Alex Tuersley Project Report.docx

by Alexander TUERSLEY

Submission date: 10-May-2021 01:30PM (UTC+0100)
Submission ID: 152495246
File name: Alex_Tuersley_Project_Report.docx (4.56M)
Word count: 23578
Character count: 117903

Student/Lecturer Booking System

KV6003: INDIVIDUAL COMPUTING PROJECT

ALEXANDER.TUERSLEY

STUDENT ID: W1708264

COURSE: COMPUTER NETWORKS AND CYBER SECURITY

PROJECT TUTOR: NICK DALTON

SECOND MARKER: NANLIN JIN

DECLARATIONS

I declare the following:

1) The materials contained in this dissertation are the product of my own work and any outside sources are stated within the references
2) The word count of this dissertation is 15,520 excluding appendices, and references
3) I have read the Northumbria University/Engineering and Environment Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been addressed within this research.

ACKNOWLEDGEMENTS

- My first acknowledgement is to my supervisor, Nick Dalton, who advised and guided me through this process.
- My second acknowledgement is to the participants of the usability testing, as without them valuable data could not have been possible and errors within the system may not have been found.

CONTENTS

ABSTRACT

This project aims to create a Web Booking System for students and Staff at Northumbria University. The reason to create this project is the move from class-based learning to online learning, this application could ease the transition. Another reason for this project is to gain experience in building and developing a product for a client, being able to develop existing skills, while also learning new ones. The system was built using PHP, HTML, JavaScript, and CSS, along with numerous JavaScript plugins to make the site dynamic.

1. INTRODUCTION

1.1 AIMS OF THE PROJECT

The aims of the project are:

- To build a Web Application Booking System; that shows University lecturer's timetables which they choose as time slots they are available.
- The booking system will also allow students to see a lecturer's timetable and availability and make a booking with them based on this.

1.2 PROJECT OBJECTIVES

- Create a literature review of the product.
    1. Gather information relating to the product such as calendar plugins and booking systems already in use.
    2. Look at the available JavaScript Calendar Plugins and the JavaScript Frameworks that are used with them.
    3. Find existing booking systems, identify their good and bad features, some of which could be implemented in the product.
- Create Diagrams and System prototypes
    1. Use Case Diagram
    2. Class diagram
    3. Database Design (Entity Relationship Diagram)
    4. Wireframe
    5. Navigation Diagram
    6. Flow Charts
- Create Testing Plan
- Identify requirements for product (Calendar plugin, JavaScript Framework, jQuery version)
- Create product based on specification.
- Test products
    1. Functional Testing
    2. Usability testing (students, staff, admin)
    3. Cross browser testing
    4. Vulnerability testing
- Make changes to product based on testing feedback and conclusions
- Evaluate the product
    1. Build Quality
    2. Testing
    3. Review the tools and techniques used, and state whether another approach could be used
    4. Review of the literature and how it was used in the product
    5. User Feedback
- Evaluate the project process and own performance
- Write Introduction and Abstract

1.3 PROJECT OVERVIEW

The project aimed to create a web application for both University students and members of staff. The staff members will be able to set when their availability in the week, along with holidays and the types of meetings a student can book. Students will then be able to find the staff member within their department, select the type of meeting they want to book, select a time slot that is available and make a booking. When the booking is made an email will be sent to both parties, along with an ICS file, so the meeting can be added to their calendar. The application aimed to be as secure as possible, due to it holding sensitive data about members of a University. Authentication was placed on many parts of the system, to prevent unauthorized access.

1.4 CONTEXT AND PROJECT JUSTIFICATION

Universities across the country have moved onto online learning due to the global pandemic. This is one of the main reasons that a booking system for students to book meetings with lecturers is needed. A lecturer will not always have the time in the day to respond to an email and they may even miss seeing it. The use of a Booking System eliminates this problem, as staff can set their availability and students will be able to book a meeting, when they are available. Previous work found of this type was in Turku University in Finland, where a student created an online booking system for the laboratory at their University [1]. My client has requested that there be separate interfaces for students and staff as the staff will be showing their schedule and the student will be making bookings without the need to display their schedule. The system will have a design similar to Northumbria University and permission has been given to use the logo.

1.5 SUMMARY OF TOOLS AND TECHNIQUES

The following techniques were used to build the system:

- PHP – used to build the main parts of the system, integrating HTML inside the PHP files to display data. It is also used to interact with the database, to query, update and delete data.
- JavaScript – used to create the calendar plugin to make bookings and to make the site more dynamic, reducing the need to reload the page using PHP.
- SQL – Used to create a database for the Booking system, that can store user data along with bookings and timeslots available for staff members.
- Bootstrap – Bootstrap was used as a CSS framework to make the design of the system consist throughout while maintaining a professional look.

The Scrum framework was used throughout the project, as it allowed multiple tasks from different parts of the system to be completed at once, while leaving more difficult tasks till later.

2. ANALYSIS

2.1 DEFINE PROBLEM

The problem this project is trying to solve is to create a way for students to book appointments with staff members at a University, without having to exchange emails or wait or the staff member to reply. The solution is to create a web booking system with the time slots a lecturer is available, the student can select a time and book an appointment and add any additional information needed. Previous work on a Booking System specifically for students and staff members at a University has not been conducted before

2.2 COMPETITOR REVIEW

The competitor review will consist of a look at currently available Booking Systems, features of these systems that could be created in the booking system and a review of the security risks associated with the Booking System.

2.2.1 SIMPLYBOOK.ME

One existing booking system that was tested was Simplybook.me[2]. This system creates a whole website for a company which can be fully customised to suit the Company's needs. After signing up the website, it is customised based on how many employees and services the Company has. The terminology used is geared towards businesses more than educational institutions. Each service offered can be customised such as the colour of the service that displays in the calendar and whether group bookings can be made [Appendix 2]. The duration of the service is also set by an Admin [Appendix 3]. The availability of the service is set, so when visitors click on the website it shows the time slots available for each service e.g. 9:00-17:00 [Appendix 4], this feature has been requested by the client in Requirement Specification 3.1 [Appendix 12]. After a visitor to the site makes a booking, they will receive an email to confirm the booking [Appendix 5], this booking can be added to outlook or google calendars as an event. The integration with google and outlook calendars is an optional feature in the requirements specification. The main features that could be implemented in the booking system are the option of group bookings and services to differentiate different types of bookings. One drawback is there is only one admin of the site and if this admin account became compromised the whole site is at risk. The way the system is designed is not for Universities, specifically as the services cannot be unique to each member of staff. The option is not given to add users to the site so every staff member would have the same schedule.

2.2.2 CALENDLY

Another booking system which was tested is Calendly[3]. Calendly works on a User-by-User basis, which would work at a University, although it would be time consuming for each member of staff to sign up, setup their page and create their schedule. After signing up the User can login and customise the site to suit their preferences, such as having different types of bookings [Appendix 6]. The use of different types of meetings is listed in the requirement specification 3.3 [Appendix 12], to differentiate the types of meetings staff members will be booked for. Different schedules can be created for different weeks [Appendix 7], which allows a lot of flexibility for the User to have multiple different schedules based on the week they are on. To create a booking individual must click on the User's unique Calendly link and select a time slot and day in the calendar [Appendix 8]. Guests can be added to a booking, which offers the same functionality as a group booking, with both people receiving an email confirmation [Appendix 9]. Calendly offers many useful features that could be implemented into the booking system such as booking types. The way Calendly is designed is very easy to use everything is displayed and there are help icons and pages if users get stuck. Security is a drawback with Calendly as anyone with the link can make a booking with the User, which could lead to a spamming of bookings by bots preventing legitimate booking requests from being made.

2.2.3 LANCASTER UNIVERSITY BOOKING SYSTEM

The final system looked at was the Lancaster University Booking System [4]. Lancaster Booking System has a simple design, each department is listed on the left side with members of that department within a dropdown [Appendix 10]. When a staff member is selected a calendar appears with the days available in the middle, and the times available on the selected day on the right side. Once a time has been selected simply click appointment and it will redirect to the Lancaster University login page for students and staff to sign in. This booking system is very simple and effective as it provides the minimum requirements needed and offers a level of security by linking to the University login. Any people outside the University can get onto the booking page but cannot proceed any further. The functionality of the Lancaster Booking System fits perfectly in line with many of the client's basic requirements, the main function being the use of the calendar that blocks out days based on the staff members schedule and the selection of time slots. One feature which has not been implemented is the different types of meetings as all meetings in this system are the same duration and type.

2.2.4 SECURITY RISKS

OWASP Top 10 Web Security Risks
1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with known vulnerabilities
10. Insecure logging and monitoring

Table 1 – OWASP Top Ten Security Risks

Security risks are a major factor to consider in Web Development as they can be the downfall of any site. OWASP [5], a non-profit foundation dedicated to the security of software, defined the top 10 security risks in a web application, shown in Table 1. Along with defining these security risks OWASP offers a guideline in how to mitigate and even remove some of these risks from a Web Application. Positive Technologies, a leading provider of cyber security to companies, conducted a study into Web application attacks in 2018 analysing types of attacks and the popularity of each on Web Applications[6], this found the most common cyber-attacks are SQL Injection, XSS and Path Traversal. SQL Injection is a common attack, where an SQL query is typed into an input to get the query to run on the server for example "DELETE FROM users" is typed into a form, which if not sanitised correctly will run on the server. XSS is an attack that attempts to run a script on an application by typing the script into an input, a simple example of this would be typing "<script> alert('hello') </script>" which if not sanitised will be run by the application. To mitigate the threat of an XSS attack all data input into a form will be sanitised using the filter_var PHP function. This function converts scalar values into a string, as defined in the PHP manual[7], converting potentially harmful scripts into a simple string. According to a study done by Georgiev[8] into the security of PHP applications information security can be divided into three primary categories: Confidentiality, Integrity and Availability or CIA. Confidentiality is the availability of information only to authorised persons, unauthorised access is not permitted. Integrity is the data stored in a system has not been altered by unauthorised persons. Availability is the assurance that data is available for authorised users when it is required. All three will have to be integrated in the design of the booking system, to ensure the application is suitable for the users while also keeping any data input into the system is secure. Figure 1 shows the flow of data through the system. The data requested can also include user input such as searching for a specific staff member. As mentioned above all data passed from the User through a PHP class will be filtered before it is run on the database nullifying the risk of an XSS or SQL injection attack. As the Booking System will be designed for a University consideration must be taken into the types of attacks that Universities face, with an emphasis being on the prevention of the most common attacks while also remaining secure against others. Referring to the study by Positive Technologies in 2018 Figure 2 shown a breakdown of the attacks on Educational Institutions. The prevention method for SQL Injection and XSS attacks has been mentioned above, path traversal is an attack that attempts to access files stored outside of the root folder. This attack is commonly known as "dot-dot-slash" referring to the command line use of this to get to the folder above. To mitigate this threat the use of .htaccess files can prevent sensitive or vulnerable files being added to the web root and by

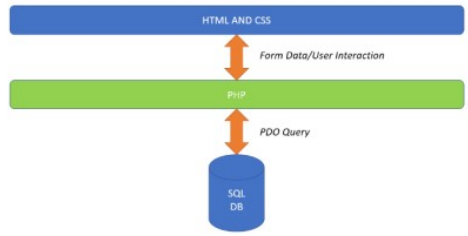


Figure 1 – Overall Structure of the System

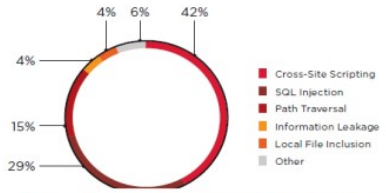


Figure 2 – Top 5 attacks on Educational Institutions

hiding the full structure of a server, using .htaccess files will also prevent Local File Inclusion. Information leakage cannot be prevented by the system as this is due to user error.

2.3 METHODS AND TOOLS

2.3.1 DATABASE

MySQL database is an Open-Source Relational Database Management System (RDBMS). MySQL is the most widely used RDBMS and is used by the likes of Facebook, Google, and many others. As of 2020 there are 205 security vulnerabilities with MySQL [9] many of which can be prevented on the client side by checking data before it is posted to the database. MySQL also offers integration with PHP. MySQL requires a server to host the database on and there are two options for this project. In this project the database will be hosted on a localhost server, due to the availability and ease of use. When testing begins users will gain remote access to the host PC to use the system. MySQL is an industry standard and has a wide availability of support and documentation available. It is due to this and the flexibility in design that MySQL offers that it will be used for the Booking System. An alternative database that could've been used is PostgreSQL[10] , which allows more flexibility than MySQL. PostgreSQL is another RDBMS and integrates many features including standard SQL syntax. Another alternative database that could've been used is MongoDB. MongoDB is a NoSQL system, as it stores data as binary JSON (BSON), instead of using tables and rows[11]. MongoDB is a lot more flexible than MySQL or PostgreSQL but does not include many safety features, as the system will be holding potentially sensitive data MongoDB wasn't used. Due to the smaller scale of this project MySQL was chosen over the other PostgreSQL.

2.3.2 PROGRAMMING LANGUAGES

PHP will be used in conjunction with HTML, CSS, and JavaScript to create the booking system. PHP will be used as a backend for the system due to its integration with MySQL, communicating with server to run queries to return, update and delete data. For the front-end HTML code will be displayed through PHP files, with JavaScript scripts being used to make the system more dynamic. This setup allows the passing of PHP and JavaScript variables into HTML code to be displayed in the Web Application. An alternative language that could have been used would be Python, this could have been done using the Django framework[12]. The Django framework allows web applications to be built rapidly, but it is not as widely used as PHP or other alternatives making documentation less readily available. Due to PHP being more widely used there are many of online resources available to speed up the project process, some of which will be mentioned in this section.

2.3.2.1 PHP

PHP is a programming language popularly used in web development, as it allows direct interaction and manipulation of SQL databases using a PDO, PHP Data Object. A survey done by W3Techs, a company that conducts software surveys [13], found that PHP is the most popular server-side programming language, shown in Figure 3. PHP offers many built in functions that aid in the functionality of the site such as the use of session handling, as defined in the PHP manual [14]. Session handling allows the preservation of data across a site, which allows features such as a user staying logged in and content displayed based on user level, which fulfils requirement 2.3 in the specification and gives users access to different features on the system based on their level. One drawback of PHP is that to reload content a whole page must be reloaded, meaning that all content must be loaded on a page at the same time. To counteract this JavaScript will be used within a PHP page to give

Most popular server-side programming languages

© W3Techs.com	usage	change since 1 November 2020
1. PHP	79.1%	+0.2%
2. ASP.NET	9.3%	-0.2%
3. Ruby	4.3%	+0.1%
4. Java	3.2%	-0.1%
5. Scala	1.8%	-0.1%

percentages of sites

Figure 3 – W3Techs Programming survey

the site a more dynamic feel. An example of this would be a HTML form displayed in a PHP file using a script written in JavaScript to alter elements on a page based on user interaction without using PHP to reload the entire page. PHP has many known vulnerabilities, but each year the list of vulnerabilities has been decreasing according to CVE, a site that allows users to report vulnerabilities in software, 23 vulnerabilities were published in 2019 as opposed to 21 in 2018 and 43 in 2017 [15], and the number dropping to only 10 in 2020. This increased security in PHP along with the integration with SQL is the main reason for using this language as opposed to other options.

2.3.2.2 JAVASCRIPT

Most popular JavaScript libraries

© W3Techs.com	usage	change since 1 November 2020	market share	change since 1 November 2020
1. jQuery	77.2%	+0.5%	96.2%	-0.7%
2. Bootstrap	21.5%	+0.2%	26.8%	-0.1%
3. Modernizr	10.3%		12.8%	-0.2%
4. Underscore	4.3%	+0.2%	5.4%	+0.2%
5. Popper	3.2%	+0.1%	4.0%	+0.1%

percentages of sites

Figure 4 – W3Techs JavaScript survey

JavaScript is another programming language popularly used in web development. JavaScript is only used in the web browser to dynamically alter content on a page without having to reload it. JavaScript offers libraries, which are a collection of JavaScript functions compacted into one file that can be referenced and used in a Web Application. One of the libraries used in this project will be jQuery. jQuery is the most popular JavaScript library according to a survey done by W3Techs, shown in Figure 4. jQuery is used to simplify the handling of events, allowing shorter lines of code to be written when handling items in the DOM [16]. jQuery will be used in the web booking system along with the jQuery UI plugin which is a library full of UI Widgets and Interactions as listed on their site [17]. One of the main widgets that will be used is the date picker, which will allow users to select a date when they want to make a booking. Bootstrap is the second most popular JavaScript library, this will also be used in the system to aid in the designing of the page as the bootstrap library offers the use of modals, which are essentially popups that appear on a page when a certain condition is met. The next JavaScript library that will be used is the jQuery tablesorter[18]. This plugin turns a standard HTML table into a sortable table that can be sorted based on the column that is selected. As there will be multiple tables displayed in the system, this plugin will be used frequently. A customised search function may also be added to tables for users to search for a specific element within the table. The final JavaScript plugin that will be used is a schedule selector plugin that uses jQuery[19]. This plugin displays selectable times of the day for each day of the week an example of this is shown in Figure 5. This plugin will be used to display the available times of a staff member to students. When an available time is selected, the student will be able to book this time slots for a meeting. This plugin was chosen for its simple but user-friendly design, as an overcomplicated design could confuse users as to how the system works. The main goal of the product is to make it easy to understand and use, while remaining responsive. All the above-mentioned plugins will be used in conjunction to aid in this goal.

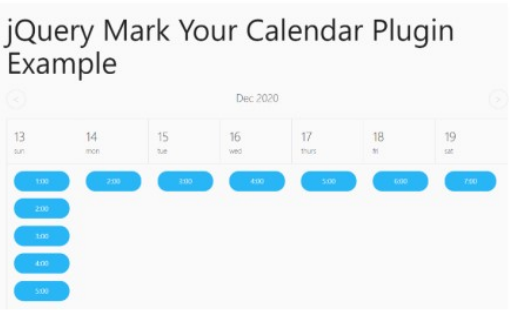


Figure 5 – Calendar Plugin Example

2.3.2.3 HTML AND CSS

HTML is the standard mark-up language for Web documents. The HTML code is sent from a server or local storage to the user's web browser to be displayed on screen. HTML is commonly used alongside a Cascading Stylesheet (CSS) which is used to alter the design of the HTML code to make it more accessible for the user. Both components will be used along with PHP and JavaScript to create the Booking System. Design will have to be carefully considered as this system could be used by a whole University, functionality and ease of use are a must.

2.3.4 PROJECT FRAMEWORK

Scrum is an agile framework for creating products created by Sutherland and Schwaber [20]. It takes a heuristic approach to design by breaking a large goal into smaller sprints. Sprints are the main event of the scrum framework, defined in their guide as a period of a month or less where a list of goals are listed to complete within this sprint. The goals can be different depending on the individual. The scrum framework can be applied to the project process as it can be broken down from a booking system into a set of individual sprints that can be completed individually to achieve the end goal.

Advantages of the scrum methodology are that they break down a seemingly daunting overall project into achievable smaller sprints, making the goal seem a lot more achievable. Applying Scrum to this project by breaking down the Booking System into smaller goals such as Sign-Up System, Sign-In System etc will be a lot more effective than trying to tackle the problem as one big entity.

The main disadvantage that many of the features of Scrum such as team meetings and group objectives will not be used. Another disadvantage of Scrum is that it is designed for teams instead of individual projects, so some of the methodology will have to be tailored to suit this project.

In this project the software used for scrum methodology will be Trello[21]. Trello is a free online project management tool that allows the creation of boards. Within each of these boards a task can be added such as Departments class. Within this Departments class a checklist can be added as a list of to dos to complete the class [Appendix 11].

2.4 REQUIREMENTS SPECIFICATION

A full list of the client's requirements are listed in the Appendix [Appendix 12]. The requirements specifiation was created from the clients requirements and broken down into four main requirements, students, staff, administrators and usability. This section will identify the key points and explain how each point will be achieved through the use of the methods and tools mentioned in section 2.3. All of the requirements are given a priority level from low to high, to differentiate which requirements are essential and which are more of a commodity. The requirements are broken down into 4 sections, 3 for the different kinds of users and one for usability.

2.4.1 REQUIREMENTS FOR STUDENTS

A key requirement for students is to see a list of available staff, as stated in requirement 5.2. The system will achieve this through the use of a bootstrap table, which lists all departments in the University. Within each department will be the staff members associated, this design will make it easier for students to find multiple staff within their department as opposed to a long list of staff members. Both the list of departments and staff members will have a search function to aid ease of use.

Another requirement for students will be to make a booking with staff members. This is defined in the Bookings Process section of Requirement Specification [Appendix 12]. The key requriements of this process are requirement 6.2 selecting a staff member to start the booking process, requirement 6.3 displaying the available meeting types of a selected staff member and requirement 6.4 the ability to display a staff members schedule

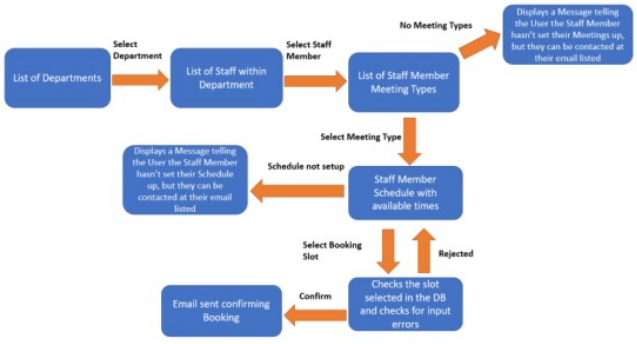


Figure 6 – Booking System Process

and booking times when the student has selected a meeting type. Together these three requirements along with requirement 5.2 create the booking process for a student, as shown in Figure 6.

A page that will list all of their confirmed bookings will be essential for students. Requirement 5.1 identifies that along with the list of bookings, the ability to edit and cancel bookings should also be available.

The final key requirement is a User page will information about the student along with the ability to edit their information. The page could be displayed as an about me page [Appendix 13] or simply list the information that is made editable to the user.

2.4.2 REQUIREMENTS FOR STAFF

A requirement for staff members the ability to set their hours of availability for each week, defined in requirement 3.1. Through the use of two times a start time and an end time staff members will be able to create slots of availability for each day of the week. Requirement 3.2 states that these slots will be displayed repeatedly for each week.

Meeting Types are an essential requirement to how the booking system will work, they are defined in requirement 3.3. Each staff member will be able to create multiple meeting types, which will then be selectable by students.

Staff members will also have a booking page available to them that shows all of their current bookings. Requirement 3.6 states that the bookings should be editable and the ability for staff to cancel a booking should also be implemented. Finally staff members will have an avout me page with information aobut themselves which is editable.

2.4.3 REQUIREMENTS FOR ADMINISTRATORS

The main requirement for administrators is the ability to edit, as stated in requirement 11.2. This will include users, departments and possibly schedules. Deleting Users will be available to administrators as well, as when staff may leave the University their account will no longer be required. Administrators will be able to send a password recovery email to users, if they have trouble logging in. Forgotten Password will also be a menu item for non logged in users.

2.4.4 USABILITY REQUIREMENTS

The main usability requirement is cross compatibility between web browsers, and the ability to use the system on a mobile, this is defined in requirement 12.2. As many students will want to book a meeting with their lecturer and some do not have access to computers this will have to be considered. Cross compatibility testing will be performed and described in a later section.

Another usability requirement will be navigation through the system, this will be performed using a menu along with icons. The icons used will be from the font awesome library[22], an open source library of icons that will help the user in understanding multiple parts of the system. Titles will also be added to the icons that will give a description of what action selecting the icon will do.

3. SYNTHESIS

3.1 DESIGN SPECIFICATION

3.1.1 WIREFRAMES

The initial wireframe design was rather basic and low level [Appendix 14]. It allowed me to create an initial idea of how the system would work, with users signing in and based on their level the system would display the correct pages e.g. for staff a schedule page to alter their schedule and meeting types. This design was later scraped for a more flexible version. The Updated wireframe homepage, Figure 7, allows Users to not only login but also sign up to the system, along with a feature that resets a forgotten password and sends it to the user's email. Font Awesome icons are added to make the menu more user friendly.

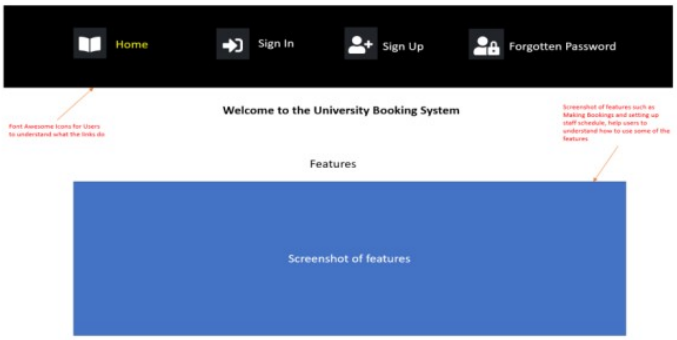


Figure 7 – Booking System Landing Page Wireframe

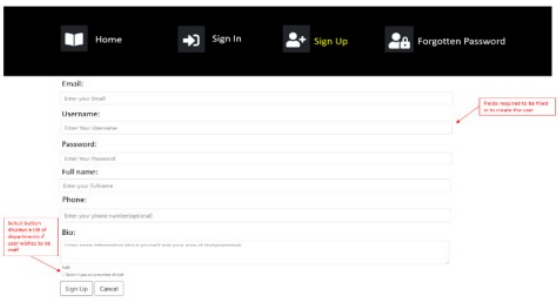


Figure 8 – User Sign Up Wireframe

The sign in page consists of two fields, one for username and one for the password and displays an error if the form is filled in incorrectly [Appendix 15]. The sign-up form is very similar, as all forms will be generated from the forms class, this displays all the information to be filled in, in order to create a user, shown in Figure 8. If the User ticks the staff tick box, a dropdown will be displayed with a list of departments for the User to select from. The final wireframe for the login system is the forgotten password wireframe [Appendix 16]. This form simply displays an email field, if the email is correct, it creates a new password for the user and sends it to that email.

Next is the Student User landing page, which is of similar design, Figure 9. The main difference is the menu and the addition of information about an upcoming Booking being displayed. The menu allows student users to view their bookings, with the bookings link, make bookings with staff link, see their user information with the user link and change their password. The bookings page, Figure 10, shows all their upcoming bookings,

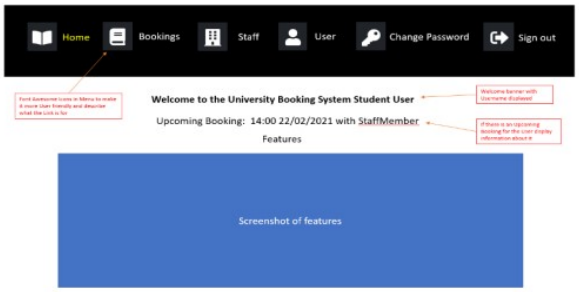


Figure 9 – Student/Staff User Landing Page Wireframe

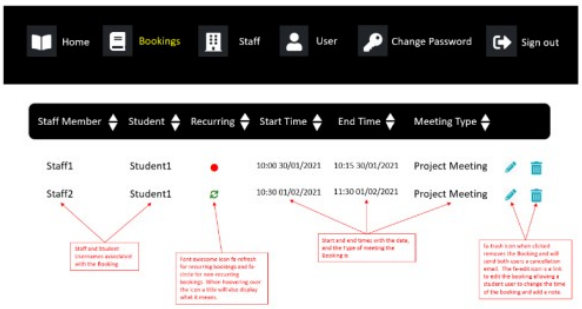


Figure 10 – Student User Bookings Page Wireframe

along with the start and end time, the type of meeting and the Staff member they are booked with. The page also shows whether the Booking is recurring or not. As shown in the Figure the font awesome icons in the table are fa-trash, that when clicked deletes the Booking and will send an email to both parties, explaining the cancellation. The fa-edit icon allows the User to edit the booking, changing the time and/or date of the booking along with changing the note associated with the booking. The Booking Time and Date can only be changed from the options in a select array, to prevent Users from booking a time outside of scheduled hours. No users can change the person they are booked with as this would break many aspects of the system, so Staff cannot change the Student they are Booked with and vice-versa. A feature may also be added to give Users the ability to search for a meeting type or possibly a start time, but this may not be implemented due to time constraints.

The next Wireframe is the design of the Staff page, which allows students to make bookings with staff members. The first page displayed is the Departments page [Appendix 17], which displays a list of selectable departments, with the number of staff members associated. When a Department page has been selected the student is put onto a staff page [Appendix 18]. The staff page displays a list of clickable staff members along with their email address, in case the student wishes to contact them directly. When a staff member has been clicked on this leads the student to the Meeting Type page. This page shows a list of available Meeting Types for the selected staff member, shown in Figure 11. This page displays the Meeting Types in a box format, along with the duration of each meeting and a description about the meeting, to explain what it is. A link back to the previous page is in the top left in case the student selected the wrong staff member. If a staff member has not created any Meeting Types a message is displayed saying this, along with their contact information. When a Meeting Type has been selected the system loads the student booking page, shown in Figure 12. The Page shows the Staff members name in the header, and a link back to the Meeting Types below. Then the JavaScript Calendar plugin displays the Staff members availability. The plugin used will be mark-your-calendar[19], which was found when researching JavaScript plugins, and stated in the analysis section. The source code will be modified to add the availability only when the time and date have not already been booking, and when the booking does

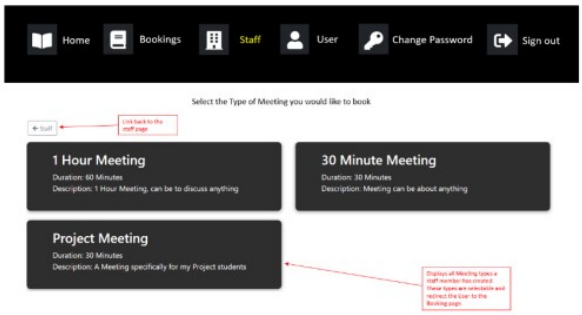


Figure 11 – Student Meeting Types Page Wireframe

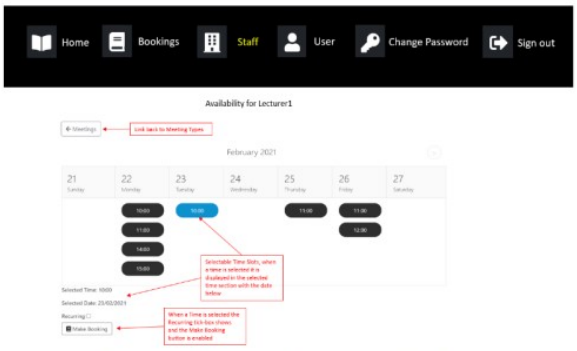


Figure 12 – Student Booking Page Wireframe

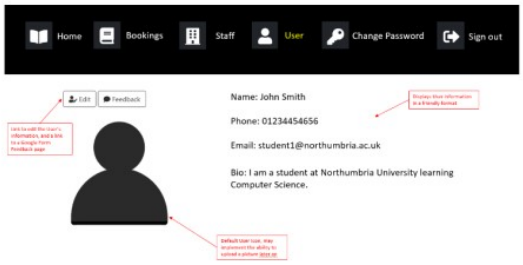


Figure 13 – Student User Page Wireframe

not interfere with a Staff member holidays. This plugin was chosen due to its simplicity, as more complicated plugins would require more coding time, and some student users may have difficulty with them. An additional feature will be to prevent bookings further than 5 weeks in advance, to prevent students from going back before the current date. When a time has been selected, the time and date will appear below the calendar, as will a recurring tick box allowing the user to make the booking recurring. After clicking the make booking button, PHP will validate the data passed through, if successful an email will be sent to both users, with an ICS file for the booking, and a message will be displayed on screen confirming the booking, if the validation fails an error message will be displayed explaining why before redirecting the user to the booking page.

The user page was designed similarly to the about me page listed in the analysis, see Figure 13. With a few changes to fit in with the current design's look. For students, the page is rather simple, with a default user icon being displayed alongside their information. An edit button is displayed with a link to a form that allows the User to edit certain information about themselves. Their user level is the only information not editable. The User edit form displays their information in fields allowing users to edit the information within [Appendix 19].

The final page for student users is the Change password page. This page has three fields, current password, new password and repeat new password. If the user types in the current password correctly and the new password correctly twice, the password will be changed, if not an error will be displayed stating the reason the password does not change, shown in Figure 14. The change password form is the same for all users, so the differences for each user will not be mentioned. The sign out button simply signs the user out and puts them onto the booking system landing page.

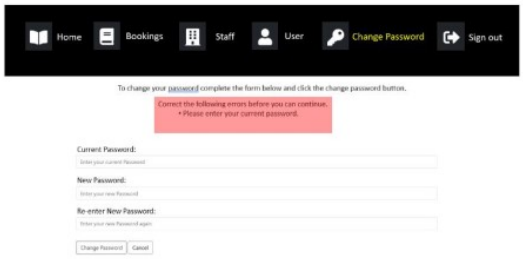


Figure 14 – Change Password Wireframe

For staff users the landing page is the same as with student users. The Bookings page has one difference, which is an add booking button, enabling the staff member to add bookings manually instead of using the booking form like students [Appendix 20]. The next difference is the staff link is replaced with a schedule link. Clicking this link brings up the staff members timeslots, as Figure 15. Each slot shows the start and end time along with the day of the timeslot, Saturday and Sunday are prevented from being used. Above the table are links to add a new schedule item, display the user's holidays and display the User's meeting types. Clicking add schedule brings up a form [Appendix 21] which has a dropdown with the days, along with two time inputs which will be created using a JavaScript plugin. This will allow the user to choose a start time between 9:00 and 16:30 using the min and max in HTML forms to prevent another time from being used. The end time can be between 9:10 and 17:00, which fits in with staff hours at University, this may be

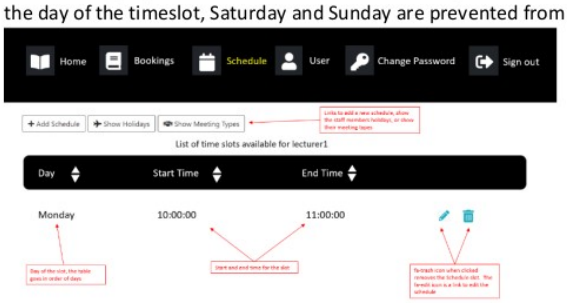


Figure 15 – Staff Schedule Wireframe

changed based on user feedback. The show holidays page is designed similarly to the schedule page, the only difference being the columns displayed will be the start and end date, shown in Figure 16. As you can see the theme is the same, the only difference being the data displayed and the configuration of the buttons, which allows navigation to the two other areas, Meeting Types and Schedule. The form for holidays will contain the start and end date fields [Appendix 22], with a staff member dropdown populated by the User, although this may be removed in the final build. The next page is the Meeting Types page [Appendix 23], which again displays the three buttons, this time with Add Meeting Types to add new Meeting Types, show holidays and show schedule. The table will display all the User's meeting types, with their name, the duration of the meeting in minutes and the font awesome icons to edit and delete the meeting type. The Meeting Type form [Appendix 24] will contain 4 fields, the staff field with the staff member within, the Meeting name field to give the meeting type a name, the duration field which will be a range from 5 minutes to 120 minutes and the description field which is an optional field to add additional information about the Meeting Type. The User link for Staff members displays a weekly schedule calendar plugin along with their information [Appendix 25]. This plugin will allow the user to see their weekly availability and if they feel it is incorrect go to the schedule page and edit it. The Change Password and Sign out links work in the same way as mentioned in the student section.

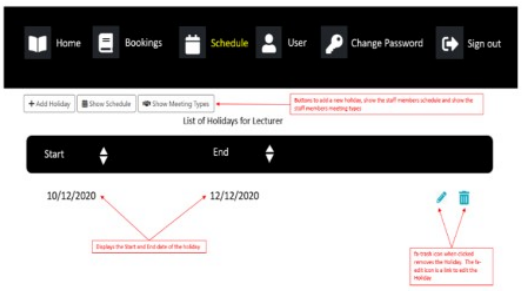


Figure 16 – Staff Holidays Wireframe

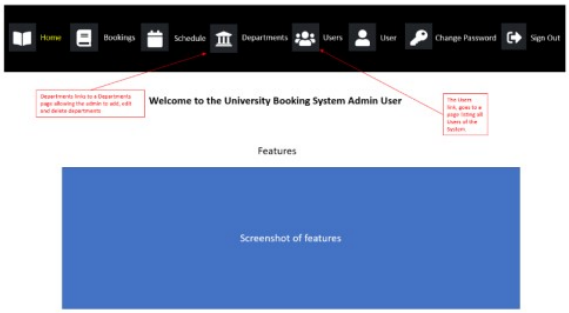


Figure 17 – Admin Landing Page Wireframe

The next design is the Admin page design. Figure 17 shows the admin landing page layout with the menu above. The menu has two new additional links, one to a Departments page, which allows the Admin to add edit and delete departments. The second new link goes to a User's page, which will display all Users in the System and allow the Admin to make changes to Users. When an Admin clicks of the Bookings link it will bring up a list of all Users with clickable links [Appendix 26]. When a link is clicked on it will bring up that User's booking page and allow them to edit their Bookings, giving admins full control over Bookings which is defined in requirement 11.2. The schedule page is slightly different as it will only display staff users, as student users cannot set a schedule [Appendix 27]. The layout will be similar with the name being a clickable link, instead of the User level the staff members department will be listed instead. Clicking on the User will bring up the staff schedule page, allowing the admin user to edit the staff members schedule, meeting types and holidays. The Departments page is next, a table is displayed with all Departments in, as shown in Figure 18. The Add Department button will put the User onto a form page with one field [Appendix 28], the department name, allowing them to add a new Department into the database. Clicking the pencil icon will allow the Department to be edited and the trash icon will delete the Department, only if there are no Users associated with it as deleting a Department with Users in would cause problems throughout the system. The Final new

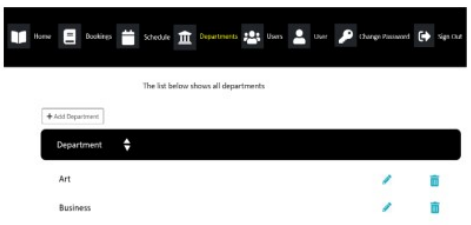


Figure 18 – Admin Departments Page Wireframe

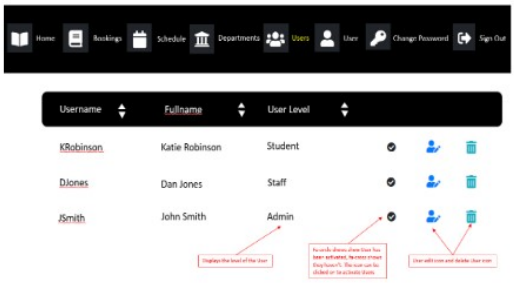


Figure 19 – Admin Users Page Wireframe

page will be the Users page, shown in Figure 19. The User table displays the user's name, full name, and user level. The icon next to the User level shows a tick circle if they are activated and a cross if not. The cross can be clicked on by an admin to activate new users. The user edit icon, next to this, allows an admin to edit all the user's information including their user level. Finally, the trash icon as with other tables allows the admin to delete a User. The edit users form for admins is the same design as the edit user form, but for administrators all fields are editable [Appendix 29]. The design of the admin user page is the same as the student user, as they do not have a schedule to display, the change password is the same format as well.

3.1.2 USE CASE DIAGRAM

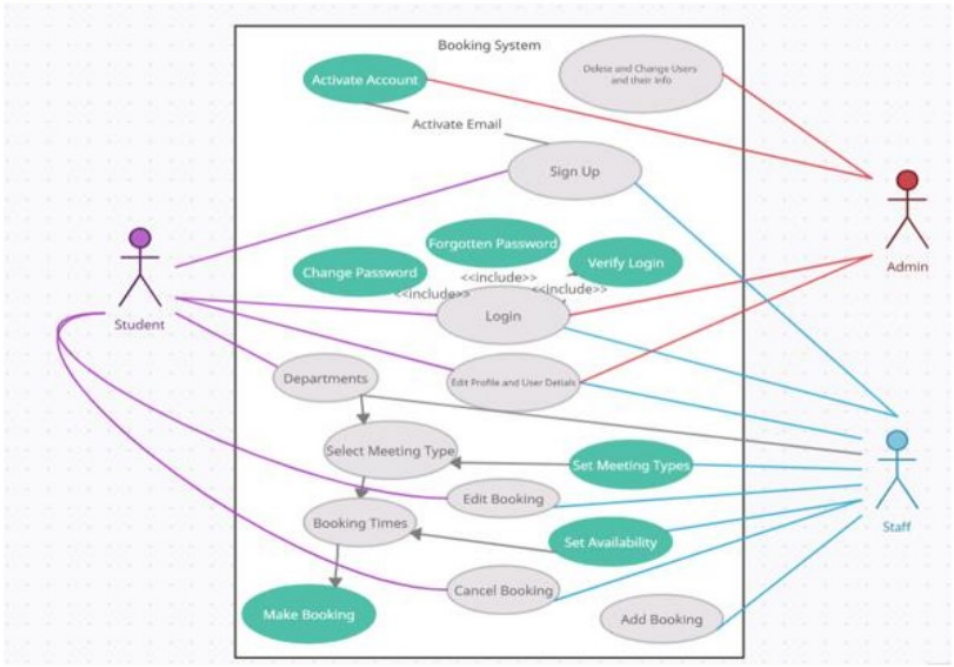


Figure 20 – Use Case Diagram

The Initial Use Case Diagram [Appendix 30] was not as detailed as the Final Use Case Diagram, shown in Figure 20. It detailed the process of Students using the Booking System but did not accurately show the sign-up options, as a User can be activated by an Admin, or use the Activation Email themselves to activate their account. The Booking process was also inaccurate, compared to the final diagram, that shows the relationship between Departments, Meeting Types and Booking. Staff members do not have to go through the booking process and can add bookings using the booking form. The final diagram shows that the user picks a Department which is associated with a Staff Member, from there the student selects the Staff member, then the type of meeting they want to book. Finally, a booking time is selected based on the Staff member's availability, after this the booking is made.

3.1.3 CLASS DIAGRAM

The initial class diagram, Figure 21, shows the first conceived structure of the system. All the main classes use the Forms class to display their forms, display class to create tables and Webpage class to create the header. The Writequery and Readquery classes are used for database queries in all the main classes. The updated class diagram, shown in Figure 22, added the ICS class. The ICS class is used to create the calendar files, which are sent with the emails to the student and staff member. The ICS file can then be added to their calendar. The major change in the updated class diagram is all the variables used are defined, along with their type. For every function, the variables passed through are also defined in the final diagram, this will help if the system is every updated as another developer can see how many of the functions work using the class diagram. The PDO connection, Readquery and Writequery functions were removed in the final class diagram and given their own class diagram separately [Appendix 31].

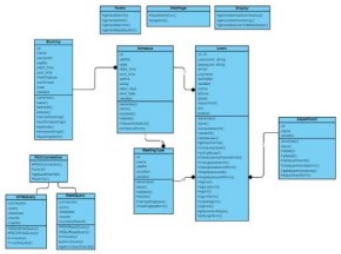


Figure 21 – Initial Class Diagram

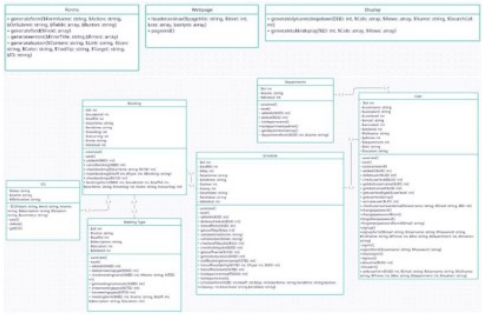


Figure 22 – Final Class Diagram

3.1.4 ENTITY RELATIONSHIP DIAGRAM

The first entity relationship diagram, Figure 23, separated staff and student users and had a simplified version of the bookings table, which only contained their ids and a confirmed field. The users also contained a loginstatus field, which was later removed as sessions can be used to manage logins. The staffschedule also only contained starttime and endtime, there were no columns for start and end dates used in holidays. Quickly realising how inflexible this design was it was improved, merging the student and staff user tables as they didn't account for admins, adding a userlevel column to define the permissions of each user. This second iteration of the entity relationship diagram [Appendix 32] added a meetingtype table for staff members to create different types of meetings for students to book, along with the departments table to define what department staff members are in, stated in requirement 4.1. The final diagram, shown in Figure 24, added some minor changes to the table, adding a recurring field which allows bookings to be repeated each week, referring to requirement 6.5. The deleted column was added to all tables to allow users to delete items from the system without removing them from the database, allowing an admin to recover any mistakenly deleted items. The photo field was removed from the userinformation table as it was not an essential feature, allowing more time to be spent on improving other aspects of the system.

Figure 23 – Initial Entity Relationship Diagram

Figure 24 – Final Entity Relationship Diagram

3.1.5 NAVIGATION DIAGRAM

The navigation diagram, shown in Figure 25, displays the structure of the navigation menu based on the type of users and when a user is not authenticated. When a user is not authenticated, their options are to sign in, sign up or fill out the forgotten password form, the home page is shown by default with some information about the system. Student Users have a link to their Bookings page, the Staff page to make a Booking with a staff member, the user page to edit their info, the change password page to change their password and the Sign out link which will put then onto the Non-authenticated user home page and sign them out. Staff Users have the same format as Student Users, apart from the Staff page is changed to a Schedule page, giving them ability to edit and create their own schedule, holidays, and meeting types. Admin users have a much larger menu, due to their ability to edit not only their information but other users. Along with links to other users' bookings and staff members schedule, there is a link to a departments page to allow admins, to add, edit and delete departments. Users is a link to a page filled with all users of the system, allowing the admin to add, edit and delete users, along with the ability to activate users. The last three links are the same as the student users.

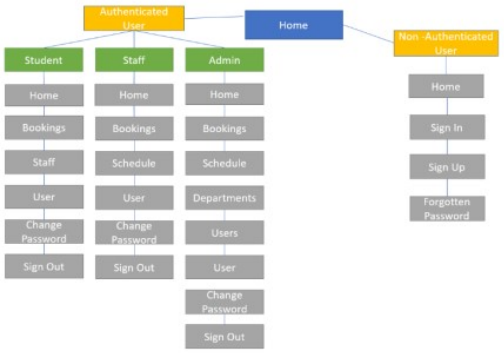


Figure 25 – Navigation Diagram

3.1.6 FLOW CHARTS

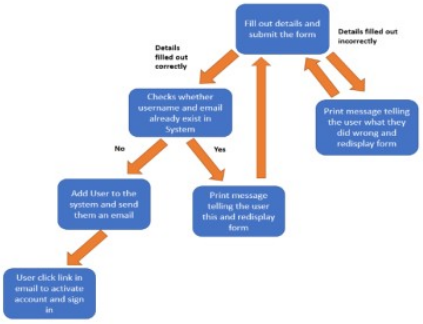


Figure 26 – Sign Up process

Flow charts will be used to describe the processes within the system, as they allow more options than sequence diagrams. Figure 26 shows the sign-up process of the system. When the user submits the sign-up form the system checks whether all the details are in the correct format, if not error messages are display. I the details are correct the system makes a call to the DB, to check if the username and email are unique within the system. If this is not the case a message is displayed to the user stating this and the form must be resubmitted. If the username and email are unique then an email will be sent to the address, so the user can activate their account. Alternatively, an admin can also activate a User's account for them if an email isn't received. The next step is the sign in process, shown in Figure 27. To sign in the user must enter their Username and password, then click the submit button. If all fields are filled out then the system will check if the username and password are correct, if not an error will display, and the form must be resubmitted. Although not implemented at this point in the future a login block could be used to prevent password attacks. This block would log the number of attempts a user makes, after a certain amount of attempts the user would be blocked from attempting to login for short period of time such as 30 seconds. The next flow chart is th Booking System process shown in Figure 28. Firstly, the User selects the department they wish to make a booking in. Then they select the staff

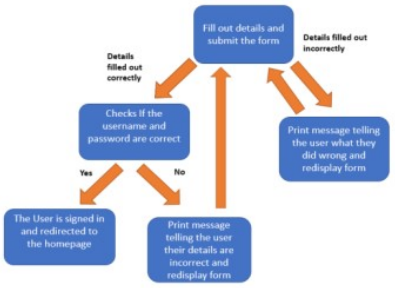


Figure 27 – Sign In process

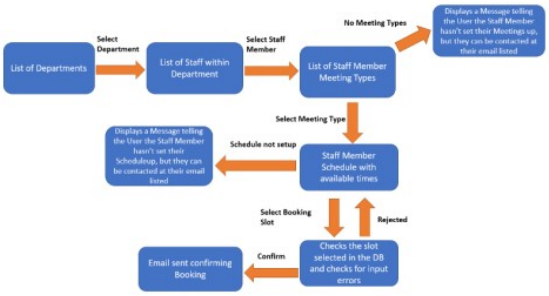


Figure 28 – Booking System Process

member they wish to make a booking with. If the staff the staff member has setup their meeting types they are displayed as clickable box links to the staff members schedule, if not a message displays telling the student this with the staff members email info to contact them directly. Next the staff members schedule is displayed in the calendar plugin, only if the staff member has setup a schedule, if not a message is displayed saying this along with the staff members' email. The plugin makes sure not to display times that are already booked, by querying the database for the staff members bookings. The plugin also makes sure no times are displayed between a staff members holiday. The user then selects a time, the system will double check that the booking does not interfere with any current bookings for the staff member, it will then create the booking in the database, send an email to both parties and display a message confirming the booking to the user. If the booking time slot interferes with current bookings an error will display, and the user will be redirected to the calendar plugin page.

3.2 PROJECT STRUCTURE

This section covers the structure of the files and folders within the project. Figure 29 shows the structure of the files within the Booking-System folder. All the php files in the folder are used by the menu to navigate through the site. The .htaccess file redirects all incorrect file paths towards the index.php file which displays the homepage. The git files are used by GitHub for Git versioning, if an error is created by a new update the system can be reverted to previous version. All the folders are self-explanatory apart from the config folder. This folder contains all two files, the config.php file and the config.ini file. The config.ini file contains all the required variables for the system to work, such as the file path, database details, mail server details and security details. This information is then loaded by the config.php file to be used as global variables. The config.php file also loads all class files within the classes folder, with .class in the filename all other files are ignored. The structure of the classes folder can be seen in Figure 30. All the classes used within the system are stored within this folder, along with a .htaccess file. The .htaccess file has a Require All Denied statement within it to prevent the classes from being shown within the source directory of a web browser, to help secure the system from potential exploitation from users viewing the source code.

.git	16/02/2021 21:27	File folder	
classes	16/02/2021 20:00	File folder	
config	04/02/2021 14:55	File folder	
css	09/02/2021 10:57	File folder	
images	08/02/2021 09:00	File folder	
js	13/02/2021 16:26	File folder	
.gitattributes	17/10/2020 13:36	Text Document	1 KB
.gitignore	08/02/2021 09:05	Text Document	1 KB
.htaccess	15/11/2020 17:48	HTACCESS File	1 KB
bookings	09/02/2021 10:57	JetBrains PhpStorm	2 KB
bookingsystem	25/01/2021 13:46	SQL File	8 KB
department	31/01/2021 21:42	JetBrains PhpStorm	2 KB
feedback	25/01/2021 13:46	JetBrains PhpStorm	1 KB
index	08/02/2021 09:40	JetBrains PhpStorm	2 KB
meetingtype	03/02/2021 17:11	JetBrains PhpStorm	2 KB
schedule	16/02/2021 19:52	JetBrains PhpStorm	3 KB
signin	18/11/2020 13:33	JetBrains PhpStorm	1 KB
user	25/01/2021 13:46	JetBrains PhpStorm	1 KB
users	26/11/2020 12:54	JetBrains PhpStorm	1 KB

Figure 29 – File and Folders Structure

.htaccess	04/02/2021 15:38	HTACCESS File	1 KB
booking.class	18/02/2021 14:22	JetBrains PhpStorm	26 KB
departments.class	04/02/2021 14:55	JetBrains PhpStorm	9 KB
display.class	04/02/2021 14:55	JetBrains PhpStorm	6 KB
forms.class	04/02/2021 14:55	JetBrains PhpStorm	13 KB
ics.class	04/02/2021 14:55	JetBrains PhpStorm	2 KB
meetingtype.class	16/02/2021 19:52	JetBrains PhpStorm	16 KB
pdoconnection.class	16/02/2021 19:52	JetBrains PhpStorm	2 KB
phpmailer.class	31/01/2021 21:28	JetBrains PhpStorm	149 KB
phpmaileroauth.class	31/01/2021 21:27	JetBrains PhpStorm	8 KB
phpmaileroauthgoogle.class	31/01/2021 21:27	JetBrains PhpStorm	3 KB
pop3.class	31/01/2021 21:27	JetBrains PhpStorm	12 KB
readquery.class	16/02/2021 19:52	JetBrains PhpStorm	4 KB
schedule.class	16/02/2021 21:27	JetBrains PhpStorm	43 KB
smtp.class	31/01/2021 21:27	JetBrains PhpStorm	44 KB
user.class	08/02/2021 09:40	JetBrains PhpStorm	46 KB
webpage.class	09/02/2021 10:57	JetBrains PhpStorm	6 KB
writequery.class	16/02/2021 19:52	JetBrains PhpStorm	4 KB

Figure 30– Class Files Structure

3.3 TEST PLANS, TEST RESULTS AND CHANGES BASED ON RESULTS

3.3.1 FUNCTIONAL TESTING

Functional testing has taken place at every step of the project process, to ensure that all functions are working in their designed way. If there are any issues the steps taken to resolve these issues are also documented. Table 2 shows the format of the functional testing, showing how issues were resolved within functions. A full log of all functional testing can be found in the Appendix [Appendix 33].

Function Tested	Expected Result	Actual Result	Resolution
Function 1	Function returns data	Function returns error	Code changed to resolve error
		Function returns wrong data	Query changed to grab correct data
Function 2	Function displays form	Function displays form	N/A

Table 2 – Functional Testing Plan

3.3.2 USABILITY TESTING

3.3.2.1 STUDENT TESTING

Task	Description	Expected outcome	Actual Outcome
User signup incorrect	user to incorrectly sign up or attempt to break the sign in	Error message will display preventing the user from signing up to the website, no account will be created on the db	
User signup correct	user to correctly sign up to the system, either get an email to activate their account or get an admin to activate their account	User account will be created in the db, a message telling the user they are successful will be displayed and they will get activated. Granting them access to the site when logging in	
User signin incorrect	user to input incorrect login details or try to login without credentials	User denied access to the site, with error messages displaying based on their input	
User sign in correct	user to input correct credentials and sign in	User logged in and redirected to home page, welcome message will display	
Forgotten Password	User input their email in the forgotten password form	An email will be sent to the user's address with a new password for them to login	
User change their details incorrect	User to try and change their details and incorrectly putting data in	Error message will display preventing the change and forcing the user to correct mistake	
User change their details correct	User to change some of their details putting in the correct input	User details will be updated in db, page is refreshed, reflecting these changes	
User find a staff member	User to go to staff page and select a member of staff	User will be able to look through departments or search staff member and find the one they want	

User make booking with staff member incorrect	User try to make a booking with a staff member incorrectly, don't input correct details for booking	Error message will display preventing the booking and say where the user has gone wrong	
User make booking with staff correct	User will input correct details to make a booking with a staff member	Booking will be created in db and email will be sent to user for booking confirmation	
User check their list of bookings	User will go to the bookings page and see the bookings they have	User will see the booking created in the previous task along with details of this and showing confirmation	
User edit booking	User will go into booking and attempt to change the time of the booking	User will change the time of the booking and will receive an email reminding them of this	
User delete booking	User goes to bookings and attempts to delete a booking	If the user inputs their password correctly the booking is deleted	

Table 3 – Student User Testing Plan

Table 3 displays the full student User testing plan. Ethical consent was obtained from all Users before testing, in line with the Universities Ethical guidelines. This plan covers all aspects of the system that the student will use, including logging in, making, and changing bookings and changing their user details. This testing was carried out by 15 students. Along with this testing there is a link to feedback in the user form, which allows Users to give more detailed feedback of the system. So far, all the feedback has been positive, with no requests for altercations to the system. This may be due to the system being quite simple in design and the computer literacy of many University students. Table 4 shows the Results of the User testing with every user being able to complete all the tasks required. A rating of the system was also taken to gauge how well Users liked it, with the lowest rating being 7 which is still very positive showing the System was well received.

	Sign Up-False	Sign Up-Correct	Sign In- False	Sign In-Correct	Change Details	Find Staff Member	Choose Meeting Type	Make Booking	Edit Booking	Delete Booking	Rate System /10
Test1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	9
Test2	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	7
Test3	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	8
Test4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	8
Test5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	8
Test6	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	8
Test7	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	9
Test8	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	8
Test9	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	9
Test10	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	8
Test11	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	7
Test12	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	8
Test13	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	9
Test14	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	10
Test15	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	8

Table 4 – Student User Testing Results

Along with the testing Users filled out a System Usability Scale Form (SUS), which is a tool used to measure the usability of a system[23]. The SUS form used for students was created in Google Forms [24] , which creates graphs based on the data collected. Using the SUS scoring scale[25] the results gathered, shown in Table 5, can be interpreted and a rating of the system can be found. The average score from the 15 users is 94.5, which shows how well liked and easy to use it was for students. As stated on the SUS site an average score of 68 and above is the sign of a good application, the average score from these results is a lot higher than that. The verbal feedback from the students highlighted this, as many of them felt that the simplicity of the design and the quick response made the site very usable.

Timestamp	I think that	I found the	I thought t	I think that	I found the	I thought t	I would im	I found the	I felt very	I needed t	SUS Score
2021/02/06	4	1	5	2	4	2	5	1	3	2	85
2021/02/07	5	1	5	1	5	1	5	1	4	2	95
2021/02/08	5	1	5	2	5	1	5	1	4	1	97.5
2021/02/08	5	2	4	1	4	1	5	2	4	2	85
2021/02/26	5	1	5	1	4	1	5	1	5	1	97.5
2021/02/27	5	1	5	1	5	1	5	1	5	1	100
2021/02/28	5	1	5	1	5	1	5	1	5	1	100
2021/03/02	5	2	4	1	5	1	5	2	5	1	92.5
2021/03/02	5	1	5	1	5	1	5	1	5	1	100
2021/03/03	5	1	5	2	5	1	4	1	4	1	92.5
2021/03/03	5	1	4	1	5	1	5	2	5	2	92.5
2021/03/04	5	1	5	1	5	1	5	1	5	1	100
2021/03/04	5	2	5	1	5	2	5	2	4	1	90
2021/03/05	5	1	5	1	5	1	5	1	5	1	100
2021/03/06	5	2	5	1	4	1	4	2	5	1	90
										Average	94.5

Table 5 – SUS Student User Testing Results

3.3.2.2 STAFF USER TESTING

Similarly, to the student user testing, 15 people tested the staff side of the system. The testing plan for the staff users requires a lot more from the users, as staff members must set up their meeting types and schedule before a booking can be made with a student user. Table 6 shows the testing plan, which goes through the same login testing process as students, but changes when it comes to bookings and setting up their schedule. The staff members are required to setup their own schedule and meeting types before a Booking can be made. The testing plan also includes testing the holidays, to make sure they can do this correctly.

Task	Description	Expected outcome	Actual Outcome
User signup incorrect	user to incorrectly sign up or attempt to break the sign in	Error message will display preventing the user from signing up to the website no account will be created on the db	
User signup correct	user to correctly sign up to the system, either get an email to activate their account or get an admin to activate their account	User account will be created in the db, a message telling the user they are successful will be displayed and they will get activated. Granting them access to the site when logging in	
User sign in incorrect	user to input incorrect login details or try to login without credentials	User denied access to the site, with error messages displaying based on their input	
User sign in correct	user to input correct credentials and sign in	User logged in and redirected to home page, welcome message will display	
Forgotten Password	User input their email in the forgotten password form	An email will be sent to the user's address with	

		a new password for them to login	
Add Slot	Add a staff slot	Slot will be added to the database and displayed in the table	
Edit Slot	Edit the time of a slot	Selected slot will be edited in the database and then displayed in the table	
Delete Slot	Delete a staff slot	Slot will disappear from the table but remain in the database in a deleted state	
Add Holiday	Add a Holiday	A holiday will be added to the database and displayed in the table	
Edit a Holiday	Change the dates of a holiday	Selected holiday will be edited in the database and then displayed in the table	
Delete Holiday	Delete a holiday	Holiday will disappear from the table but remain in the database in a deleted state	
Add Meeting Type	Add a Meeting Type to the database	Meeting Type will be added to the database and displayed in the table	
Edit a Meeting Type	Change the duration and name of a Meeting Type	Selected Meeting Type will be edited in the database and then displayed in the table	
Delete Meeting Type	Delete a Meeting Type	Meeting Type will disappear form the table but remain in the database	
User change their details incorrect	User to try and change their details and incorrectly putting data in	Error message will display preventing the change and forcing the user to correct mistake	
User change their details correct	User to change some of their details putting in the correct input	User details will be updated in db, page is refreshed, reflecting these changes	
User check their list of bookings	User will go to the bookings page and see the bookings they have	User will see the booking created in the previous task along with details of this and showing confirmation	
User add Booking	User click the add booking button and add a booking with a student	A new Booking will be added to the database with the student and staff member and an email will be sent to both	

User edit booking	User will go into booking and attempt to change the time of the booking	User will change the time of the booking and will receive an email reminding them of this	
User delete booking	User goes to bookings and attempts to delete a booking	If the user inputs their password correctly the booking is deleted	

Table 6 – Staff User Testing Plan

After 15 users tested the system, the results were gathered in an excel file, along with another rating of the system. The results of which can be seen in table 7. As shown, there were slight issues with the holidays which were fixed after the second test, and one easy fix for deleting meeting types. there were no problems with staff members using the system and setting up their own schedules. The system rating was slightly lower for this part, as it requires more of the user than the student side of the system, with the average rating of the feedback being 7.53 [Appendix 34]. This could be changed in the future to be more automated, but at this point the system is working correctly.

	Sign Up-False	Sign Up-Correct	Sign In- False	Sign In-Correct	Change Details	Add Slot	Edit Slot	Delete Slot	Add Holiday	Edit Holiday	Delete Holiday	Add Meeting Type	Edit Meeting Type	Delete Meeting Type	Add Booking	Edit Booking	Delete Bookin
Test1	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y
Test2	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y
Test3	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test6	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test7	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test8	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test9	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test10	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test11	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test12	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test13	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test14	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Test15	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Table 7 – Staff User Testing Results

The final part of the testing used the SUS scale form. This form was again created in google forms but is separate from the student user scale form to get accurate results for both aspects of the system. The 15 users were asked to fill out the form, the results are shown in Table 8. The SUS ratings were lower for the staff users, especially at the start of testing, as there were issues with the Holiday part of the system and the delete meeting type did not work correctly on the first test. Additionally, Users were required to complete a lot more tasks for the staff side of the system, this may explain the lower results for the system. Some users did not understand the staff side of the system and had to be helped. Despite this the average score ended up being 85.13 which shows that even the staff side is built well according to Users. Based on the User feedback the holidays schedule and meeting types could be given their own pages to separate them.

Timestamp	I think that	I found the	I thought t	I think that	I found the	I thought t	I would im	I found the	I felt very	I needed t	SUS Score
2021/02/17	4	3	4	2	4	2	5	2	4	3	72
2021/02/17	5	2	4	2	5	1	4	2	4	3	80
2021/02/18	5	1	4	2	4	1	4	2	3	2	80
2021/02/18	5	1	5	2	4	1	4	1	5	2	90
2021/02/21	5	2	4	1	5	1	4	1	4	2	85
2021/02/21	4	2	4	1	5	1	4	1	5	1	90
2021/02/23	5	2	3	2	4	1	4	2	4	1	80
2021/02/23	5	1	5	1	5	1	5	1	5	1	100
2021/03/03	5	2	4	2	5	1	5	2	4	2	82.5
2021/03/03	4	1	5	2	4	1	5	2	4	2	82.5
2021/03/04	4	1	4	1	4	2	4	1	4	2	82.5
2021/03/04	5	1	5	1	5	1	5	1	5	1	100
2021/03/05	5	2	4	1	4	1	5	1	4	2	87.5
2021/03/07	4	2	5	1	4	2	4	1	4	1	87.5
2021/03/07	5	2	3	2	4	2	5	1	3	1	77.5
										Average	85.13

Table 8 – SUS Staff User Testing Results

3.3.2.3 ADMIN USER TESTING

For the Admin User testing a thorough plan was created, shown in Table 9. This plan was carried 5 times, to make sure that all features work correctly. As shown in the table these tests require a good knowledge and grasp of the system. Through this testing any bugs that need fixing with the administrator side of the site will be found

as it is performed 5 times, if there are errors discovered after each test, the errors are fixed and the test is ran again, by the fifth test all functions should be working correctly.

Task	Description	Expected outcome	Actual Outcome
User signup incorrect	user to incorrectly sign up or attempt to break the sign in	Error message will display preventing the user from signing up to the website, no account will be created on the db	
User signup correct	user to correctly sign up to the system, either get an email to activate their account or get an admin to activate their account	User account will be created in the db, a message telling the user they are successful will be displayed and they will get activated. Granting them access to the site when logging in	
User sign in incorrect	user to input incorrect login details or try to login without credentials	User denied access to the site, with error messages displaying based on their input	
User sign in correct	user to input correct credentials and sign in	User logged in and redirected to home page, welcome message will display	
User change their details incorrect	User to try and change their details and incorrectly putting data in	Error message will display preventing the change and forcing the user to correct mistake	
User change their details correct	User to change some of their details putting in the correct input	User details will be updated in db, page is refreshed, reflecting these changes	
Admin edit user booking	User to go to bookings page, select a user and edit their booking	Admin will edit the User's booking changing the note to something else	
Admin delete user booking	User goes to bookings and attempts to delete a booking	If the user inputs their password correctly the booking is deleted	
Admin add staff schedule	Add a new timeframe to a staff members schedule	New schedule added for staff member	
Admin edit staff schedule	Edit a Staff members current schedule and change it	Existing schedule will be edited	
Admin delete staff schedule	Click on the delete button in schedule menu	Schedule item set to deleted	
Admin add staff holiday	Add a new holiday for a staff member	New holiday added to system	
Admin edit staff holiday	Edit the dates of a holiday	Holiday dates will change in db	
Admin delete staff holiday	Delete holiday from table	Holiday set to deleted in db	
Admin add staff Meeting Type	Add a Meeting Type to the database	Meeting Type will be added to the database	

		and displayed in the table	
Admin edit staff Meeting Type	Change the duration and name of a Meeting Type	Selected Meeting Type will be edited in the database and then displayed in the table	
Admin delete staff Meeting Type	Delete a Meeting Type	Meeting Type will disappear form the table but remain in the database	
Add new department	Add a department to the system	Department added to system	
Edit department	Edit the name of a department	Department name will change in db	
Delete department	Delete a department from the table	If department has no staff members inside it will be deleted, if not message will display	
Delete User	Go to Users menu and delete user	User will be removed from the system but remain in the DB	

Table 9 – Admin User Testing Plan

As with the staff testing there were a few issues with the staff holidays, but this was due to how the data was formatted when it was passed to the validation functions. This was resolved after the second test and every test after had no issues, as shown in Table 10. As a result of the first test the delete holiday and edit holiday functions could not be tested as a holiday needed to be added first. In the second test the add and delete worked, but the edit didn't as some of the troubleshooting had resulted in an error. After this all tests ran smoothly and there were no issues, showing that all the admin functions work correctly.

Table 10 – Admin User Testing Results

3.3.3 VULNERABILITY TESTING

All the forms are built within the same forms class, so they will all respond the same way to an attack, the results will apply throughout the system. The vulnerability testing plan consists of an SQL Injection attack, an XSS attack and a URL attack, a detailed description of the testing be seen in Table 11. The main threats of a website have been broken down into four tests, one for each main threat to the system.

Task	Description	Expected outcome	Actual Outcome
XSS Attack within form	Type in a simple script to create an alert	The Script will be filtered before it can be run, preventing the attack	
SQL Injection Attack	Create an SQL Injection in the form to run when it is submitted	The class will filter the SQL Injection and prevent it being run on the database	
URL Attack	Using the URL try to gain access to restricted parts of the site	The spoofing will not work as the user will not have permission to view the restricted parts of the system	

Input Incorrect data to break the application	Input incorrect data into a field to break the system	The incorrect data will be identified, and an error message will be returned	

Table 11 – Vulnerability Testing Plan

The first test was performed on the sign in page, a simple hello script was input into the username field to see if it would run on the system, Figure 31 shows the results of this. Not only was the attack blocked but it also filtered out the harmful characters of the script, due to the use of the filter_var() function built into PHP. This function checks a variable against a user defined filter[7]. The second attack, an SQL Injection was again performed on the sign in form. Figure 32 shows the attack being performed, if successful all data in the users table would be deleted [Appendix 35], but this was prevented using PDO statements. PDO statements are used instead of standard SQL queries, which are very vulnerable to these attacks. As stated in the OWASP SQL Injection cheat sheet prepared statements are one way around the attack hence why they are used in this system. The next attack performed on the system was a URL attack, this where a client tries to gain access to a part of the system, they do not have permission for. The example in Figure 33 is the student user, with an id of 9, manually inputting another user's id in the URL to try and change their user details. This attempt is prevented by the system as the user does not have the correct permissions to alter other users' information. A second URL attack was performed on the make booking function to try and get a calendar to show for a user that does not exist. The system queried the user, as they did not exist a message displayed showing this and preventing invalid data from showing [Appendix 36]. The next test tried to input invalid data into the system, that either saves the data or returns an SQL error. This test was used on the schedule form to see if a time can be pushed onto the database or create an error, but the system prevented this. One weakness found in the system is the ability to input a custom time into the make booking part of the system. This is shown in Figure 34, as after one booking has been made the user can input a time into the URL, enabling them to make another booking at a time of their choosing.

Figure 31 – XSS Attack on System

Figure 32 – SQL Injection Attack on System

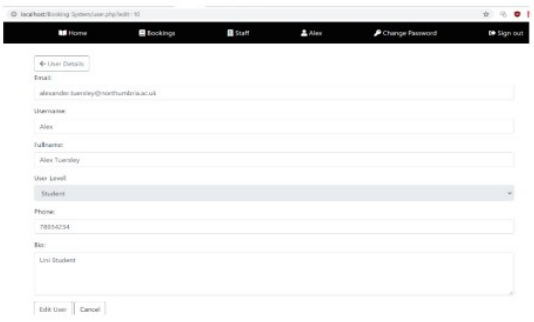


Figure 33 – URL Attack on System

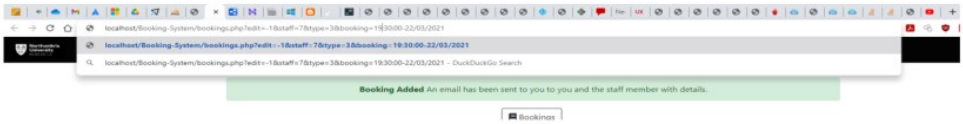


Figure 34 – Successful URL attack

This could be a major threat to the system as student users, if they figure this out, could book anytime they want and circumvent the schedule the staff member has put in place [Appendix 37]. Figure 35 shows the form after the incorrect time has been submitted. Using HTML validation, the form has not been submitted and the time has been flagged as wrong. The next step of this test was for the user to remove the HTML validation and see if the server can identify the incorrect time as well. The server also recognised the time as incorrect and flagged it as so, giving the system two levels of validation to prevent incorrect data being submitted. From the testing done the main weakness identified was the visible items of the URL. One way to mitigate this threat would be to store the booking time in a session, this would prevent the user seeing and altering the booking time but would leave the system vulnerable to session hijacking.

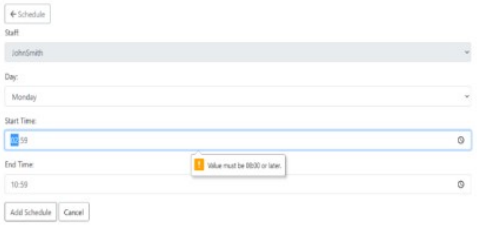


Figure 35 – Input Incorrect Data into System

3.3.4 CROSS BROWSER TESTING

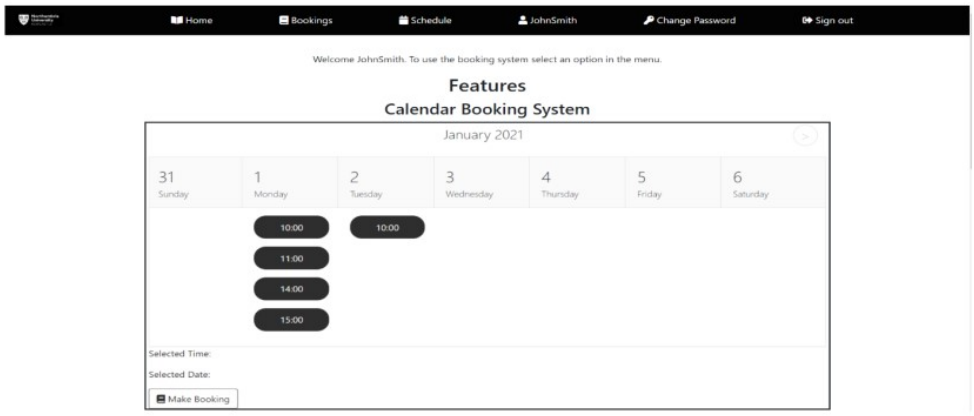


Figure 36 – The System in Chrome

For cross browser testing 5 different browsers were used, 4 of them being the most used browsers in 2021[26], according to W3Counter. The system was tested in Firefox [Appendix 38], Chrome, Edge, Opera [Appendix 40] and Brave [Appendix 41]. Due to lack of access to a Mac it could not be tested in Safari, so Brave was used instead. Most of the testing has occurred within Chrome as it is used by 65% of people and would give the most accurate representation of what the site will look like to most users. As shown in Figure 36 and 37 there are no standout differences in the layout of the system between Chrome and Edge.

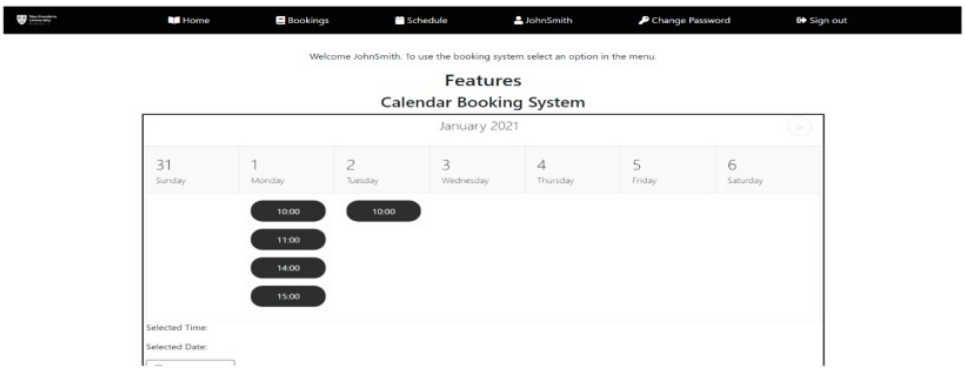


Figure 37 – The System in Edge

The forms were looked at in all the browsers and the only difference was in the layout for Firefox [Appendix 39], the select bars appeared differently in Firefox. The other difference was that Firefox did not use the step HTML attribute to increment the times in the Schedule Form. Although it does not increment correctly the Browser recognises the format it should be in and displays and error in the form, as shown in Figure 38. Due to this no action needs to be taken to make the system anymore cross browser compatible.

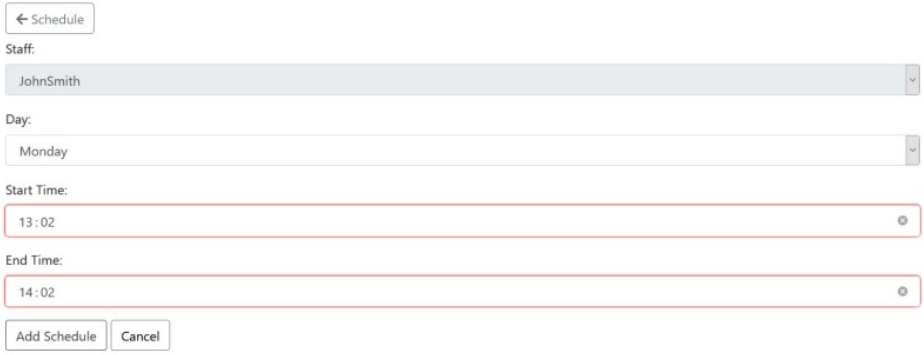


Figure 38 – Booking System Form in Firefox

4. EVALUATION AND CONCLUSIONS

4.1 EVALUATION

4.1.1 PRODUCT EVALUATION

4.1.1.1 BUILD QUALITY

The quality of the system made sure that all the main parts of the requirements were met, along with adding a few additional features that were not described but seemed to fit in the system. An example of this is displaying a staff members weekly schedule to them in their user details. Another good feature is the error handling on the system which catches all error or exception messages and displays a friendly message to the User [Appendix 42].

The UI was built to a good standard using bootstrap and font awesome icons, allowing the creation of a flexible and consistent design throughout the system. The colour scheme was designed with Northumbria University in mind and is reflected throughout.

4.1.1.2 TESTING

The user testing was insightful, being able to get 15 participants for both the staff and student side of the system. Although the actual testing plan for administrators was in depth, more user testing would've been beneficial. The methods used for the testing could've been improved by giving users a sheet which would help them use the system, as opposed to giving them verbal hints.

Unit testing was not implemented into the system due to time constraints and the functional testing that took place. It didn't seem necessary to create unit tests when actual tests were being performed in a localhost environment. The functional testing also went well as when a new issue was found, if it were similar to a previous issue the resolution could be used again to fix it.

Vulnerability testing was not as thorough as it could have been due to lack of experience, but issues were identified. If done again a more detailed vulnerability plan would be implemented with the tests being run on multiple different forms in the system.

Cross browser testing did not highlight any issues but showed the minor changed in the structure when using different browser.

4.1.1.3 PRODUCT REQUIREMENTS

Most of the original requirements were met with only a few optional and low requirements not being met. Feedback from the employer when shown the system was also very positive.

4.1.1.4 USER FEEDBACK

The user response to the system was very positive, with many of them saying that they would for this system to be implemented. The users felt that the simple and sleek design was easy to understand, and bookings were very easy to arrange. The main issue was on the staff side, as the staff setup requires them to setup a schedule and meeting types before a booking can be made.

4.1.1.5 LITERATURE

This section refers to the literature review conducted in at the start of the project process and analyses how effective this literature was in terms of use in the product.

4.1.1.5.1 SECURITY

The security side of the system was a key part, as it identified ways to prevent multiple kinds of attacks. An example of this research being implemented was the use of prepared statements to prevent an SQL attack, as shown in the vulnerability testing.

Some of the security aspects were not implemented such as login blocking after a few attempts, due to time constraints and prioritising the main requirements being usable. In the future the login block would be implemented along with validating the make booking function further to prevent attacks, and other aspects identified in the research.

4.1.1.5.2 DEVELOPMENT

Research into development allowed me to identify the Scrum framework for the development process. Although the waterfall methodology could've been used, this approach is linear whereas the Scrum framework is very flexible. Trello allowed me to breakdown the requirements into smaller sub tasks, making everything more manageable and giving myself small tasks to focus on daily. Figure 39 illustrates this; the Booking class was broken down into functions based on requirements.

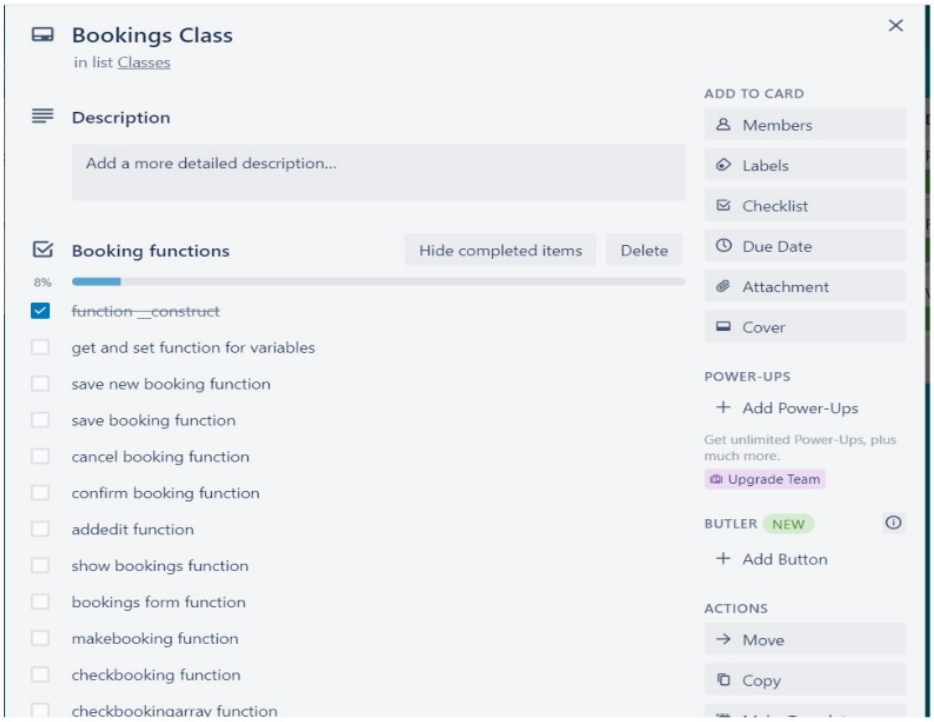


Figure 39 – Trello Booking Class

If the project were done again, Trello would be used as it is a very flexible application and helped greatly in the creation of the product. Other methodologies were identified but did not suit this project as many of them not suited to the design of the system.

4.1.6 TOOLS AND TECHNIQUES

The tools and techniques chosen at the start of the process allowed the development of an effective Booking-System, that aligned with the client's major requirements, creating an application that could be used by a University to allow students to create bookings with clients.

4.1.6.1 SQL DATABASE

SQL allowed the system to manipulate all data as required, which contributed to the great user feedback from the application. Although there a few issues with database queries these were quickly resolved as the project got into the latter stages. Due to my vast experience of SQL the other options though theoretically better would

have resulted in a worse application. The time taken to learn and understand a new database type would've eaten into development time and I may have not completed all of the application. Another option was MongoDB which is more flexible but less secure and security is a big part of the product.

4.1.6.2 PHP

PHP allowed me to create both the front and backend of the application. Class files were used throughout, which allowed me to call functions from a class repeatedly instead of rewriting the code over and over in other files. An example of this was the navigation menu, which displayed a different menu based on user level [Appendix 43].

If the application were done again, Laravel would be used. Laravel allows applications to be created quickly with built-in login and signup functions, along with easy ways to validate data and navigation done using API routes [27]. In standard PHP, each function had to be designed and written out, with some of code being quite long, the user class had over 1000 lines. Laravel would've allowed me to make the application more secure, preventing the threat of both SQL and XSS attacks with a few lines of code, making it much more efficient than standard PHP. Alternatively, Laravel could've been used as an API in conjunction with a JavaScript framework.

4.1.6.3 JAVASCRIPT

JavaScript helped make the system more fluid, altering data on the page without having to reload it, which would've been necessary in standard PHP. The calendar Plugin also allowed me to pass through arrays to it which then displayed a user's schedule based on the data inside. An example of this is the schedule array and holidays array when a schedule item is between any start and end date of the holidays array it is not displayed [Appendix 44].

If the project were done again, a JavaScript framework could be used such as React. React is very popular JavaScript framework and allows the passing and manipulation of data very easily. React also has many user built calendar plugins such as React dates[28]. This would give the system a more up to date and responsive feel, while also reducing the amount of code needing to be written.

4.1.6.4 VISUAL STUDIO CODE AND BOOTSTRAP

Visual studio code allowed me to identify errors very easily, using extensions such as PHPIntelliSense. It is a very flexible code editor and would be used again if the project were re done. Visual studio code also has extensions for Laravel, which makes it even more usable.

Bootstrap helped make the application look very professional and cross browser friendly. Due to my experience with it, it was very easy to design many aspects of the system, although if the project were done again, I would try another framework such as Materialize to see whether it could be more effective.

4.1.2 PROCESS EVALUATION

This section evaluates the project planning process and the execution of said plan was used to create the Booking System.

4.1.2.1 TIME PLAN

The time plan changed throughout the project due to some tasks taking longer than expected, a good example of this was building the system which took longer due to fixing errors and making sure the system was completed to a high quality.

Figure 40 – Initial Time Plan

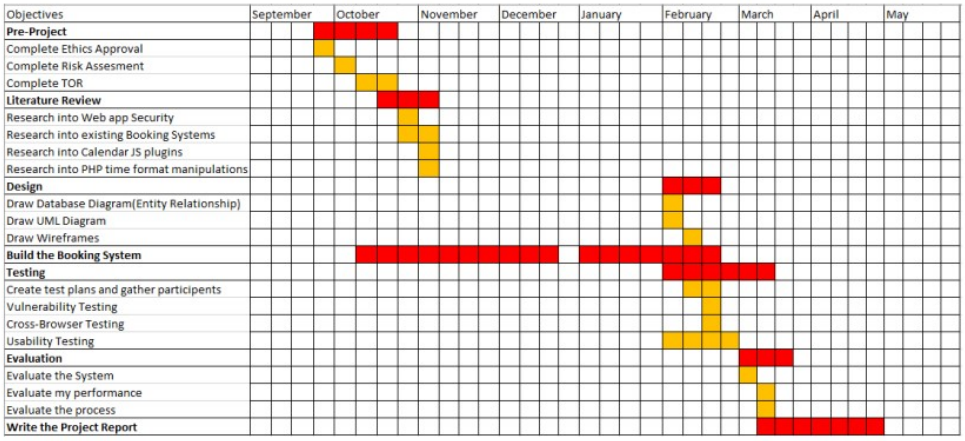


Figure 41 – Actual Time Plan

Pre project was finished in the same way to the initial time frame and the literature review didn't take as long as expected, shown in Figure 41. This allowed me to start creating the system earlier than expected, but the actual building of the system took longer than expected, when compared to Figure 40. The main factor of the longer timeline was to make the system less vulnerable as some areas were very exposed and required further validation to secure.

Usability testing took longer due to the global pandemic, people's schedules and making sure the system was functional for users to test.

The project did manage to be completed at the same time as predicted although the progress of different tasks varied.

4.1.2.2 SKILLS

Many skills were learned and developed during the project process, such as communication as this was required during the User testing.

A skill that was learned during this process was designing. Having minimal previous design experience, two wireframes were created during this process, one high and one low fidelity. Multiple entity relationship diagrams

were created, showing the relationship of the tables on the server. UML class and case diagrams were also made to show how the system works. All the above show the progress that has been made in design.

Planning was another skill learned during this project, as each part of the system was broken down classes and within these classes functions to achieve a goal. Trello was used for this and it helped massively in making the creation of the system manageable and giving a clear goal at the end.

A skill that was enhanced during the project process was PHP coding. Having a lot of experience previous at the end of the project these skills have still improved, as all the naming conventions, functions and comments used in the files are clearly laid out with descriptions explaining the use of each one [Appendix 45].

A technical skill developed during the project was UI design, as an effective and easy to use UI was created that could be used by a University. The theme was consistent throughout with a clear colour scheme and easy navigation through the system, showing the progress made.

SQL querying was a skill improved during the project, as multiple queries had to join tables and return data based on very specific conditions such as getting all bookings after the current data and time.

Adaptability was developed as the system had to fit in the defined client requirements, flexibility was essential as the PHP code had to be moulded to create a system the client would be happy with.

JavaScript was an existing skill advanced to create the calendar plugin for the system. Manipulating the arrays passed to the plugin and changing it from a simple display into a complex system that hides already booked timeslots and ones between holidays. Using adaptability the mark your calendar plugin was taken and altered to fit the client's requirements.

4.1.2.3 PROJECT LIFE CYCLE, METHODS AND TOOLS USED

The life cycle used in this project worked well. The scrum methodology allowed clear goals to be made using Trello, shown in Figure 42. Setting daily tasks, the day before gave a goal to strive for, even if one task was not completed as long as another was forward progress was made.

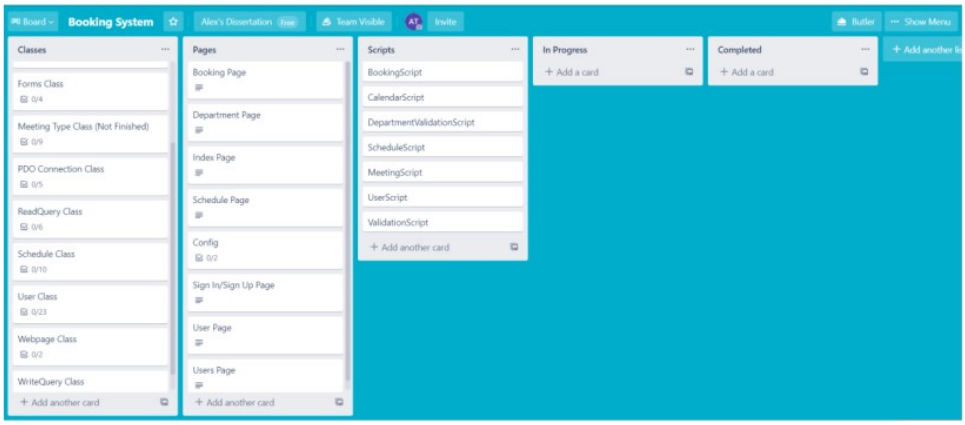


Figure 42 – Trello Booking System

As shown in the Figure each Class, Page and Script was clearly defined. Using this along with scrum created a flexible development cycle, as certain tasks in one class would be completed, then tasks in another class would be completed instead of completing everything in one class all together, Figure 43.

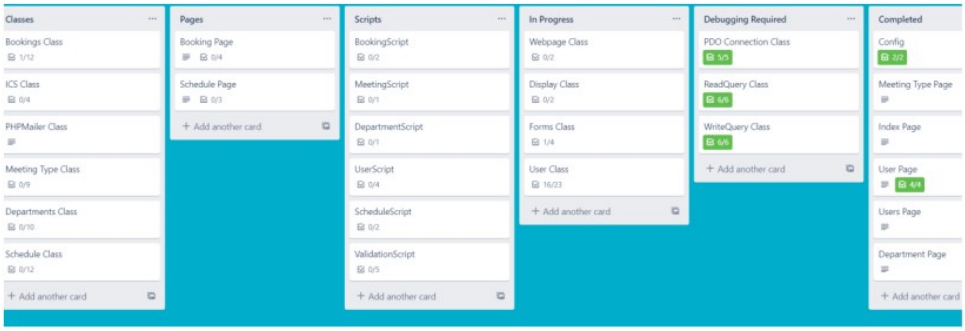


Figure 43 – Trello Booking System Progressing

Figure 43 shows the project progressing with a new field introduced, debugging, to add a bit of testing before the function would be pushed onto the GitHub server. GitHub allowed the creation of multiple different instances of the system, one for testing purposes, one for alternative designs of the project, such as using a different CSS framework and a master instance that will be used in the final project.

PHP, SQL, and JavaScript being used created an interactive system that fit the client's requirements. If the system were created again alternative methods would have been used, to see if a better system could be created, as stated in the above sections a Laravel API would be created, with a JavaScript framework would be used to display the data.

4.1.2.4 LEARNING PROCESS

The main thing learnt during this process was how to create a system to fit a client's requirements. Before this a project of this scale had not been completed. Additionally, a bespoke system had not been created without assistance. Skills such as planning, and design had to be learned to make sure the project was completed.

Multiple software engineering techniques were used such as User Testing and functional testing to help create the system and make sure it was finished to a high standard. The literature review helped as much of the information researched was used to create the application, such as the OWASP security threats and the ways to mitigate them.

Another key thing learnt during this process, was the ability to adapt free source code and make it work within a system. The best example of this was the calendar plugin, which was taken from the internet, the design was changed to fit into the colour scheme and the code was changed to accept multiple arrays. Another example of this was jQuery as multiple functions offered by jQuery were implemented into the system.

During the learning process the PHP Laravel framework was learned. It has built in functions such as User Login, User Sign Up and a design tool called Blade[29]. The inbuilt login/signup system would have saved a lot of development time, which could have been focused on improving other areas. Using blade would have also saved time by eliminating the need to write out multiple HTML lines of code for the system. CSS can also be used with Blade as can jQuery, if the project were to be done again many of the features created could be ported into a Laravel Web Application. User permissions can be set very easily using the models and controllers available in Laravel, eliminating the need to store data inside a session. Laravel also offers the ability to check data that will be submitted to the database before it is submitted, using PHP request files, problems such as SQL Injection and incorrect data being submitted to the database are eliminated.

4.2 CONCLUSIONS

4.2.1 DELIVERABLES

The Deliverables at the start of the project were:

- Create a web booking system with secure login, sign up and change password capabilities. Emails may be sent to users when a password is forgotten or to activate account, if Emails are not enabled an admin can activate the account.
- Using a JavaScript calendar plugin and PHP, a page for staff users to set a schedule up and set dates they are away in the booking system.
- A page for student users to view all staff members and create bookings with them based on their schedule.
- The system will be styled appropriately for ease of use.
- Testing plan for student and staff users to test the system.
- Testing results for both staff and student users based on a predetermined testing plan, the system may change based on testing feedback.
- Flow Charts, Class diagram, Entity relationship diagram, and a wireframe will be created for the system to outline the design and how the whole system will work.

The first deliverable of the project was met, as the system currently has a secure login, sign up and change password capabilities. Along with a forgotten password function, admins can activate user accounts if users do not receive an email.

The second deliverable was created as the system allows staff users to create their schedule. The use of a calendar plugin was not needed as a form was used to allow staff to create their schedule. A page for students to make bookings with staff members was made and is fully functioning, with the addition of times being hidden if they are already booked.

The system has a consistent style throughout, with the Northumbria logo and a dark theme. A testing plan for student and staff users was created, fulfilling this deliverable.

Testing was done on 15 people for both the staff and the student sides of the system, changes were made based on user feedback.

Finally, multiple flow charts, class diagrams, entity relationship diagrams and wireframes were created, shown in section 3.1 Design Specification.

4.2.2 OBJECTIVES

Objectives of the Project:

- Introduction and Abstract.

An introduction and abstract were created to explain the purpose of the project. The abstract gives a brief description and the introduction outlines why the project was created, what tools were used and the aims and objectives of this project.

- Create a literature review for the product.

A thorough literature review was created, similar systems were identified, along with the positives and negatives of each system and how some of their features could be integrated into the product. Security risks of the system were identified and methods to secure the system were also found and implemented into the system.

- Create Diagrams and System prototypes.

Multiple diagrams have been created of each type required in the deliverables. For example, 3 entity relationship diagrams were made to show the progression of the database structure from the beginning all the way to the finished product.

- Create Testing Plan.

A testing plan was created for staff, student, and admin users. Each plan has a name and description of the task and expected outcome as shown in the Usability Testing section of this dissertation.

- Identify requirements for product (Calendar plugin, JavaScript Framework, jQuery version)

Through conversations with the client about what kind of product they wanted, a requirements specification was created [Appendix 12], along with the methods and tools needed to fulfil the requirements. Research was also done into third party software that would be used in the product such as Bootstrap for styling purposes.

- Create product based on specification.

Using the specification created, a product was built to meet these requirements along with a few additional features. Some of the minor requirements could not be met, such as admins being able to add users using an email address, this was a minor requirement and would not have been possible in the current system. But overall, the system works as it should and is fit for purpose.

- Test products.

15 users tested both the student and staff sides of the system, with valuable feedback and data being obtained. The admin side was tested 5 times due to the length of the plan and issues were identified and resolved. The SUS scale form was used to identify how fit for purpose the system is, and additional feedback was given by testers when necessary.

- Make changes to product based on testing feedback and conclusions.

Changes were made to the admin side, to allow admins to add and edit and delete holidays correctly. Added validation was added to the make booking function so as not to allow bookings that were out of hours if the user was not a staff member.

- Evaluate the product.

The product has been evaluated in this section, with ways in which it could have been improved along with the strengths of the final product.

- Evaluate the project process and own performance.

The project process has been evaluated along with my own performance. The initial and actual time plans have been compared and the differences have been explained. The skills used and learned during the process have been stated as has the methods and tools used.

4.2.3 CHANGES AND FURTHER WORK

All the deliverables and objectives have been completed and the system has been completed to a good standard, wherein it could be deployed by a University. One change that would have been made was the use of Laravel for an API, it would have allowed development to be a lot quicker, due to the inbuilt login system and authentication. Another change that could have been made on the UI side would be to use React as a front-end

framework, this would have allowed the website to feel more modern and interactive as React could call the API to gather data without reloading a page.

Further work to be carried out on the current system would be the ability for a user to add a photograph to their profile. Another feature to add would be for students to view staff members profiles before making a booking with them. The reliance on sessions could be changed to storing data in the database, this would be more secure and reduce the risk of session hijacking. The login system would be replaced and integrated with Northumbria University's own student and staff login system, the code which is used in the system would be adapted to make sure that student and staff members are redirected to the right pages. Further security measures would be added to the system in the future to increase security, such as a login block after a number of failed attempts and the use of tokens from a database to authenticate users.

5. REFERENCES

[1] J. Kujanpaa, "Developing a Booking System-Case :Tuas Cisco Laboratory," *Dev. a Book. Syst.*, p. 35, 2015.

[2] "SimplyBook.me - Free Appointment Scheduling Software." https://simplybook.me/en/ (accessed Oct. 24, 2020).

[3] "Free Online Appointment Scheduling Software - Calendly." https://calendly.com/ (accessed Oct. 22, 2020).

[4] "Make an Appointment - Digital appointment using Microsoft Teams - Events - Lancaster University." https://lancaster-uk.libcal.com/appointments/digital-appointment (accessed Oct. 29, 2020).

[5] "OWASP Top Ten Web Application Security Risks | OWASP." https://owasp.org/www-project-top-ten/ (accessed Oct. 23, 2020).

[6] P. Technology, "Attacks on web applications: 2018 in review," p. 14, 2019, [Online]. Available: https://www.ptsecurity.com/ww-en/analytics/web-application-attacks-2019/.

[7] "PHP: filter_var - Manual." https://www.php.net/manual/en/function.filter-var.php (accessed Dec. 13, 2020).

[8] G. Georgiev, "Information Security of PHP Applications," *2019 Int. Conf. High Technol. Sustain. Dev. HiTech 2019*, 2019, doi: 10.1109/HiTech48507.2019.9128275.

[9] "Mysql Mysql : List of security vulnerabilities." https://www.cvedetails.com/vulnerability-list.php?vendor_id=185&product_id=316&version_id=0&page=1&hasexp=0&opdos=0&opec=0&opov=0&opcsrf=0&opgpriv=0&opsqli=0&opxss=0&opdirt=0&opmemc=0&ophttprs=0&opbyp=0&opfileinc=0&opginf=0&cvssscoremin=0&cvssscoremax=0&year=0&cweid=0&order=1&trc=205&sha=db4d7f9ce3bd6b34321f8d0190d4c1dc2fe6e3a8 (accessed Dec. 07, 2020).

[10] "PostgreSQL: The world's most advanced open source database." https://www.postgresql.org/ (accessed Apr. 22, 2021).

[11] "What Is MongoDB? | MongoDB." https://www.mongodb.com/what-is-mongodb (accessed Dec. 07, 2020).

[12] "Django overview | Django." https://www.djangoproject.com/start/overview/ (accessed Apr. 22, 2021).

[13] "W3Techs - extensive and reliable web technology surveys." https://w3techs.com/ (accessed Dec. 11, 2020).

[14] "PHP: PHP Manual - Manual." https://www.php.net/manual/en/ (accessed Dec. 08, 2020).

[15] "PHP PHP : CVE security vulnerabilities, versions and detailed reports." https://www.cvedetails.com/product/128/PHP-PHP.html?vendor_id=74 (accessed Dec. 07, 2020).

[16] "jQuery." https://jquery.com/ (accessed Dec. 13, 2020).

[17] "Download Builder | jQuery UI." https://jqueryui.com/download/ (accessed Dec. 07, 2020).

[18] "tablesorter | jQuery Plugin Registry." https://plugins.jquery.com/tablesorter/ (accessed Dec. 07, 2020).

[19] "Pick Hours of Availability For Each Day - Mark Your Calendar | Free jQuery Plugins." https://www.jqueryscript.net/time-clock/pick-hours-availability-calendar.html (accessed Dec. 07, 2020).

[20] K. Schwaber and J. Sutherland, "The Scrum Guide™ The Definitive Guide to Scrum: The Rules of the Game," 2017.

[21] "Trello." https://trello.com/ (accessed May 07, 2021).

[22] "Font Awesome Icons." https://fontawesome.com/v4.7.0/icons/ (accessed Dec. 07, 2020).

[23] A. S. for P. Affairs, "System Usability Scale (SUS)," Sep. 2013.

[24] “SUS Scale Form - Google Forms.” https://docs.google.com/forms/d/1_fPZxe0nTTP-PyL6OOdtJa3jC5GAUpsoetojBLtNsBE/edit (accessed Feb. 05, 2021).

[25] “System Usability Scale online with analytics | usabiliTEST.” https://www.usabilitest.com/system-usability-scale (accessed Mar. 02, 2021).

[26] “W3Counter: Global Web Stats.” https://www.w3counter.com/globalstats.php (accessed Mar. 08, 2021).

[27] “Laravel - The PHP Framework For Web Artisans.” https://laravel.com/ (accessed Mar. 15, 2021).

[28] “react-dates - npm.” https://www.npmjs.com/package/react-dates (accessed Mar. 15, 2021).

[29] “Blade - Laravel guide.” https://laravel-guide.readthedocs.io/en/latest/blade/ (accessed Apr. 22, 2021).

6. APPENDICES

APPENDIX 1 – TERMS OF REFERENCE

A: Project Title: Web Application Booking System for University

B: Background to Project

As many Universities across the country are moving online, students can no longer walk into a lecturer's office and ask for an appointment or simply see them at University to arrange for a meeting. Thus the need for a booking system has arisen, where a lecturer can set when they are available each day of the week and if they are away or unavailable. This reduces the amount of time both students and lecturers need to spend emailing back and forth to arrange a meeting time, or in some cases a student waiting for a lecturer to reply to an email. The only previous work found of this type was in Turku University in Finland, where the student created an online booking system for the laboratory at their University [1]. This product is slightly different as it is a web booking system to arrange meetings between students and lecturers, and potentially between two lecturers but this could be further work. My client has requested that there be separate interfaces for students and staff as the staff will be showing their schedule and the student will be making bookings without the need to display their schedule.

The project will be to create a Web Application booking system, built on the front end with PHP and JavaScript. The backend will be a MySQL database, which will store all the relevant data. There will be separate classes for each part of the system such as the departments, users etc. PHP will be used to retrieve, update, and insert all data from the MySQL database, this will be done through the use of classes. PHP Classes will also be used to display data in a relevant format for example a User class will take data from the database and display it in a User information table which shows the user's name, email, biography, and timetable etc. JavaScript will mainly be dealing with the calendar plugin and passing the data retrieve by PHP from the database onto the calendar to display for users. The most difficult part of the system will be integrating the JavaScript plugin and getting it to display all the saved data in the database in the format required, along with details of any bookings the user has. JavaScript will also be used to make API calls to a PHP controller, that will run SQL queries to make the page more dynamic. When the system has been built, user testing will be implemented to see if improvements can be made. Staff and student feedback will be essential, to see if they are any obvious issues have bene missed and how the system can be made more user friendly.

As the booking system will require timetables to be displayed, either a calendar plugin or a self-made calendar must be used. There are numerous good calendar plugins available in the JavaScript library. One such plugin is full calendar which is a lightweight plugin offering a lot of customization, such as the design of the calendar format. Full calendar also has a lot of documentation written which makes it easier to integrate into a system, although it is a paid plugin, they do offer a free version which could be used. Getting the data gathered from PHP queries JavaScript can format it to fit in line with the full calendar display, which will then be presented to the user. Another good plugin is Bootstrap calendar which is a calendar based on the HTML bootstrap framework and offers integration with its namesake. Bootstrap calendar also uses AJAX to update its events, so a PHP API could be setup to run certain queries to update ethe calendar when a user makes a booking.

I am very interested in this type of project as I spent last year on placement as a Junior Developer mainly working on a Web System built in PHP and JavaScript. The manipulation of data especially in a time format is something I covered, as I created a login block for the login system that triggered when a certain number of incorrect attempts had been reached, when triggered the user is blocked from inputting a password for 30 seconds. The Web System also had a Calendar for events built in JavaScript, so I have experience in manipulating data from a database and putting it into a calendar.

The Booking system will specifically be geared towards a University as the client requested, which differentiates itself from many other booking systems that are available. Using Northumbria University as an example, it has no system currently in place, students usually email the lecturer they want to arrange a meeting with, this process can become time consuming for both parties, so the need for a booking system has arisen. My supervisor

suggested this idea as he currently uses an external booking system but feels that a better system could be created within the University. A system where lecturers can set time frames where they are available in the day eliminates the need for a back and forth between the student and lecturer making the whole process a lot easier. The user interface will also be easy for everyone to understand as some booking systems can look complicated and daunting to new users. This type of booking system could not only be useful to Northumbria University but others as well, as there isn't a booking system in place at many universities across the UK.

After researching there are a few systems that could be used by Northumbria University, but they do not have integration with the University login system; so anyone with access to the link would be able to book a meeting, which is not ideal. Lancaster University has its own booking setup that is linked to their University login and potentially this is what this project could be depending on how it turns out [2]. Simplybook.me is the best system for this purpose as it creates a whole website for the user/business, where the times available are set and anyone with the link can book a meeting [3]. One drawback is that every lecturer would have to create their own site in simplybook.me which is quite time consuming and tedious. A great feature of simplybook.me which could implement in the project is the option to make a booking recurring, so the same booking can be made every week until a certain point. Another good system is Calendly which my supervisor currently uses to book appointments, though as he said there are a few drawbacks of this system [4]. Like Simplybook.me it uses a unique link for each user for example Calendly/alextuersley which clients will use to book appointments. Calendly allows the user to create different types of meetings such as 30 min or 60 min meetings so it's very customisable but again is not built for the university infrastructure, anyone can use this link to book a meeting so there is limited security, although the user has the option to cancel this meeting and block potential fake meetings from happening. The user interface for Calendly is lot more user friendly than Simplybook.me as it uses a range of bright colour to identify different types of meetings and keeps the booking system as simple as the user wants it with some customisability.

C: Proposed Work

The proposed work of this project is to create a web application booking system for University students to book meetings with lecturers.

This Booking system will be built using a MySQL Database to store the data. This data will include user information, staff schedules, bookings, and departments. The database design will reflect this, as many of the tables will be linked through ID columns.

The clients for this project will be my supervisor, and potentially Northumbria University. User testing will be performed will Northumbria University students, as many of them may be using the system in the future they can provide insight into improving the application. Staff at Northumbria will also be contacted to take part in the testing, especially useful would be Computer Science staff as they can identify any weaknesses in the application and possible improvements.

As the system could potentially hold sensitive user information such as email addresses and phone numbers, security will be and essential part of the application. Current knowledge and research into this field will be required to make the system as secure as possible, although no system is ever 100% secure. OWASP is a company that is seen by developers across the world as the first step towards secure coding. OWASP has a top 10 security risks for web applications, and this will used as a guideline to help making the product more secure [5].

The system will consist of 3 sides, the admin side, the staff side, and the student side. The staff side will be dealing with creating their schedule and changing it based on what they have going on, such as setting days they are away. Staff will be able to setup time slots they are available each day, like opening times for shops, so students can see when they are free. A photo and bio can also be created by the staff member if they choose. The student side will consist of a list of departments and sub dropdown of the lecturers within. The student can then select the lecturer they want to book a meeting with, see their availability and other information and make a booking in their timetable. An email will be sent to both participants when the student has booked. The admin side will mainly deal with the adding and deleting of users and departments.

D: Aims of the Project

- To build a Web Application Booking System; that shows University lecturer's timetables which they choose as time slots they are available.
- The booking system will also allow students to see a lecturer's timetable and availability and make a booking with them based on this.

E: Objectives

- Create a literature review of the product.
    4. Gather information relating to the product such as calendar plugins and booking systems already in use.
    5. Look at the available JavaScript Calendar Plugins and the JavaScript Frameworks that are used with them.
    6. Find existing booking systems, identify their good and bad features, some of which could be implemented in the product.
- Create Diagrams and System prototypes
    7. Use Case Diagram
    8. Class diagram
    9. Database Design (Entity Relationship Diagram)
    10. Wireframe
    11. Navigation Diagram
    12. Flow Charts
- Create Testing Plan
- Identify requirements for product (Calendar plugin, JavaScript Framework, jQuery version)
- Create product based on specification.
- Test products
    5. Functional Testing
    6. Usability testing (students, staff, admin)
    7. Cross browser testing
    8. Vulnerability testing
- Make changes to product based on testing feedback and conclusions
- Evaluate the product
    6. Build Quality
    7. Testing
    8. Review the tools and techniques used, and state whether another approach could be used
    9. Review of the literature and how it was used in the product
    10. User Feedback
- Evaluate the project process and own performance
- Write Introduction and Abstract

F: Skills

<u>Software Design</u>

As this is a software project, database designs, UML diagrams and a wireframe will be created. I have previously designed a wireframe but not on this large of a scale. UML diagrams are a new concept but with some research I will be able to figure them out and I have designed database tables before on placement, so I will find a good database design software and use that.

<u>PHP</u>

I have been coding in PHP since first year and spent a year in the industry as a Web Developer so many of the skills needed for this project, I am already familiar with. The hardest skill will be manipulating the staff time slots

from time slots through PHP into data that can be stored in an SQL database. The use of PHP classes will be essential as they will be used to run each different aspect of the system.

JavaScript

I also have previous knowledge of JavaScript, but as a Calendar plugin will be used this is a new area for me. The only previous plugin I have used if calendar.js but I will find a suitable plugin for this system and get to grips with how it works and how I can display the staff schedules and bookings. This will require some research and the plugin I choose must have a lot of documentation, as this will be needed to complete many tasks.

SQL

I have previous done a lot of SQL both on MySQL and Microsoft Web Servers, which are very similar. I will need to design appropriate queries for the PHP classes to pull and update from the database. The database will also have to be designed appropriately to store all of the user data.

CSS

I have previous knowledge of CSS, but some learning and re-learning may be required to get the booking system looking professional. As this may be used by the University the website should be able to be used both on web browsers and on mobile, so some knowledge of CSS is essential.

G: Sources of Info/bibliography

[1] J. Kujanpaa, "Developing a Booking System-Case: Tuas Cisco Laboratory," *Dev. a Book. Syst.*, p. 35, 2015. (accessed Oct. 21, 2020).

[2] "Appointments: Using the Appointments Booking Explorer - Help Center." https://ask.springshare.com/libcal/faq/1521 (accessed Oct. 22, 2020).

[3] "SimplyBook.me - Free Appointment Scheduling Software." https://simplybook.me/en/ (accessed Oct. 24, 2020).

[4] "Free Online Appointment Scheduling Software - Calendly." https://calendly.com/ (accessed Oct. 22, 2020).

[5] "OWASP Top Ten Web Application Security Risks | OWASP." https://owasp.org/www-project-top-ten/ (accessed Oct. 23, 2020).

H: Resources/Statement of Hardware/ Software Required

- Visual Studio Code will be used as the code editor as it has PHP plugins to assist with coding. An alternative code editor to use would be PHPStorm as it has Intellisense, and error detection built in
- GitHub desktop will be used store the booking system as there can be different branches for testing features and it can be used on multiple devices. A backup will also be stored on a hard drive, in case GitbHub servers go down.
- Mamp will be used as a localhost to test the booking system without users as this would be a conflict of the GDPR regulations. xammp is an alternative localhost server that can be used if Mamp isn't working
- When the product is being tested by users it on will be newnumyspace, the actual user data will only be on this database. It will be removed after testing has been completed. An alternative live server could be used as a VM Webserver can run the product if newnumyspace goes down.
- FileZilla will be used to put the booking system onto newnumyspace.
- To create an entity relationship diagram dbdiagram.io will be used as it is free software and the diagram create can be very detailed.
- UML diagram software will be used either for free or through the use of a trial period.

- Chrome will be used to test booking system as it is the most widely used Web Browser. Cross browser testing will involve Firefox and potentially Microsoft Edge.
- A personal PC will be used to create and run the code, along with a personal laptop, as the code is on GitHub an alternative computer could be used in the event of both devices breaking.

I: Structure and contents of project report

1) Abstract
2) Introduction
   a) Aims and Objectives
   b) Product Overview – features, purpose, and characteristics
   c) Methods and tools
3) Analysis
   a) Literature Review - Similar Booking Systems, possible approaches
   b) Requirements based on Specification
4) Synthesis
   a) Design – UML Diagrams, Entity Relationship Diagram, Wireframe
   b) Product Code
   c) Testing – Plans, Results
   d) Changes made after testing
5) Evaluation
   a) Evaluation of the Product
   b) Evaluation of the Process
6) Conclusion
   a) Aim
   b) Issues
   c) Changes to the project if repeated
   d) Further Work
7) References
8) Appendices

<u>Appendices</u>

Terms of Reference
Existing Booking System screenshots
Diagrams
Booking System
Test Plan
Code Snippets
Test Results

J: Marking Scheme

Software Engineering Project

Report: 40%

Abstract & Introduction	5%
Analysis	30%
Synthesis	30%
Evaluation & Conclusions	30%
Presentation	5%

Product 50%

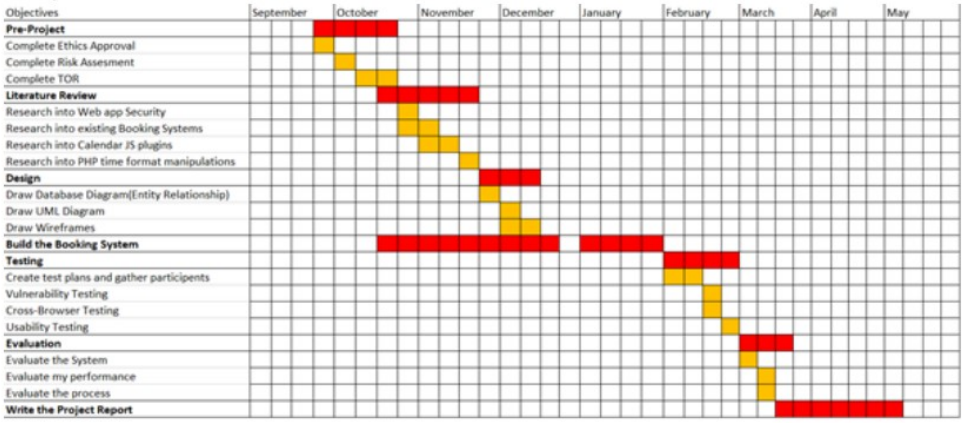
Fitness for Purpose	40%
Build Quality	60%

Viva 10%

K: Deliverables

- Create a web booking system with secure login, sign up and change password capabilities. Emails may be sent to users when a password is forgotten or to activate account, if Emails are not enabled an admin can activate the account.
- Using a JavaScript calendar plugin and PHP, a page for staff users to set a schedule up and set dates they are away in the booking system
- A page for student users to view all staff members and create bookings with them based on their schedule
- The system will be styled appropriately for ease of use
- Testing plan for student and staff users to test the system
- Testing results for both staff and student users based on a predetermined testing plan, the system may change based on testing feedback
- Flow Charts, Class diagram, Entity relationship diagram, and a wireframe will be created for the system to outline the design and how the whole system will work.

L: Project Plan



Appendices

Ethics Form

	Complete
Full Name	Alexander Tuersley
Northumbria email address	alexander.tuersley@northumbria.ac.uk
Faculty	Engineering and Environment
Department	Computer and Information Sciences
Submitting as (choose one from the drop down list →)	Undergraduate student

Has this project received full ethical approval from an external organisation?	
Module code (UGT/PGT only)	KV6003
Module Tutor (UGT/PGT only)	Xiaomin Chen
Research Supervisor	Nick Dalton
Co-Investigators	Nanlin Jin

GENERAL AIMS AND RESEARCH DESIGN OF YOUR PROJECT

Title of your research project	Web Booking System for University
Outline General Aims and Research Objectives (*State your research aims/questions (maximum 500 words). This should provide the theoretical context within which the work is placed, and should include an evidence-based background, justification for the research, clearly stated hypotheses (if appropriate) and creative enquiry.)*	• To build a Web Application Booking System; that shows University lecturer's timetables which they choose as time slots they are available. • The booking system will also students to see a lecturer's timetable and availability and make a booking with them based on this
Research Activities (Please give a detailed description of your research activities **Please provide a description of the study design, methodology (e.g. quantitative, qualitative, practice based), the sampling strategy, methods of data collection (e.g. survey, interview, experiment, observation, participatory), and analysis. Do sensitive topics such as bereavement; sexuality; drug use; abuse; body image; trauma; pornography; physical or mental health; religious beliefs; bullying; whistle-blowing; prisons; criminality; cybersecurity inform the research? Are vulnerable participants, and or, those lacking the ability to give informed consent participating? If so, how will you address these issues?)**	Research activities will be User testing which will only include Northumbria students and Staff. I can also use dummy data and not actual personal data for testing to avoid GDPR issues.

Does your project involve:		**RESPONSE**
1	Gathering data or information from human participants (e.g. via questionnaire / interview/survey/experiment/ social media/ VR)	No
2	Collecting personal data, i.e. name, email, home address, computer IP address, phone number etc.	Yes

3	Analysis of secondary data **NOT** in the public domain (e.g. archive material that require organisational membership)	No
4	The collection or use of information which is 'commercially sensitive'	No
5	Financial inducements other than expenses and compensation for time.	No
6	Ministry of Defence, EU Security Funding	No
7	Cybersecurity issues including cyberattack	No
8	The collection of data/information that might be confidential or classified (e.g. protected by the Official Secrets Act)	No
9	Gathering data/information at a location external to Northumbria University campuses and franchised locations.	No
10	Collection of samples such as plants, soils etc, that might disturb the environment or archaeological remains.	No
11	Research involving animals or materials derived from animals	No
12	Anything else which means that the research poses greater than minimal ethical risk	No

PROJECT SCORING OUTCOME	MEDIUM/HIGH

Answering 'NO' to questions 1 - 12 above renders the project 'LOW' ethical risk.

Please now complete Mandatory Sheets 3 and 5.

If one or more 'YES' answers have been given in questions 1 - 12, your project is 'MEDIUM' or 'HIGH' ethical risk.
Please continue answering questions below to determine the appropriate risk level for proportionate review.

Does your project involve:		**RESPONSE**
13	Sensitive topics e.g. bereavement; sexuality; sexual behaviour, drug use; abuse or exploitation; body image; trauma; pornography; physical or mental health; religious beliefs; gender, ethnicity, bullying; whistle-blowing; prisons; criminality; cybersecurity etc.	No
14	Potentially vulnerable people or groups, for example children and young people (under 18's), those with a learning disability or cognitive impairment.	No

15	Intrusive interventions: the use of drugs or other substances (e.g. food, drink, placebos, or drugs); procedures involving physical distress (e.g. prolonged or repetitive testing); emotional distress (e.g. stress or anxiety)	No
16	Analysis or direct observation of activities during which criminal offences may occur (e.g. hunting; drug dealing).	No
17	Collection of data relating to extremism, radicalisation, or terrorism.	No
18	Funding from a source that may be controversial (e.g. due to the nature of the funder or a conflict of interest.)	No
19	Covert methods of investigation or deception	
20	International partners or research undertaken outside of the UK where there may be issues of local practice and political sensitivities. In these instances it will be necessary to act in accordance with the legal and ethics review requirements in the countries included in the research and demonstrate awareness of these.	No
21	Research where participants or other individuals may be identifiable in the material used or generated either directly (images or sound recordings) or indirectly (through aggregating separate data sources)	No
22	Access to records of personal or sensitive confidential information, including genetic or other biological information concerning identifiable individuals.	
23	Participants recruited or identified through the internet e.g. closed discussion groups	No
24	Individuals or groups where permission of a gatekeeper is normally required for initial or continued access to participants (e.g. adult professionals, family member, community leader)	No
25	Sharing of data or confidential information beyond the initial consent given?	No
26	Other considerations that mean that this research should be treated as 'high risk'	No
27	Persons who lack capacity to make decisions, e.g. people with dementia or learning disabilities.	No
28	Recruitment of patients, staff or volunteers through the NHS, or the use of NHS patient data?	
29	The collection of bodily tissue e.g. blood, saliva, urine samples from living or deceased persons.	No
31	Will the study involve participants, or their data, from adult social care, including home care, or residents from a residential or nursing care home?	No
32	Collection of data from patients or staff recruited via the NHS or Health and Social Care agencies.	No

33	Funding from the Department of Health?	No
34	A health related study or clinical trial of an investigational medicinal product or a medical device?	No
35	Funding, sponsorship, or the involvement of Ministry of Defence.	No
36	Direct testing on animals or materials derived from animals	No

PROJECT SCORING	MEDIUM
EXTERNAL ETHICAL REVIEW REQUIRED	NO

RESEARCH DATA MANAGEMENT PLAN		**Complete**
1	**Anonymising Data** Describe the arrangements for anonymising data and if not appropriate explain why this is and how it is covered in the informed consent obtained.	If any un-anonymous data is used in my project, informed consent will be required through the use of a form, signed by the participant.
2	**Storage Details** Describe the arrangements for the secure transport and storage of data collected and used during the study. Please explain the kind of storage you intend to use, e.g. cloud-based, portable hard drive, USB stick, and the protocols in place to keep the data secure. If you have identified the requirement to collect 'Special category data', please specify any additional security arrangements you will use to keep this data secure.	The data stored in my booking system will only be on my newnumyspace account within the University. This is protected by my login and only those who know how to use the system can gain access to the server data. After user testing has been completed the personal data will be removed from the database. Dummy data will be used for testing on my localhost.
4	**DECLARATION** I confirm that I will comply with the University's data retention schedule and guidance.	Yes

RESEARCH PROJECT TIMESCALE

5	Proposed start date	23/10/2020
6	Proposed end date	24/05/2021

SECURITY SENSITIVE INFORMATION (INCLUDING PREVENT-RELATED MATERIAL)

7a	Will you require access to, or use of, material that relates to extremism, radicalisation and/or terrorism (including extreme/terror groups)?	No
7b	If yes, please provide details	
8a	Was your research commissioned by the Military or under an EU security call?	No
8b	If yes, please provide details.	
9a	Will you require access to, or use of, material that is prohibited/restricted (e.g. under Government security classifications or the Official Secrets Act)?	No
9b	If yes, please provide details.	

HEALTH AND SAFETY

Before completing this section please refer to the Guidance in this workbook

10a	Are there PHYSICAL risks associated with the research project work?	No
10b	If Yes, I confirm that a risk assessment has been approved and will be attached when submitting this workbook	
10c	If No, I can confirm that there are no physical risks associated with this project and so no risk assessments are required.	Yes
11	I confirm that I have read and understood the University's Health and Safety Policy (see links on Guidance page of this workbook)	Yes
12	I confirm that I have read and understood the University's requirements for the mandatory completion of risk assessments in advance of any activity involving potential physical risk.	Yes

INSURANCE

13	I confirm that I have read and understood the University Insurance guidance document *(see links on Guidance page of this workbook)*	Yes
14	I confirm my work is covered by University Insurance. I confirm an **insurance** risk level of	Medium

EXTERNAL ETHICAL APPROVAL

15a	Has external ethical approval been granted for this project e.g. NHS REC, MoD REC?	Not applicable
15b	Please confirm which organisation has approved ethics documentation for this project?	
15c	Have you uploaded confirmation of external approval with this application?	Not applicable

ADDITIONAL INFORMATION

16a	Has your project received external funding?	No
16b	If Yes, confirm the name of the funder and any other details you have	
17a	Is there a Collaborative Venture (Franchise Programme Organisation) organisation involved in this research project?	No
17b	If Yes, please provide details	

PEOPLE AND/OR PERSONAL DATA		**Complete**
1a	Does your project involve people and/or personally identifiable data? (including where individuals may be identified in the material used or generated either directly (sound or images) or indirectly through aggregating separate sources of data).	Yes
1b	If Yes, please provide details of the sample groups that will be involved in the study and include details of their location (whether recruited in the UK or from abroad, or through the internet, or internet discussion forums) and any organisational affiliation. For most research studies, this will cover: the number of sample groups; the size of each sample group; the criteria that will be used to select the sample group(s) (e.g. gender, age, sexuality, health conditions). If the sample includes vulnerable participants, please state this clearly. If the sample will include NHS staff or patients, please state this clearly. If this is a pilot study and the composition of the sample has not yet been confirmed, please provide as many details as possible.	The sample groups will be Northumbria University students and staff.
2	Does your project involve data pertaining to **Living Individuals**? *(including still or moving images)*	Yes

2b	If Yes, please specify the nature of this data (i.e. name, email, home address, computer IP address, phone number, genetic or biological data), and (if appropriate) include details of the relevant individuals who have provided permission to utilise this data; attach evidence of these permissions when submitting this workbook for review.	As it is a booking system user's names, email address and phone number will be stored on the system, They can also choose to upload a photo.
3a	Does your project involve any **Special Category data?**	No
3b	If Yes, please provide details of any Special Category Data. *If you will be collecting data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, data concerning health or data concerning a natural person's sex life or sexual orientation, please specify which categories you will be using.*	
4	Please confirm the **legal process for processing personal data**. *Under the General Data Protection Regulation and the UK Data Protection Act 2018 any organisation processing personal data of EU citizens for any purpose (including research) must have an appropriate legal basis for this and communicate it to all participants. For research, in most cases the appropriate legal basis will be "Article 6(1) e: processing is necessary for the performance of a task carried out in the public interest". If you are collecting special categories of personal data (see above) then you will need an additional legal basis. For research, in most cases the above) then you will need an additional legal basis. For research, in most cases the appropriate additional legal basis will be "Article 9(2) j: processing is necessary for scientific and historical research purposes". Further detailed guidance on this is available in the Research Ethics and Governance Handbook* https://www.northumbria.ac.uk/research/ethics-and-governance/	processing is necessary for scientific and historical research purposes
5	Will you be using secondary data which is **NOT** in the public domain?	No

6	**Data Gathering Materials** Provide a detailed description of what the participants will be asked to do for the research study, including details about the process of data collection (e.g. completing how many interviews / assessments, when, for how long, with whom). Please submit any relevant documentation with this workbook for review.	Student participants will be asked to sign up to the system, login to the system. They will then select a staff member and create a booking. Staff participants will be asked to sign up to the system, login, and setup their schedule in the system.
7	Please confirm the type of Consent related to your project. Please specify if the sharing of data goes beyond the original consent given. Note: Copies of information sheets and consent forms must be submitted with this workbook. If the study involves participants who lack the capacity to consent, procedures in line with sections 30-33 of the Mental Capacity Act will need to be put in place. If you are using alternative formats to provide information and /or record consent (e.g. images, video, or audio recording), provide brief details and outline the justification for this approach and the uses to which it will be put. If you are using a covert method of research, you must address how this is integral to the research.	Informed Consent via consent form, none of the participant data will be shared
8	**Researcher and Participant Safety Issues** If there any risks the research could cause any discomfort or distress to participants (physical, psychological, or emotional) describe the measures that will be put in place to alleviate or minimise them. Please give details of the support that will be available for any participants who become distressed during their involvement with the research.	

9	**Recruitment** Describe the step-by-step process of how you will contact and recruit your research sample and name any organisations, gatekeepers, or groups that will be approached. Your recruitment strategy must be appropriate to the research study and the sensitivity of the subject area. You must have received written permission from any organisations or groups before you begin recruiting participants. Copies of draft requests for organisational consent must be attached to this workbook when submitted together with copies of any recruitment emails/posters that will be used in your study.	For the Northumbria students I will ask people who I know from my course and people within my household to participate. As for the staff I will send emails to staff members requesting participation, mainly to staff in the Computer Science department who I am familiar with.
10a	**Remuneration** Will you make any payment or remuneration to participants or their carers/consultees?	No
10b	If Yes: Please provide details/justifications. *Note that your Faculty may have specific guidelines on participant payments/payment rates etc and you should consult these where appropriate.*	
	Commercially Sensitive Data	
11a	Does your project involve commercially sensitive data?	No
11b	Please provide details of how this data will be managed.	No
	Criminal Activity	
12a	Does your project involve analysis or direct observation of activities during which criminal offences may occur (e.g. hunting; drug dealing.	No
12b	Please provide details of the type of activity and how this will be managed.	
	Environmental Data	
13	Does your project involve the collection of **environmental** data? Please provide details of how you will minimise any disruption to the environment or archaeological artefacts.	No
	Human Tissue & Physical Procedures	
14a	Does your project involve the collection and or use of **human tissue**?	No

14b	Use of food products and/or drinks (e.g. herbal supplements, medicine, placebos, drugs, or other substances).	No
14c	Invasive, intrusive, potentially harmful, or painful procedures (e.g. giving injections, or exposure to ionising radiation)	No
14c	Collection of bodily tissue e.g. blood, saliva, urine samples.	No
14e	Procedures involving physical distress (e.g. prolonged or repetitive testing); emotional distress (e.g. stress or anxiety)	No
14f	A Clinical Trial or a Clinical Trial of an Investigational Medicinal Product (CTIMP); or the use of medical devices.	No
14g	Describe the measures and controls that will be used in the project to mitigate physical risk to participants and to alleviate any discomfort.	
Animal Subjects		
15a	Does your project involve experimenting on **animals**, or **material derived from animals**? *N.B. Northumbria University is not an ASPA designated establishment therefore any regulated procedures must be performed, under full licensing conditions, at an appropriate designated establishment (usually another* UK university).	No
15b	What process are in place to minimise and mitigate any harm or discomfort that the project may involve? Full details of experimental design with consideration of 3R's and relevant H.0. licencing details must be supplied (e.g. PPL/PIL numbers).	
International Partners		
16a	Does your research involve working with International partners or research undertaken outside of the UK where there may be issues of local practice and political sensitivities?	**No**
16b	Please describe how you will act in accordance with the legal and ethical review requirements in these countries?	

17	Potential Ethical Issues Please describe any potential ethical issues the project may have which are not covered above (e.g. funding source), and how you have sought to minimise these.	

DBS (Disclosure and Barring Service) Clearance Requirements (previously Criminal Records Bureau Check)

18a	Does your research involve children (aged under 18) and/or vulnerable adults?	No

DECLARATION

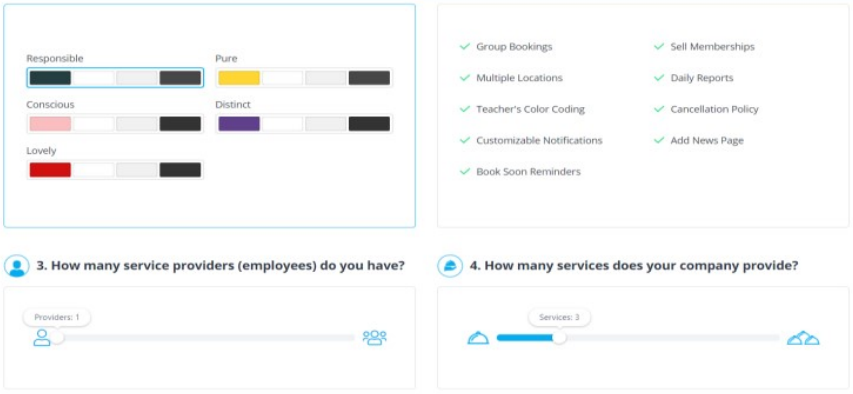
Staff/Students should read the Declaration(s) then complete blue fields

Student Declaration	I confirm my supervisor has reviewed the contents of this document

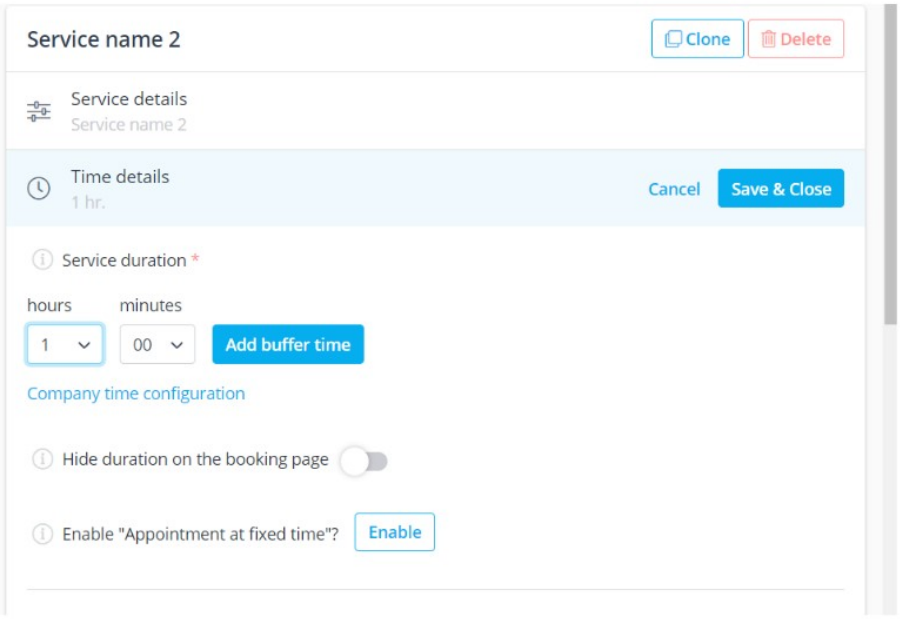
Student and Staff Declaration	I confirm I have assessed the ethical risk level of my work correctly and answered the above sections as fully and accurately as possible.

	Complete
Full Name	Alexander Tuersley
Date	23/10/2020

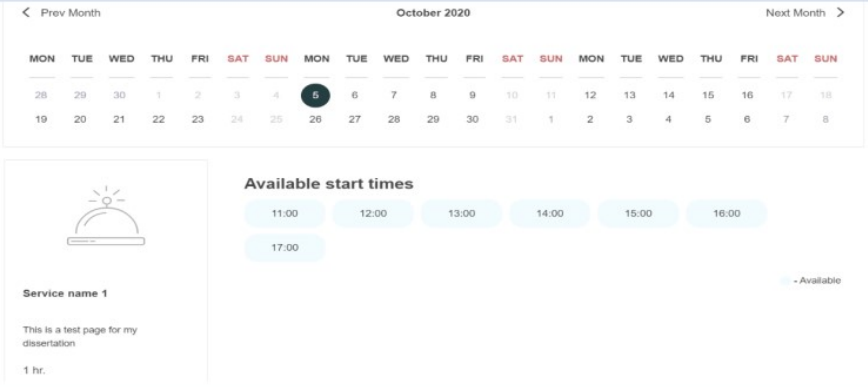
APPENDIX 2 – SIMPLYBOOK.ME DESIGN



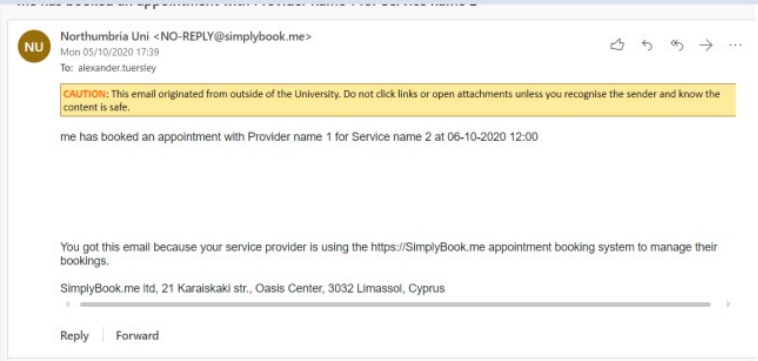
APPENDIX 3 – SIMPLYBOOK.ME SERVICE



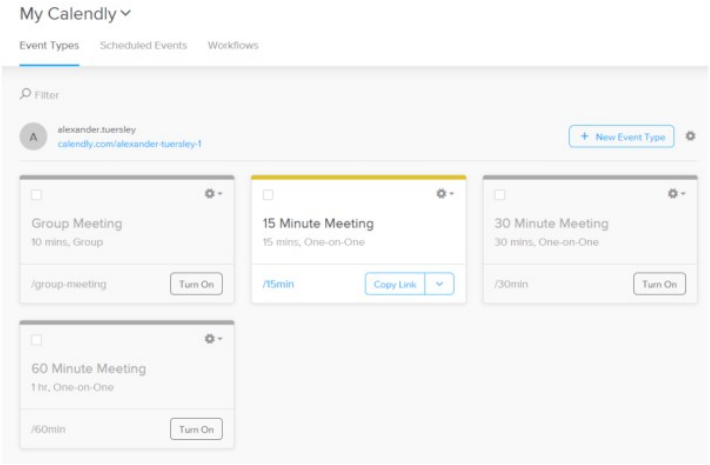
APPENDIX 4 – SIMPLYBOOK.ME SCHEDULE



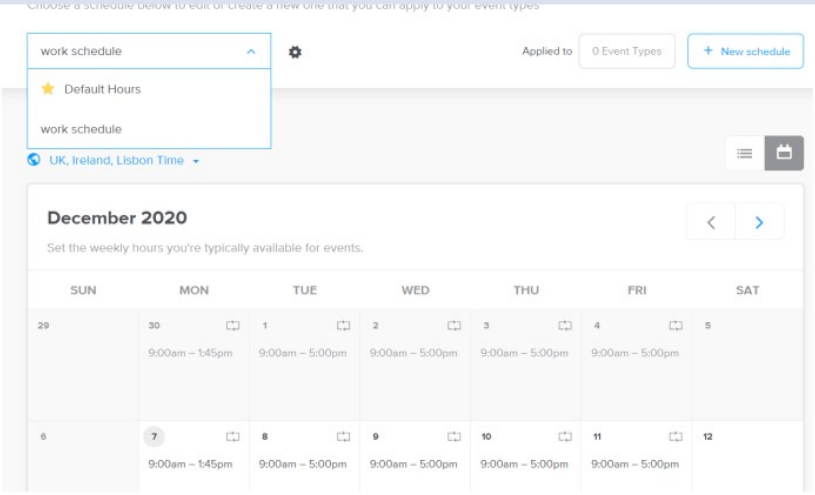
APPENDIX 5 – SIMPLYBOOK.ME EMAIL



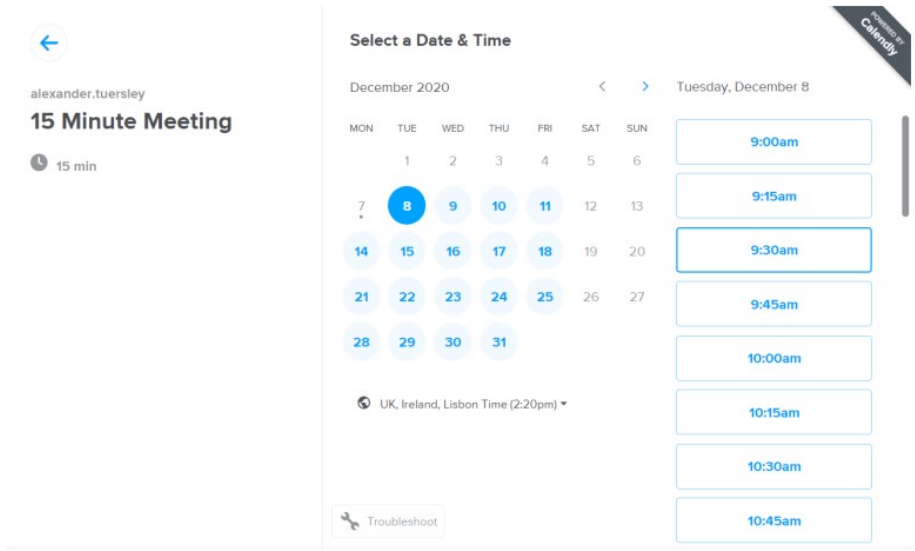
APPENDIX 6 – CALENDLY MEETINGS



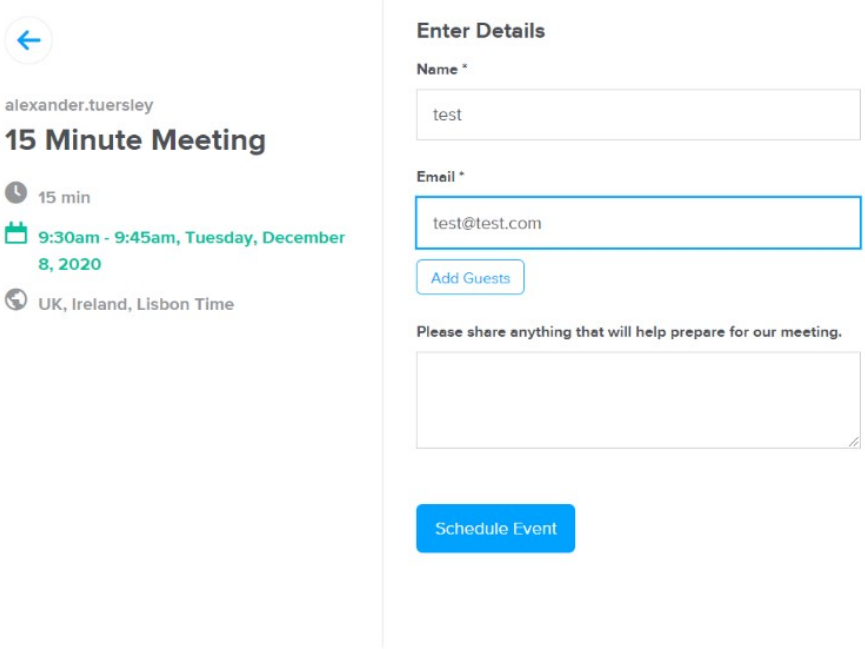
APPENDIX 7 – CALENDLY SCHEDULE



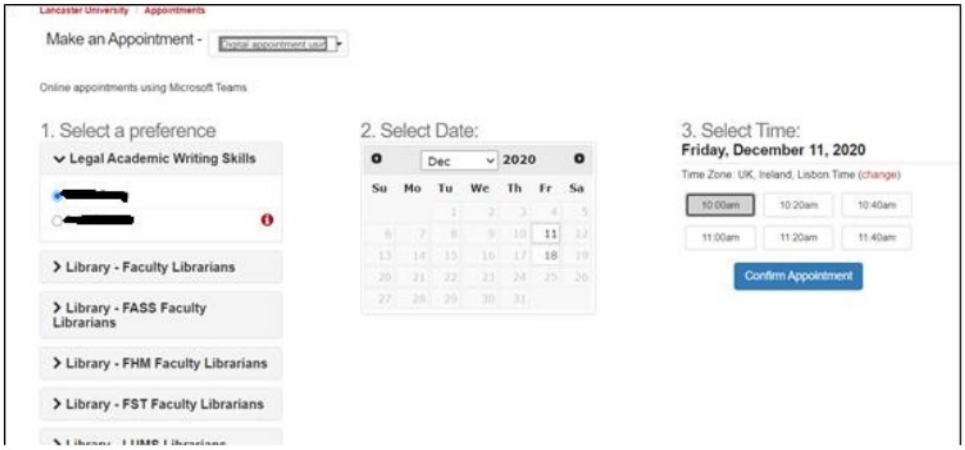
APPENDIX 8 – CALENDLY CALENDAR



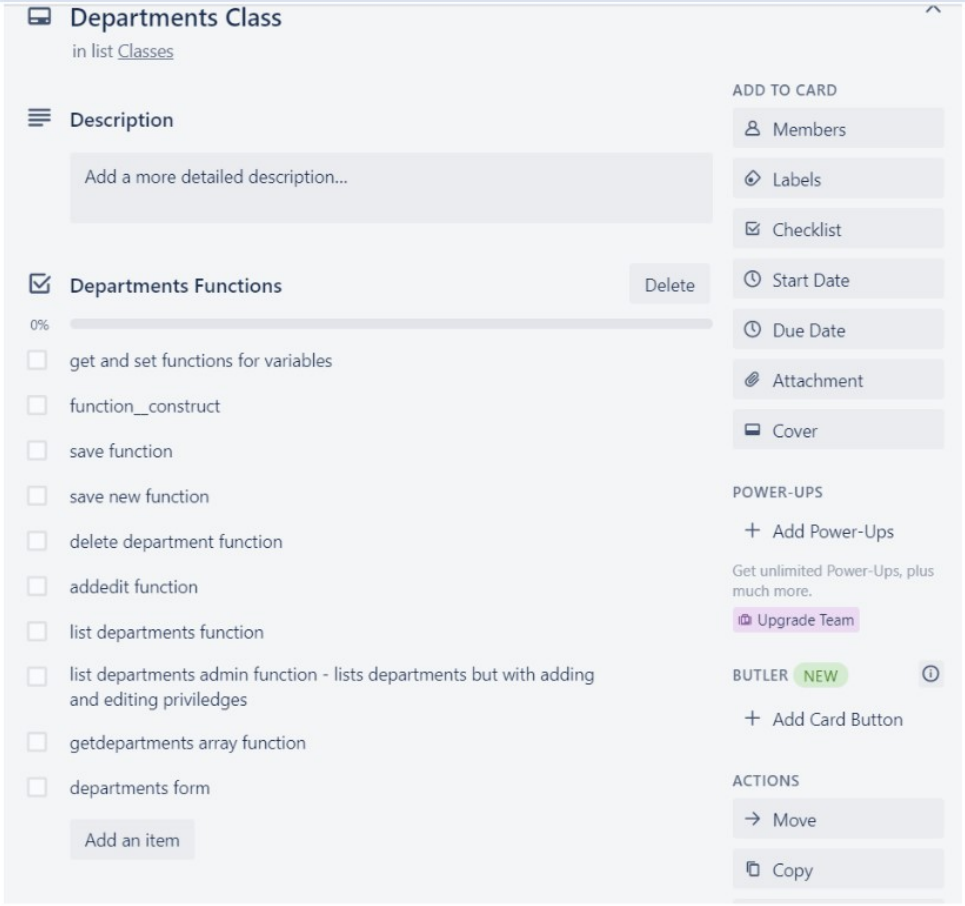
APPENDIX 9 – CALENDLY BOOKING



APPENDIX 10 – LANCASTER BOOKING SYSTEM



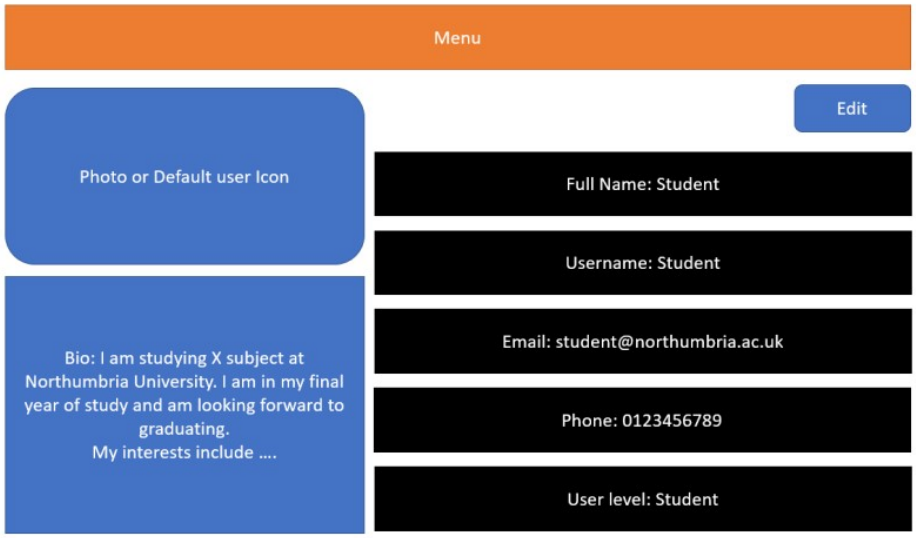
APPENDIX 11 – TRELLO DEPARTMENTS CLASS



APPENDIX 12 – REQUIREMENTS SPECIFICATION

No	Requirement	Priority
1	**Separate Interfaces for Staff, Students and Administrators**	**Low**
2	**Login and Sign up System**	**High**
2.1	Password Recovery	Medium
2.2	Admin to be able to add new Users	Low
2.3	Use cookies to allow system to remember User without logging in twice	Low
3	**Setup for Staff**	**High**
3.1	Staff to be able to setup hours of work for each day (Monday- Friday), can be separate times slots e.g. 10-11 and 1-3 for availability to book meetings	High
3.2	Hours are repaeated over each week unless specified by staff member	Medium
3.3	Meeting Types coulld be used to differentiate between different types of meetings	High
3.4	Meetings schedule to the nearest 5 mins	Low
3.5	Possibly email a link to the booking for students and staff	Low
3.6	Bookings page with all available bookings and ability to edit and cancel booking	High
4	**Staff Page as viewed by Student**	**Medium**
4.1	Name, Department, Location (Office or Teams profile), contact details and information about staff member should be available to view. Possiby a photo as well	Medium
4.2	Ability to set how long in advance students can book	Low
4.3	Ability for students to book by meeting type, should be named and have individual time periods	Low
4.4	Group meetings(Optional)	Low
5	**Setup for Students**	**High**
5.1	Student page to show bookings with ability to edit and cancel	High
5.2	Staff page to show list of staff members available for bookings	High
5.3	About me page with their information	Medium
5.4	Possibly link to a calendar(Optional)	Low
6	**Booking Process**	**High**
6.1	List of Staff Departments with Staff members inside	High
6.2	Select Staff member to make a booking	Medium
6.3	When Staff member is selected shows the staff members available meeting types	Medium
6.4	After selecting a meeting type shows the staff's schedule and available booking times	High
6.5	Option to make booking weekly	Low
6.6	Option for group bookings(Optional)	Low
6.7	Create notes for booking with email being sent when a new note is added(Optional)	Low
7	**Staff Booking**	**Medium**
7.1	Staff can make booking with student if they have an email	Medium
7.2	Staff can add booking outside of established hours	Low
8	**Day of Booking**	**Low**
8.1	Email/Text to remind Staff/Student of booking on the day of booking about 8:00	Medium
9	**Canceling**	**Medium**
9.1	Students/Staff can cancel anytime via the webpage	Medium
9.2	Email sent to both about the cancellation	Low
10	**Re-arranging**	**Low**
10.1	A mix of cancelling and re-booking for students, staff could edit the booking(Optional)	Low
10.2	Sends Email about the change to both parties	Low
11	**Administrator**	**Medium**
11.1	Ability to delete Staff/Students	Low
11.2	Ability to make any edits	Medium
11.3	Ability to reset user passwords	Medium
12	**General**	**Medium**
12.1	Compliant with GDPR	High
12.2	Works with phone and desktop browsers	Medium
12..3	Northumbria Branding	Low

APPENDIX 13 – ABOUT ME PAGE DESIGN



APPENDIX 14 – INITIAL WIREFRAME DESIGN

Student Booking System

Northumbria University Logo

username

password

Login button

Not registered? Sign up(link to sign up page)

Student home

Search bar

List of Departments:

Staff listed by department.

Dropdown list changes based on department clicked.

Staff Member Information

Photo(if not provided default image)

Bio (bit of information about the lecturer)

Click to goto student booking page

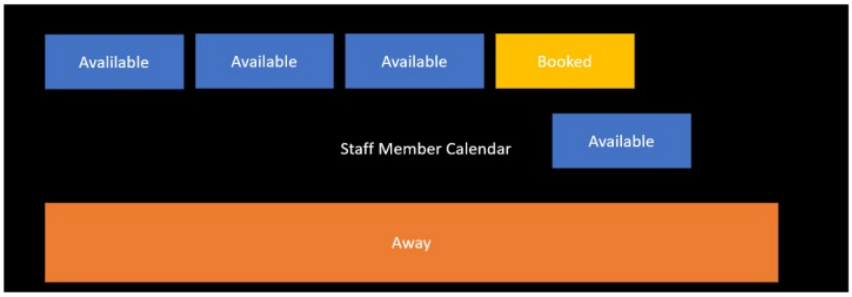
Email

Phone Number

Department

Student Booking

Student double clicks available booking to goto booking page



Booking Confirmation (student)

Booking time shown e.g. 10:30-11:30

Add note to lecturer Recurring booking (tick box)

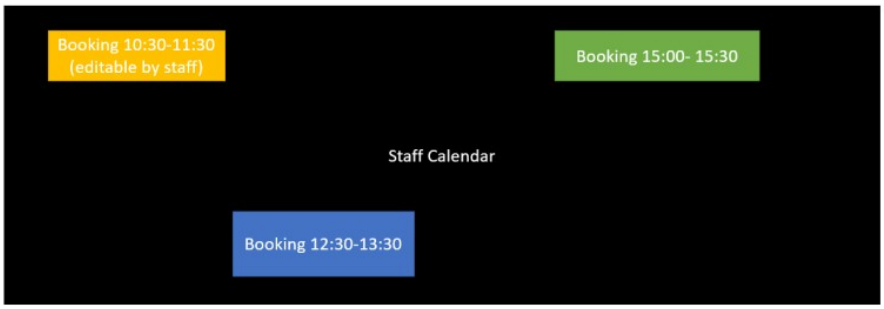
Send Reminder(tick box)

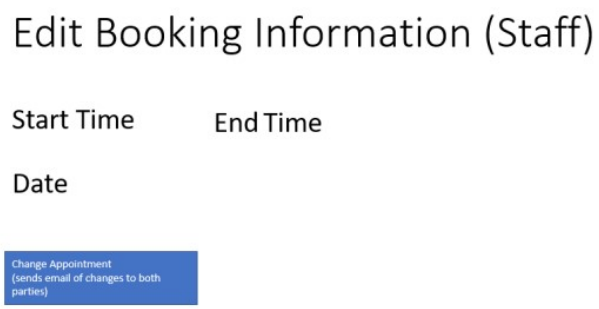
Confirm

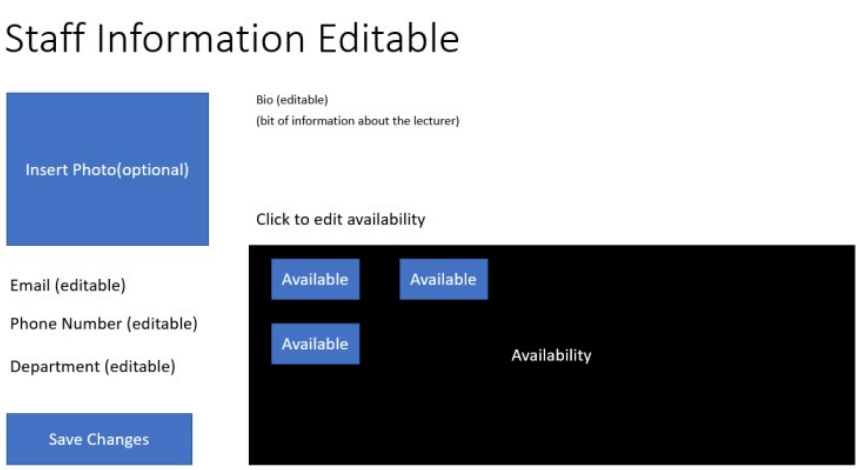
Staff home

Number of bookings for the week

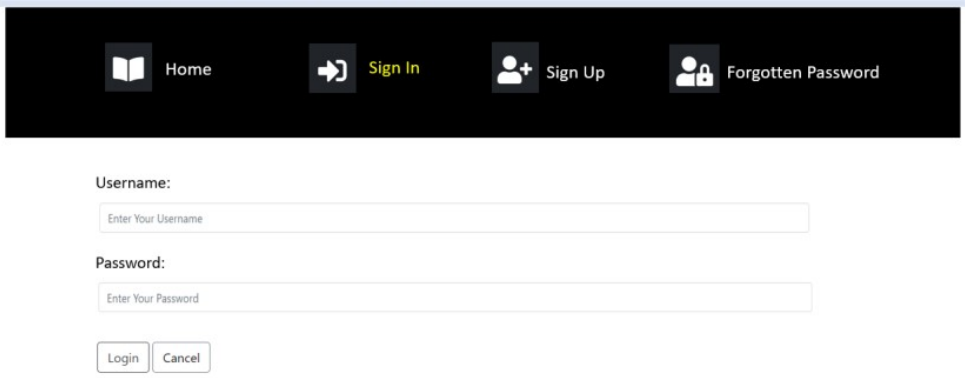
Next appointment and information about appointment



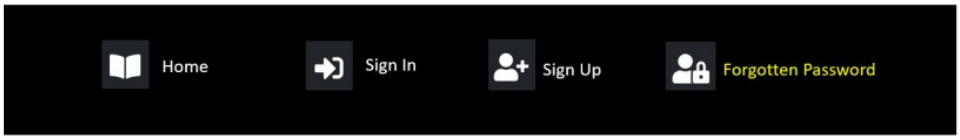




APPENDIX 15 – SIGN IN PAGE WIREFRAME



APPENDIX 16 – FORGOTTEN PASSWORD WIREFRAME



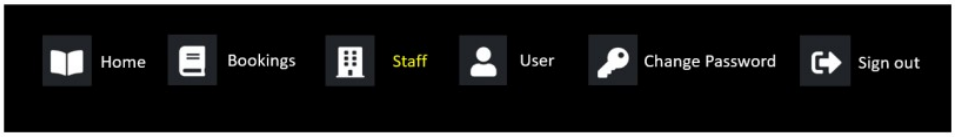
If you have forgotten your password, please complete this form and your password will be sent to your email.

Email:

Enter your Email

Forgotten Password | Cancel

APPENDIX 17 – DEPARTMENTS PAGE WIREFRAME



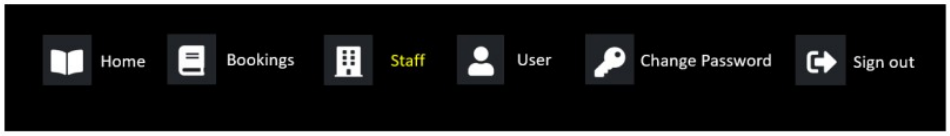
The list below shows all departments. Click on a Department to see the Staff members.

Department	Number of Staff
Business	10
Art	5

Displays the Departments name with a clickable link

Number of Staff Members within the Department

APPENDIX 18 – DEPARTMENT USERS PAGE WIREFRAME



The list below shows all Staff within the Business department. Click on a Staff member to see their schedule.

← Departments

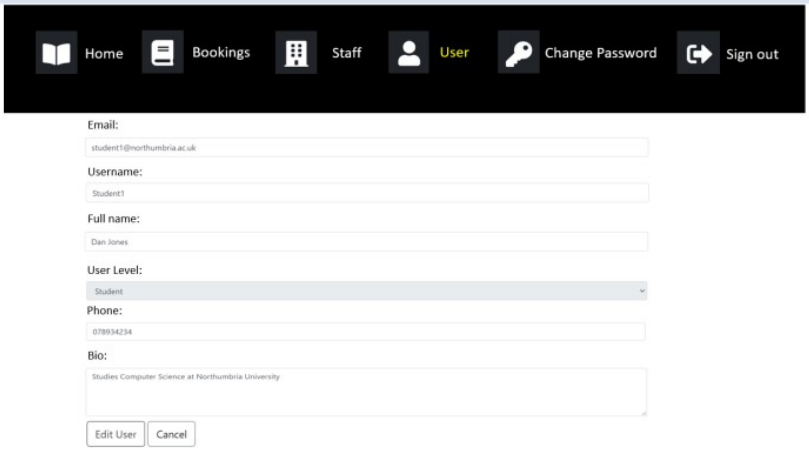
Shows the name of the department the User has selected

Staff Member	Email
Lecturer1	lecturer1@northumbria.ac.uk
Lecturer2	lecturer2@northumbria.ac.uk

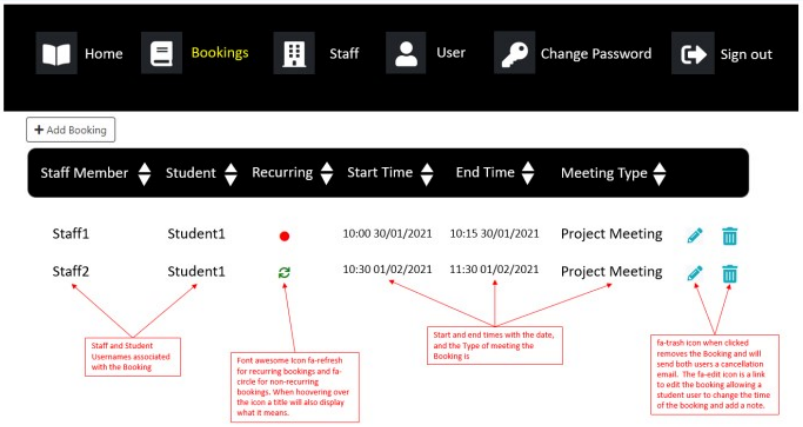
Displays the Staff Members name with a clickable link

Displays their email address in case the student wishes to contact them directly

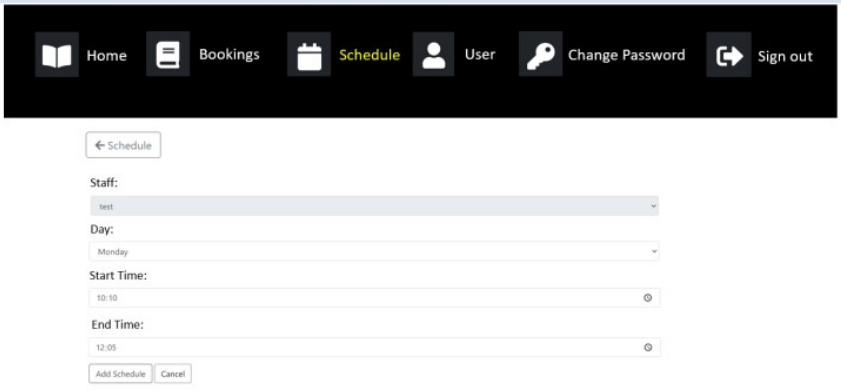
APPENDIX 19 – USER EDIT PAGE WIREFRAME



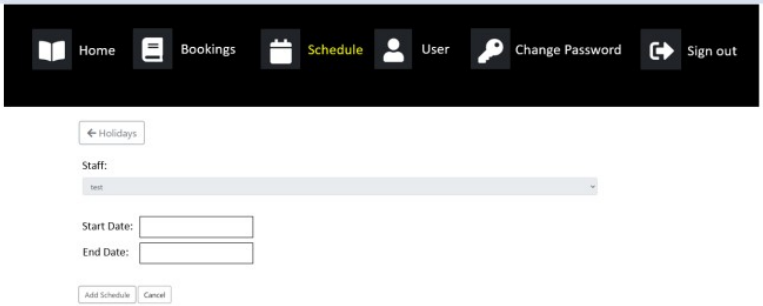
APPENDIX 20 – STAFF BOOKINGS PAGE WIREFRAME



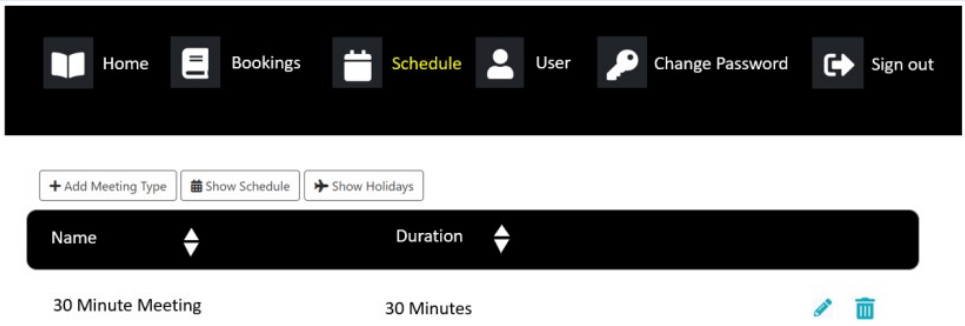
APPENDIX 21 – ADD SCHEDULE WIREFRAME



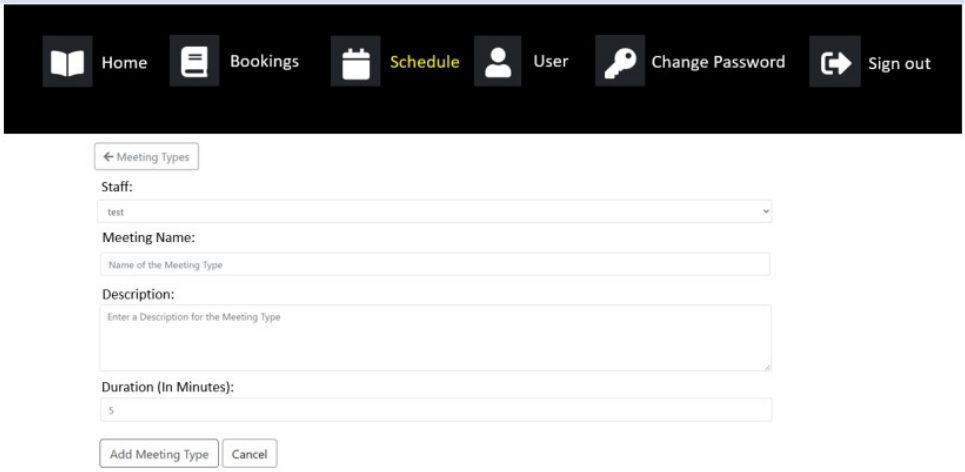
APPENDIX 22 – HOLIDAYS FORM WIREFRAME



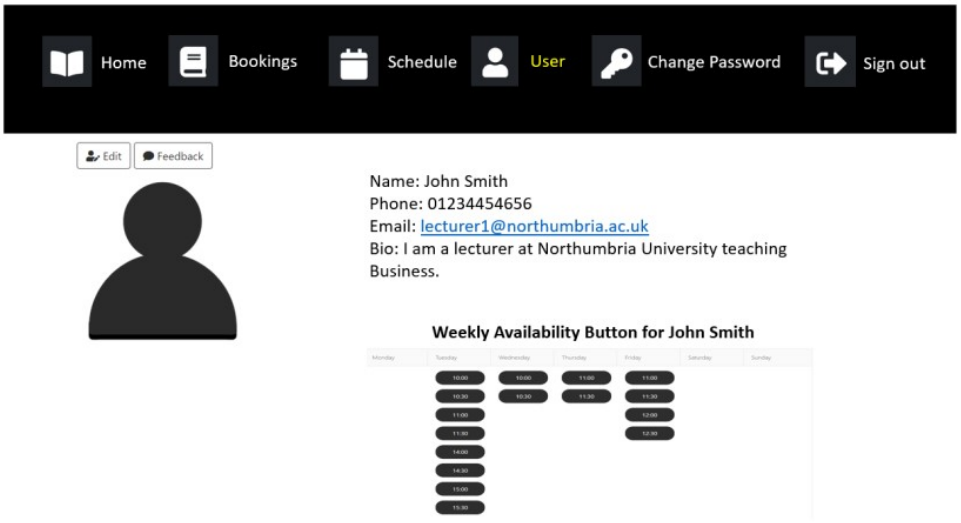
APPENDIX 23 – MEETING TYPES WIREFRAME



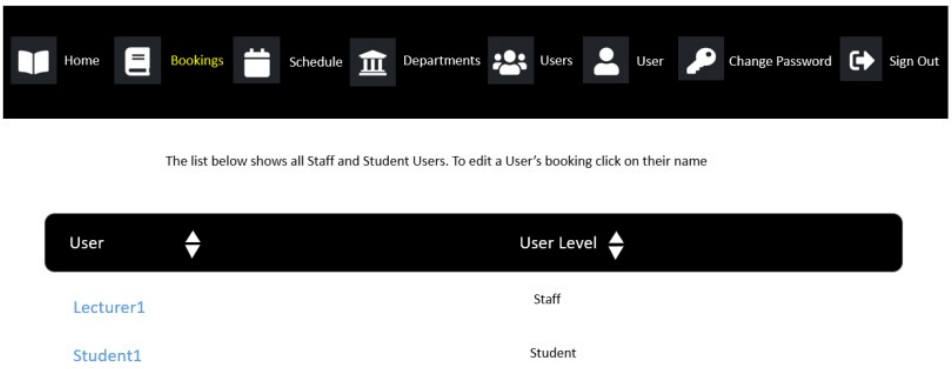
APPENDIX 24 – MEETING TYPES FORM



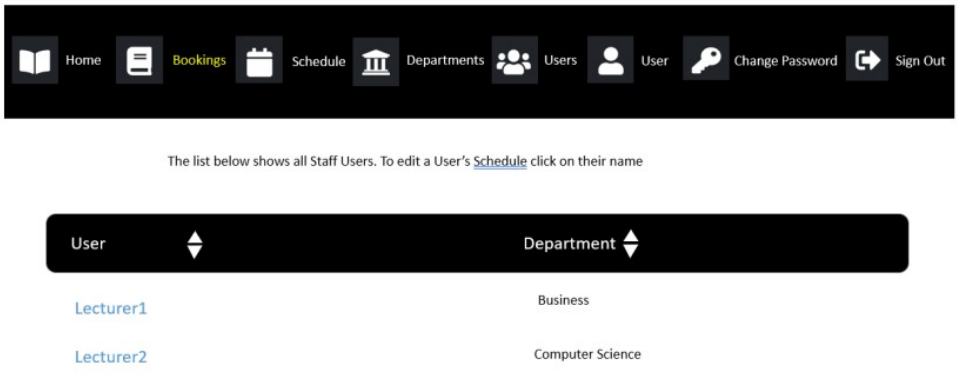
APPENDIX 25 – STAFF USERS PAGE



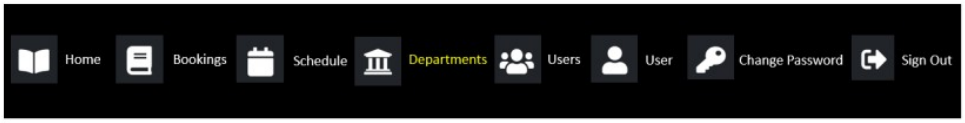
APPENDIX 26 – ADMIN USER BOOKINGS PAGE



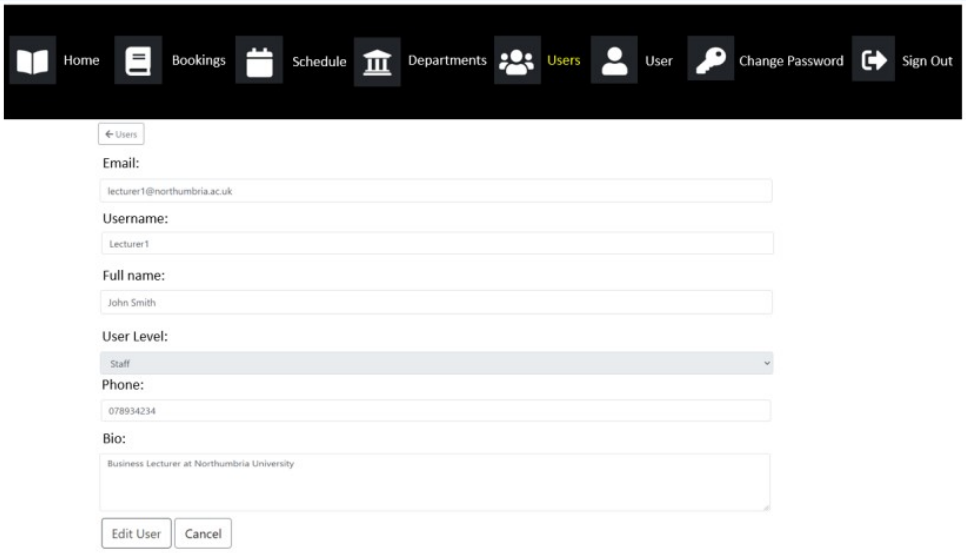
APPENDIX 27 – ADMIN USER SCHEDULE PAGE



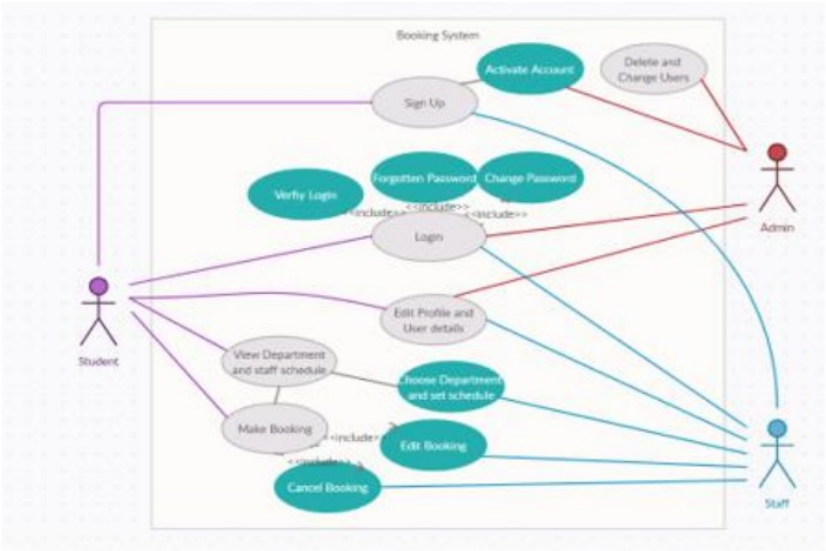
APPENDIX 28 – DEPARTMENTS FORM WIREFRAME



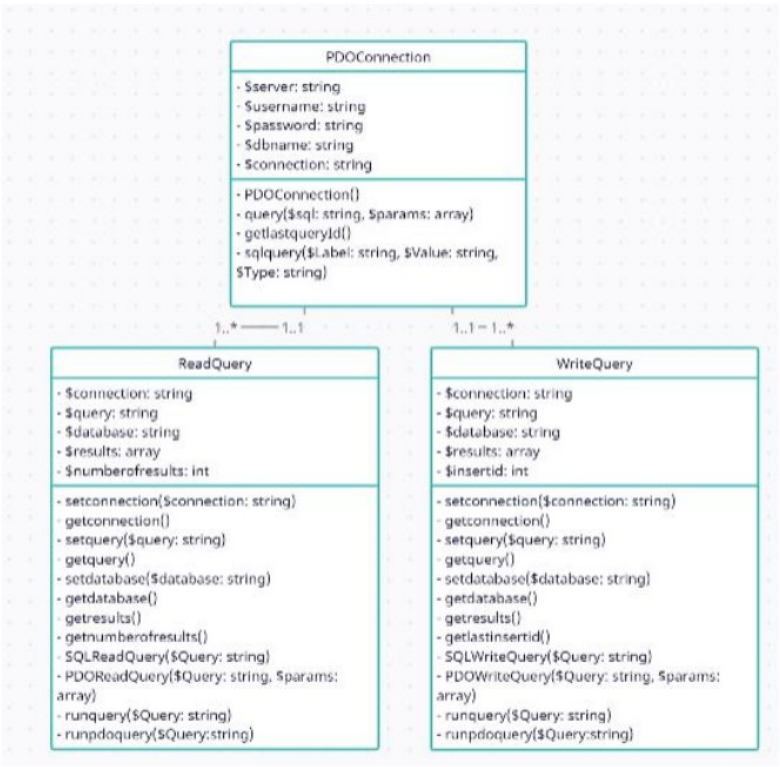
APPENDIX 29 – ADMIN EDIT USERS FORM WIREFRAME



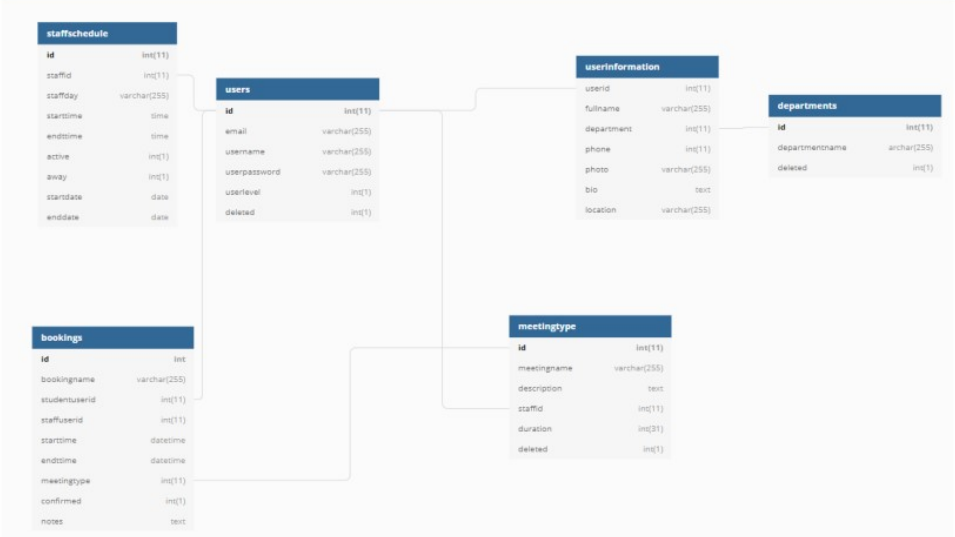
APPENDIX 30 – INITIAL USE CASE DIAGRAM



APPENDIX 31 – PDO CLASS DIAGRAM



APPENDIX 32 – SECOND ENTITY RELATIONSHIP DIAGRAM



APPENDIX 33 – FUNCTIONAL TESTING RESULTS

Function Tested	Expected Result	Actual Result	Resolution
loginform ()	Login form displays when function is called	Login form displays when function is called	N/A
checklogin ()	If login details are correct user is logged in if incorrect error displays	If user details are correct, logs in, if not refreshes user form	Calls the generateerrors function and prevents refresh if details are incorrect
signupform ()	Displays a form for users to signup, different for staff and students	Displays form but doesn't give staff option to select departments	Tick box added for staff members to select a department
checksignup ()	Gets all the data posted from the form. If it is all correct send the user an email to activate their account, if not displays errors showing the user what actions they need to take.	Errors displayed are incorrect	Added for if statements to display correct errors
		User doesn't receive mail.	Gmail details added to xammp to enable the sendmail function, now user receives email
forgottenpasswordform ()	Displays a form with an email field and submit button	Displays a form with an email field and submit button	N/A
forgottenpassword ()	If email is in database reset password to a random password and send an email	Sends an email but doesn't reset the password	Changed the password creation to work correctly and update query to change the password correctly
signout ()	Destroys the session, signing the user out	Destroys the session, signing the user out	N/A
listbookings ()	Displays a list of user bookings with edit and delete buttons	Displays a list of user bookings with edit and delete buttons	N/A
deletebooking ()	Sets a booking to deleted in the DB and sends an email to users	Sets a booking to deleted in the DB, but email doesn't send	Changed the email part to work correctly
editbooking ()	Allow user to edit booking	Error occurs when editing booking	Changed the format to correctly allow editing of the booking
makebooking ()	Allows student users to make a booking with staff and sends an email to both users	Allows student users to make a booking with staff and sends an email to both users	N/A
liststaffschedule ()	Shows either the staff schedule, their holidays or their meetings types allowing them to add, edit and delete all of them	Shows either the staff schedule, their holidays or their meetings types allowing them to add, edit and delete all of them	N/A
editschedule ()	Edit an existing schedule or holiday	Edit an existing schedule or holiday	N/A
addschedule ()	Add a new schedule item, either a schedule or holiday	Adds a new item but some fields are missing in the Db	Changed the insert query to add the missing fields

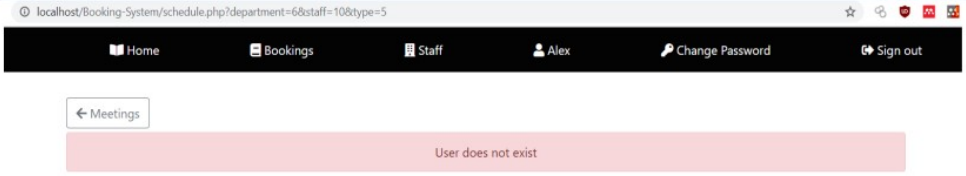
deleteschedule ()	Sets a schedule item to deleted in the DB	Sets a schedule item to deleted in the DB	N/A
editmeetingtype ()	Edit an existing Meeting Type	Edit an existing Meeting Type	N/A
addmeetingtype ()	Add a new Meeting Type	Add a new Meeting Type	N/A
deletemeetingtype ()	Sets a meeting type to deleted in the DB	Sets a meeting type to deleted in the DB	N/A
liststaffavailability ()	Displays a Javascript calendar with a staff members availability, if a time slot is booked it shouldn't display and timeslots shouldn't display if they are between a holiday	Displays the calendar but slots are not on the correct day	Put the slots into a day array and pass the array to the plugin
		Slots display on the correct day but still display if they are booked	Pass a bookings array to the plugin if slot is equal to booking array item don't display it
		Slots still display between holidays	Add holidays array and if slot is between a holiday item in the array don't display
editdepartment ()	Edit an existing Department	Edit an existing Department	N/A
adddepartment ()	Add a new Department	Add a new Department	N/A
deletedepartment ()	Sets a department to deleted in the DB	Sets a department to deleted in the DB	N/A
edituser ()	Edit a user's details	Edit a User's details	N/A
deleteuser()	If user is an admin, then delete selected user	Delete selected user if user is admin	N/A
listbookingusers()	If user is an admin display a list of users with links to their bookings	If user is an admin display a list of users with links to their bookings	N/A
listscheduleuser()	If user is an admin display a list of staff with links to their schedule	If user is an admin display a list of staff with links to their schedule	N/A

APPENDIX 34 – STAFF TESTING FEEDBACK

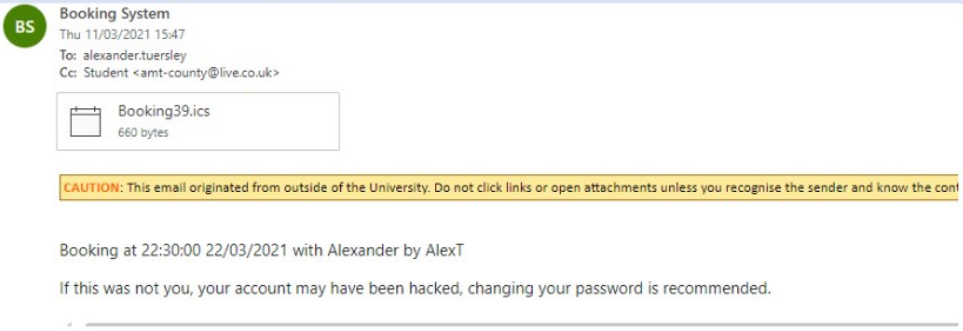
Rating /10
6
7
7
8
7
8
7
9
8
7
8
8
7
7
9
7.53

APPENDIX 35 – USER TABLE AFTER SQL INJECTION

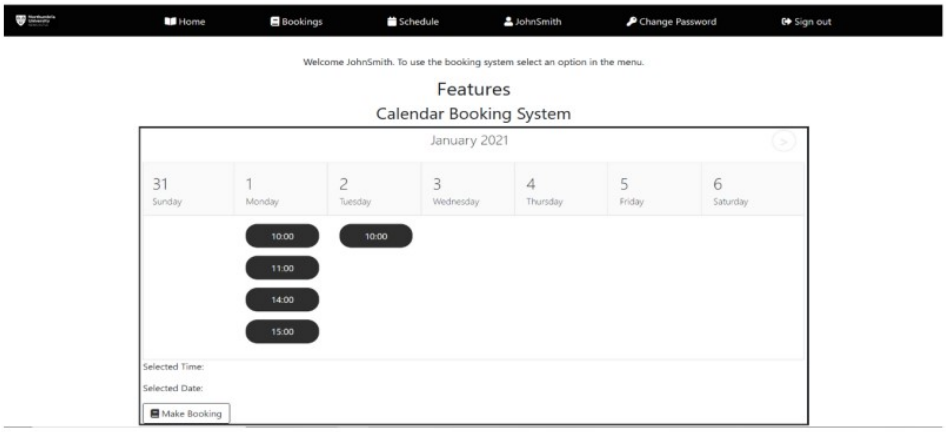
APPENDIX 36 – URL ATTACK IN MAKE BOOKING



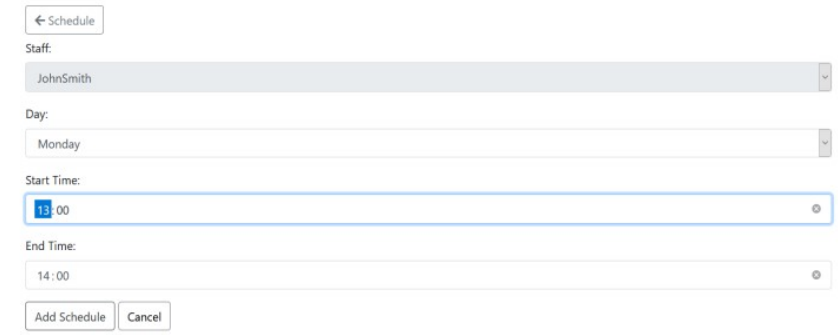
APPENDIX 37 – OUT OF HOURS BOOKING BY STUDENT



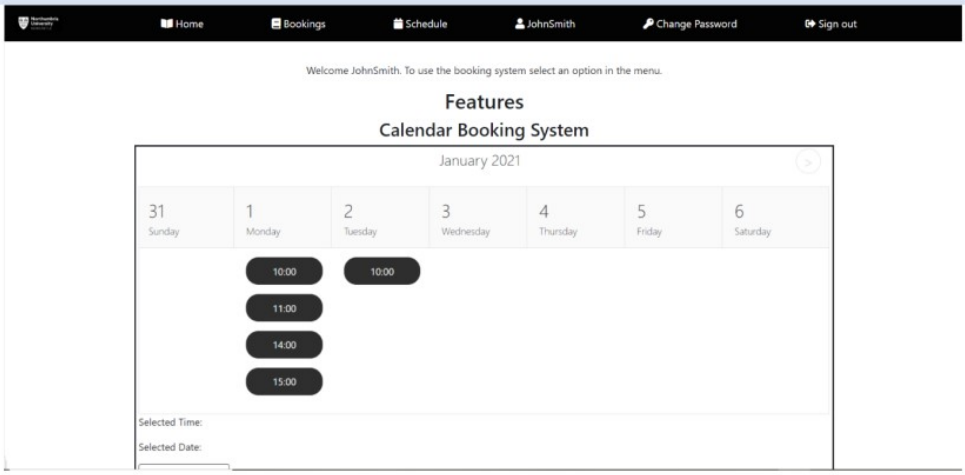
APPENDIX 38 – HOME PAGE IN FIREFOX



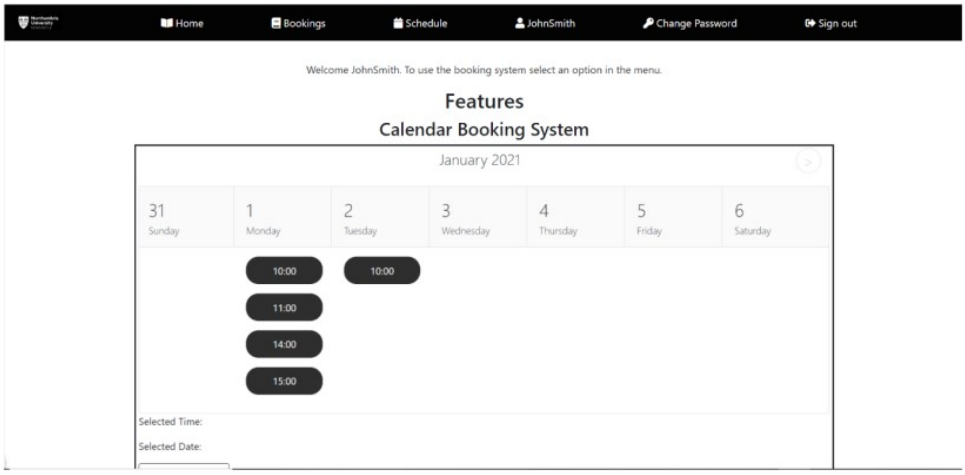
APPENDIX 39 – FORM IN FIREFOX



APPENDIX 40 – HOME PAGE IN OPERA



APPENDIX 41 – HOMEPAGE IN BRAVE



APPENDIX 42 – ERROR HANDLING IN CONFIG.PHP

```php
/**
 * This function handles exceptions, logging the detailed exception to a file and displaying a basic message to the user
 */
function exceptionHandler($e) {
    $msg = array("status" => "500", "message" => $e->getMessage(), "file" => $e->getFile(), "line" => $e->getLine());
    $usr_msg = array("status" => "500", "message" => "Internal Server Error");
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");
    header("Access-Control-Allow-Methods: GET, POST");
    echo json_encode($usr_msg);
    logError($msg);
  }

  /**
  * This function is an error handler that shwos the user a basic message and logs the detailed error to a file.
  */
  function errorHandler($errno, $errstr, $errfile, $errline) {
    if ($errno != 2 && $errno != 8) {
      throw new Exception("Fatal Error Detected: [$errno] $errstr line: $errline", 1);
      logError("Fatal Error Detected: [$errno] $errstr line: $errline");
    }
    else{
      logError("Fatal Error Detected: [$errno] $errstr line: $errline");
    }
  }
```

APPENDIX 43 – NAVIGATION MENU FUNCTION

```
static public function headerandnav($pageTitle,$level,$css ="",$scripts=""){
    print("<!DOCTYPE html>
          <html lang='en'>
          <head>
          <meta charset='utf-8' />
          <title>" $pageTitle."</title>
          <link rel='stylesheet' href='/Booking-System/css/style.css'>
          <link rel='stylesheet' href='/Booking-System/css/bootstrap.min.css'>
          <link rel='stylesheet' href='css/bootstrap-grid.min.css'>");
          if($css != ""){
           foreach($css as $style){
              print("<link rel='stylesheet' href=".$style.">");
           }
          }
    print("<script type='text/javascript' src='js/jquery-3.4.1.min.js'></script>
          <script type='text/javascript' src='js/bootstrap.min.js'></script>
          <script type='text/javascript' src='js/bootstrap.bundle.min.js'></script>
          <script src='https://kit.fontawesome.com/45be65547d.js' crossorigin='anonymous'></script>");
          if($scripts != ""){
           foreach($scripts as $script){
              print("<script type='text/javascript' src='".$script."'></script>");
           }
          }
          print("
          </head>

          <nav class='navbar navbar-expand-lg'>
          <img style='width:5%; max-height:60px;' class='logo' src='".LOGO."' alt='logo'>
          </img>
          <div class='container'>
          ");

    switch($level){
        case 1:
            $navList = array(array("index.php","Home","fa-book-open","Return to homepage"),array("bookings.php","Bookings","fa-book","View your Bookings"),array("schedule.php","St
            break;
        case 2:
            $navList = array(array("index.php","Home","fa-book-open","Return to homepage"),array("bookings.php","Bookings","fa-book","View your Bookings"),array("schedule.php","Sc
            break;
        case 3:
            $navList = array(array("index.php","Home","fa-book-open","Return to homepage"),array("bookings.php","Bookings","fa-book","View your Bookings"),array("schedule.php","Sc
            break;
        default:
            $navList = array(array("index.php","Home","fa-book-open","Return to homepage"),array("signin.php","Sign In","fa-sign-in-alt","Sign in to the system"),array("signin.php
            break;
    }
```

APPENDIX 44 – CALENDAR PLUGIN AVAILABLE TIMES FUNCTION

```
this.getAvailableTimes = function() {
    var tmp = ``;
    var today = new Date();
    console.log(settings.availability);

    for (i = 0; i < 7; i++) {
        var tmpAvailTimes = ``;
        $.each(settings.availability[i], function() {
            var booked = 0;
            var holiday = 0;
            var newDate = new Date(settings.startDate.addDays(i).toDateString()+" "+ this);
            var timestamp = newDate.getTime() / 1000;

            for(k = 0; k < settings.holidays.length; k++){
                if(timestamp > settings.holidays[k][0] && timestamp < settings.holidays[k][1]){
                    holiday = 1;
                }
            }

            for(j = 0; j < settings.bookings.length; j++){
                if(timestamp >= parseInt(settings.bookings[j][0]) && timestamp < parseInt(settings.bookings[j][1])){
                    booked = 1;
                }
            }
            if(newDate.getTime() > today.getTime() && booked == 0 && holiday == 0){
                tmpAvailTimes += `
                <a href="javascript:;" class="myc-available-time" data-time="` + this + `" data-date="` + formatDate(settings.startDate.addDays(i)) + `">
                    ` + this + `
                </a>
            `;
            }
        });

        tmp += `
            <div class="myc-day-time-container" id="myc-day-time-container-` + i + `">
                ` + tmpAvailTimes + `
                <div style="clear:both;"></div>
            </div>
        `;
    }
    return tmp
}
```

APPENDIX 45 – PHP FILE LAYOUT

```php
/**
 * Uses the Calendar Plugin to create an interactive Booking Calendar to make a Booking with a Staff Member
 * @param int $SID - Id of the Staff Member
 * @param int $Type - Meeting Type of the Booking
 * @param int $DID - Id of the Department associated with the Booking
 */
static public function liststaffavailability($SID,$Type,$DID){
    Forms::generatebutton("Meetings","schedule.php?department=".$DID."&staff=".$SID,"arrow-left","secondary");
    if($SID && $Type){
        $Duration = Schedule::getstaticduration($Type);
        if($Duration > 0){
            $schedule = Schedule::liststaffslots($SID,$Duration);
            $bookings = Schedule::staffbookingstimestamp($SID);
            if($schedule){
                $name = Schedule::getstaffname($SID);
                if($name){
                    print("<p class='welcome'>Availability for ".$name."</p>
                    <div id='picker'></div>");
                    print("<p>Selected Time: <span id='selected-time'></span></p>
                            <p>Selected Date: <span id='selected-date'></span></p>
```