SWANSEA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

PROJECT IMPLEMENTATION AND DISSERTATION

DEMONSTRATORS ALLOCATION WEB APPLICATION

Submitted By :

Michalis Mylonas

914079

Supervisor: Dr. Mike Edwards
Computer Science Department

Project Dissertation submitted to Swansea University
in Fulfilment for the Degree of Bachelor of Computer Science

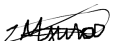
Abstract

The WWW has been and will continue to be a rapidly developed infrastructure. Web applications are a fundamental part of it as well. Despite the humongous success and use of the WWW, our university lacks a web application for assisting people who are interested in demonstrating a module. That creates a slow and difficult operation. Taking into consideration the time-consuming process of managing those applications, this project was proposed. The project itself has developed a website to help resolve this matter in a creative and imaginative way. The program established is beneficial for reducing the time that normally takes to process the applications and assign demonstrators where necessary. Some features this website facilitates include, potential modules for demonstration as a way for future demonstrators to choose modules. Year, module codes, total hours selected and the number of maximum students allowed to be allocated are included as well. All the above are constrains to help automate the procedure and simplify a vigorous task.

Declaration

This work has not previously been accepted in substance for any degree and is not being currently submitted for any degree.

April 2, 2020

Sign: 

Statement One

This dissertation is being submitted in partial fulfilment of the requirements for the degree of a Bachelor in Computer Science (Honours).

April 2, 2020

Sign: 

Statement Two

This dissertation is the result of my own independent work / investigation, except where other-wise stated. Other sources are specifically acknowledged by clear cross referencing to author, work, and pages using the bibliography / references. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure of this dissertation and the degree examination as a whole.

April 2, 2020

Sign: 

Statement Three

I hereby give consent for my dissertation to be available for photocopying and for inter library loan, and for the title and summary to be made available to outside organisations.

April 2, 2020

Sign: 

Acknowledgements

The original idea for this web application came from Dr. Mike Edwards and this work was achieved under his supervision. The author would like to acknowledge this wonderful collaboration. and to personally thank him for his advice, valuable guidance and insightful comments.

The author also wishes to thank his second marker for his detailed informative feedback and suggestions for this dissertation, Prof. Markus Roggenbach.

The author has received a lot of guidance and assistance throughout this dissertation's entire writing and would really like to thank colleagues, relatives and friends for their support and understanding.

Contents

List of Listings **7**

1 Introduction **11**
1.1 Problem introduction . . . 12
1.1.1 Explaining the background of demonstrator allocation in universities . . . 12
1.1.2 Current Procedure . . . 13
1.2 Project definition . . . 14
1.3 Tools Utilised . . . 14

2 Background research **15**
2.1 Current tool . . . 16
2.2 Current procedure . . . 16
2.3 Alternatives . . . 17
2.4 Our own tool . . . 19

3 Specification **20**
3.1 Problem Specification . . . 21
3.2 UML Diagrams . . . 22
3.2.1 Activity Diagram . . . 22
3.2.2 Interaction Diagrams . . . 24
3.3 Proposed Functionality . . . 25
3.4 Proposed solution . . . 26
3.5 Proposed timeline . . . 26
3.6 Risk Assessment . . . 28

4 Design **30**
4.1 Class structure / System overview diagrams . . . 31
4.2 Prototype design workflows . . . 32
4.3 Screenshots of different workflows . . . 33
4.3.1 Admin Home Page . . . 33
4.3.2 Creating labs . . . 35
4.3.3 Adding/Removing Demonstrators . . . 36

4.3.4 Lab Details . . . 37
4.3.5 User CRUD Actions . . . 38
4.3.6 Roles CRUD Actions . . . 40
4.3.7 Student/Normal User Login . . . 42

5 Implementation **43**
5.1 Tools used to develop solution . . . 44
5.1.1 Version control . . . 44
5.1.2 MVC . . . 44
5.1.3 Visual Studio . . . 45
5.2 Security Infrastructure . . . 45
5.3 Backend development of database . . . 47
5.4 Email Server . . . 48
5.5 Implementing the CRUD actions . . . 49
5.5.1 Create Service . . . 49
5.5.2 Update Service . . . 50
5.5.3 Delete Service . . . 50

6 Software Testing **51**
6.1 Design and implementation of the tests and their results . . . 52
6.2 Functionality Testing . . . 52
6.2.1 Links . . . 52
6.2.2 Forms . . . 52
6.2.3 Cookies . . . 52
6.2.4 HTML, CSS . . . 53
6.3 Unit tests . . . 53
6.4 Input Validation . . . 54
6.5 Compatibility Testing . . . 55
6.6 Crowd-Source Testing . . . 55

7 Reflection on work done **56**
7.1 Risk Evaluation . . . 57
7.2 Timeline Evaluation . . . 57
7.3 Functionality . . . 59
7.4 Areas for expansion and improvement . . . 59
7.5 Conclusion . . . 60

8 Appendix **64**
8.1 Appendix A (Introduction) . . . 65
8.1.1 Procedure in excel . . . 65
8.2 Appendix B Background Research . . . 67
8.2.1 Proposed Alternative Solutions . . . 67
8.3 Appendix C (Specification) . . . 68

8.3.1 Proposed Risk Analysis 68
8.4 Appendix D (Design) 69
8.4.1 Prototype Mock-ups 69
8.5 Appendix E (Implementation) 75
8.5.1 Module Controller Code 75
8.6 Appendix F (Testing) 77
8.6.1 Boundary Value Testing 77

List of Figures

1.1 Spreadsheets used . . . 13
1.2 Example timetable . . . 13

2.1 VisiCalc spread-sheet . . . 16
2.2 Spreadsheets Used . . . 17

3.1 Project aims . . . 21
3.2 Application work-flow . . . 22
3.3 Student action sequence . . . 23
3.4 Administrator action sequence . . . 23
3.5 Student graphical overview . . . 24
3.6 Admin graphical overview . . . 24
3.7 Proposed Timeline . . . 27

4.1 Class structure diagram . . . 31
4.2 Timetable and module selection . . . 32
4.3 Final lab selection . . . 32
4.4 Administrator login design . . . 33
4.5 How to create a lab . . . 35
4.6 Process of selecting applicants . . . 36
4.7 Lab details . . . 37
4.8 Links to CRUD services . . . 38
4.9 Editing a user . . . 39
4.10 Roles CRUD ACTIONS . . . 40
4.11 Services for each role . . . 41
4.12 Student log in design . . . 42

5.1 Structure diagram . . . 47
5.2 Email server confirmation . . . 49

6.1 Unit testing code . . . 53
6.2 Unit testing results . . . 53

6.3 Invalid code . . . 54
6.4 Invalid name . . . 54
6.5 Invalid email . . . 54

7.1 Actual timeline . . . 58

8.1 Spreadsheets for allocation . . . 65
8.2 Student's timetable . . . 66
8.3 Alternative tools . . . 67
8.4 A table presenting risk rating. . . . 68
8.5 Proposed risks . . . 68
8.6 Log in Page . . . 69
8.7 Time-table of modules . . . 70
8.8 Module selection . . . 71
8.9 Adding a Demonstrator . . . 72
8.10 Informative Pop-Up . . . 73
8.11 Confirmation prompt . . . 74
8.12 Input limitations . . . 77

List of Listings

4.1 Relevant information algorithm 34
5.1 Anti-forgery token 45
5.2 Prevent cookie theft 46
5.3 Error reporting 46
5.4 HTML encoding 46
5.5 Email server 48
5.6 Confirmation email 48
5.7 New labs 49
5.8 Edit labs 50
5.9 Delete lab 50
6.1 Input limitations 54
8.1 Module controller 75

List of Tables

2.1 Alternative tools 18

3.1 Risks table 28
3.2 Risks continued 29

7.1 Objectives fulfilled 59

List of Abbreviations

API	Application programming interface
CRUD	Create read update delete
CSS	Cascading style sheets
GUI	Graphical user interface.
HR	Human resources
HTML	Hypertext markup language
HTTP	HyperText transfer protocol
MVC	Model view controller
PHP	Hypertext preprocessor
SQL	Structured query language
SSL	Secure Sockets Layer
UML	Unified modeling language
URL	Uniform resource locator
WWW	World wide web
XSS	Cross site scripting

Chapter 1

Introduction

Approximately three decades ago, Tim Berners-Lee, a scientist at CERN revolutionised the modern world of technology by creating the World Wide Web. Tim Berners-Lee invented HTML, URL and HTTP. His aspiration was to help his fellow colleagues share information between them in a fast and easy way [28]. Since then websites have become a part of our life. Especially nowadays, there is a variety of technologies that make use of them. For instance, any work we want to do on the WWW can be done with a few clicks in an easy and simple way. Speaking of which, new technological advances and demands, will further help in the development of this functionality, resulting in World Wide Web use in most if not all the situations [38].

Initially, the WWW introduced only web sites. Most of these were mainly data repositories. The invention of the web browser allowed access to this knowledge. However, the most interesting innovation was when this sites when from one-way flow of information to two-way [18]. Allowing the user, not only to receive information but also upload. A transformation that made the WWW unrecognisable compared to its earlier form.

This introduced a variety of technologies and supported the growth and creation of new software now referred as the ecosystem of web-based applications [2]. Like most of the times new technologies bring with them new, ingenious, ground breaking features and possibilities. Web application revolutionised our world by giving us the possibility of cost effective development, the ability to access such applications from anywhere, sustain and maintain became much easier than before while more improvements and conveniences greatly boost the quality of our life [20].

1.1 Problem introduction

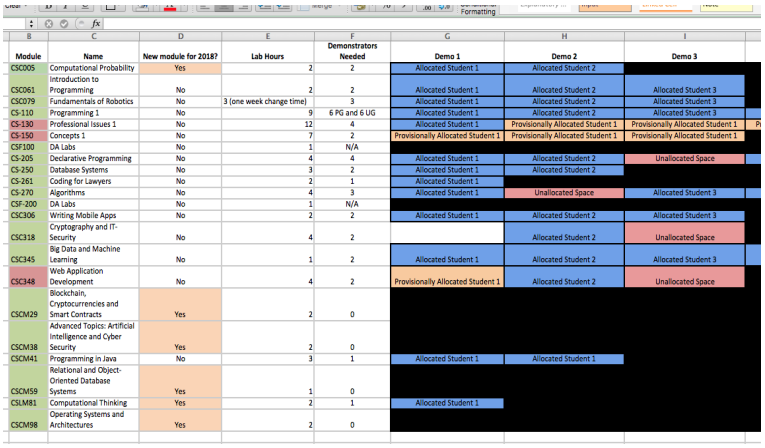
Swansea University has recently increased its number of students in many departments due to their way of teaching, reputation and the novelties in the technological field. Even though its new campus is equipped with the best technological teaching tools and administrative processes, the demonstrator theme for students is outdated and manually performed. People which are responsible for demonstration allocation have to do that twice annually. Thus, an assisting tool would be very useful and time-saving for both the demonstrator and the responsible agent as manually would require even months in some cases to allocate everyone correctly. Considering that there is no software that does this, we would like to help find a solution for it.

1.1.1 Explaining the background of demonstrator allocation in universities

To begin with, in every university there are practical sessions for all appropriate modules. Those practical lessons are usually run by the lecturer and depending on the number of attendees a relevant number of demonstrators is needed to help assist and guide students during practicals. What most of us do not take into consideration though, is how those demonstrators are assigned to those practicals. For that reason, there is a selection process. Undergraduates, post-graduates, post-doc or PhD students are welcome to apply for this. Given some criteria they undergo through a selection process. Eventually, every practical session will be assigned its own demonstrators. That is what we are after for, to provide a practical solution to this problem which will allow a simplification of the procedure through a web application.

1.1.2 Current Procedure

Currently, no software can do that process. Instead, there are spreadsheets as shown below. After someone has shown interest in demonstrating, the person responsible will take a look at the timetable as shown in figure 1.2 to make sure that there will be no clashes. If that is accomplished then a few other constraints need to be checked as well. Such as the year of studies and the maximum number of students that can be allocated to that specific module. With the spreadsheets in figure 1.1, it is clearly conveyed that it requires a lot of effort. Firstly, all the constraints need to be set and all the data has to be present. Such as, module code and module name, lab hours and how many demonstrators are needed. After these are established, the procedure of finding demonstrators begins. A time-consuming and complex operation as shown in the following **low resolution** figures.



Module	Name	New module for 2018?	Lab Hours	Demonstrators Needed	Demo 1	Demo 2	Demo 3
CSC005	Computational Probability	Yes	2	2	Allocated Student 1	Allocated Student 2	
CSC061	Introduction to Programming	No	2	2	Allocated Student 1	Allocated Student 2	Allocated Student 3
CSC079	Fundamentals of Robotics	No	3 (one week change time)	3	Allocated Student 1	Allocated Student 2	Allocated Student 3
CS-110	Programming 1	No	9	6 PG and 6 UG	Allocated Student 1	Allocated Student 2	Allocated Student 3
CS-130	Professional Issues 1	No	12	4	Allocated Student 1	Provisionally Allocated Student 1	Provisionally Allocated Student 1
CS-150	Concepts 1	No	7	2	Provisionally Allocated Student 1	Provisionally Allocated Student 1	Provisionally Allocated Student 1
CSF100	DA Labs	No	1	N/A			
CS-205	Declarative Programming	No	4	4	Allocated Student 1	Allocated Student 2	Unallocated Space
CS-250	Database Systems	No	3	2	Allocated Student 1	Allocated Student 2	
CS-261	Coding for Lawyers	No	2	1	Allocated Student 1		
CS-270	Algorithms	No	4	3	Allocated Student 1	Unallocated Space	Allocated Student 3
CSF-200	DA Labs	No	1	N/A			
CSC306	Writing Mobile Apps	No	2	2	Allocated Student 1	Allocated Student 2	Allocated Student 3
CSC318	Cryptography and IT-Security	No	4	2		Allocated Student 2	Unallocated Space
CSC345	Big Data and Machine Learning	No	1	2	Allocated Student 1	Allocated Student 2	Allocated Student 3
CSC348	Web Application Development	No	4	2	Provisionally Allocated Student 1	Allocated Student 2	Unallocated Space
CSCM29	Blockchain, Cryptocurrencies and Smart Contracts	Yes	2	0			
CSCM38	Advanced Topics: Artificial Intelligence and Cyber Security	Yes	2	0			
CSCM41	Programming in Java	No	3	1	Allocated Student 1	Allocated Student 1	
CSCM59	Relational and Object-Oriented Database Systems	Yes	1	0			
CSLM81	Computational Thinking	Yes	2	1	Allocated Student 1		
CSCM98	Operating Systems and Architectures	Yes	2	0			

Figure 1.1: Spreadsheets used to allocate demonstrators in different labs. A **higher resolution** figure can be found here 8.1.

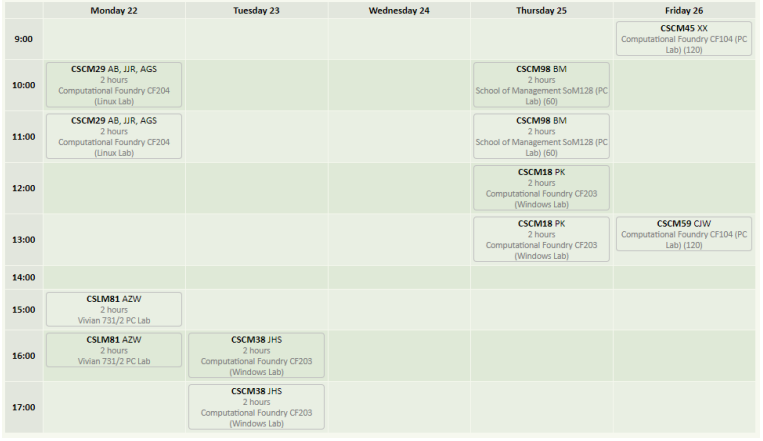


Figure 1.2: Example of a student's timetable. A **higher resolution** figure can be found here 8.2.

1.2 Project definition

Having to manually allocate all demonstrators creates a massive time-consuming approach. Thus, this project proposes a web application which will be the most logical thing to do. As automated procedures will be carried out by the application where demonstrators will be able to apply for a module they are interested in.

It is obvious that such a program simplifies this procedure greatly and reduces potential errors while saving time and effort. In addition, the application would allow for a lot of customisability and will give the opportunity to the person in charge to make changes in a simple and effortless way.

Overall, a website which would help people apply for demonstrating a module they are interested in will be useful not only for them as they will gain experience from helping other people but also for our university as it will increase productivity. Comparatively, helping our department automating such procedures means that those responsible for that will have more time to work on other duties making it beneficial for everyone.

1.3 Tools Utilised

This section briefly describes some initial steps regarding all the tools required for the design, implementation and finally the deployment of this application.

For the design face, prototyping tools were used to give us an idea of how the program would look like and how the interactions would take place. For example, draw.io [16] was used for the design of prototypes. Such prototypes are listed in Appendix 8.1. Further discussion of these can be found in section 4.2.

Choosing the right framework, programming language and tools in general is a difficult choice that many developers either new or even existing ones undertake frequently during their career. Not to mention that, when we realise that there are truly several tools to choose from and that all have their very own advantages and disadvantages it tends to be very hard to choose the most suitable one.

Given this circumstances, the most preferred framework for this web application as it turned to be, was visual studio. It is a really useful tool as it takes care for a lot of messy stuff like SQL by itself. On top of that, some basic skills of JavaScript, CSS and C# where invaluable for the completion of this application.

Chapter 2

Background research

2.1 Current tool

The current tool that is used is a very primitive but at the same time an exceptionally practical way to apply the procedure required. Spreadsheets are electronic documents great for organisation and data storage in a tabular form. VisiCalc[6] was the first spreadsheet tool that became widely known to the public.

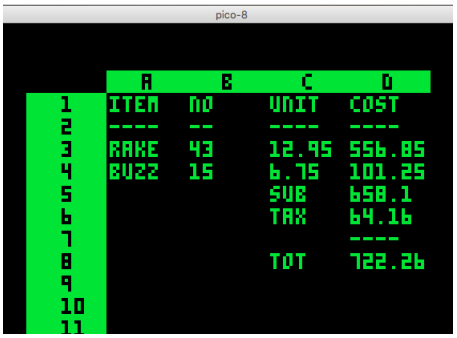


Figure 2.1: VisiCalc was one of the first spread-sheets computer programs for the era of personal computers [36].

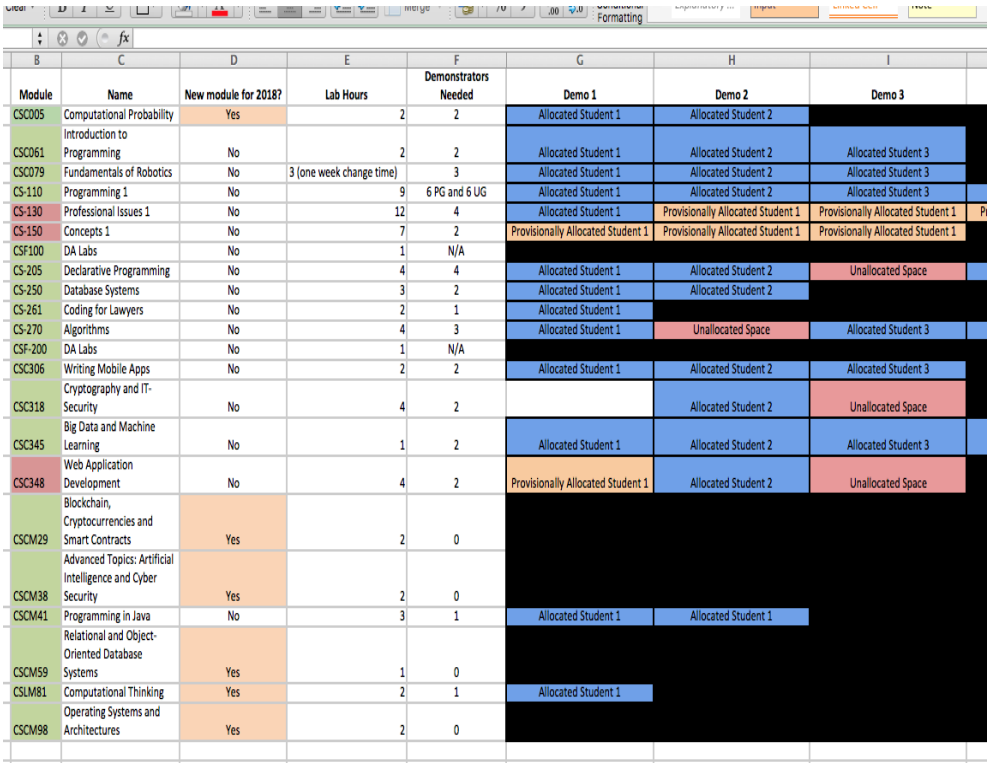
Since then we have come a long way. Nowadays with spreadsheets provided by Google we can access them anywhere from a lot of devices which have internet accessibility.

This was exceptionally important for the proposed solution. It was fundamental to retain this mobility while improving other aspects that were not useful at all. As an illustration, all the necessary effort will be reduce to the absolute minimum by the proposed solution.

2.2 Current procedure

The current procedure is a very complex one because the allocation of all demonstrators to each module requires considerable time and effort.

- The first step in this process, is to fill in a spreadsheet where all the modules will be listed as conveyed in the figure 2.2 which is below.
- In the next step, the person responsible has to fill in all the necessary details about every module. For instance, if this is a new module, the lab hours, demonstrators required and any other special request such as an equivalent number of post-graduate and under-graduate students.
- The following step is by far the most time-consuming. The procedure of finding people which requires a lot of communication.
- Finally, if there are no changes all the labs with their demonstrators should be at last completed. Doing this though, takes a couple of months, thus an immensely time consuming task.



Module	Name	New module for 2018?	Lab Hours	Demonstrators Needed	Demo 1	Demo 2	Demo 3
CSC005	Computational Probability	Yes	2	2	Allocated Student 1	Allocated Student 2	
CSC061	Introduction to Programming	No	2	2	Allocated Student 1	Allocated Student 2	Allocated Student 3
CSC079	Fundamentals of Robotics	No	3 (one week change time)	3	Allocated Student 1	Allocated Student 2	Allocated Student 3
CS-110	Programming 1	No	9	6 PG and 6 UG	Allocated Student 1	Allocated Student 2	Allocated Student 3
CS-130	Professional Issues 1	No	12	4	Allocated Student 1	Provisionally Allocated Student 1	Provisionally Allocated Student 1
CS-150	Concepts 1	No	7	2	Provisionally Allocated Student 1	Provisionally Allocated Student 1	Provisionally Allocated Student 1
CSF100	DA Labs	No	1	N/A			
CS-205	Declarative Programming	No	4	4	Allocated Student 1	Allocated Student 2	Unallocated Space
CS-250	Database Systems	No	3	2	Allocated Student 1	Allocated Student 2	
CS-261	Coding for Lawyers	No	2	1	Allocated Student 1		
CS-270	Algorithms	No	4	3	Allocated Student 1	Unallocated Space	Allocated Student 3
CSF-200	DA Labs	No	1	N/A			
CSC306	Writing Mobile Apps	No	2	2	Allocated Student 1	Allocated Student 2	Allocated Student 3
CSC318	Cryptography and IT-Security	No	4	2		Allocated Student 2	Unallocated Space
CSC345	Big Data and Machine Learning	No	1	2	Allocated Student 1	Allocated Student 2	Allocated Student 3
CSC348	Web Application Development	No	4	2	Provisionally Allocated Student 1	Allocated Student 2	Unallocated Space
CSCM29	Blockchain, Cryptocurrencies and Smart Contracts	Yes	2	0			
CSCM38	Advanced Topics: Artificial Intelligence and Cyber Security	Yes	2	0			
CSCM41	Programming in Java	No	3	1	Allocated Student 1	Allocated Student 1	
CSCM59	Relational and Object-Oriented Database Systems	Yes	1	0			
CSLM81	Computational Thinking	Yes	2	1	Allocated Student 1		
CSCM98	Operating Systems and Architectures	Yes	2	0			

Figure 2.2: Spreadsheets Used to allocate demonstrators

Therefore, the spreadsheets were studied in depth and evaluated in such a manner that they guided the project to obtain the best overall result as they highlighted any requirements and needs that could be used by the application.

2.3 Alternatives

There are a lot management systems that could combine systems and processes. Human resources can be easily managed by such systems. A list of such systems is shown below.

- BambooHR in an applicant tracking system.
- Zoho People which among others includes time management.
- Dayforce HCM provides web and mobile access to a cloud based application.
- SAP SuccessFactorshas a target in promoting the talented employees.
- Kronos Workforce Central can be configured to serve different purposes such as labour cost, risk, productivity, etc.

In practise there are numerous software solutions that could provide assistance. With that said, we have compiled a list of advantages and disadvantages that give us a more in depth view on those software systems on table 2.1.

Alternatives	Advantages	Disadvantages	Verdict	APP
BambooHR	It covers the most basic and some additional components. It is a very well-structured design. Effortless to set it up. Has an open API that make it easy to integrate existing organisation software [33].	Expensive compare to other rivals. Other antagonist offer administration benefits [33].	It might not be the cheapest option but is well worth it. It is well organised, visually appealing and simple to use [33].	X
Zoho People	Allows for customizability in various areas. This can include custom forms and fields [31].	Small technical issues. Both web and mobile applications need improvement [31].	A very cost effective system. User friendly in creating and managing online forums [31].	✓
Dayforce HCM	Beautiful appearance and intuitive GUI make's it very simple to use especially for people with technical abilities. Ability to manage task progress among devices [5].	A few technical issues concerning HTML [5].	We can actually anticipate some quirks with HTML timesheets. However a highly configurable tool [5].	✓
SAP Success Factors	It can handle quite a lot of management systems and can be used as a learning platform as well [27].	System workflow is confusing and too much information is being displayed to the user which end up making the system even more complicated [27].	The absence of a mobile platform does not make it appealing to everyone or at least practical to a lot of organisations [27].	X
Kronos Workforce Central	An extremely flexible tool that is simple to set up and run for every user [19].	An expensive human resource management product. Sometimes the system speed is an issue [19].	Simple and quality product. However is best suited for large multi location companies [19].	X

Table 2.1: A table comparing advantages and disadvantages of alternative tools.

Even though some of these HR solutions are great, they are not what we are after for. This solutions combine a great deal of software which is great for some companies but in our case it would just made our solution more complicated. For that reason and after a lot of consideration we knew exactly what was required. All the solutions which were compared and discussed in table 2.1 had some very good components. Therefore we used them as a very good example on which our own solution was based.

2.4 Our own tool

Through the selection process of alternative solutions we were naturally led to think that the only working solution will be our own. Since what we were looking for did not exist, because a lot of the compared solutions had features which would have been never used or would have been of great use of a company in a much greater scale than our department.

Therefore what we propose is a web application. Because being on the web allows everyone to have access from everywhere. Making the entire process a lot easier and effortless. Comparably, assisting our department to automate such operations implies that those liable for it have even more time to spend on other tasks, making it advantageous to all.

There were a lot of reasons for this choice. It was vital to make certain that the university data will not be used by any other company. Evermore, the cost of this software would have yet another expense for our department. On top of that we knew that there was not any software which met our needs exactly and that any system we could have purchased would have been excessively modified to serve its purpose. For these and many other reasons we chose to develop our own software solution for this time-consuming task.

Chapter 3

Specification

3.1 Problem Specification

Figure 3.1 conveys the specifications agreed to create a nice, simple and above all functional web application, all vital components of a successful product. Along with them there are a couple of words on how these specifications influenced the web application. Although, all of these specifications are discussed in chapter 7 and critically evaluated according to the work that was supposed to be done and what we have actually done.

- **Automated Procedures**

  All the procedures have been automated as much as possible for improved productivity.

- **Easy to use GUI**

  This enabled easy use of the system, taking advantage of the WWW and providing simple means for a time-consuming problem.

- **Easy to collect the applications**

  With the website, is much easier for any student to apply and thus it is effortless to collect the demonstrator's applications.

- **Flat Design**

  The GUI is as minimal as possible and helps users to familiarise themselves with the site and navigate it [43].

- **Three-step Application Procedure**

  Since everything is done automatically, the entire allocation procedure is much simpler.

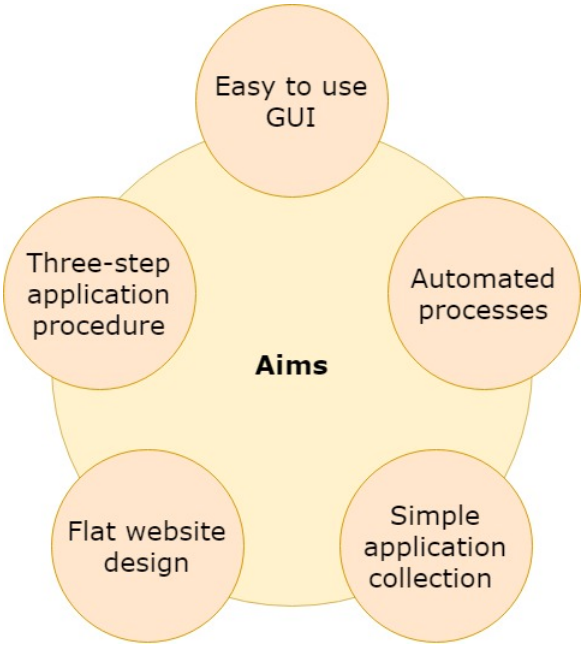


Figure 3.1: The project aims in a graphical representation.

3.2 UML Diagrams

This section provides system scenarios illustrated through used case and activity diagrams. These figures 3.2, 3.5 give a clear insight of the different uses of the system.

3.2.1 Activity Diagram

The diagram below represents the workflow of this system in a graphical way [7]. It describes the functions taken, actions and the outcomes of them.

The first activity is the log in. Following the flow of the system the student will be represented with a list of available modules which require a demonstrator. During this activity a student will be able to apply for a lab.

The system will check again that the lab is still available and it will submit the application. The administrator will be able to view the new demonstrator. A confirmation email will also be sent to the applicant.

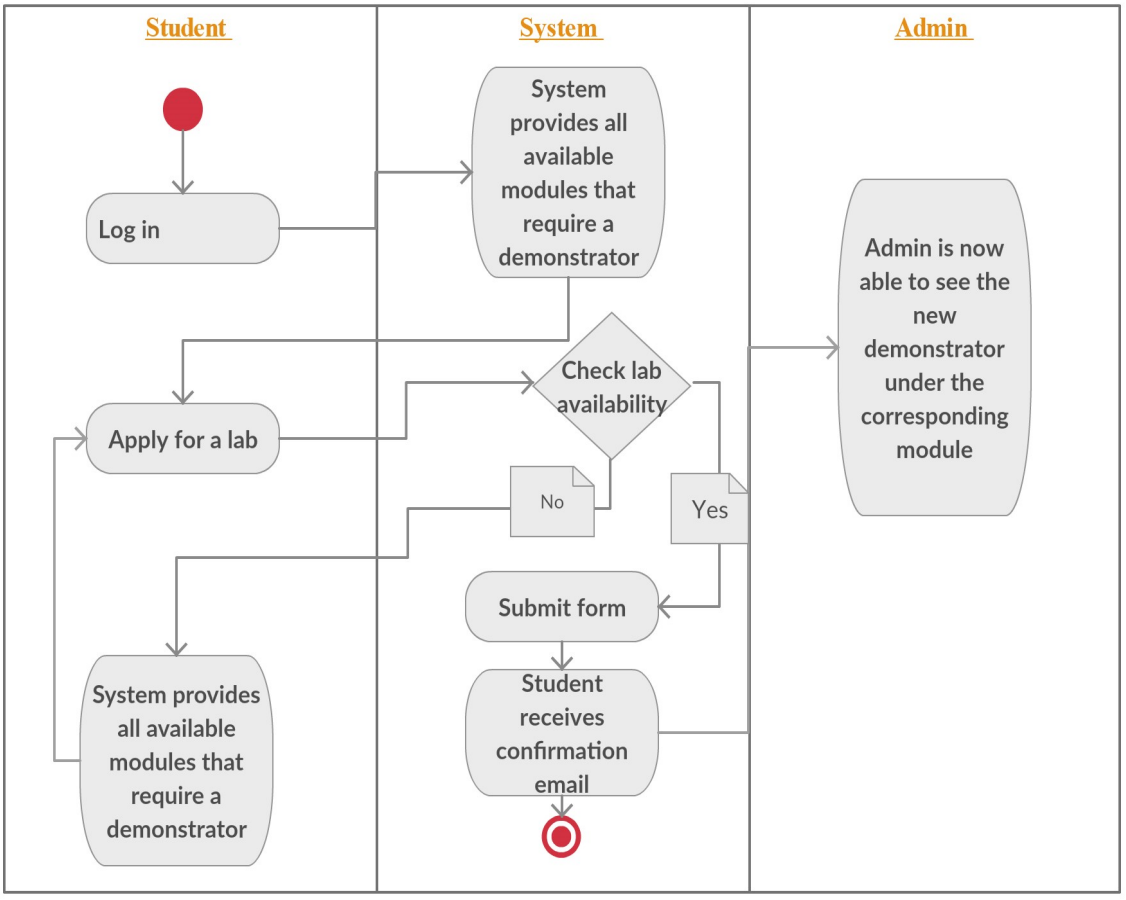


Figure 3.2: Workflow diagram explaining the application procedure [40]

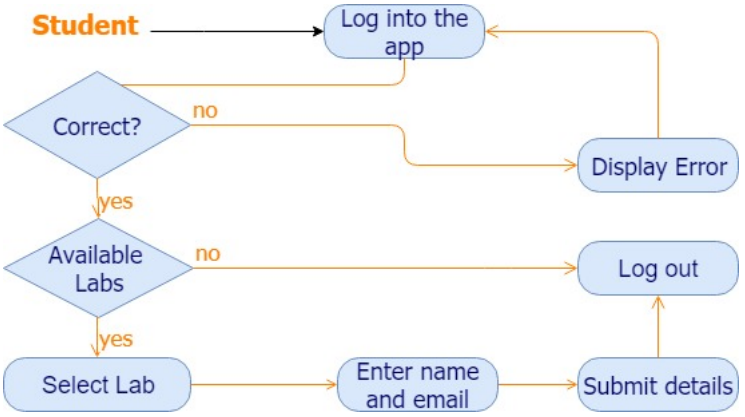


Figure 3.3: Sequence diagram explaining a workflow of a student.

The above use case scenario describes the process a student undergoes to apply for a lab. By using the university credentials the user can log in on the website. Then, the user gets shown a list of available modules depending on the constraints applied by the application. The next step of this scenario is to select a lab. If there is no lab, the student will not be able to select anything. Consequently, the user will log out of the program. On the other hand, if there is a lab it will be selected and the student will be able to apply for it. Finally, the last step of the process is a confirmation email which includes all the necessary details of the lab.

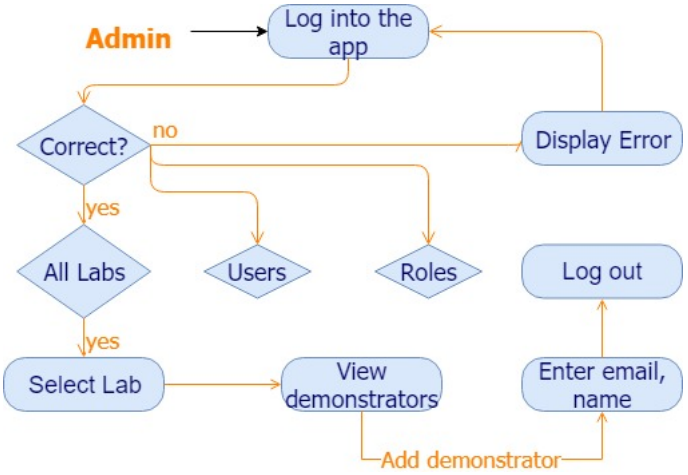


Figure 3.4: Sequence diagram explaining a workflow of an administrator.

Figure 3.3 provides a detail diagram of an example use case. The person in charge of the allocations logs in the website. The home page of the website is made up of a list of labs. These can include either available or not available labs for demonstrators. In our example case, the user wants to add a new demonstrator to a specific lab. In this scenario, the corresponding lab needs to be selected. The website will redirect us to a new page where we can enter the details of the new demonstrator. These include the name and university email. Following this action we simply submit the details of the student. After this point no additional action is required. The student will receive a confirmation email which includes the lab details.

3.2.2 Interaction Diagrams

The interaction diagrams shown below represent different functions by those interacting with the system. .

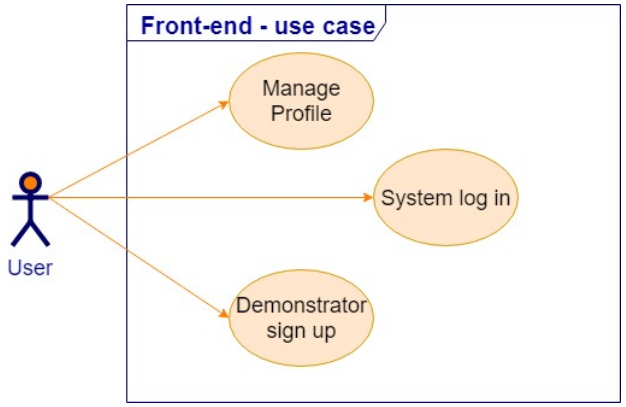


Figure 3.5: Graphical overview diagram representing the front-end user model.

Here we have a diagram conveying all the possible actions a student can take. Those are the options to login, edit profile details or select and apply for the module they would like to demonstrate.

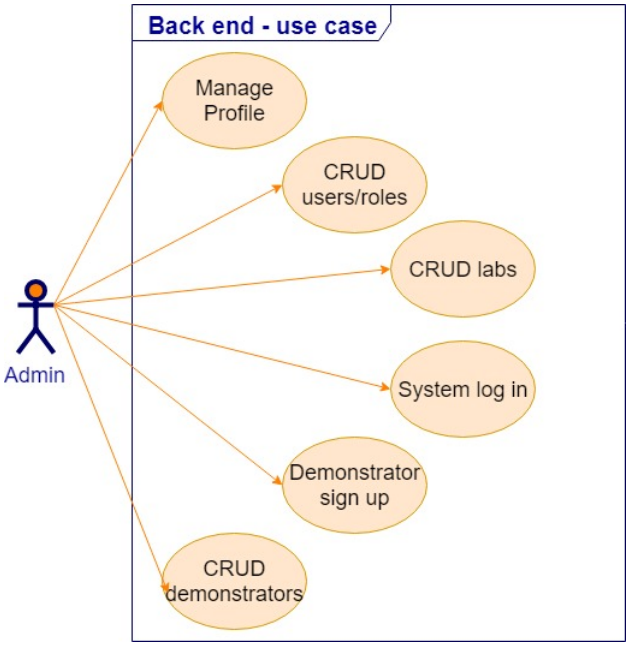


Figure 3.6: Graphical overview diagram representing the back-end user model.

This figure 3.6, represents all the actions the web page administrator is able to do. These among other include, the ability to manage the user profile, all the CRUD services available for users and the option to add or delete demonstrators.

3.3 Proposed Functionality

Below is a collection of goals suggested for this project.

- For the application, a database language will be used as requests will be made to get information back to the site to fulfil the constraints. Hence, **SQL** will be used [12]. With SQl we aim to store users, modules and roles on the database.
- After SQL, it will also be necessary to **implement** the database with Visual Studio. To test functionality, the database will need to be seeded.
- **Visual Studio** is a software development framework design by Microsoft. First, with the framework, it offers a substantial level of security. Second, creating amazing designs is relatively easy. It also supports the MVC structure system, which is not supported by many frameworks [29].
- **Graphical User Interface** when implementing a website is extremely important to pay attention to the design. A responsive design means that every device will make the website accessible. Furthermore, a minimalist design is a feature that attracts many people because it also makes it user-friendly [24].
- The **security infrastructure** is a vital component of a web application. This project will verify that there are no vulnerabilities such as cross site scripting, cross site forgery, over posting attacks, cookie theft or any error reporting risks [42].
- Another goal is an **auto-generated email**. The person in charge will be delighted to know that an email will be sent to all those involved in a specific module with a click of a button.
- In addition, another challenge will be that anything entered should be **validated**. For instance, they need to enter their email address when they log in. If they don't enter their email, an error message will tell them what went wrong. [3].
- There will also be a small **contact us** section. Having this section will help in resolving any issues that may arise. It will be useful for people which have log in issues, selection issues or any module issues. Therefore keeping a channel of communication between the people who will interact with the website and those responsible for is essential. Hence, what makes a nice and functional website.
- **Version Control** is a great way to record programme changes or recall a specific version later in the development process.
- **MVC** is useful to enrich the web application design. This architectural pattern is explained in depth in section 5.1.2 with an example to help clarify its use.
- **CRUD** services assist in creating, editing, deleting or updating information from the database. An extensive description of each action and how it is implemented can be found under section 5.5.
- **Users and roles** provide different levels of authorisation across the application. Section 5.3 provides a detail explanation of how the authorisation has been implemented for this project.
- **Verification email** provides an additional security layer for the application. Listed under section 5.4, there is a detail explanation of how the verification email was implemented.

3.4 Proposed solution

This project aims to provide a solution to solve the time-consuming approach currently taken to the problem. A web application is easily accessible from everywhere. This gives the opportunity for easy and effortless access to the system. Allowing future applicants to easily apply for a lab.

Any student will be able to sign in to the application by using the same credentials as the one's used to log in to the university intranet. The home page will include all available modules to that individual. Taking into account the year of studies, any other work and how many demonstrators are required for the module. Future users will be able to choose the lab they want and apply for it with just a single click. Following this action, a confirmation email will be sent to them.

On the other hand, the person in charge will have the ability to manage all the demonstrators in a much easier and unchallenging way. The system will recognise the administrator and provide the means to make any necessary changes. For example, add or remove students as well as create or edit lab details.

3.5 Proposed timeline

After the design of the first schedule there were some tasks which were not included. Some of them were not scheduled but following the project needs they were required for the completion of this application. The figures 3.7, 7.1 convey the proposed timeline and the actual timeline respectively. Figure 7.1, which is discussed in chapter 7 clearly contains a more detailed schedule with an in-depth timeline .

Tasks two to four included the time required to allow the developer to familiarise itself with new programming languages and frameworks,

Task six was the framework choice for the web application development. For that reason a substantial amount of time was devoted to learn this framework. This was proven to be a wise choice as it provided time to use a different framework and as a result complete the project.

The Gantt chart also includes a few tasks which were not important for the project completion but helped to clarify some initial checkpoints. This included, exam preparation during Christmas and summer and the holiday period accordingly. Even though, these were not vital for the completion of the application, they had a positive impact on the project.

Task thirteen included the GUI design. This turn out to be quite a lot of milestones. For example, it should have included many little refines and milestones to achieve this checkpoint. In the chart 7.1 those have been included.

Proposed Schedule

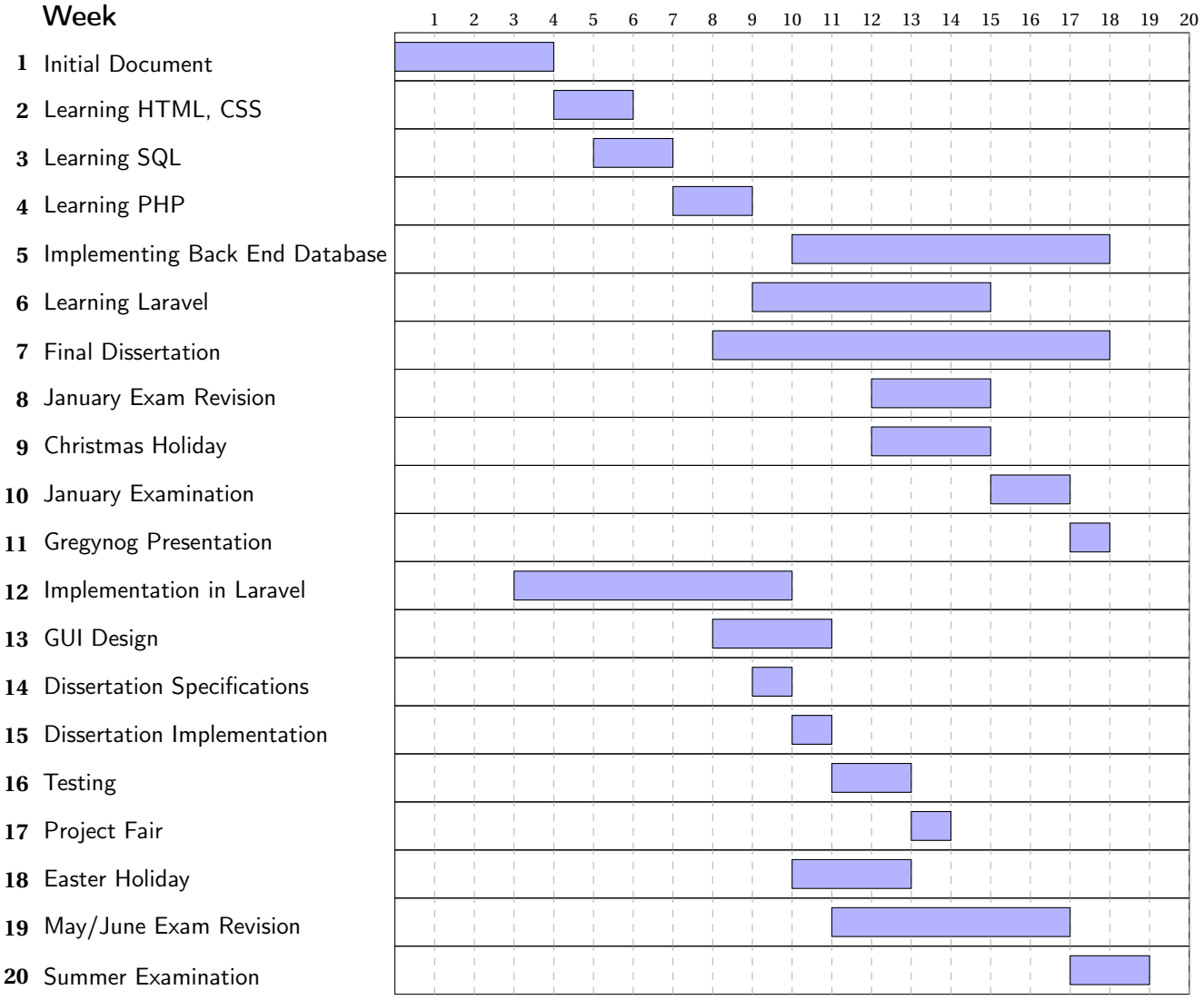


Figure 3.7: Proposed Timeline

It was almost impossible to know prior of implementation and the dissertation write up the small things that could delay the schedule. On top of that, there were a few changes that had to do with the software used. As mentioned earlier, those are thoroughly discussed in chapter 7.

3.6 Risk Assessment

During this project risk tables were produced. These include potential risks, the risk rating for the specified risk, a proposal on how to avoid the risk, potential impact that the risk may have on the project and finally, a mitigation action in the scenario which we have to deal with a risk. Everything mention above is conveyed in tables 3.7, 7.1.

ID	Risks	Impact	How to avoid	Risk rating	Mitigation Action
1	Stress/Family Issues	Delay in the completion of the project.	Never delay the project intentionally. In that way there is play of time to catch up.	Low	With careful planning and plenty of time for each task none of these issues can interfere with the schedule.
2	Being sick	Cause proposed timeline disruption which will end up in delayed components of the project	Be as healthy as possible	Low	Work hard to get back on track with the project.
2	Software / Hardware failure	Lose all the work that has been done so far.	We can actually anticipate when and if there is going to a a hardware or software failure.	Low	Use the previous backups to restore the project.
3	Requirements interpreted wrong	Misinterpreted requirements could result in a major liability. They can also cause a huge amount of work accomplished to be undertaken on a different platform.	Spend plenty of time establish requirements.	Medium	Allocate enough time to be able to re-implement any misconceive requirement
4	Wrong software model	This would result and bad software management.	Research software model in depth to make the best possible decision.	Medium	As long as the implementation proceeds this will not pose a thread to the completion of the application.

Table 3.1: Risks their impact how to avoid them and the mitigation action if required.

ID	Risks	Impact	How to avoid	Risk rating	Mitigation Action
5	Underestimate development time	Fell behind on schedule or spend less time on more important parts of the project.	Work according to plan or even be ahead of the scheduled plan.	High	More work during those days to keep up with the proposed timeline.
6	More expectations than what is possible	Incomplete project or not all the objectives are fulfilled.	Work hard to have enough time to fulfil all objectives.	High	Think of other possible ways to fulfil incomplete objectives
7	Laravel is the best option for web development	The project will not be completed on time or at all.	Take enough time to compare frameworks.	High	Use the second best option for this project in order to deliver an application.

Table 3.2: High rating risks

Each project entails risks. These dangers vary greatly in software engineering. However, a typical procedure should be carried out regardless of the risk [13].

Any significant risk could very well lead into an unfinished or missing piece of work. It is therefore essential to have a schedule, restrict, mitigate, monitor, and minimise the risk or, if required, produce an alternative solution for the successful completion of the project [10].

Tables 3.1, 3.2 provide a detail discussion of risks. Including, a critical analysis for each one of them while stating the mitigation action. Table 3.1 contains only low and medium rating threads to the completion of the project, whereas table 3.2 is made of of the the high rating risks. A critical reflection on the risks and their impact on the project is given under section 7.1.

Chapter 4

Design

4.1 Class structure / System overview diagrams

In order to capture the whole system structure we have produced the following diagram.

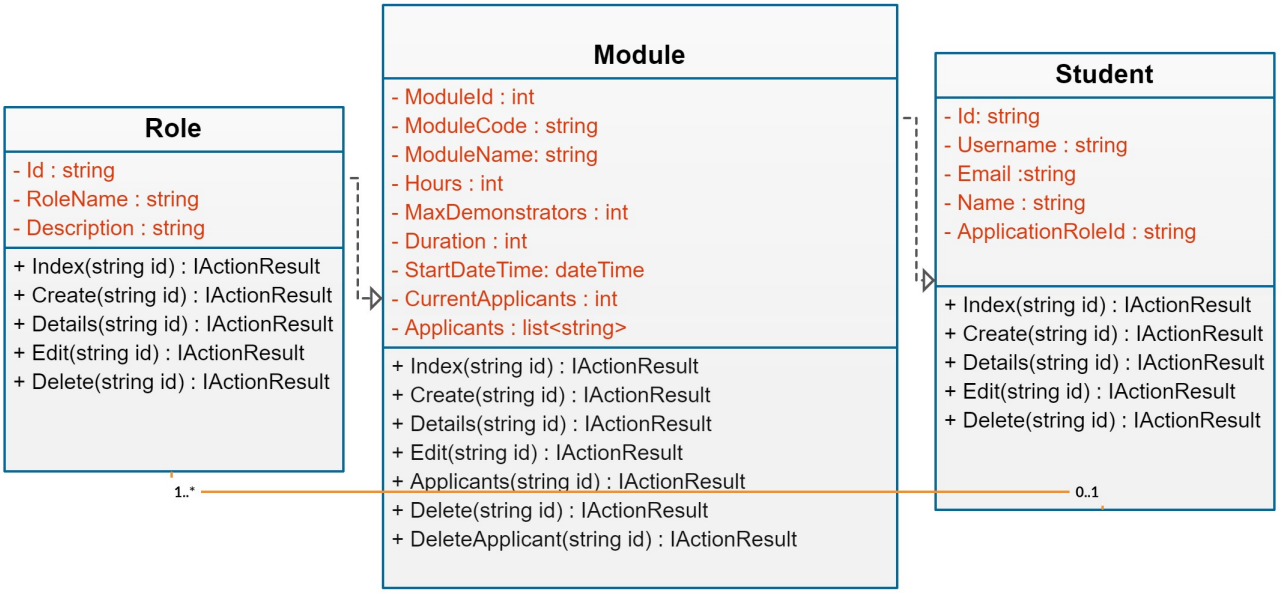


Figure 4.1: Class structure diagram of the three main controllers of the web application along with methods and attributes for each class.

Because visual studio uses the MVC architecture the developer has compiled class diagrams that overlay the class architecture by combining the models, views and controllers that the MVC framework benefits from, The MVC architecture is clearly explained with in depth examples on the next chapter.

The class in the middle(module) has the most attributes and methods as it is the main class of the system. Here we have methods which perform the task of applying for a module or methods to provide students the most relevant labs according to their year of study, available labs, PhD restrictions, etc.

Figure 4.1 also illustrates a graphical overview of the other two supporting classes and their structure. The class student and roles contains important attributes to CRUD actions on the users according on the data they handle. This method and attributes are also mentioned in the next chapter and where they are relevant.

4.2 Prototype design workflows

The following section is about the current design workflows the functionality and the overall functionality of each separate action compared to early designs. Prior to these designs we had created a prototype mock-ups that illustrated the look and feel of the application. These mock-ups are listed under Appendix section 8.1. These were an initial design step to assist the whole process of software development.

Each prototype illustrates different actions taken either by a student or by a person responsible to do the demonstrator's allocation. Each action describes with small and easy steps the procedure that needs to be followed.

A good example is figure 8.7. This was our initial idea of how the main page of the whole web application would look like. However after a lot of trials and errors this design had to be dropped. That had mainly to do with the framework limitations and the additional process that was required to go through. The final design is displayed in figure 4.4. How this page operates is discussed meticulously in the next section.

For simple comparison and to give the reader an idea of the difference between the prototype and final design, we have included the figures discussed above in **low resolution** to portrait the difference in appearance.

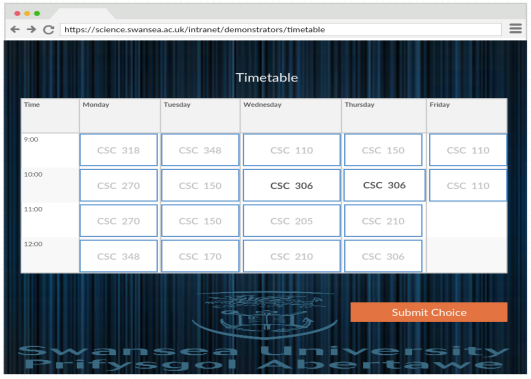


Figure 4.2: Initial Timetable and module selection prototype

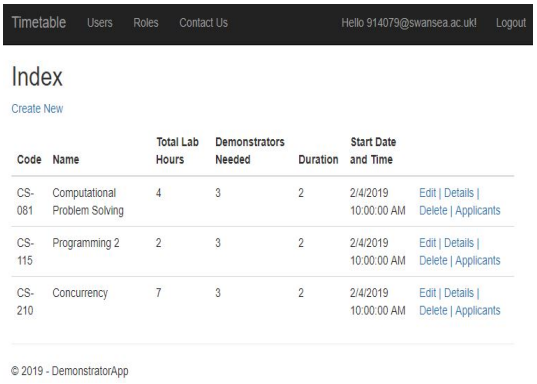


Figure 4.3: Final lab selection design overview

The process of producing early mock-ups was proven to be extremely useful for the project, Visualising the final product assists to comprehend the design process. Getting early feedback on the looks and feels of the application proved to be useful on gathering feedback. This led to identifying the prospective users and served really well on collecting ideas which were utmost important. Generally, user feedback was a very simple but at the same time a great way to validate the prototypes. Having said that, real user experience worked only in our favour and we are delightful for that.

In appendix section 8.1 there is a more detailed discussion about the designs and what did not work quite as expected with each one of them. As a result, this had an impact on the final product. Section 4.3 clearly demonstrates this impact with detailed and informative explanation under each final design.

4.3 Screenshots of different workflows

This section is about individual ideograms corresponding to each function that a student or the page administrator may attempt. This also includes a brief description of what is going on especially as far as the code is concerned.

4.3.1 Admin Home Page

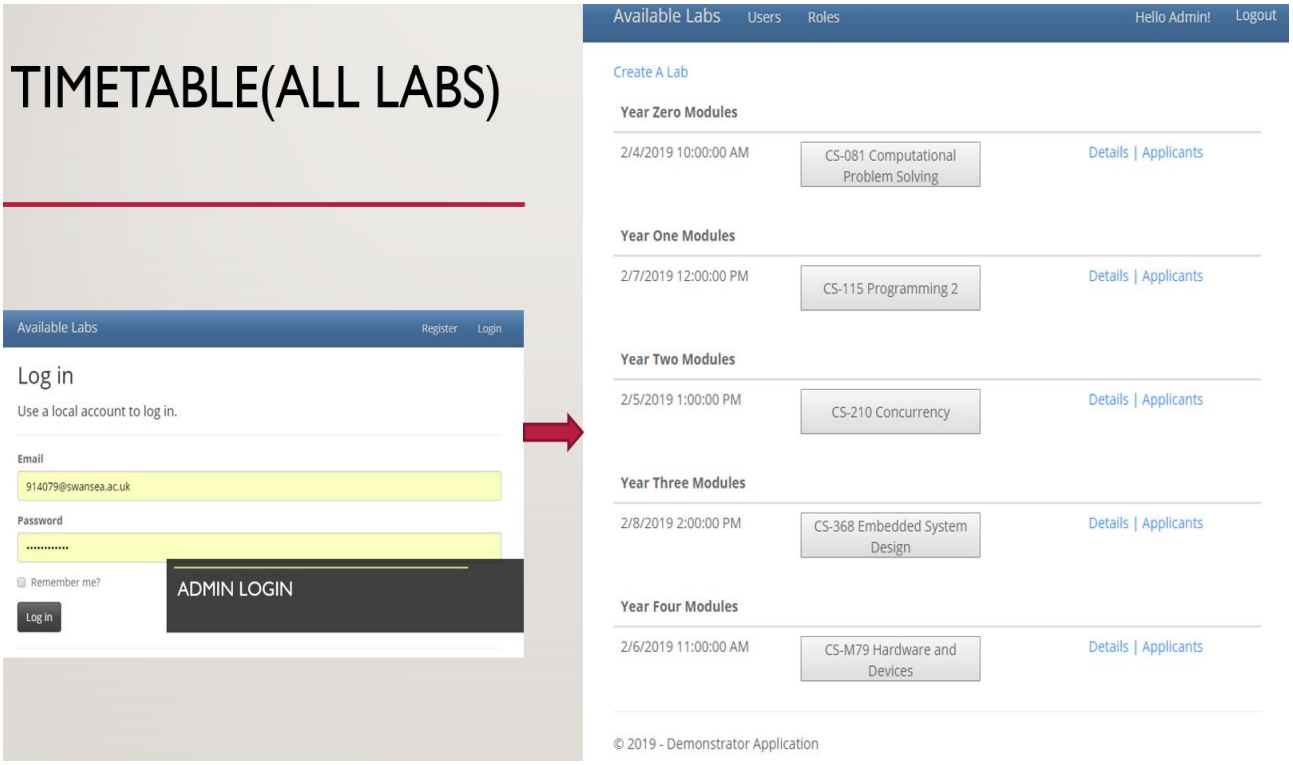


Figure 4.4: This is from the administrator perspective view of the login page and the homepage where a user is redirected after login. The homepage consist of a list with all the modules and links to CRUD services for each one individually.

The above figure 4.4 represents the initial page that the admin will view after logging in. There is no need for a sign up button or page as everyone can log in with the university credentials.

Onto the right of this page, we can distinguish different buttons. Those represent a module. The following screenshot portrays the outcome of that button been pressed.

This was one of the first designs in visual studio. The first step of the whole application was to set up the the log in page as shown in 4.4. The module controller explained in listing 4.3.1 was vital for the correct function of this page.

```
1         // GET: Module
2         public Task < IActionResult > Index ()
3         {
4
5             ApplicationUser u = _userManage.FindByEmailAsync(User.Identity.Name);
6
7             if (user.Year  == Admin)
8             {
9                 // condition to check if user is  admin
10                return View(await _context.Module.ToListAsync());
11            }
12            else  if (user.Year == PhD)
13            {
14                 // Condition to check for PhD students
15                var UnderUserYear = _context.Module.Where(m => m.Year < user.Year && m.CurrentApplicants < m.
      MaxDemonstrators && m.Hours<maxHours);
16                return View(await UnderUserYear.ToListAsync());
17            }
18            else
19            {
20                // condition to check all other students
21                var UnderUserYear = _context.Module.Where(m => m.Year < user.Year && m.CurrentApplicants < m.
      MaxDemonstrators);
22                return View(await UnderUserYear.ToListAsync());
23            }
24        }
```

Listing 4.1: The screenshot of this part of the code contains the algorithm which displays relevant information to the user.

Figure 4.3.1 conveys the part of the code which is liable for presenting relevant data to the user. The full code can be found in appendix 8.5. As we have seen in figure 4.4 there was a list of available labs. This list was compiled by the above code. The module controller displays to us available modules through a view model.

The first condition will check if the user is the administrator and if it is all labs will be displayed. This is there to ensure that only the administrator will have full control of the labs.

After that we check if the user is a PhD student. According to that we need to verify which labs are available and if the PhD student is allowed to teach for that lab by taking into account that maximum number of hours a PhD student is allowed to work per week.

Last but not least on the third condition we display labs depending on the user year of study and which module still needs demonstrators. The last condition will only display relevant information if the student number is verified. Otherwise, nothing will be displayed. This is useful to ensure our data is only shown to relevant students and staff.

4.3.2 Creating labs

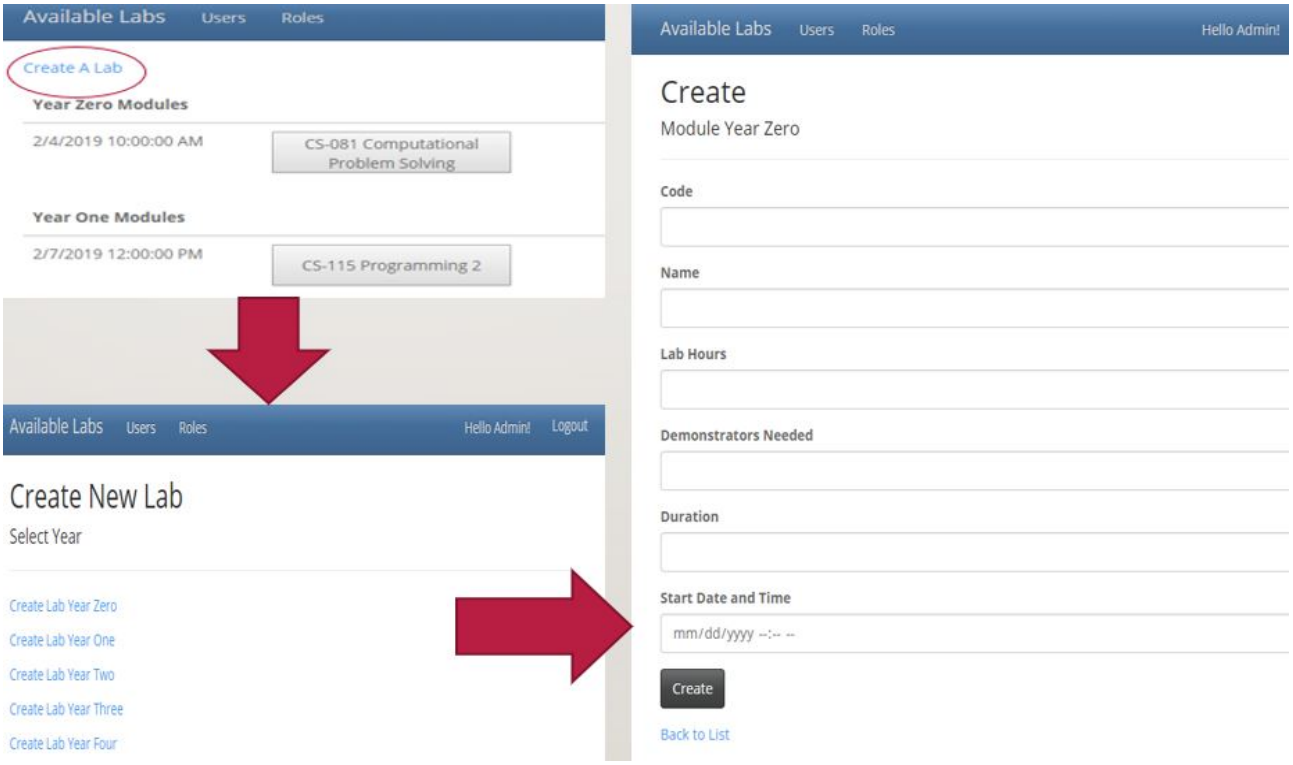


Figure 4.5: The three pages above illustrate how to create a lab by selecting the lab button, the year of the lab and then filling out all details related to the lab.

The figure 4.5 illustrates the procedure required to create a lab if necessary. The button for this can be observed in our screen's top left corner. Upon pressing the button, we are rerouted to a new page.

On that page, we can view buttons related to the year we want to create the new lab.

On the selection of the lab year we are redirected to a new page where we can fill in the details such as code, name, lab hours, etc.

Finally, we get rerouted to the home page. There, we can view the lab we have created with this simple procedure.

This part of the implementation was very important. A lot of the procedures required were linked with each other. Therefore, this process enable us to proceed to the next part of the implementation. Figure 4.6 clearly conveys and explains how this next part works.

4.3.3 Adding/Removing Demonstrators

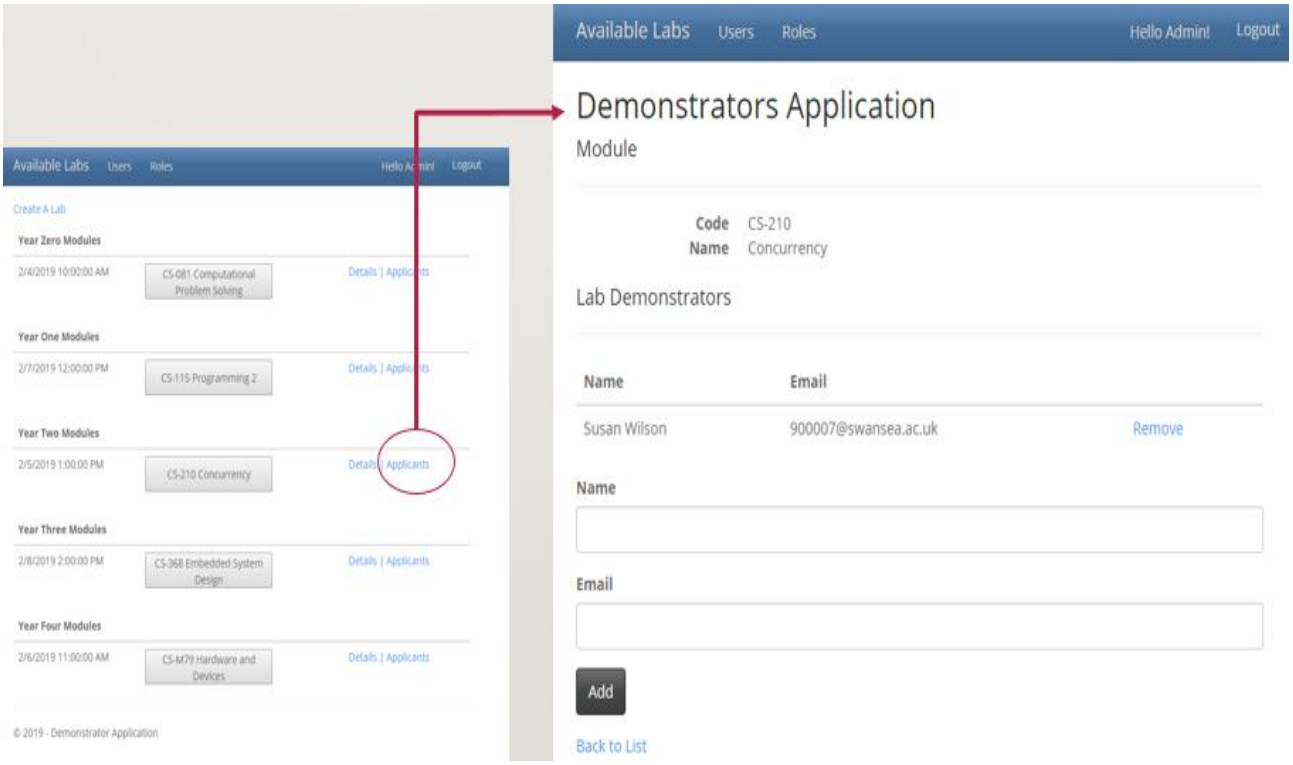


Figure 4.6: This figure demonstrates the process of selecting the applicants page and on the right side the page includes all the applicants for the module and if necessary the means required to add one.

When the applicants button is pressed, we are presented with a list of all the students who have already signed up for this lab demonstration. The admin has the ability to remove or add a demonstrator.

The purpose of the application is to allow easy manipulation of data. Additionally, to assist in this procedure as much as possible. For that reason, the program recognises the maximum number of demonstrators and provides the user with an adequate informative message.

For instance, if a lab has already reached the maximum number of demonstrators it will not be included in the list of available labs. Thus, simplifying the process and reducing the possibility of mistakes or errors.

The prototype design of this device can be found in figure 8.8.

4.3.4 Lab Details

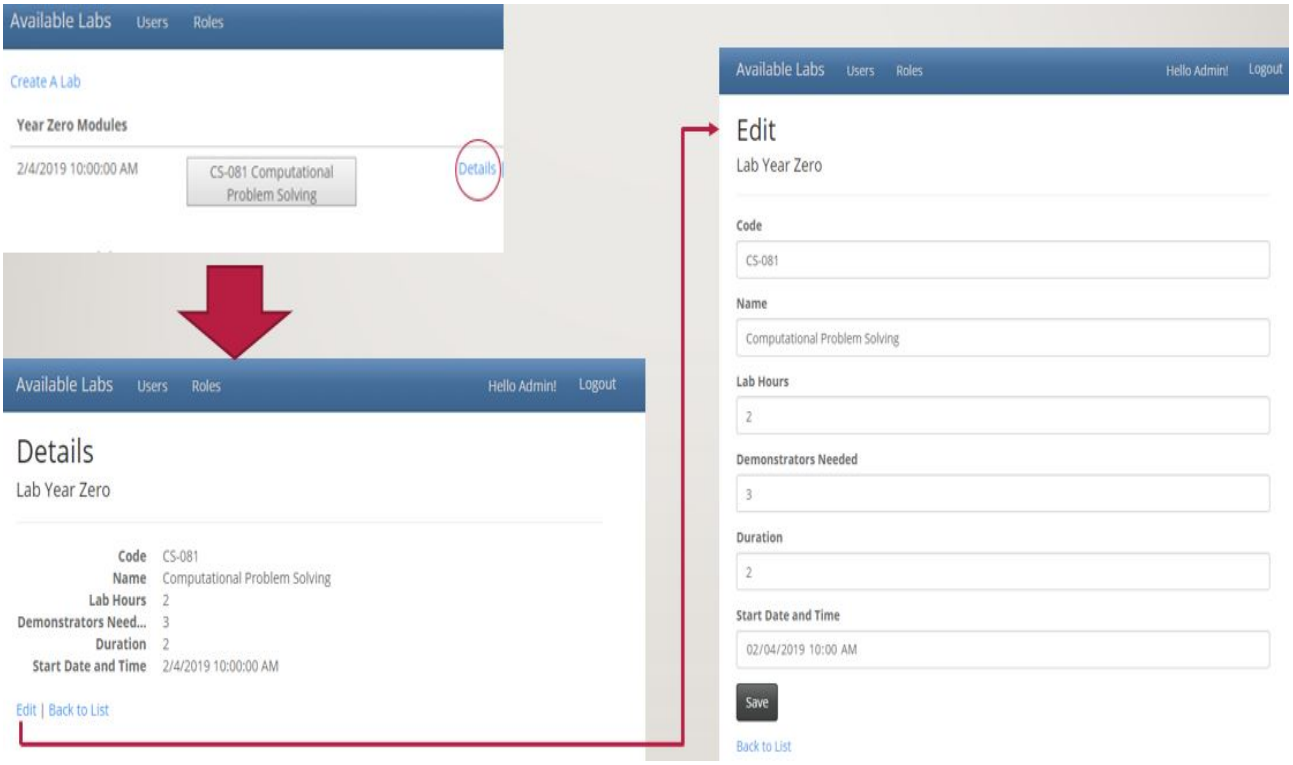


Figure 4.7: Administrator perspective of lab detail and how to update them.

The above figure 4.7 describes how we can edit the details of an existing lab. Our aim was to make this experience as simple as possible.

The button shown in the red circle contains the link to the details page. When pressed the user is redirected to that page. There we can review all the relevant information of a lab.

On the bottom left corner of the website we can view the "edit lab details" button. If necessary as mention earlier we are able to edit them. The red arrow demonstrates where we get redirected in order make any necessary changes.

There for example, we change the number of maximum demonstrators to allow for a greater number of students to apply or we can edit the time and date that the practicals take place.

4.3.5 User CRUD Actions

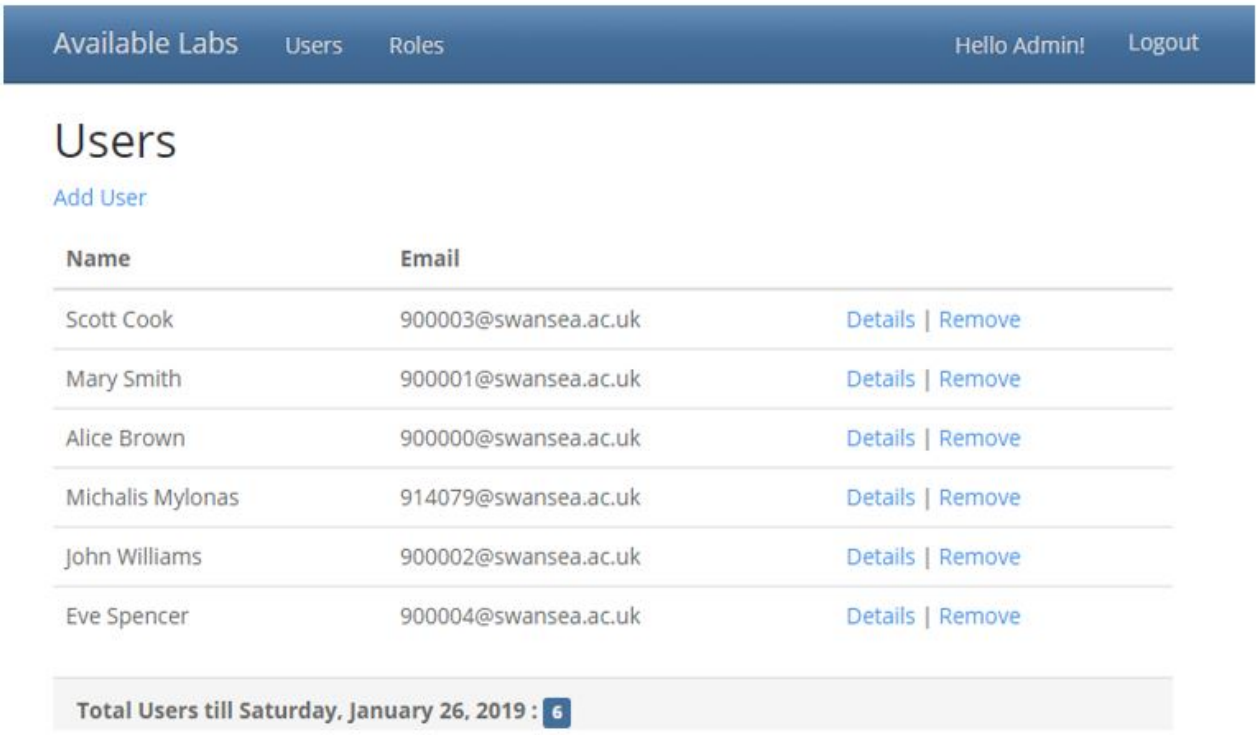


Figure 4.8: All users signed in for lab demonstrations and links to relevant CRUD actions about them.

Figure 4.8 represents all the users currently signed in a lab demonstration. The person in charge has the ability to apply any of the CRUD services. As shown in figure 4.9.

From the figure 4.8 we can select a user and for example update the role of that user. That is a very likely scenario to happen as the roles of each user give them access to different modules according to their year of study.

The button add a user is located under the users page title in figure 4.9. The user gets redirected to a form where all the relevant details need to be submitted.

If we do not want to create a new user and we just want to edit an existing one, all we have to do is press the details button which is clearly visible on the screen.

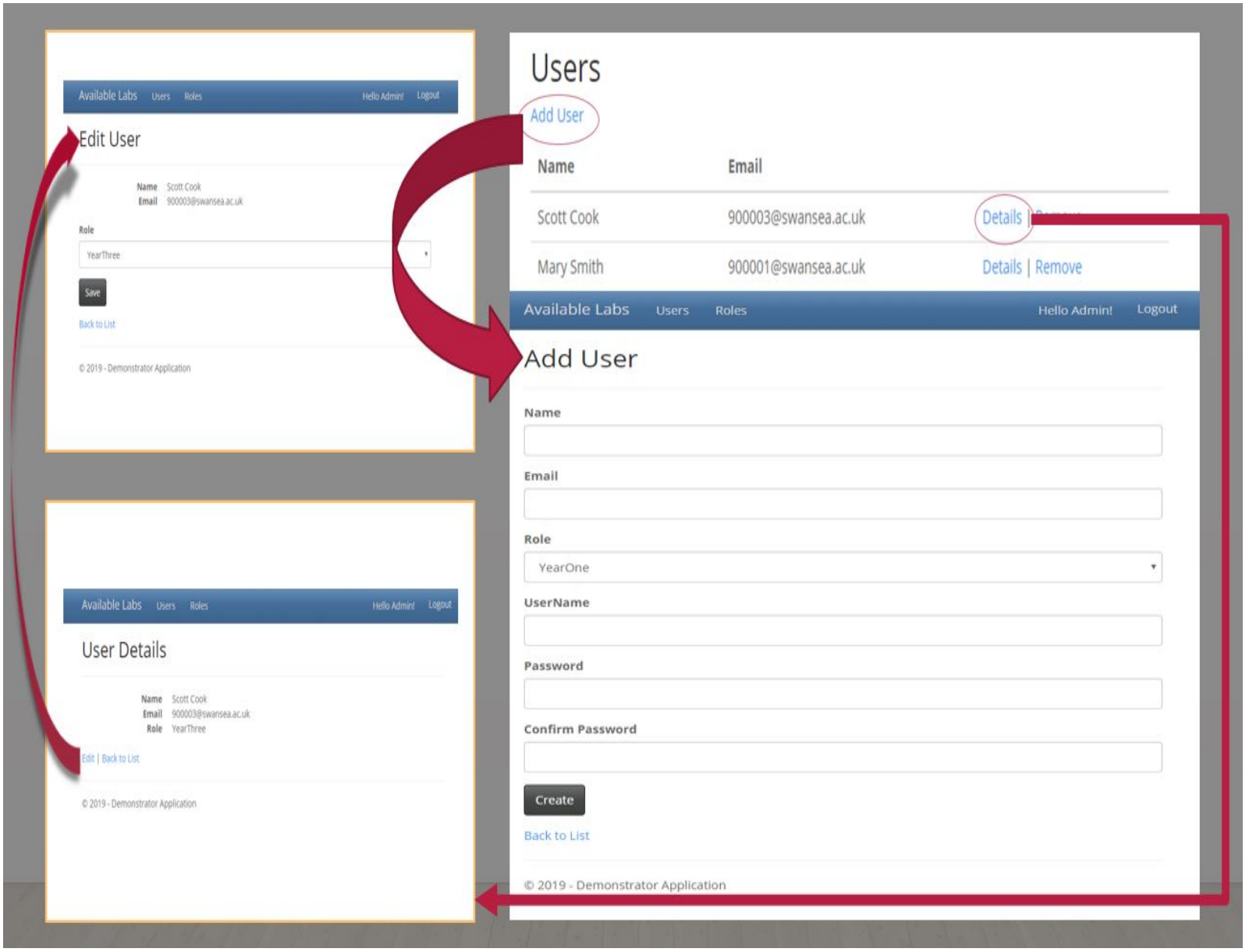


Figure 4.9: Here is a detail ideogram with arrows explaining the process of editing or adding a user on the database.

4.3.6 Roles CRUD Actions

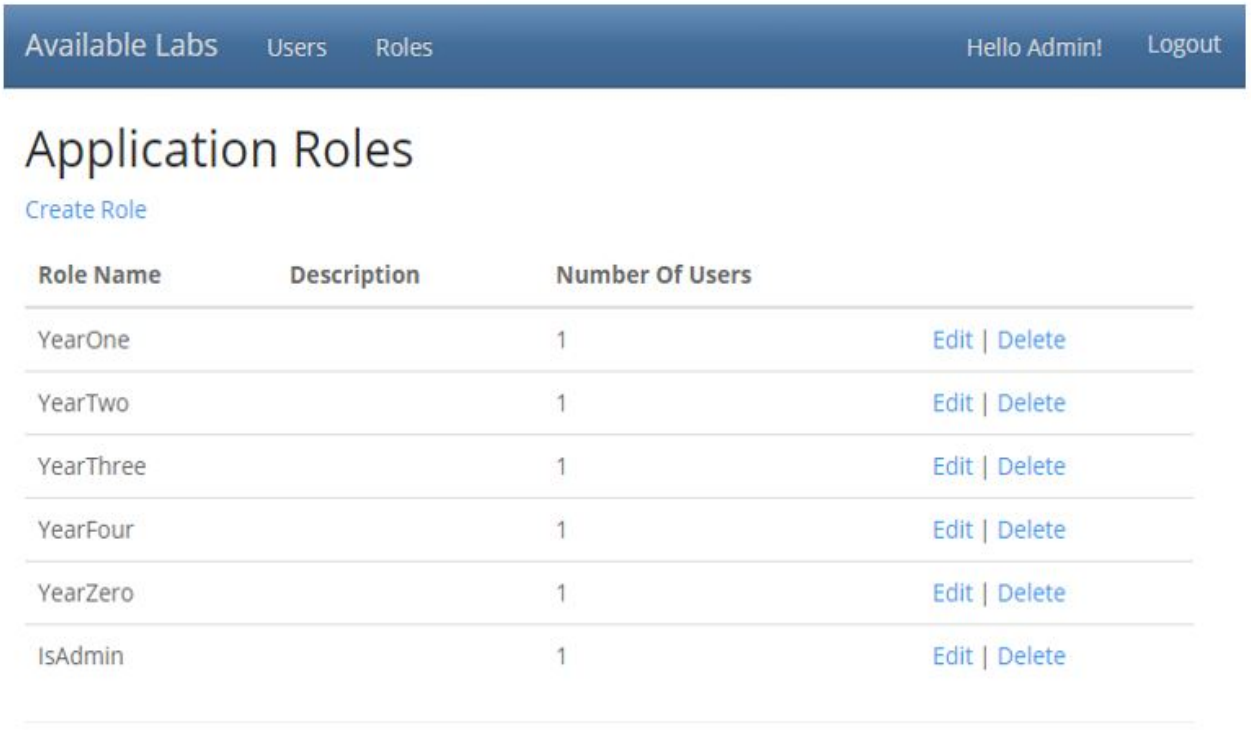


Figure 4.10: The administrators point of view on users roles along with links to apply CRUD actions for each individual role.

Figure 4.10 conveys all the roles that currently exist on the system. As mentioned earlier, the application manager will be able to apply any of the CRUD actions as shown in the figure 4.11.

Only the administrator is able to access all of these pages and up to this one. We do not want any of the students messing around with authorisation/permissions. This was the same reason why it was important to include such privileges for the administrator.

Despite the fact that this was not a major university system, it was still wise to include as much flexibility and customizability as possible.

The figure below, provides an informative example on how to edit a role, if necessary how to delete one or clear cut examples on how to create a role.

The completion of the roles and users shown in 4.10, 4.8 accordingly was crucial for the project completion. Those were extremely important to milestones. Mainly, because they allowed further development of the project.

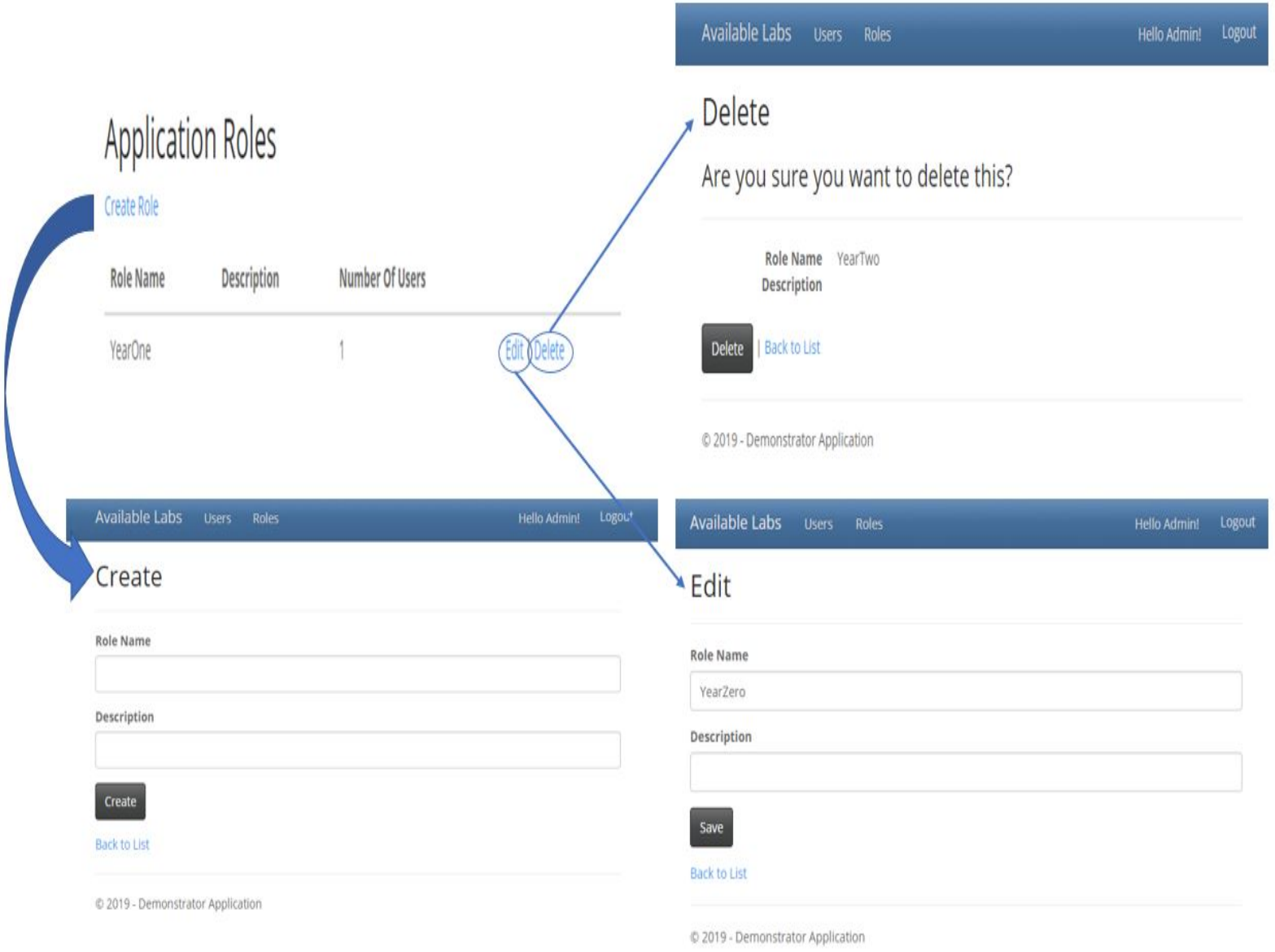


Figure 4.11: A detailed ideogram explaining with arrows all the CRUD actions that can be taken for each role individually.

4.3.7 Student/Normal User Login

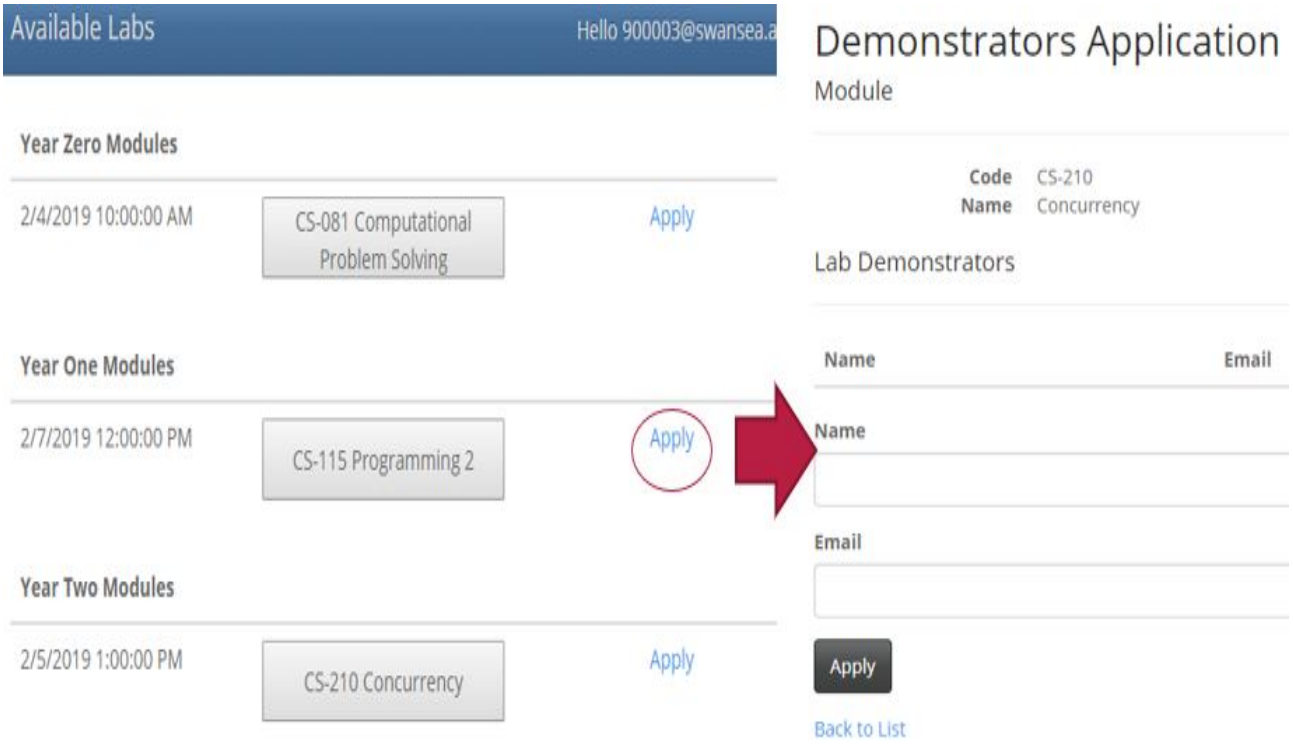


Figure 4.12: The web application from a student's point of view, along with arrows to illustrate the process of applying for a lab demonstration.

The list on the left hand side is compiled by the same code shown in figure 4.3.1. This was the last part of the core functionality. With this done all the core functionality was completed. Allowing the developer to focus on other functions which were not vital but were nice additions to the whole application.

Last but not least, in the figure 4.12 we are able to observe the procedure in which a student is able to apply for a lab. After a user has logged in, gets displayed a list with all the available labs that are suitable for him/her.

Next to each selection there is a button to apply(red circled button). When the button is pressed, the user gets redirected to a new page.

This page is similar to the same page shown in figure 4.6. The only difference has to do with the authorisation levels. For instance, the delete button is not shown because we do not want a student to be able to delete an applicant.

Therefore, the page consists of text boxes to submit the name and the email. Consequently, with just the apply button a user can register for a lab with this simplified way.

Chapter 5

Implementation

5.1 Tools used to develop solution

The following section is about the main tools used to implement the web application. Among others, these include version control, MVC and visual studio. Visual studio is composed of a lot of other small tools which can be seen under section 5.1.3.

5.1.1 Version control

To begin with version control is a way to record changes on a project and then have the ability to recall a specific version later on in the development procedure [8].

A simple way to control versions of your project is to use a time-stamped directory to back up your project's current status. However, this way is incredibly prone to error because the file you are in is easy to be overlook and as a result, you can edit the wrong directory.

The version control used for this project was the one mentioned above. The reason for that was that uploading the files in the cloud means that it is out of our control to keep them safe. Thus, the backups of the project were saved locally in two different hard disk drives.

Additionally, since the project was not supposed to be shared online for collaborative work, it did not make a big difference having a local version control. Distribute version control systems like GIT exist for that purpose and are extremely helpful for large organisations and businesses. In our case, it does not make a lot of difference since there are two backups of the project in different drives [35].

5.1.2 MVC

The MVC design structure was utilised for the application implementation [26]. This design structure can be divided into three different stages model, view and controller. The controller interacts with the model or view. This is effective for the project since this model is used to enrich the configuration of the web service. This also enables to develop software quickly and makes it possible for the website to be compatible with a variety of browsers [9].

On top of that, MVC architectural pattern divides the control of the application as mention above into three main logical sections(model, view, controller). Thus, not allowing the user to interact with sensitive data. For instance, the controller takes requests from the user. These can be requests regarding module data. Then the controller manipulates this requests to the model. This means that the model uses any available resources, such as the database. It looks to get the right information from that request through this data. Once the information has been processed , the view will be created from the model and the controller will address it. Thereby, the web application will display it to the user.

A clear example where the MVC has been used in the project has been demonstrated in figures 4.9, 4.11. Because, one of the view models is utilised when a user clicks the create button. Consequently, the controller takes the input from the view with an action and returns the view model by using the model.

5.1.3 Visual Studio

Visual Studio is an excellent framework for developing web applications, especially if you want to create a web application that is highly marketable and secure. The level of security integrated and the things that the framework does to assist the developer is astounding. To begin with it assists on creating the CRUD actions. To be honest, for simple things the developer has to specify a model and then visual studio creates both the view and the controller. Obviously for more complicated methods the developer has more work to do but at least for some simple methods everything is made by the framework.

Another important aspect that the framework takes care for the developer is the security of the website. The framework provides a command called anti-forgery token which ensures that no cross-site forgery will happen. With another command(httpOnlyCookies) it prevents cookie theft. Generally, visual studio has supplied many simple and useful commands to developers for any security thread.

On top of that, another handy aspect of this program is the native ability to change the application layout as the browser window resizes. This is provided by the default bootstrap from visual studio, and as long as we do not mess with it, it works perfectly. Apparently, if you experiment with this feature it can be broken in a very ugly and repelling style.

5.2 Security Infrastructure

Because security is very important in web applications. Actually, a lot of users will not notice that you have done a good job with security but they will notice if you do not.

Two very simple rules we have taken into account while implementing this application were:

- We should never assume that the framework, visual studio in our case will take care of everything for us.
- Secondly, we should never under any circumstances trust any data we are given by our users.

Here is a list of all the security risks that have been taken care of:

- There is no obvious cross-site forgery risk.

```
1 // POST: Module/Demonstrators
2         [Validate-Anti-Forgery-Token]
3         public Task<ActionResult> Demonstrators(
4         ModuleDetailsViewModel viewModel)
5         {
6             if (ModelState.IsValid)
7             {
8             // creating a new application user
9                 ApplicationUser user = new ApplicationUser
```

Listing 5.1: The anti-forgery token is used to overcome the cross-site scripting or over-posting requests.

- Actions were taken to prevent theft of cookies.

```
1
2           // Action to prevent cookie theft.
3           public void Configure( services )
4           {
5               s.Configure < CookiePolicyOptions > ( opt =>
6               {
7                   //Consent for non essential cookies
8                   opt.CheckConsentNeeded = context => true;
9                   opt.MinimumSameSitePolicy = SameSiteMode.None;
10              }) ;
```

Listing 5.2: Only cookies allowed, prevents cookies theft.

- Steps have been taken to mitigate the risks posed by error reporting.

```
1   // GET: Module/Demonstrators/5
2           public Task<Result> Demonstrators ( int userId)
3           {
4               if  (userId == null)
5               {
6                   return Not Found();
7               }
```

Listing 5.3: Whenever there is an error the user will be redirected to the iconic "404 page not found".

- HMTL encoding everything has mitigated the requests for XSS and over posting. This is the task of adjusting all characters designated for HTML with codes in order to try and make them as a string.

```
1               <th>
2                   @Html.Display(m => m.ModuleCode)
3               </th>
4               <th>
5                   @Html.Display(m => m.ModuleName)
6               </th>
7               <th>
8                   @Html.Display(m => m.Hours)
9               </th>
10              <th>
11                  @Html.Display(m => m.MaxDemonstrators)
12              </th>
```

Listing 5.4: Encoding to overcome cross-site scripting requests.

5.3 Backend development of database

Any substantial changes or updates of any kind should be made by the application administrator. These include adding or removing users from the database, the modules or the roles. This is essential since the website must be fully controlled by the admin.

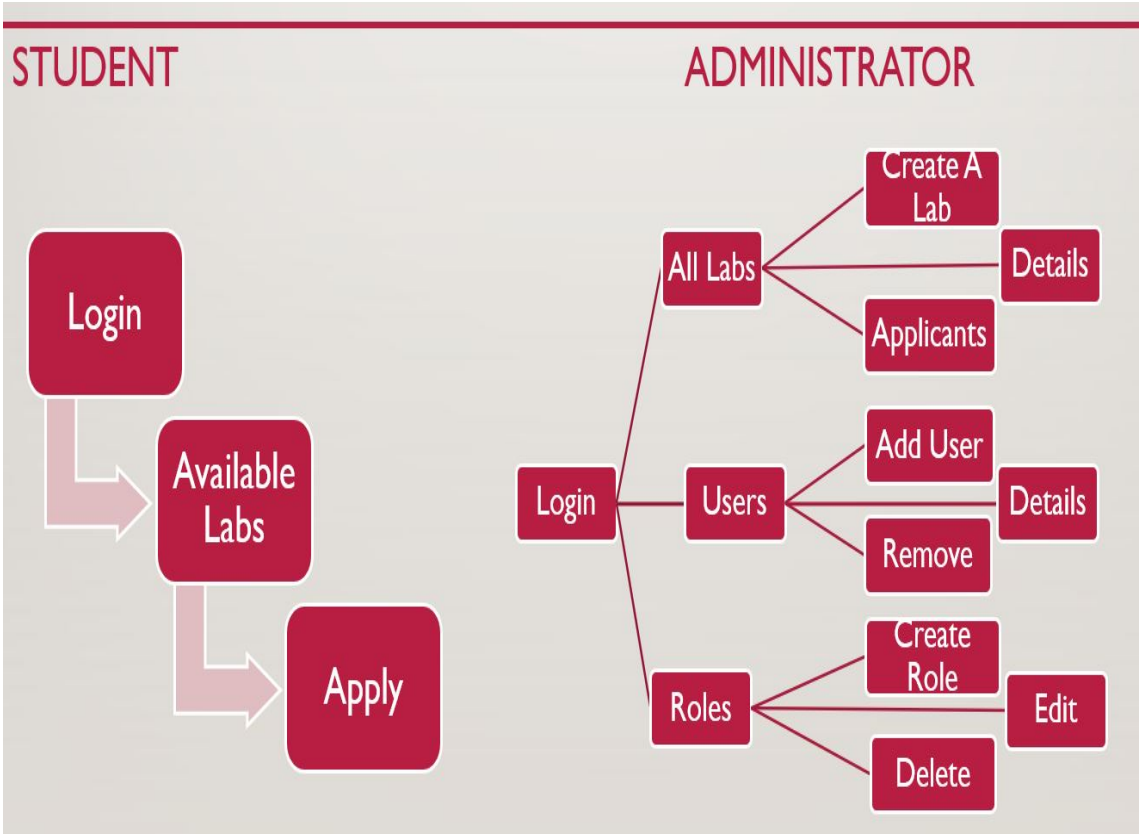


Figure 5.1: Class structure overview diagram depicting what a student and what a demonstrator can access within the application

Figure 5.1 illustrates the different level of authorisation deployed in this web application. A student will only be able to select from a list of available labs. Whereas, the administrator will have the ability not only to view those labs but also edit them. On top of that, roles and users will be accessible by the application admin in case there is a need to modify or delete a user, a role or a module.

Evermore, depending on the student year of study the system will provide labs that correspond to that particular individual. Additionally, any labs that have already reached the maximum number of demonstrators will not be displayed. Another possibility for a lab to not be displayed is if we have a PhD student. They are only allowed to work 6 hours per week. Therefore, if they have already reached that, they will have no lab to select from.

Thus, the application uses user accounts to differentiate users. Because of that, users have different levels of authorisation. Additionally, there is an interface for site owners to create and edit users and user roles.

5.4 Email Server

```
1                namespace DemonstratorApp.Email
2 {     public class DevEmailSender : IEmailSender
3     {           public Task SendEmailAsync(string email, string sub, string htmlMessage)
4         {                email client = new SmtpClient("smtp.gmail.com") // Email server
5             {
6                 UseDefaultCredentials = false,
7                 Credentials = new NetworkCredential("applicant.confirmation.noreply@gmail.com",
8                 "!Password123!"), // Email client
9                 Port = 587, // Port used
10                EnableSsl = true, // SSL
11
12            };
13            var mailMessage = new MailMessage // new email obejct
14            {
15                From = new MailAddress("applicant.confirmation.noreply@gmail.com")
16            };
17            mailMessage.To.Add(email); // Recipient
18            mailMessage.Subject = subject; // Subject
19            mailMessage.Body = htmlMessage; // Body
20            return client.SendMailAsync(mailMessage);
21        }    }    }
```

Listing 5.5: IEmailSender is the class responsible of the email server.

The procedure of sending an email in visual studio ASP.Net Core is kind of complicated but doable. Firstly, we had to set up a new class called DevEmailSender. This class includes the credentials of an email address, the port used, if we want SSL to be enable and the client server we are going to use. Luckily, Google allowed us to use their email client with a trial account. There was a limitation of 500 emails per day which did not pose a problem for us [14].

```
1             Email.DevEmailSender devEmailSender = new Email.DevEmailSender();
2               await devEmailSender.SendEmailAsync("kuhgyv@gmail.com", "Demonstrator Application Confirmation",
3                   "You have succefully applied to demonstrate: \r\n" + "Module Name: " + module.ModuleName+ "\r\n"
4                   + "Module Code: " + module.ModuleCode+ "\r\n" + "Day and Time: "+module.StartDateTime+ "\r\n");
```

Listing 5.6: Code added to the demonstrators method in order to send a confirmation email to each applicant.

Figure 5.4 conveys the code that was required to allow the application to send a confirmation email to applicants when they register themselves for a lab demonstration. This code manages the information which the email will include regarding the module each user has selected.

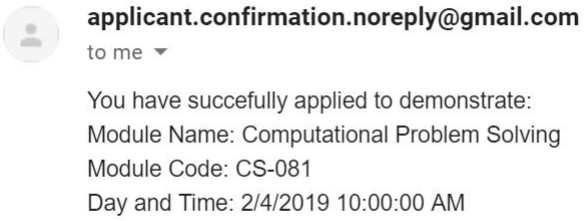


Figure 5.2: Proof that the email server is working.

Finally, in figure 5.2 we have the confirmation email. We thought that it is important to include the module name, code and the hours that this module takes place in the confirmation email.

5.5 Implementing the CRUD actions

This section will represents the code part of figures 4.9, 4.11 which depicts the CRUD actions.

Figure 4.3.1 represents the read action. Therefore, this sections will focus on the rest of the CRUD services.

5.5.1 Create Service

```
1        // A method to create new modules
2         public Task < IActionResult > Create([bind("ModuleCode,ModuleName,Hours,MaxDemonstrators,Duration,
       StartDateTime")] Module module)
3           {
4               if (ModelState)
5               { // Adding data to the databa
6                   _data.Add(module);
7                   await _data.SaveChanges();
8                   return RedirectTo ( ( Index ) );
9               }
10              return View(module);}
```

Listing 5.7: Create method to create new labs

This method simply creates a new module by taking all relevant information from the view. The view is where the user submits module name, code,etc. This information is then passed to the controller where the module gets created and then saved in the database.

5.5.2 Update Service

```
1   // A method to edit module details
2   public Task < Action > Update ( int moduleId )
3   {
4       if (moduleId== null)
5       {
6           return Not Found();
7       }
8
9       module = data.Module.Find(moduleId);
10      if (module == null)
11      {
12          return Not Found();
13      }
14      return view module;
15  }
```

Listing 5.8: Edit method to edit lab details

The edit task is used to change the details of a module. For instance, if we want to change the maximum allowed demonstrators we simply click the edit button as illustrated in 4.7 and then we are redirected in the edit details page where we can easily amend one of the current information fields about the module.

5.5.3 Delete Service

```
1   // A method to delete modules
2   public Task < Action > Delete( int moduleId)
3   {
4       if ( moduleId == null )
5       {   return Not Found();   }
6       Module module = _data.Module(module => module.ModuleId == id);
7       if ( module == null )
8       { return Not Found(); }
9       return view (module ) ; }
```

Listing 5.9: Delete a lab from the list of module to be demonstrated

Last but not least, the delete service lets the administrator to remove a lab if it is ever required. This button along with the rest of the services mentioned above are only visible to the administrator of the page. Additionally, only this person has access to perform those actions. These can be verified with roles which are nicely referenced in figure 4.10.

Chapter 6

Software Testing

6.1 Design and implementation of the tests and their results

Software testing helps to improve a product's quality, while finding faults / bugs is one of the software testing purposes [1], it is not the sole purpose. Verifying and validating that this web application meets the stated requirements / specifications is extremely important [21]. As this will result, in quality improvements which will help later in the process to drastically reduce support, service / maintenance costs after release [30].

6.2 Functionality Testing

This type of testing checks whether functional requirements/specifications are met by testing various parts of the application[34]. The components listed below were some of the most important elements which we had to ensure that they were functioning properly.

6.2.1 Links

At every step of the implementation it was important to test if the implemented part was working [45]. For example, when the email confirmation system was implemented, all the links leading to sending the email confirmation were tested. After a few errors, the application was able to properly send an email. Thus, after each implementation the application was tested to confirm that it behaves the way it is supposed to.

6.2.2 Forms

Forms were tested-checked to verify that they were functioning correctly and were readable [39]. For instance, if a user does not check a box which is a required field, an error notification will be displayed. Hopefully, this was included with ASP.NET CORE and as long as these limitations were specified in code, the users were notified with informative feedback why there was an error.

The next step had to do with the default and required values. It was essential that these values were saved in the database as indented. Following a quick look on the database data we were able to validate that no one was able to submit data without submitting all the required values [32].

6.2.3 Cookies

Cookies provide a way to keep track of our preferences [23]. Here the goal was to check if they are populated properly. As a paradigm when testing if our credentials were saved, we were able to log out and log in without needing to type our credentials again. When deleted, they were properly deleted and we were asked to resubmit them the next time we visited the website.

6.2.4 HTML, CSS

Sometimes a misplaced column on the CSS, HTML source code can and will lead to an error which will in some circumstances cause the entire page to be presented in a weird configuration. In order to verify the site will not have such errors, we checked for any syntax errors, then we made sure that all the colours are readable and the page layout is how it is supposed to be [37].

Practically it is impossible to cover every possible component of testing with the same accuracy. Therefore, we used all the help we could get from external resources as explained in section 6.6.

6.3 Unit tests

The purpose of unit testing is to make sure that a method, class, etc. is functioning the way they are supposed to [25]. This may indeed included not only valid but also invalid inputs. It is important that our code functions properly before someone else is able to use it. Figure 6.2, 6.1 asserts that the classes function the way they were expected.

```
[Fact]
public void Task_GetModuleById_Return_OkResult()
{
    //Arrange
    var controller = new ModuleController(repository);
    var moduleId = 2;

    //Act
    var data = controller.GetModule(moduleId);

    //Assert
    Assert.IsType<OkObjectResult>(data);
}

[Fact]
public void Task_GetModuleById_Return_NotFoundResult()
{
    //Arrange
    var controller = new ModuleController(repository);
    var moduleId = 3;

    //Act
    var data = controller.GetModule(moduleId);

    //Assert
    Assert.IsType<NotFoundResult>(data);
}
```

Figure 6.1: Code sample of unit testing module controller.

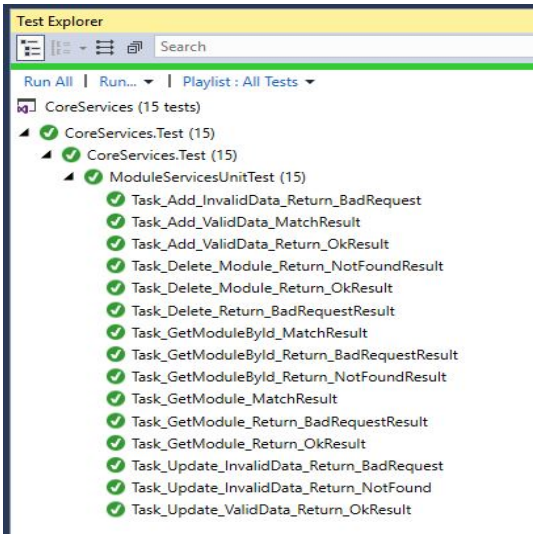


Figure 6.2: Proof of unit testing of the module controller.

These tests were useful in testing and debugging our code. Automating test cases reduced the effort required on performing tests and allowed us for a more thorough evaluation of our project. Resulting in a higher quality product [15]. Many bugs were found during the process. Some of them were complex while some others were an easy fix. Overall, it was exceptionally important for code quality.

6.4 Input Validation

Input validation is vital for the security of our application. Multiple attacks can often be launched towards a web app that insert malformed information [44]. With visual studio we are allowed to set limit on the input values.

```
1 // Sets string length
2         [MinLength(6), Required]
3         [( DisplayName = "Module-Code" ) ]
4         Module ModuleCode { get ; set ; }
5
6         // Sets string length
7         [MinLength(2), Required]
8         [ ( DisplayName = "Module-Name" ) ]
9         Module ModuleName { get ; set ; }
```

Listing 6.1: Input Limitations, setting module code to be a a fix size of characters,

Figure 6.4 clearly displays the limitations added to check the input values. Firstly, the module code and name need to submitted and cannot be left blank. Additionally, the module code needs to be at least six characters and the module name must be at least two, as figures 6.3, 6.4 illustrate.

Figure 6.3: Proof of invalid input check on module code.

Figure 6.4: Proof of invalid input check on module name.

Figure 6.5: Proof of invalid input check on user email.

This example represents the user email address. In case we want to edit the user email address, the new email address has to be in the correct format. If we do not do that the application states that we have made a mistake. As figure 6.5 conveys, the application has validated the email address. Because the input was incorrect we have a nice informative message stating that the email address entered was not valid.

6.5 Compatibility Testing

For the purpose of this test suite, we used different browsers to make sure that the application functions properly in every single one of them. We decided to use the 5 most used web browsers [46].

- Chrome
- Safari
- Firefox
- Edge
- Opera

The outcome of this test was not surprising at all. ASP.NET core was supposed to work on these browsers as long as we do not mess up anything with the JavaScript, CSS and the bootstrap which is responsible for the resizing of the window. Indeed it worked properly on every occasion. What was surprising was that Google's Chrome could load the application much faster than the other browsers.

It would have been extremely useful if we were able to test for smartphone compatibility but unfortunately we were unable to test for mobile functionality as this website is still on a local emulator server.

6.6 Crowd-Source Testing

This is an upcoming concept of user testing. Usually, from this concept we are able to uncover many small defects which otherwise would have been unnoticed [11].

These test cases included a single-user testing on the front-end of the application. Users were able to test the responsiveness of the system from an emulated web server. The web application was slowing down when loaded for the first time. Luckily, there was not any other delays in the redirection from one page to another. This was extremely useful because there was no need to fix any client-side code [22].

However, there were a few little problems. For example, in a particular page the resizing function did not work properly. Fortunately, this was an easy fix. Another problem, was that the delete function did not work even though it was working absolutely perfectly before. The fault had to do with pre-populating the database. This was caused only when a new user was added to the database. After quite a lot of trials and errors it was finally fixed. The problem was caused by a particular function which created incomplete entries in the database. Other than that the application function properly.

Chapter 7

Reflection on work done

7.1 Risk Evaluation

Comparing the risks there were some which were unidentified but thanks to the quality management put for this project there was enough time to overcome them and deploy the application.

During the design face of the project we had a prototype timetable page. This page was supposed to contain all the labs in a very nice format as illustrated in figure 8.7. During the implementation stage our main goal was to get a big part of the core functionality before implementing any extra features. Creating the timetable page required a lot of research because there was not any documentation available on such a design.

The project was developed on ASP.NET core 2.2 which was released in December of 2018 [17]. Therefore, it was not really any time at all to release documentation on different features. Luckily, there was plenty of time allocated for exactly this kind of situation. The mitigation action as mentioned earlier was to take more time and research on this topic. This resulted in some solutions that required payment such as syncfusion [41]. Syncfusion provides a variety of user interface solutions but in an excessive price.

The Second step was to built it. After a while we managed to have a working example. The only thing left was to integrate it with the project. Our timetable was written on bootstrap 4 and ASP.NET core uses bootstrap 3. Because of that, the integration was not possible.

After many trials and errors, we decided to have a simple but very well organised user interface instead of a calendar view of the application. In the future it would be a really nice addition.

This would not have been possible if it was not of the really good and effective planning that basically provided a time frame to succeed or find another solution to this objective.

7.2 Timeline Evaluation

The projects proposed timeline differ a lot from the actual timeline. However, all tasks were completed in time but not necessarily with the proposed order. Additionally, a few tasks arose during the implementation phase and were added to the final timeline 7.1. All the tasks listed in the project schedule were completed.

Initially, there was not any dissertation write-up in the timeline. During the progression of the project it became obvious that the dissertation will be written in small parts in parallel of the project implementation. This was included in the final timeline. Allocating extra time in the initial schedule was proven to be of great aid to the whole process.

Quality time management was a key feature of this project. Good organisation of the various tasks was vital for this project completion. Even though a few activities were not completed on time, such as the timetable mentioned above. However, in the end, there was sufficient time to continue as planned and finish the project in time.

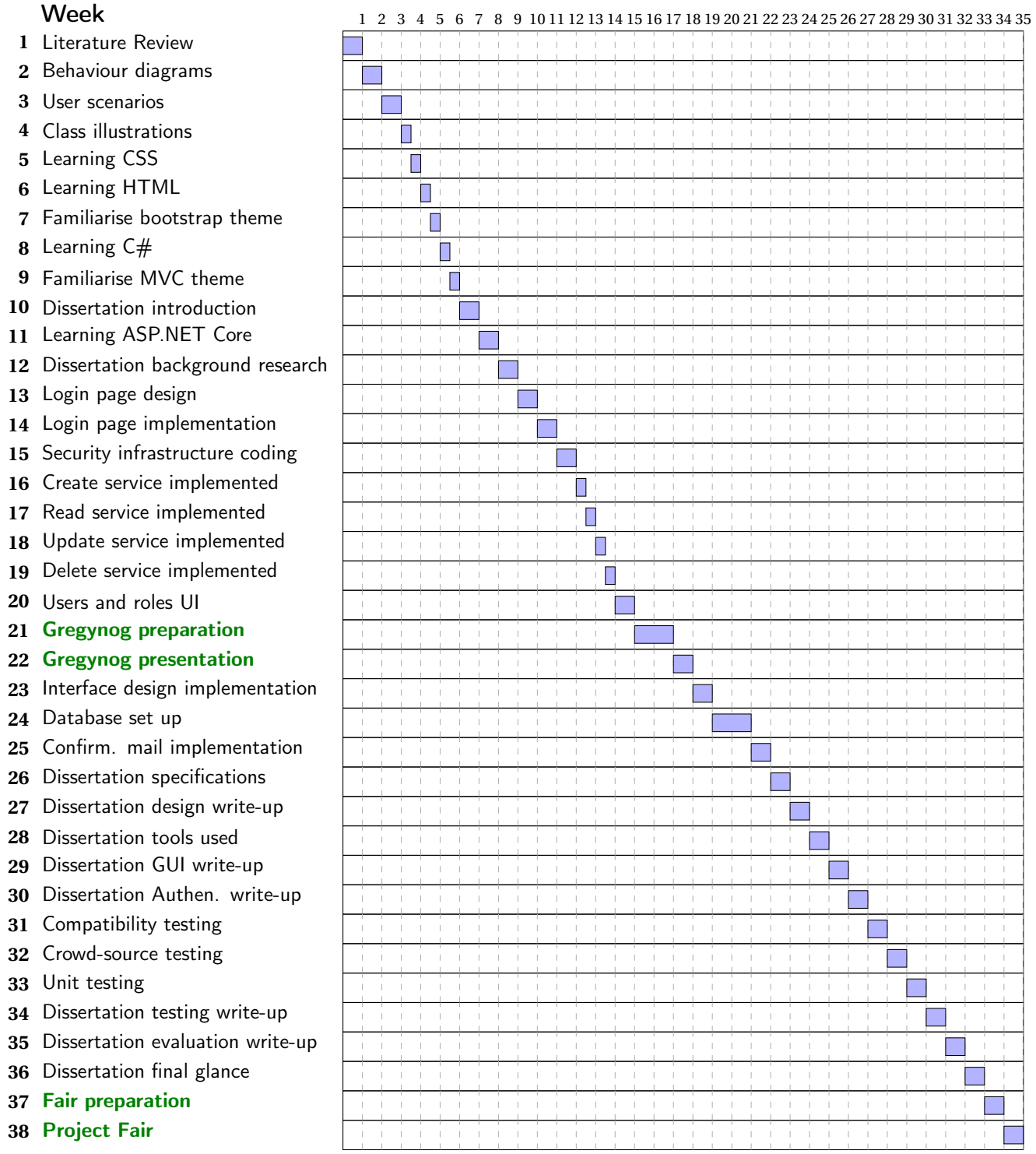


Figure 7.1: Actual timeline which keeps track of the dissertation and implementation phase.

7.3 Functionality

This project aimed at producing a web application where students can apply for a module of their choice. By completing the project and delivering a functional web application, our goal was fulfilled.

Table 7.1 provides a list of all the objectives and specifies which of those were completed or not. These objectives were covered in section 4 and rigorously discussed in section 5.

Only two objectives were not completed and are presented with an X mark. Those did not tamper with the back-end of the application which is fully functioning. Instead they were extra features targeted to support the front-end user.

The implementation of the timetable user interface was linked with the implementation of selenium. Even if selenium was implemented it would not serve any purpose because without the timetable the data collected by selenium would have been useless. Therefore, not implementing the timetable user interface impact the implementation of selenium. Given this reason, we can actually understand why at the end two of the objective were not fulfilled.

Objective	Fulfilled
CRUD Modules	✓
CRUD Roles	✓
CRUD Users	✓
Store Modules	✓
Store Roles	✓
Store Users	✓
Store lab demonstrators	✓
Automatically seed database	✓
Implement Selenium	X
Email confirmation	✓
Assign user roles	✓
Remove user roles	✓
Different levels of authorisation	✓
Claim authorisation system	✓

Objective	Fulfilled
Timetable GUI	X
Apply for a Module	✓
View Applicant	✓
Delete Applicant	✓
PhD constraint	✓
Max demonstrators constraint	✓
Version Control	✓
MVC pattern	✓
Repository Pattern	✓
Cross site scripting	✓
Cross site forgery request	✓
Over posting	✓
Cookie theft	✓
Error reporting	✓

Table 7.1: Table listing all objective and stating the outcome of them.

Hence, core features were prioritised and as result we have the table 7.1 listed above. Additionally, the unfulfilled objectives could be great for future work as mentioned in the next section.

7.4 Areas for expansion and improvement

It is the developer's hope that the development of this project will continue and on the next demonstrators allocation occurring during this summer it will proved to be a useful addition.

A web application always has some area of expansion and improvement, because new framework editions and constant updates improve the current technology and introduce new features which could translate in endless possibilities.

This application would greatly benefit from a mobile application. Giving the opportunity of a wider access infrastructure will simplify even more this process.

Moreover, this program could benefit from an automatic tool such as selenium, which could take necessary data to create labs automatically.Consequently , providing an even more simplified version of the current application.

The user interface could be improved by providing a calendar view of the available labs. This would simplify even more the look and feel while supplying crucial information to the user with a quick glance.

7.5 Conclusion

This final year project report focuses on researching the current demonstrator allocation procedure to develop a web application. A brief background of the WWW was highlighted together with our history's events and accomplishments. Several related technologies have been meticulously compared and assessed. Through the comparison process we established our requirements, aims and objectives. We designed several UML diagrams to give us a clear insight of the different uses of the system. Finally, system development was straight forward as we dealt with a typical administrative problem. With testing, we verified and validated that the web application meets the stated requirements and specifications. Time management took place as an integrated and effective project element. Risks were assessed and assisted in completing the project without significant interruption in the work-flow of development.

Consequently, the main focus of the project laid on selecting the best possible technology and the delivery of a quality assurance. Thereupon, our purpose was fulfilled by producing a utilitarian web application to allocate demonstrators in a hassle-free and user-friendly environment.

Finally, upon extensive study and implementation of the project it was doable to list a number of conclusions. The programming infrastructure is very flexible allowing rapid software development. Except from the idea of the MVC system structure, the rest of the programming languages were simple and easy to use and understand. Last but not least, visual studio, ASP.NET core and C# are a great modern option for developing web applications.

Bibliography

[1] Ghahrai A. Why do we test? What is the Purpose of software testing?, 2018. https://www.testingexcellence.com/why-do-we-test-what-is-the-purpose-of-software-testing/ [Accessed: May 2019].

[2] McPeak A. A Brief history of web browsers and how they work, 2018. https://crossbrowsertesting.com/blog/test-automation/history-of-web-browsers/ [Accessed: May 2019].

[3] Weyers B., Burkolter D., Kluge A., and Luther W. User-centered interface reconfiguration for error reduction in HCI. In *Third international conference on advances in human-oriented and personalized mechanisms, technologies and services*, pages 52–55, 2010.

[4] Reid S. C. An empirical analysis of equivalence partitioning, boundary value analysis and random testing. In *Proceedings fourth international software metrics symposium*, pages 64–73, 1997.

[5] Stephen A. S. C. Dayforce HCM, 2016. https://www.softwareadvice.com/hr/dayforce-hcm-profile/ [Accessed: May 2019].

[6] Bricklin D. VisiCalc: Information from its creators, 1999. http://www.bricklin.com/visicalc.htm [Accessed: May 2019].

[7] Mello R. M. d., Pereira W. M., and G. H. Travassos. Activity diagram inspection on requirements specification. In *Brazilian symposium on software engineering*, pages 168–177, 2010.

[8] Spinellis D. Version control systems. *IEEE Software*, 22(5):108–109, 2005.

[9] Zhang D., Wei Z., and Yang Y. Research on lightweight MVC framework based on spring MVC and mybatis. In *Sixth international symposium on computational intelligence and design*, volume 1, pages 350–353, 2013.

[10] Anunciacion E. Building a risk assessment matrix, 2016. https://www.workiva.com/blog/building-risk-assessment-matrix, [Accessed: May 2019].

[11] Dolstra E., Vliegendhart R., and Pouwelse J. Crowdsourcing GUI tests. In *IEEE Sixth international conference on software testing, verification and validation*, pages 332–341, 2013.

[12] Kemnitz G. Which is the most popular database language?, 2017. https://www.quora.com/Which-is-the-most-popular-database-language [Accessed: March 2019].

[13] Roy G. G. A risk management framework for software engineering practice. In *Australian software engineering conference*, pages 60–67, 2004.

[14] Google. Email server limitations, 2019. https://support.google.com/mail/answer/22839?hl=en [Accessed: May 2019].

[15] National instruments. Prove it works: using the unit test framework for software testing and calidation, 2019. https://quickdraw.withgoogle.com/ [Accessed: May 2019].

[16] Draw IO. Quick start on draw IO, 2018. https://quickdraw.withgoogle.com/ [Accessed: May 2019].

[17] Damian J. Announcing ASP.NET core 2.2, 2018. https://devblogs.microsoft.com/aspnet/asp-net-core-2-2-available-today/ [Accessed: May 2019].

[18] Emspak J. and Zimmermann K. A. Internet history timeline, 2019. https://quickdraw.withgoogle.com/ [Accessed: May 2019].

[19] Hostmark J. Kronos workforce central software, 2017. https://www.softwareadvice.com/uk/hr/kronos-workforce-central-profile/ [Accessed: May 2019].

[20] Hurst J. Fifty killer apps that greatly boost your quality of life, 2018. https://www.lifehack.org/articles/technology/50-killer-apps-that-greatly-boost-your-quality-life.html [Accessed: May 2019].

[21] McCluskey E. J. Verification testing. In *19th Design automation conference*, pages 495–500, 1982.

[22] Sonmez J. *Common types of software testing*, 2013. https://usersnap.com/blog/software-testing-basics/ [Accessed: May 2019].

[23] LaCroix K., Loo Y. L., and Choi Y. B. Cookies and sessions: a study of what they are, how they work and how they can be stolen. In *International conference on software security and assurance*, pages 20–24, 2017.

[24] Strickler K. Top ten website trends, 2016. https://www.commonplaces.com/blog/10-website-trends-for-2016 [Accessed: March 2019].

[25] Isaacs M. Five key software testing steps every engineer should perform, 2015. https://techbeacon.com/app-dev-testing/5-key-software-testing-steps-every-engineer-should-perform [Accessed: May 2019].

[26] Jailia M., Kumar A., Agarwal M., and Sinha I. Behavior of MVC based web application developed in PHP and .NET framework. In *International conference on ICT in business industry government*, pages 1–5, 2016.

[27] Mohammad M. SAP Success factors, 2017. https://www.softwareadvice.com/hr/successfactors-software-profile/ [Accessed: May 2019].

[28] Toyoda M. and Kitsuregawa M. The history of web archiving. *Proceedings of the IEEE*, 100(Special Centennial Issue):1441–1443, 2012.

[29] Microsoft. Visual studio framework, 2019. https://visualstudio.microsoft.com/ [Accessed: May 2019].

[30] Kumar N. Why is software testing necessary?, 2016. https://www.utest.com/articles/why-is-software-testing-necessary [Accessed: April 2019].

[31] Finance online. Zoho people, 2018. https://reviews.financesonline.com/p/zoho-people/ [Accessed: May 2019].

[32] Indumathi C. P. and Begum A. S. Web database testing using ER diagram and state transition model. In *International conference on communication and signal processing*, pages 1937–1942, 2016.

[33] Marvin R., Michelle R., Martinez J., and Rafter V. BambooHR, 2019. https://uk.pcmag.com/cloud-services/76322/bamboohr [Accessed: May 2019].

[34] Vogels R. A six step guide to web application testing, 2013. https://usersnap.com/blog/web-application-testing/ [Accessed: May 2019].

[35] Chacon S. and Straub B. *PRO GIT Everything you need to know*, 2014. https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control [Accessed: May 2019].

[36] Levy S. A spreadsheet way of knowledge, 2014. https://www.wired.com/2014/10/a-spreadsheet-way-of-knowledge/ [Accessed: May 2019].

[37] Mahajan S. and Halfond W. G. J. WebSee: A tool for debugging HTML presentation failures. In *IEEE 8th International conference on software testing, verification and validation*, pages 1–8, 2015.

[38] Murugesan S., Rossi G., Wilbanks L., and Djavanshir R. The future of web apps. *IT Professional*, pages 12–14, 2011.

[39] Roy Choudhary S., Versee H., and Orso A. A cross-browser web application testing tool. In *IEEE International conference on software maintenance*, pages 1–6, 2010.

[40] Sengupta S. and Bhattacharya S. Formalization of UML use case diagram notation based approach. In *International conference on computing informatics*, pages 1–6, 2006.

[41] SyncFusion. The only UI component suite that you will ever need, 2019. https://www.syncfusion.com/ [Accessed: May 2019].

[42] Dorsey T. 18 new tools and extensions for visual studio, 2018. https://visualstudiomagazine.com/articles/2018/06/21/visual-studio-toolbox.aspx [Accessed: May 2019].

[43] May T. The beginner's guide to flat design, 2018. https://www.creativebloq.com/graphic-design/what-flat-design-3132112 [Accessed: May 2019].

[44] Rutter T. Validate your input!, 2005. https://www.sitepoint.com/validate-your-input/ [Accessed: May 2019].

[45] Software testing. Web application testing complete guide, 2018. https://www.softwaretestinghelp.com/web-application-testing/ [Accessed: May 2019].

[46] W3Counter. Browser & platform market share, 2013. https://www.w3counter.com/globalstats.php [Accessed: May 2019].

Chapter 8

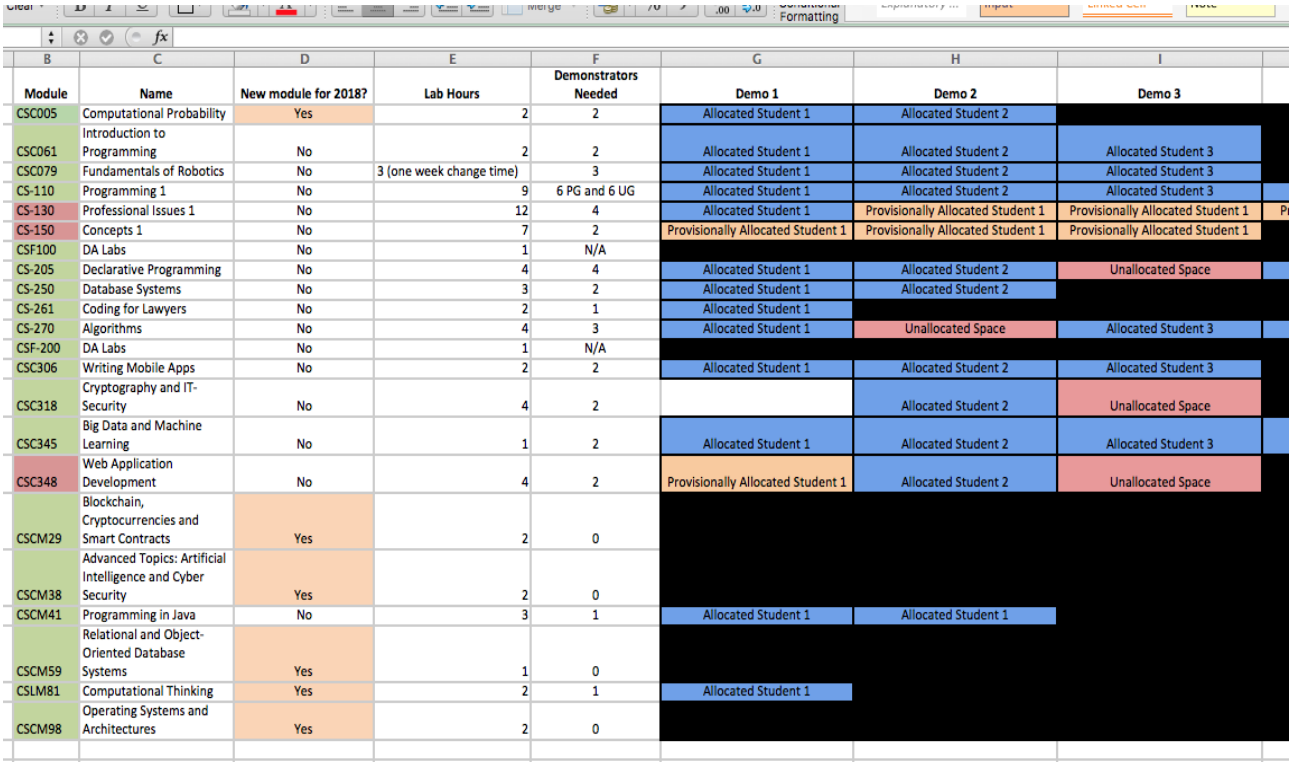
Appendix

8.1 Appendix A (Introduction)

8.1.1 Procedure in excel

Related work-spredsheets **Date: 23/10/2018**

Procedure With Spreadsheets



Module	Name	New module for 2018?	Lab Hours	Demonstrators Needed	Demo 1	Demo 2	Demo 3
CSC005	Computational Probability	Yes	2	2	Allocated Student 1	Allocated Student 2	
CSC061	Introduction to Programming	No	2	2	Allocated Student 1	Allocated Student 2	Allocated Student 3
CSC079	Fundamentals of Robotics	No	3 (one week change time)	3	Allocated Student 1	Allocated Student 2	Allocated Student 3
CS-110	Programming 1	No	9	6 PG and 6 UG	Allocated Student 1	Allocated Student 2	Allocated Student 3
CS-130	Professional Issues 1	No	12	4	Allocated Student 1	Provisionally Allocated Student 1	Provisionally Allocated Student 1
CS-150	Concepts 1	No	7	2	Provisionally Allocated Student 1	Provisionally Allocated Student 1	Provisionally Allocated Student 1
CSF100	DA Labs	No	1	N/A			
CS-205	Declarative Programming	No	4	4	Allocated Student 1	Allocated Student 2	Unallocated Space
CS-250	Database Systems	No	3	2	Allocated Student 1	Allocated Student 2	
CS-261	Coding for Lawyers	No	2	1	Allocated Student 1		
CS-270	Algorithms	No	4	3	Allocated Student 1	Unallocated Space	Allocated Student 3
CSF-200	DA Labs	No	1	N/A			
CSC306	Writing Mobile Apps	No	2	2	Allocated Student 1	Allocated Student 2	Allocated Student 3
CSC318	Cryptography and IT-Security	No	4	2		Allocated Student 2	Unallocated Space
CSC345	Big Data and Machine Learning	No	1	2	Allocated Student 1	Allocated Student 2	Allocated Student 3
CSC348	Web Application Development	No	4	2	Provisionally Allocated Student 1	Allocated Student 2	Unallocated Space
CSCM29	Blockchain, Cryptocurrencies and Smart Contracts	Yes	2	0			
CSCM38	Advanced Topics: Artificial Intelligence and Cyber Security	Yes	2	0			
CSCM41	Programming in Java	No	3	1	Allocated Student 1	Allocated Student 1	
CSCM59	Relational and Object-Oriented Database Systems	Yes	1	0			
CSLM81	Computational Thinking	Yes	2	1	Allocated Student 1		
CSCM98	Operating Systems and Architectures	Yes	2	0			

Figure 8.1: Spreadsheets used to allocate demonstrators in different labs. A higher resolution figure can be found here

Description:

The spreadsheets are used to accomplished the constraints which are required. Such as the study year and the maximum number of students that can be assigned to that particular module. It is clearly conveyed with the spreadsheets above that it requires a great deal of effort. The process of finding demonstrators begins after these are established. A time-consuming and complex operation as shown in the figures below.

Student Calendar view **Date: 23/10/2018**

Student's Timetable

	Monday 22	Tuesday 23	Wednesday 24	Thursday 25	Friday 26
9:00					**CSCM45** XX Computational Foundry CF104 (PC Lab) (120)
10:00	**CSCM29** AB, JJR, AGS 2 hours Computational Foundry CF204 (Linux Lab)			**CSCM98** BM 2 hours School of Management SoM128 (PC Lab) (60)	
11:00	**CSCM29** AB, JJR, AGS 2 hours Computational Foundry CF204 (Linux Lab)			**CSCM98** BM 2 hours School of Management SoM128 (PC Lab) (60)	
12:00				**CSCM18** PK 2 hours Computational Foundry CF203 (Windows Lab)	
13:00				**CSCM18** PK 2 hours Computational Foundry CF203 (Windows Lab)	**CSCM59** CJW Computational Foundry CF104 (PC Lab) (120)
14:00					
15:00	**CSLM81** AZW 2 hours Vivian 731/2 PC Lab				
16:00	**CSLM81** AZW 2 hours Vivian 731/2 PC Lab	**CSCM38** JHS 2 hours Computational Foundry CF203 (Windows Lab)			
17:00		**CSCM38** JHS 2 hours Computational Foundry CF203 (Windows Lab)			

Figure 8.2: Example of a student's timetable. A higher resolution figure can be found here

Description:

While finding out when the student is available, the timetable must be taken into consideration. After somebody has contacted the person in charge and has shown interest in lab demonstrations, the administrator will take a look at the schedule as conveyed above to determine if there are any clashes.

8.2 Appendix B Background Research

8.2.1 Proposed Alternative Solutions

Alternative solutions **Date: 23/10/2018**

Table of Alternative Systems

Alternatives	Advantages	Disadvantages	Verdict	Android/Apple App
BambooHR	Covers the basics and then some, with a well-organized, clean design. Easy to set up. Open API makes it easy to integrate with existing human resource tech vendors.	Priced higher than many competitors. Doesn't include administration benefits offered by rivals.	Is not the cheapest option but is well worth it. A well-organized, visually appealing tool that is simple to set up and run. Open API allows the software to be integrated with a company's existing HR tech vendors.	✗
Zoho People	Can be customized in several areas including custom forms and fields. Custom modules can also be created and assigned to parent categories within the user interface.	Both mobile and web page applications need to be simplified. Small technical issues.	A very cost effective and versatile system overall. Very user friendly to create online forums. It saves lots of valuable time for feeling time sheets in spreadsheets. All these features are done here dynamically.	✓
Dayforce HCM	Beautiful appearance. and very Intuitive user interface that is easy to use for people with various technical abilities. Easily view task progress across devices	A few technical issues especially with HTML.	It is highly configurable. It handles changes due to union contracts with ease. However, the new HTML timesheets have some quirks.	✓
SAP Success Factors	It can manage multiple Human Resource related processes like performance Management and can be used as a Learning Management System.	Confusing workflow or too much information which ends up being more complicated.	Unintuitive design. And absence of a mobile platform does not make this software appeal to everyone.	✗
Kronos Workforce Central	Very flexible system which is easy to learn and mange both from user and manager perspective.	Very expensive product. System speed sometimes possess an issue.	Simple and accurate product. However, is best suited for large, multi-location companies.	✗

Figure 8.3: A table comparing advantages and disadvantages of alternative tools.

Description:

In practise there are numerous software solutions that could provide assistance. With that said, we have compiled a list of advantages and disadvantages that give us a more in depth view on those software systems on table 2.1.

Even though some of this HR solutions are great, they are not what we are after for. This solutions combined a great deal of software which is great for some companies but in our case it would just made our solution more complicated. For that reason and after a lot of consideration we knew exactly what was required. All the solutions which were compared and discussed in table 2.1 had some very good components. Therefore we used them as a very good example on which our own solution was based.

8.3 Appendix C (Specification)

8.3.1 Proposed Risk Analysis

Proposed Risk Analysis **Date: 23/10/2018**

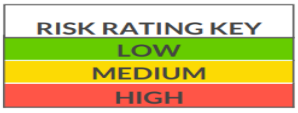


Figure 8.4: A table presenting risk rating.

Table of risks

		RISK ELEMENTS		OVERALL
REFERENCE	POTENTIAL RISKS	IMPACT	PROBABILITY	RISK RATING
GE				
GE1	Stress/Family Issues	L	L	L
GE2	Being Sick	L	L	L
GE3	Underestimate Time	H	L	M
PS				
PS1	Software/Hardware Failure	M	L	M
PS2	Requirements Interpreted Wrongly	H	L	M
PS3	Wrong Software Model	L	L	L
PS4	Implementation phase takes more time	M	M	M
PS5	New Material Slow Me Down	M	M	M
PS6	Integration with University Database	H	M	H
PS7	Malicious Attack	M	L	M
PS8	Broken Access Control	M	L	M
PS9	Future Website Compatibility	M	M	M

Figure 8.5: A table presenting all risks their impact, probability risk rating and their mitigation action.

Description:

Each project entails risks. These risks vary greatly in software development. Although, we should follow a typical procedure should be followed regardless of what the risk is [13].

Each project entails risks. These dangers vary greatly in software engineering. However, a typical procedure should be carried out regardless of the risk [13]. It is therefore essential to have a schedule, restrict, mitigate, monitor, and minimise the risk or, if required, produce an alternative solution for the successful completion of the project [10].

8.4 Appendix D (Design)

8.4.1 Prototype Mock-ups

Prototype: Login Page **Date: 23/10/2018**

Log in Page

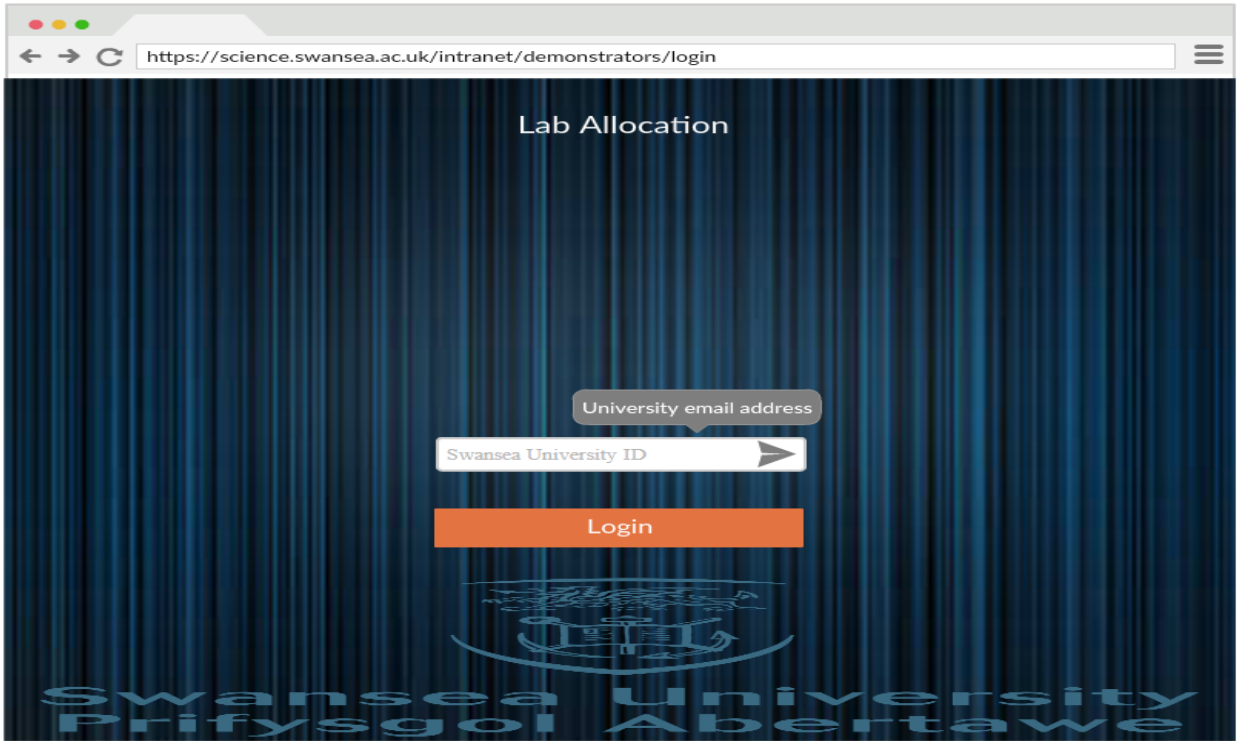


Figure 8.6: Log in Page

Description:

A Member **login page** is essential because all relevant information must be acquired by the application and all necessary restrictions applied.In the figure above, an approximation is shown. That demonstrates that if credentials are required is easy to understand where they are needed. Because there is only one text container that demands data.

Prototype: Time-table Page **Date: 23/10/2018**

Time-table Page

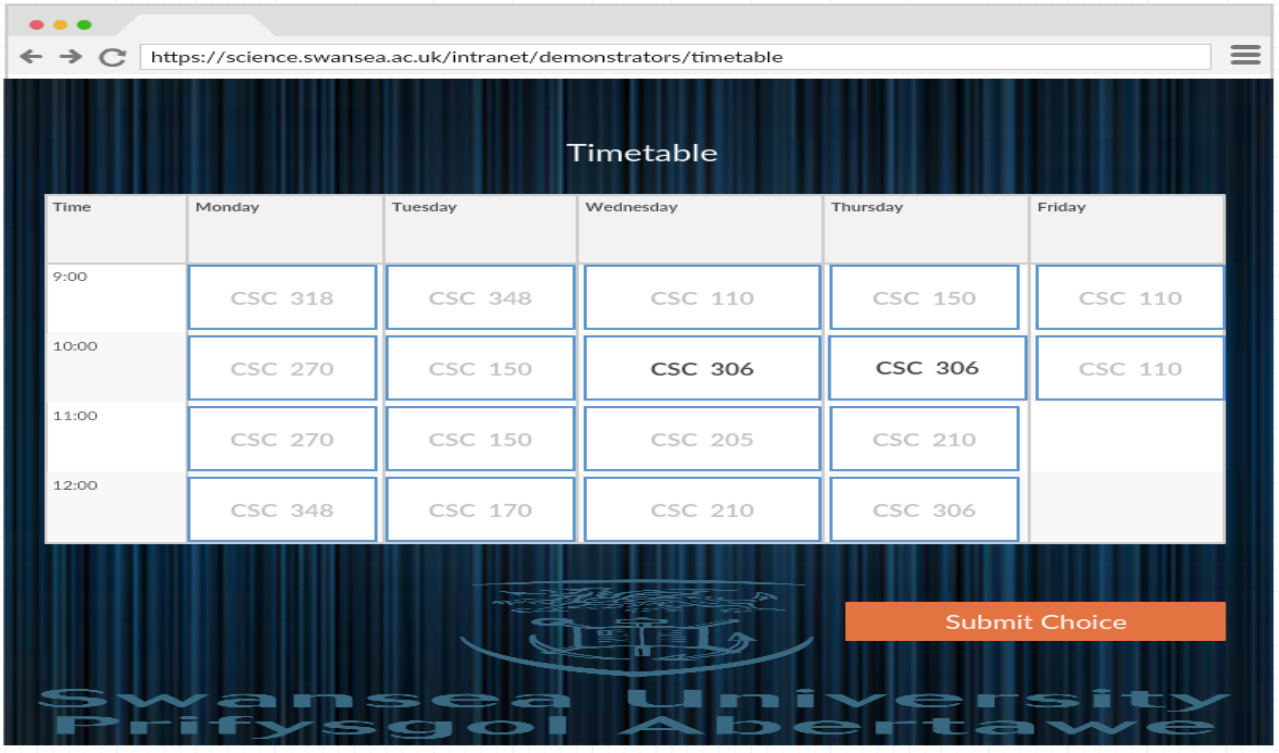


Figure 8.7: Time-table of modules

Description:

Onto the right of this page, we can distinguish different buttons. Those represent a module. The following screenshot portrays the outcome of that button been pressed.

This was one of the first designs in visual studio. The first step of the whole application was to set up the the log in page as shown in 4.4. The module controller explained in listing 4.3.1 was vital for the correct function of this page.

Prototype: Timetable Page **Date: 23/10/2018**

Back-End Page

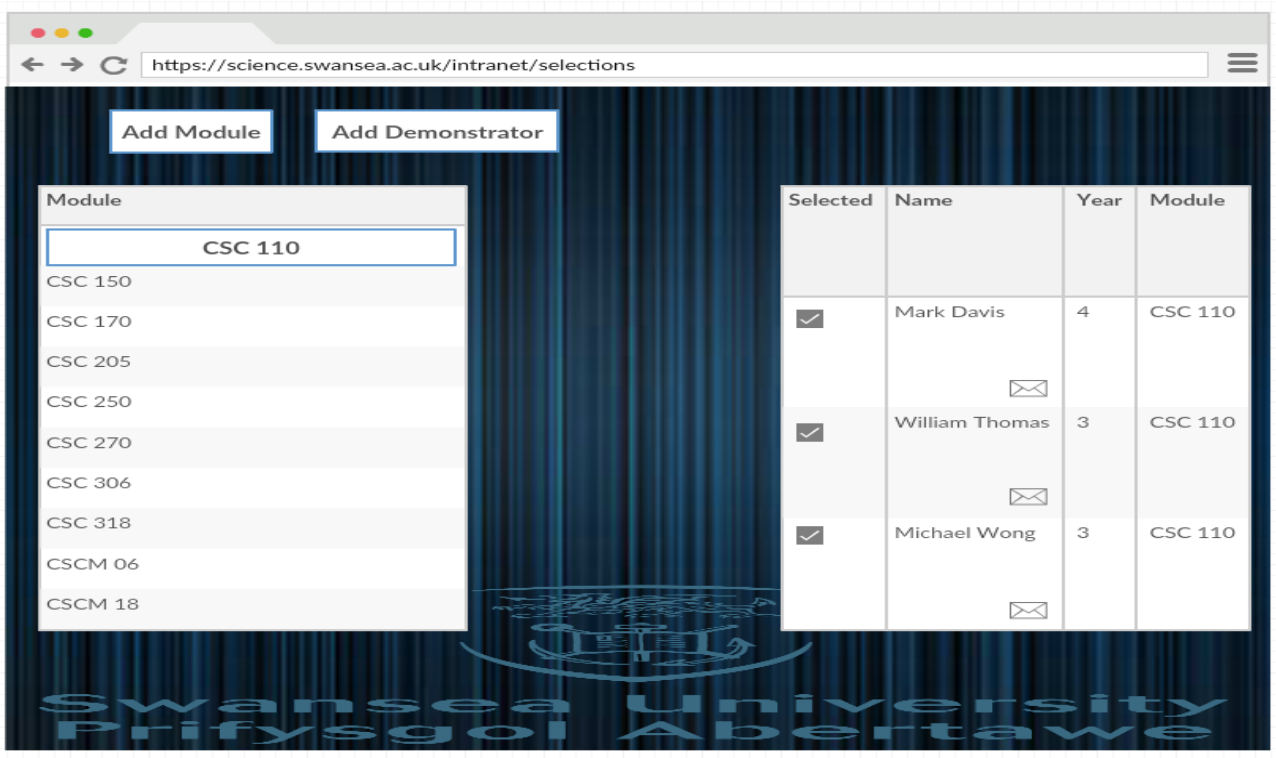


Figure 8.8: Module selection

Description:

When the applicants button is pressed, we are presented with a list of all the students who have already signed up for this lab demonstration. The admin has the ability to remove or add a demonstrator.

The purpose of the application is to allow easy manipulation of data. Additionally, to assist in this procedure as much as possible. For that reason, the program recognises the maximum number of demonstrators and provides the user with an adequate informative message.

For instance, if a lab has already reached the maximum number of demonstrators it will not be included in the list of available labs. Thus, simplifying the process and reducing the possibility of mistakes or errors.

Prototype: Back-Page **Date: 23/10/2018**

Demonstrator Form

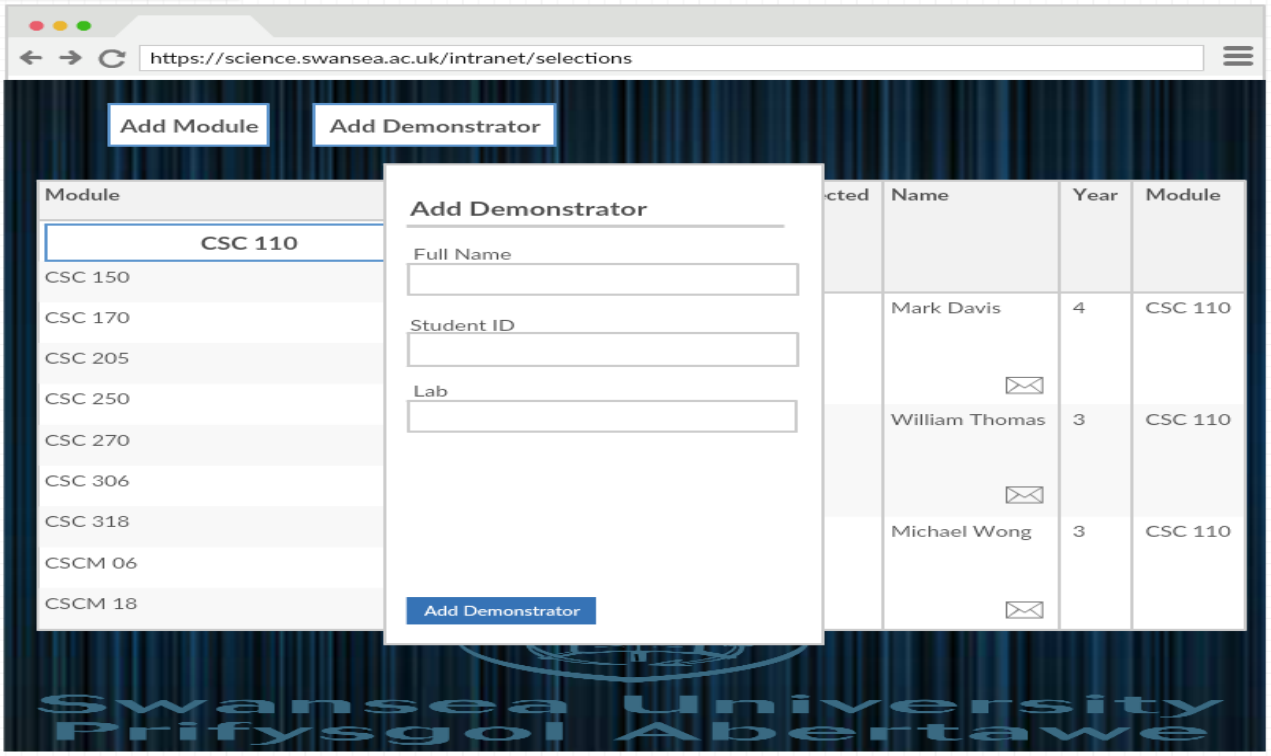


Figure 8.9: Adding a Demonstrator

Description:

Last but not least, in the figure 4.12 we are able to observe the procedure in which a student is able to apply for a lab. After a user has logged in, gets displayed a list with all the available labs that are suitable for him/her.

Next to each selection there is a button to apply(red circled button in figure 4.12. When the button is pressed, the user gets redirected to a new page

This page is similar to the same page shown in figure 4.6. The only difference has to do with the authorisation levels. For instance, the delete button is not shown because we do not want a student to be able to delete an applicant.

Prototype: Email Confirmation Page **Date: 23/10/2018**

Email Confirmation

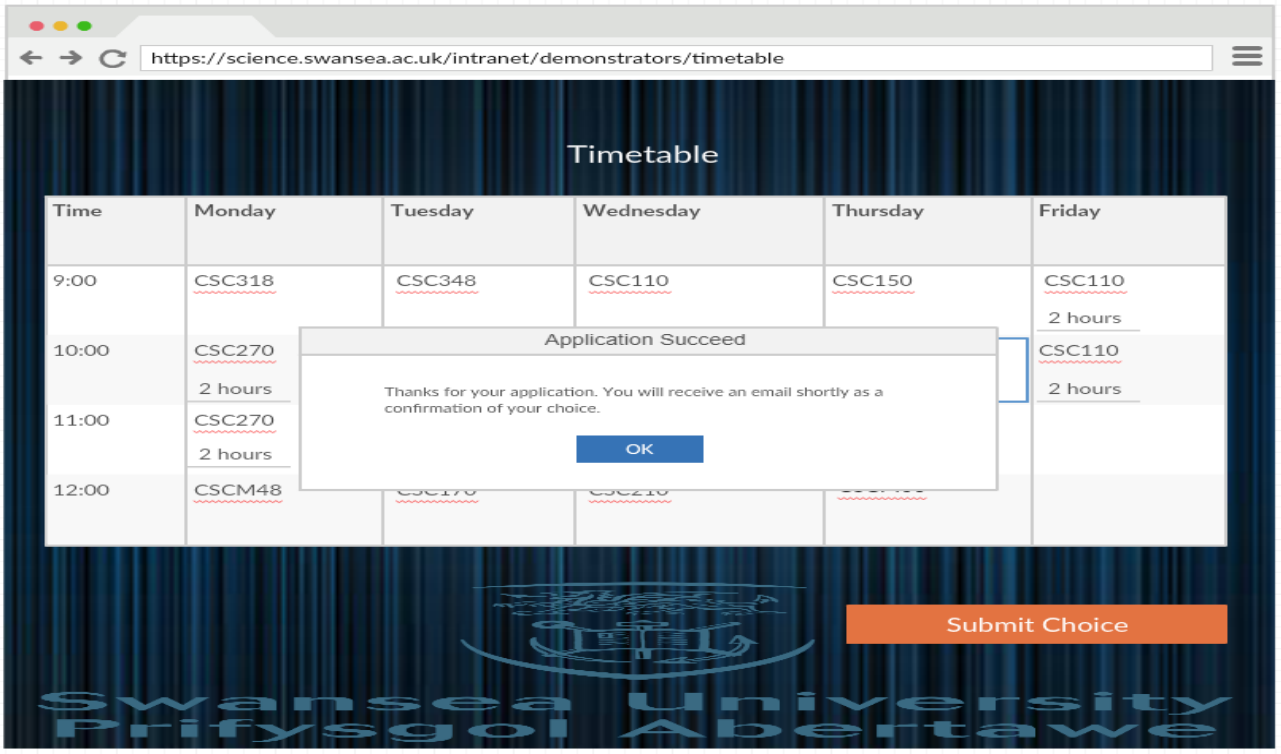


Figure 8.10: Informative Pop-Up

Description:

The procedure of sending an email in visual studio ASP.Net Core is kind of complicated but doable. Firstly, we had to set up a new class called DevEmailSender. This class includes the credentials of an email address, the port used, if we want SSL to be enable and the client server we are going to use. Luckily, google allowed us to use their email client with a trial account. There was a limitation of 500 emails per day which did not pose a problem for us.

Prototype: Email Confirmation Page **Date: 23/10/2018**

Choice Confirmation

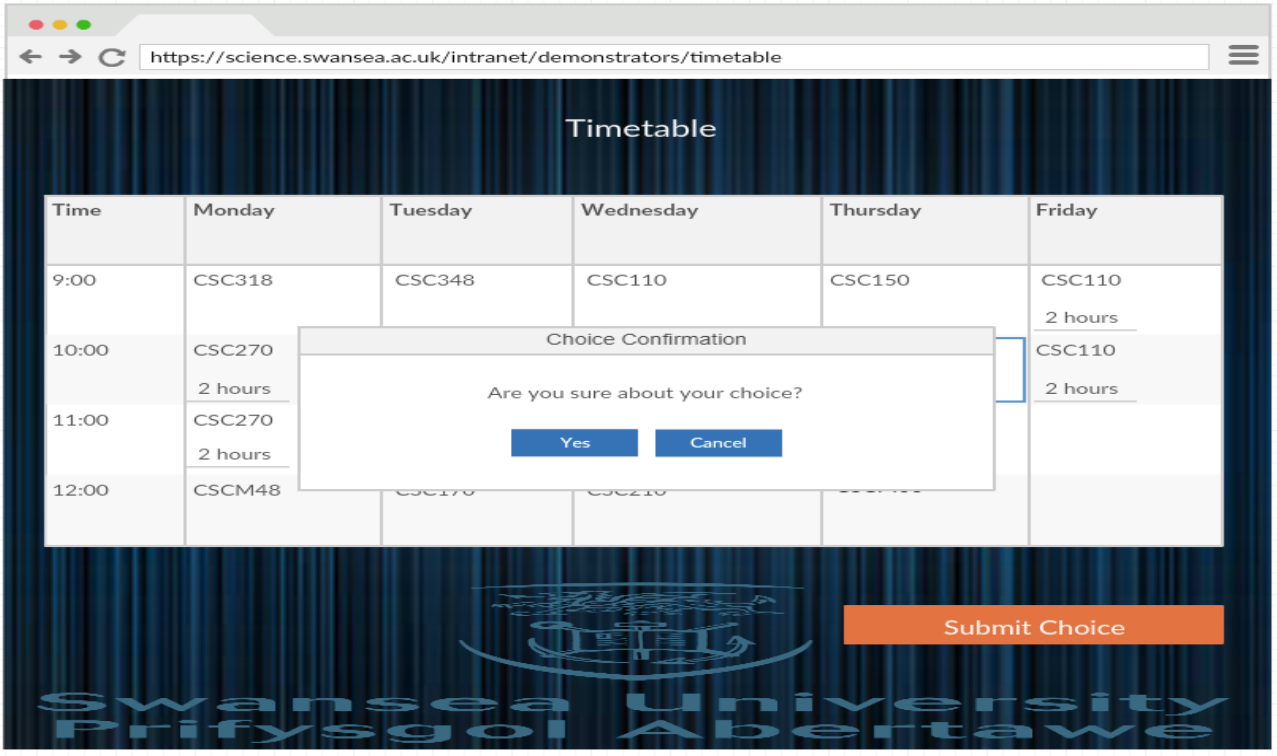


Figure 8.11: Confirmation prompt

Description:

This **confirmation email** conveys the process of which a demonstrator has already applied for a module. In figure 8.11 a confirmation box is shown.

This was the initial design to confirm the user choice. At the end we change it and we implemented a an email confirmation which is much better as the user receives an email directly to their account..

8.5 Appendix E (Implementation)

8.5.1 Module Controller Code

Module Controller Implementation **Date: 19/01/2019**

```
1
2  namespace DemonstratorApp.Controllers
3  {    [Authorize]
4
5          // POST: Module/Demonstrators
6          public Task < IActionResult > Demonstrators([ bind ("ModuleID,Name")]
7          ModuleDetailsViewModel viewModel)
8          {
9              if (ModelState)
10             {
11                 ApplicationUser user = new ApplicationUser
12                 {
13                     Name = viewModel.Name,
14                     Email = viewModel.Name,
15                     UserName = viewModel.Name
16
17                 };
18
19                 Module module = await _context.Module
20                 .FirstOrDefaultAsync(m => m.ModuleId == viewModel.ModuleID);
21
22                 List<ApplicationUser> applicationUsers = await _context.ApplicationUser.Where
23                     (m => m.MyModule == module).ToListAsync();
24                 // Set the list size
25                 module.CurrentApplicants = applicationUsers.Count();
26
27                 module.CurrentApplicants++;
28
29                 if (module == null)
30                 {
31                     return NotFound();
32                 }
33
34                 user.MyModule = module;
35                 _context.ApplicationUser.Add(user);
36                 _context.Module.Update(module);
37                 await _context.SaveChangesAsync();
38
39                 viewModel = new ModuleDetailsViewModel
```

```
40                  {
41                      Module = module,
42                      Name = user.Name
43                  };
44
45                  applicationUsers = await _context.ApplicationUser.Where
46                      (m => m.MyModule == module).ToListAsync();
47
48                  Email.DevEmailSender devEmailSender = new Email.DevEmailSender();
49                  await devEmailSender.SendEmailAsync("kuhgyv@gmail.com", "Demonstrator Application Confirmation",
50                      "You have succefully applied to demonstrate: \r\n" + "Module Name: " + module.ModuleName+ "\r\n"
51                      + "Module Code: " + module.ModuleCode+ "\r\n" + "Day and Time: "+module.StartDateTime+ "\r\n");
52
53                  viewModel.ApplicationUsers = applicationUsers;
54              }
55              return View(viewModel);
56          }
```

Listing 8.1: All the code used to create the module controller

Description:

This class has the more attributes and methods as it contains the most vital information's of the system. We have methods such as the task of applying for a module or methods to provide students the most relevant labs according to their year of study, available labs, PhD restrictions, etc.

8.6 Appendix F (Testing)

8.6.1 Boundary Value Testing

Boundary Testing **Date: 10/04/2019**

Input limitations

```
10      public class Module
11      {
12          [Key, Required, ScaffoldColumn(false)]
13          public int ModuleId { get; set; }
14
15          [MinLength(6), Required]
16          [Display(Name = "Code")]
17          public string ModuleCode { get; set; }
18
19          [MinLength(2), Required]
20          [Display(Name = "Name")]
21          public string ModuleName { get; set; }
22
23          [ Required]
24          [Display(Name = "Total Lab Hours")]
25          public int Hours { get; set; }
26
27          [Required]
28          [Display(Name = "Year")]
29          public int Year { get; set; }
30
31          [Required]
32          [Display(Name = "Demonstrators Needed")]
33          public int MaxDemonstrators { get; set; }
34
35          [Required]
36          [Display(Name = "Duration")]
37          public int Duration { get; set; }
38
39          [Required]
40          [Display(Name = "Start Date and Time")]
41          public DateTime StartDateTime { get; set; }
42
43          [ScaffoldColumn(false)]
44          public int CurrentApplicants { get; set; }
45
46          [NotMapped]
47          [ScaffoldColumn(false)]
48          public List<String> Applicants { get; set; }
49
50
51      }
52  }
53
```

Figure 8.12: Another example of limitations applied on input for the module controller.

Description:

Boundary testing is used to validated user input. This is done by taking values below and above from what is considered valid input, marginally valid input and not applicable input[4]. Fortunately, there was no need for boundary testing because every value entered in the system is checked and then is added in the database.

Figure 6.4 clearly displays the limitations added to check the input values. Firstly, the module code and name need to submitted and cannot be left blank. Additionally the module code needs to be at least 6 characters and the module name must be at least 2.