($con->link);
                    mysqli_stmt_prepare($delete_lab, "DELETE FROM labs WHERE labID = ?");  //Query deletes labs that match the labID
                    mysqli_stmt_bind_param($delete_lab, "i", $this->labID);
                    mysqli_stmt_execute($delete_lab);
                    $deletion = true;
                }
                mysqli_close($con->link);
            }
        }
        return json_encode(array("success" => $deletion));
    }
```

Listing 5.22: Delete Lab - PHP

The “deleteLab” function first checks that the user is a lecturer using the Security Class functions, this prevents unauthorised people from performing the delete function. It then checks that the user is actually a lecturer of the course to which the lab belongs, therefore 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 whether to display an error or not.

5.8 Admin Panel

I will not go into detail about the admin panel as it is mostly used for testing of the system. The only notable features it provides are: the ability to assign lecturers to courses, remove/add students to the database and change a users access-level.

The Admin Panel did not have an initial design. This was because I was not certain what functionality it would have and therefore was uncertain of a design. Instead I designed it as I progressed 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.

Chapter 6

Testing

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

6.1 Development Testing

The development of individual piece of functionality was an iterative cycle. This cycle consisted of developing the functionality and then testing that it performed what it was expected to do. If the function did not work correctly I would further develop it and then retest it, repeating the cycle until the function works the way it was intended and was reliable.

This process was vital to the development of the lab marking system.

6.2 Unit 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. Unfortunately due to time constraints of the project there were less unit tests developed than were initially intended. Because of the time constraint the unit tests that were developed focused on the core functionality required to make the system work. Some of the unit tests that were developed are explained below.

6.2.1 Security Class Tests

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

To test the security class the hasAccessLevel and hasGreaterAccessThan functions were tested. Since the hasAccessLevel function checks if a user has equal to or greater access than that being requested,

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(6.1).

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

Listing 6.1: hasAccessLevel Function Test

6.3 User Testing

The final testing of the system was done by allowing users to use the system, and report any bug found by so that the source of the problem could be found. This testing was done as part of the usability case study which will be explained more in the next chapter.

Chapter 7

Evaluation

This section shows how I evaluated the lab marking system and helps determine how successful I was at implementing the system, and what future improvements are required to make the system more useful and workable.

7.1 Usability Case Study

The marking system was evaluated with 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 gain the maximum amount of feedback about the lab marking system.

The usability case study consisted of four sections: student, lab helpers, lecturers and mobile. Each section had specific tasks that participants were required to perform, they then evaluated 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 and identified themselves as a particular type of user (student, lab helper and lecturer) and the final section is testing the lab marking system on a mobile device.

(Appendix **??**)

7.2 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, only twelve people took part in the case study. 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 obtained 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 section 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.

7.2.1 Quantitative Questions

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

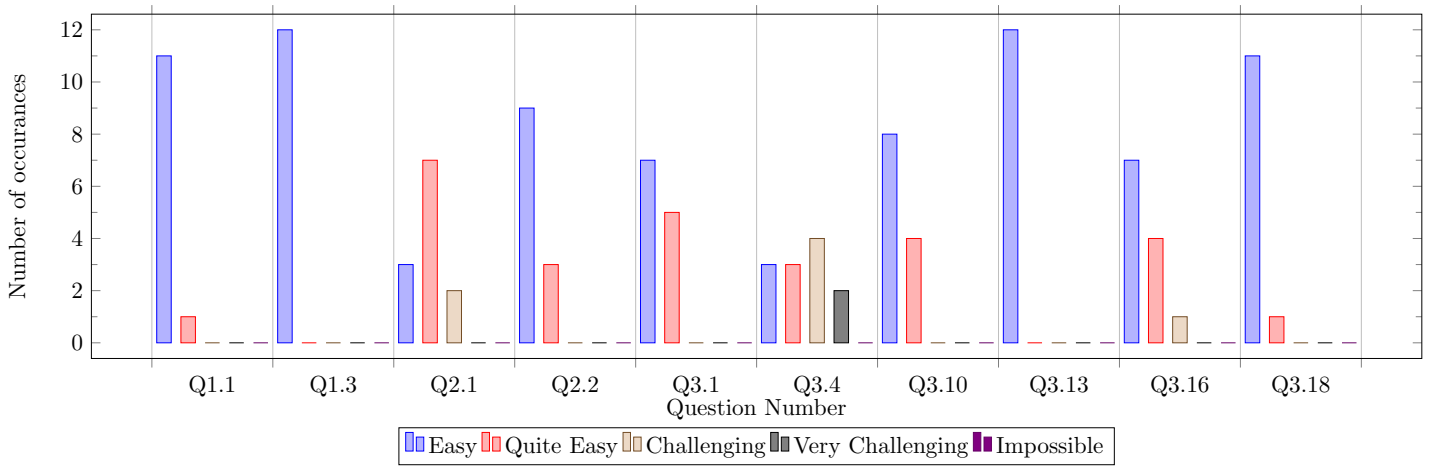
Scale questions were used to allow participants to evaluate how difficult a specific task was and ranged from one to five. One being very easy, three being challenging and five meaning the task was impossible to complete. The table below (table 7.1) contains the mean, median, mode and overall difficulty of the task for each of the scale questions in the case study. The difficulty is decided by rounding the mean value up or down.

Table 7.1: 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 7.1) displays how challenging all the participants found each of the questions. The y-axis represents the number of participants and how they assessed the level of difficulty of a specific task. The x-axis represents each of the questions and is broken into five bars - each representing a difficulty. The bar on the extreme left represents easy and as the bars progress to the right the difficulty increases until the final one which represents impossible.

Figure 7.1: Bar Graph of Question Results



The first thing that can be noticed is that overall the participants found the lab marking system fairly easy to use and understand. The two tasks 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 difficult to understand at first. Though it can be seen with the "update student mark" task that once the participants have marked one student they now found it much easier to mark other students.

It can be seen in figure(7.1), that many users found the creating of new labs to be a difficult task to complete. While watching participants I observed 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(7.1) and the graph(7.1) shows all participants found this to be easy to do. This is useful to know, 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 their marks is very straight forward.

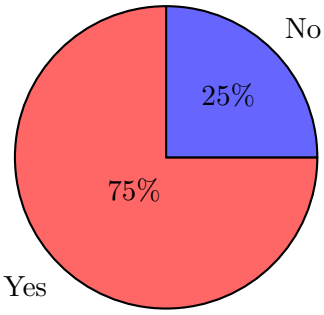
Yes/No Question

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

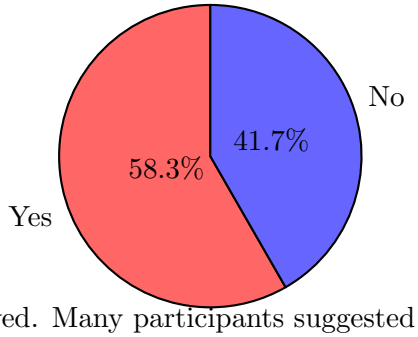
The first question was to check if participants understood what the different coloured buttons for students represented and 75% did understand what they represented. The fact that 25% of the participants did not understand what the legend shows and the colours to which they correlate means that improvement is required.

Figure 7.2: Q2.3: Know what the different student colours mean?



Q3.6: Visibility Button

Figure 7.3: Q3.6 Know what visible / not visible means?

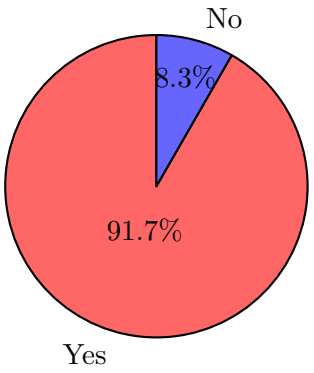


The second yes/no question focused on the creation of new labs and the understanding of the effect of the visible / not visible button on individual questions. Only 58.3% of participants understood what this button meant. Since this is a key piece of functionality and allows lecturers to control what 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.

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 7.4: Q3.8: Did an alert appear



7.2.2 Qualitative Questions

As part of each of the task participants were asked to provide feedback about the task and any improvements that could be made to each of the different parts of the lab marking system. These qualitative questions allowed participants to provided useful feedback and improvements that could be made. The summarised feedback will be broken down into the four main pages: Results, Marking, Lab Creation and Lab Manager.

Results Page

The qualitative questions relating to results page come from both the student section and lecturers as they are both based on a similar design. The overall design of the two results pages was liked by the participants and many stated that they "liked the transition when you drop down the box". Though some participants did not immediately understand that labs were click-able and suggested that on hovering over the lab that "the mouse should change to a hand" to show that it is click-able. What was noticeable from the feed back was that users were finding difficulty reading the text inside the result tables. Many participants suggested "increasing the fonts" or "make the font bold". This is easily done by adjusting the CSS responsible for the font size.

This feedback solely relates to the student version of the results page but there was little unique feedback for the student version since it has a very simplistic design. The only feed back that was given was that there was "No indication of drop-down menu until clicked on" this applied to labs where the student had not been marked yet and was meant to discourage users from clicking those labs.

Finally the feed back about the lecturers version of the results page which overall showed that users found it "Easy to find student and view marks", though many participants had suggestions for improving the results pages. The largest comment was "A search bar to search to shorten the student list" would be useful to have as we have upwards of 160 student and scrolling though 160 names looking for one specific name would be challenging and reduce the systems overall usability. Multiple participants also stated that "clicking on the marks closed the lab", I had not thought that users would be clicking the marks but one participant stated that they would not be able to highlight and copy marks as they cannot click on it. So this should be corrected for the next version.

Lab Marking Page

The lab-helper section was where participants were asked to test out the marking functionality of the lab marking system. The task to mark students had the second highest difficulty of any task and therefore has a large amount of feedback about parts they did not understand and how to make certain sections easier to understand.

The first things participants were saying was "make it more obvious where the marking section is", I noticed that quite a few participants did not notice the "mark lab" button in the navigation bar. Some even suggested that lab-helpers should be "directed to the marking page when they logged in" and not to the results page. Both of these issues could be corrected by the inclusion of a button in the results page that takes the user to the lab marking page. It easier to navigate too and allows users a quick way to access it when they log in.

Something multiple participants raised was "confusion of the student colour coding" as they they did not know the difference between fully marked and just being marked. It was suggested that the use of tool tips on the on the colour code legend would be useful to explain the different colours.

The most common issue stated was that "Initially they did not know they could click the yes/no buttons": this was in reference to the boolean type questions where you could click the button to swap between yes and no. In addition to this participants did not know if they were setting student mark to no as it was quite ambiguous. Some participants suggested some solutions to the issues which were the use of a "tick box or a drop down selection" instead of the button, both of these options would make it easier to tell which option has been selected and know that they can interact with it.

Some participants found the marking of students became easier once they had an idea of how it functioned, stating "Once you learn how to use it the system is very smooth to use". This shows that while the marking of students has an initial learning curve as with many systems, once users have time to practice with it they can use it without many problems.

While the marking of students has a few issues that need to be resolved, participants did not find any issues that severely impacted the usability and functionality of the lab marking system.

Lab Creation Page

The feedback from participants showed that they found the creation of labs to be the most challenging aspect of the lab marking system. The complexity of making labs resulted in a large amount of comments

on how to improve the design to make it more intuitive to use. Most of the issues raised by participants were the same, showing that these issues have a major impact on the marking of labs.

Participants provided many comments about the selection of question types because they felt that the buttons to select question types "are not obvious". It is not clear what the buttons in the side bar are meant to do. Some participants also stated that they did not know what "the different types of questions do" and suggested the use of on hover tool-tips to give an explanation of what each question type is.

When it came to the question tiles participants also had issues and suggested improvements. The most immediate issue participants thought was that "the question tiles were pop-ups" and from personal observation tried to submit labs thinking that they would then be able to select the next question. This issue will need to be solved to make the creation of labs easier to understand and perform. The rewording of the question adding buttons would probably help but the most important would be the redesign of question tiles to look more interlocking and less closely resemble pop-ups.

The fact that participants had issues which severely impacted on the usability of the lab creation page shows that it needs major improvements. Participants did say that "they could create labs once it was explained to them" therefore showing that the lab creation page does fill the requirement it needs to, but the learning curve for using it needs to be reduced.

Lab Manager Page

Participants did not raise any issues with the lab management page feeling that all the functionality "worked very well and was easy to perform", although they did provide some suggestions to help make the lab manager better. A few suggested that the "lab management page should be the home page for lecturers", I can understand this idea as much of the functionality for lecturers is based on this page and would make it quicker for lecturers to perform actions. One participant suggested the inclusion of "a pop-up to state where a lab has been exported to" this would be useful in helping lecturers find labs quickly that they have exported. Final thing that was suggested was that it should be possible to "export the total mark for a course", this would make it easy to compare the whole course's results in a spread sheet form.

7.3 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 increase 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.

7.4 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 digital lab marking system.

To evaluate if the project matches the requirements of the digital lab marking system, the functional requirements originally shown in section 3.2 have been repeated here. Though 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 7.2: 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 markable lab. 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, 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 why a mark would need to be deleted. Currently the only way to delete a mark completely from the database is to remove a student from a course and then add them back on to it. This has its own issues because 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

At the start of the project I stated that there would be as many "could" and "should" requirements implemented as possible. There were 17 should/could functional requirement of these 17 only 9 (52.9%) were implemented. Of these nine requirements only one of them was a "should" and that was to "Generate end of year spread sheets" which is doable thanks to the "IO" class which can output spread sheets. All that is needed is the data to put into it and that can be recovered using a mysqli query.

The other eight requirements that were not included were "could" priority requirements which are the first requirement to be abandoned if time does not allow. So it is not surprising that the largest number of not implemented requirements are "could" requirements, these are requirements that can be implemented in a later version and their absence from the lab marking system does not have a large effect on its overall usability and functionality.

7.5 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 we must 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". This project was successful in designing and implementing a digital lab marking system. However to assess if the system has enhanced the speed of the process of marking a more thorough study would require to be run. Though it can be seen that the time between marking a student's work and them viewing their marks has become almost instantaneous.

Now for the objectives: this project has seven objectives it set out to complete. Firstly "Simplify the way that lab marks are currently processed" the developed system does this by allowing marks to be instantly seen by lecturers and students, removing the time it took for lab-helpers to hand in marking sheets and for the lecturer to transfer the marks to a spread sheet.

The second aim was "Allow lecturers to create marking schemes on-line that lab-helpers can access": this aim is met in the lab creation page where lecturers can create new labs and the lab marking page where the labs can be accessed.

The third and fourth aims are "Lab helpers can mark students using the marking schemes" and "Lab Helpers are able to mark labs using an on-line system". Both of these aims are successfully met by the lab marking page, lab helpers can select students and mark them and even update current marks and since the lab marking system is a web application lab helpers can do this on-line.

The fifth aim was to "Allow students to see the mark they achieved from the lab instantly", this is done automatically by the system meaning the moment that a student has been marked they can go to their results page and see their results for the lab.

The sixth aim was to "Provide useful statistics and graphs for lecturers and students". The lab marking system does provide statistics for both students and lecturers, but does not provide a graphical representation of these stats and is something that should be added in a later version. Additionally lecturers could be given a wider variety of statistics as they only have stats on a lab by lab basis not statistics for the whole course.

The seventh and final aim was to "Provide different views for student, lab helpers and lecturers", this

is done in both the results page and lab marking page. On the results page students are given a different layout compared to lecturers- they can only see their own results while lecturers are able to select students whose results they wish to view. While on the marking page lecturers are able to select from any labs on a course while lab helpers can only see and select labs that the lecturer has declared are mark-able.

7.6 Overall Evaluation

Overall the evaluation of the system shows it to be a good implementation of a digital marking system. The usability case study showed that users found it easy to understand and quickly picked up how to use it, while also helping to point out improvements that could be make to the system. The system also met a majority of the requirements needed in a digital marking system, thus showing it has the functionality required to be a useful system. Finally the lab marking system as a whole meets the aim and objective of this projects, showing that development of it was a success.

It however needs some improvements, especially the lab creation section which was shown to be a little complex to understand and use. Additionally search bars can be added to areas where there is a need to select students, to help make it easy to find students in large classes. Finally the addition of more statistics would greatly improve the usefulness of the system for lecturers and students.

Chapter 8

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 which covers all the different parts of the project.

8.1 Development

8.1.1 Issues

Originally I had be planning to develop the lab marking system using cakePHP,but 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 not to use it and used PHP with no framework or template.

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 unexpectedly and give the website a bad appearance. There is a way to make d3 graphs responsive but given the amount graphs this would add to the lab marking system I decided instead to focus my time on more important functionality.

8.2 Limitations

The aim of this project is to 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, along with how to solve them or mitigate the effect they have on the system as a whole.

8.2.1 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.

8.2.2 Student Only Lab Helpers

During the usability study it was commented that the current lab marking system does not allow for lab markers who are not lecturers or a student lab helper. This causes an issue for research fellows who may be marking a lab as there would be no way to provide them access. The creation of a new account type that is just for marking that can be provided to users who are not students or lecturers, would solve this problem and should be implemented for later versions.

8.2.3 No Encryption

Currently all the data stored by the lab marking system is kept unencrypted. This was to make it easier to develop the system as I could tell exactly what was being stored when functions were being run, therefore making it easier to test and tell if something had gone wrong. For a final release version encryption would have to be included, one-way encryption should be used for the password to prevent passwords being stolen. The lack of encryption does not affect the system in its development stages but not including it in the release version would break the data protection act.

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

8.3.1 Peer Marking

Suggested during the requirements analysis was the ability for the lab marking system to be used to allow students to peer mark each other. The inclusion of this functionality in the future would be very useful for lecturers as it would speed up the time taken to process student peer reviews. It would also make the system as a whole more useful as

8.3.2 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.

8.3.3 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.

8.4 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 cannot not be sorted out in the next version of the system.

Appendix A

Additional Code Listings

A.1 "createLab" Function

```php
public function createLab(){
$successful = false;
$con = new ConnectDB();
$this->link = $con->link;

$qNum = 1;          //Question Number
$minPos = 0;        //Array position of minMarks
$maxPos = 0;        //Array position of maxMarks

$booleanTypeID = $this->getTypeID("boolean");   //ID value of boolean type
                                                        :
$textTypeID = $this->getTypeID("text");         //ID value of text type

if ($this->validInput()) {  //Checks that input is valid before attempting to insert
    mysqli_autocommit($con->link, FALSE);  //Sets up transaction for database insertion

    $labID = $this->insertLabName($this->course, $this->lab_name);   //Inserts new lab in to labs table and gets the new ID it creates

    if ($labID !== false) { //Checks that lab was successfully inserted

        foreach ($this->types as $t) { //Loops through each question by its type
            switch ($t) {  //Case statement checking what type each question is
                case "boolean": //Inserts boolean type questions
                    $successful = $this->insertQuestion($labID, $booleanTypeID, $qNum, $this->questions[$qNum - 1], NULL, $this->
                        max_marks[$maxPos], $this->visibility[$qNum - 1]);
                    $maxPos++;
                    break;
                                                        :
                                                        :
                default:
                    echo "default";  //Default if type doesn't exist
                    mysqli_rollback($con->link); //Undoes all inserts into the database during the transaction
                    $successful = false; //Sets successful to false
            }

            if (!$successful)   //Checks if insertion was successful
                break;
            $qNum++;  //Increments the question number
        }
    }
}

mysqli_commit($con->link);
mysqli_close($con->link);

($successful) ?  $redirect = "../../html/pages/labmanager.php" : $redirect = "../../html/pages/labmaker.php";
header("Location: " . $redirect);   //Redirects to webpage
}
```

Listing A.1: createLab Function - PHP

A.2 Expanding Row Size

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

Listing A.2: Expanding Row Size - JavaScript