



# Online Booking System for After-School Childcare

Simon Herrick

Supervised By: Prof David W Walker

Moderated By: Dr Kirill Sidorov

Module Code: CM3203

Module Title: One Semester Individual Project

Module Credits: 40

# Contents

- 1 Introduction ..... 6
- 2 Background ..... 7
  - 2.1 The Problem ..... 7
  - 2.2 Existing Problems ..... 7
  - 2.3 Methods & Tools ..... 11
- 3 Specification & Design ..... 12
  - 3.1 System Requirements ..... 12
    - 3.1.1 Online Booking System .....12
    - 3.1.2 Mobile Application .....12
    - 3.1.3 Hosting Website .....13
  - 3.2 Overview ..... 13
    - 3.2.1 Online Booking System .....13
      - 3.2.1.1 Login & Registration Portal ..... 13
      - 3.2.1.2 Book Event ..... 14
      - 3.2.1.3 Create Event ..... 14
      - 3.2.1.4 My Created Events ..... 14
      - 3.2.1.5 My Booked Events ..... 14
      - 3.2.1.6 Messages ..... 14
      - 3.2.1.7 Accounts ..... 14
      - 3.2.1.8 Profile ..... 15
      - 3.2.1.9 Settings ..... 15
    - 3.2.2 Mobile Application ..... 15
      - 3.2.2.1 URL Entry Screen ..... 15
      - 3.2.2.2 Login Screen ..... 15
      - 3.2.2.3 Book Event ..... 15
      - 3.2.2.4 Create Event ..... 15
      - 3.2.2.5 My Created Events ..... 16
      - 3.2.2.6 My Booked Events ..... 16
      - 3.2.2.7 Messages ..... 16
      - 3.2.2.8 Profile ..... 16
      - 3.2.2.9 Settings ..... 16
    - 3.2.3 Hosting Website ..... 17
      - 3.2.3.1 Homepage ..... 17
      - 3.2.3.2 About Page ..... 17
      - 3.2.3.3 Docs Page ..... 17
      - 3.2.3.4 Videos Page ..... 17
      - 3.2.3.5 Frequently Asked Questions Page ..... 17
  - 3.3 Changes from Initial Plan ..... 17
  - 3.4 Design Principles ..... 18
    - 3.4.1 Online Booking System ..... 18
    - 3.4.2 Mobile Application ..... 18
    - 3.4.3 Hosting Website ..... 18
  - 3.5 Visual Designs ..... 19
    - 3.5.1 Online Booking System ..... 19
    - 3.5.2 Mobile Application ..... 23
    - 3.5.3 Hosting Website ..... 25

3.6	Use Cases .....	25
3.6.1	Online Booking System .....	25
3.6.2	Mobile Application .....	31
3.7	Database Design.....	37
3.7.1	Overview.....	37
3.7.2	Field Types.....	37
4	Implementation .....	39
4.1	Online Booking System .....	39
4.1.1	Login & Registration System.....	39
4.1.1.1	Overview.....	39
4.1.1.2	Registration.....	40
4.1.1.3	Login.....	41
4.1.1.4	Forgotten Password.....	42
4.1.1.5	Notes .....	42
4.1.2	Book Event Page.....	43
4.1.2.1	Overview .....	43
4.1.2.2	Notes .....	44
4.1.3	Create Event Page .....	44
4.1.3.1	Overview .....	44
4.1.3.2	Notes .....	44
4.1.4	My Created Events Page.....	45
4.1.4.1	Overview .....	45
4.1.4.2	Notes .....	45
4.1.5	My Booked Events Page.....	45
4.1.5.1	Overview .....	45
4.1.5.2	Notes .....	46
4.1.6	Messages Page .....	46
4.1.6.1	Overview .....	46
4.1.6.2	Notes .....	46
4.1.7	Accounts Page.....	47
4.1.7.1	Overview .....	47
4.1.7.2	Notes .....	47
4.1.8	Profile Page.....	48
4.1.8.1	Overview .....	48
4.1.8.2	Notes .....	48
4.1.9	Settings Page.....	48
4.1.9.1	Overview .....	48
4.1.9.2	Notes .....	49
4.1.10	Book Function.....	49
4.1.10.1	Overview .....	49
4.1.10.2	Attendee Reservations .....	49
4.1.10.3	Room or Seat Reservations.....	49
4.1.10.4	Best Fit Allocation Algorithm .....	49
4.1.10.5	Applied In .....	50
4.1.11	Create Event Function.....	50
4.1.11.1	Overview.....	50
4.1.11.2	Applied In .....	50
4.1.12	Cancel Event Function .....	51
4.1.12.1	Overview.....	51
4.1.12.2	Applied In .....	51
4.1.13	Message Function .....	51
4.1.13.1	Overview.....	51

4.1.13.2	Applied In .....	51
4.1.14	Cancel Booking Function .....	52
4.1.14.1	Overview .....	52
4.1.14.2	Cancelling Attendee Booking .....	52
4.1.14.3	Cancelling Seat or Room Booking .....	52
4.1.14.2	Applied In .....	52
4.1.15	Make/Remove Admin Function .....	52
4.1.15.1	Overview .....	52
4.1.15.2	Applied In .....	53
4.1.16	Remove Attendee Function .....	53
4.1.16.1	Overview .....	53
4.1.16.2	Applied In .....	53
4.1.17	Delete User Function .....	53
4.1.17.1	Overview .....	53
4.1.17.2	Applied In .....	53
4.1.18	Update Contact Info Function .....	53
4.1.18.1	Overview .....	53
4.1.18.2	Applied In .....	54
4.1.19	Change Password Function .....	54
4.1.19.1	Overview .....	54
4.1.19.2	Applied In .....	54
4.1.20	Update Settings Function .....	54
4.1.20.1	Overview .....	54
4.1.20.2	Applied In .....	54
4.2	Mobile Application .....	54
4.2.1	Overview .....	55
4.2.1.1	Overview .....	55
4.2.2	URL Entry Screen .....	55
4.2.2.1	Overview .....	55
4.2.2.2	Functions Used .....	56
4.2.3	Login Screen .....	56
4.2.3.1	Overview .....	56
4.2.3.2	Functions Used .....	56
4.2.4	Book Event Screen .....	56
4.2.4.1	Overview .....	56
4.2.4.2	Functions Used .....	57
4.2.5	Create Event Screen .....	58
4.2.5.1	Overview .....	58
4.2.5.2	Functions Used .....	58
4.2.6	My Created Events Screen .....	59
4.2.6.1	Overview .....	59
4.2.6.2	Functions Used .....	60
4.2.7	My Booked Events Screen .....	60
4.2.7.1	Overview .....	60
4.2.7.2	Functions Used .....	61
4.2.8	Messages Screen .....	61
4.2.8.1	Overview .....	61
4.2.8.2	Functions Used .....	61
4.2.9	Profile Screen .....	62
4.2.9.1	Overview .....	62
4.2.10	Update Info Screen .....	63
4.2.10.1	Overview .....	63
4.2.10.2	Functions Used .....	63

4.2.11	Change Password Screen .....	63
4.2.11.1	Overview .....	63
4.2.11.2	Functions Used.....	63
4.2.12	Accounts Screen.....	64
4.2.12.1	Overview .....	64
4.2.12.2	Functions Used.....	64
4.2.13	Settings Screen.....	65
4.2.13.1	Overview .....	65
4.2.13.2	Functions Used.....	65
4.3	Installation Script.....	65
4.3.1	Overview.....	65
4.3.2	Functionality .....	66
5	Testing.....	66
5.1	Overview.....	66
5.2	User Testing.....	67
5.3	Reflection on Testing Results .....	72
6	Future Work .....	72
7	Conclusion .....	73
	Reflection.....	74
	References.....	75

# 1 Introduction

The aim of this project is to provide a solution for schools and parents wanting to book children into after-school childcare.

The intended audience for the system will primarily be schools looking to offer parents a simple way to manage booking their children into childcare. The system should give the school and parents peace of mind that the child is expected and that contact details can easily be found.

The original scope of the project was to design and implement an online booking system that allowed parents to book their children into after-school child care. The system would also allow the childcare administrators to view children that were booked in, as well as contact details for the child's parents. After some thought, I decided that the end result should be a more universal system that was not just limited to after-school childcare. The system would be a single system that accommodated multiple applications which could be used by a much wider audience. I also wanted the booking system to include mobile applications so that users could quickly and easily access the information they required from wherever they were. At minimum I wanted the system to:

- Be easy to install.
- Have a login and user registration system.
- Allow events to be created and cancelled.
- Have multiple user privileges.
- Allow users to book into events, and remove their booking.
- Allow attendees to be viewed.
- Allow users to be managed.

This list provides a very high level view of the system requirements. Refer to section 3.5 for a more in-depth review of the system requirements.

The system itself would be downloaded from a website as a package and then installed on the user's webserver. The system would have a professional look and promote ease of use, with clear application.

The scope changed slightly throughout the project but this remained the foundation that the result was built on.

There are some minor assumptions that the project relies on. The end user must have a webserver. This webserver must be running a minimum of PHP 5.5 for security reasons that will be discussed later in this report. The user must also have access to a MySQL database with knowledge of the database username and password.

The final outcome can be divided into a number of goals:

- The planning stage which will involve the design of the system, the database design, the interface design and branding, the mobile application design and the design of the website on which the system will be hosted.
- The implementation stage which will involve the creation of the system and mobile applications as well as the content for the hosting website.
- The testing phase which will involve testing all aspects from installation and usage.
- Explanation of how the system could be expanded in the future for more application.
- Concluding the project and contrasting against the original proposal.

## 2 Background

### 2.1 The Problem

The popularity of web applications over recent years has hit a massive surge. When Google released the Chromebook in 2011 [1], which is a notebook that allows the user to operate their day-to-day tasks solely from web applications, the possibilities of the use of web applications in the mainstream really became apparent. The idea of web applications arrived with the birth of the dynamic web which many consider to be attributed to the release of CGI (Common Gateway Interface) in 1993 [2]. CGI allowed websites to run scripts, primarily Perl, and then display the output. This opened the gates for many server-side technologies to be developed such as PHP, Java, Python and Ruby to name a few.

When looking more specifically at booking systems many, especially small organisation or single-employee businesses, still employ an offline solution choosing to manually take note of bookings and reservations, whether it be over the phone or face to face. This can present numerous problems, for instance, time restriction. Someone must be available to take the booking which, for a small organisation may mean paying an extra employee or, for a single-employee may involve the inconvenience of always be reachable, otherwise a loss of business could occur. This solution also has restricted access in the sense that a reservation or booking diary may be used by only one employee at a time to access information. Furthermore, if this diary was to be lost, the organisation would suffer a major inconvenience. This project aims to solve this by providing users with an 'off-the-shelf' web application that can be tailored to suit their needs. This will allow users to offer their chosen audience the convenience of 24/7 access to obtain the information they require from a web application or mobile device.

### 2.2 Existing Problems

There are a number of existing projects that go some way to providing a solution to the problem but the majority are either too specific to a single problem rather than offer a more universal solution, or don't quite fit the brief. Some examples are:

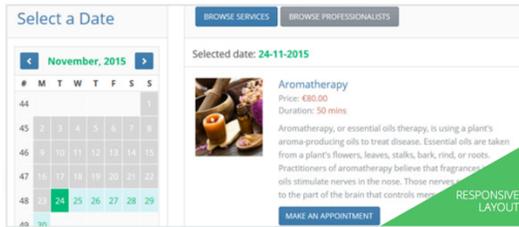
#### - Appointment Scheduler by PHPJabbers [3]

Appointment Scheduler is a booking system written in PHP that provides its users with the ability to allow their clients to book appointments on their website. The package includes a front-end layout which displays bookable appointments and a back-end layout which is an administration portal allowing the moderation of appointments and services. It boasts such features as:

- Appointment Booking
- Responsive Design
- 10 Colour Themes
- Reporting Features
- Email & SMS Notifications

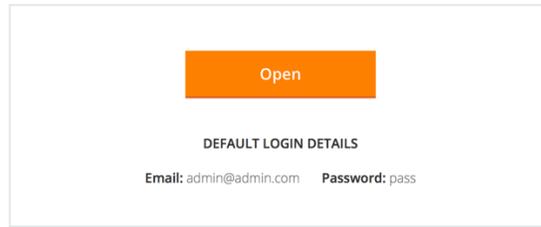
Front-end

Open administration page to view available color themes for the front end.



Administration page

Click the button below to open the admin panel of the php appointment script.



Appointment Scheduler goes some way to solving the problem. It takes an in-depth approach to offering very specific features such as employee scheduling by adding business hours, working hours and separate employee accounts. It falls short in the context of the problem because it does not offer solutions to other states requirements. For instance, Appointment Scheduler allows for events to be created, but only one person may book on to the event. Appointment Scheduler also does not offer reservations in terms of rooms or seats. The layout for Appointment Scheduler is responsive so is accessible on mobile devices but no mobile applications or ways to connect a mobile application are offered.

Appointment Scheduler is priced at \$49 for a user’s licence, or \$99 for a developer licence.

- phpMyReservation by Ole Jon Bjørkum [4]

phpMyReservation is a reservation system based on HTML5, PHP and MySQL. The original scope of the project had a similar base to this project scope in that it was designed to replace a paper-based reservation system used by the developer’s local launderette. After logging in the user is presented with a calendar where they are able to book in to any hour timeslot. There is an administration portal which allows user administration and a few basic modifications to the system such as reservation settings and control. It boasts features such as:

- Login/Registration system
- Yearly view
- Price per reservation
- User/Admin Control Panel
- Reservation reminders by email



phpMyReservation again goes some way to solving the problem. Like Appointment Scheduler the system purely focuses on reservation booking, as such each reservation is restricted to one person. It falls short on the stated requirements in a number of ways. Each reservation in phpMyReservation has to be an hour long. If an event was to be created that was two hours long it would count as two reservations. When creating an event in phpMyReservation, it is required that the event runs throughout the day on consecutive hours. If, for instance, an event was to be created that only ran between 4pm and 5pm on a Friday phpMyReservation would not be a solution. phpMyReservation also does not provide time zone support. This means that your webserver must be located in the same country as the one you are providing the appointments for. If, for instance, the webserver was issued by an American hosting company the reservation schedule could be up to eight hours out of time.

phpMyReservation is free to use but is not maintained so does not provide support further than the Wiki.

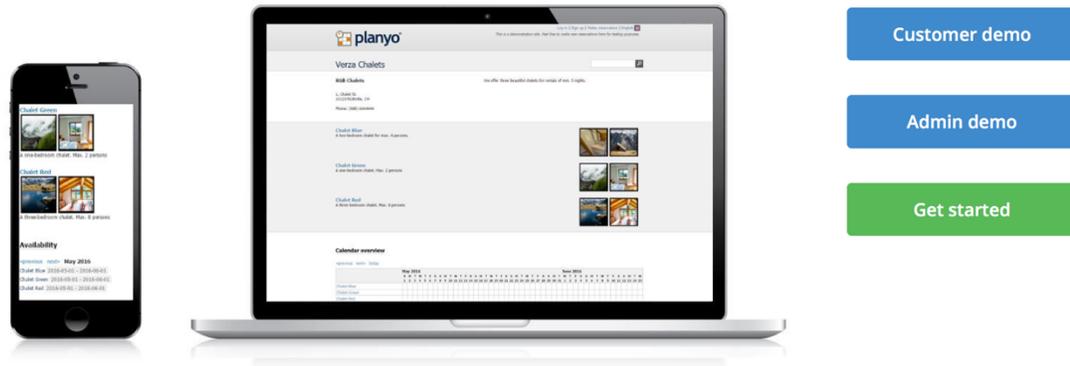
### - Planyo Free by Planyo [5]

Planyo offers a range of different costing options for their online reservation system, and although titled “Planyo Free”, the customer when booking must pay a €1 booking fee for every reservation. This fee gets paid directly to Planyo.

Planyo Free integrates into the users current site by providing html widgets that the user can implement in their site code. Planyo Free allows customers to make reservations that focus around hotel booking. Planyo Free provides an in-depth administration portal, but the portal itself is hosted on the Planyo website. From this administration portal the user can access numerous features such as reservation management, schedule management and a customer list. SMS and Email notifications can also be set up, but again they cost per notification which must be paid to Planyo in “Planyo Coupons”. Planyo Free boasts features such as:

- Integration
- Recurring reservations
- Multiple language support
- Customizable forms
- Set-up assistant
- Responsive design
- Mobile Applications

## PLANYO FREE - FREE ONLINE RESERVATION SYSTEM



Planyo Free offers a different scope than the other existing projects and focuses more on location based reservations than appointment based reservations. However, it still does not offer both. Planyo Free is not a suitable solution for a number of reasons. It does not offer event booking, only accommodation based reservations. Planyo Free claims to be responsive but their demo, when at certain dimensions, does not fit into the constraints. If Planyo Free was to be implemented, the user must be very careful of the hidden fees. Planyo “Free” appears to charge for many of its functions, and if used frequently, could incur a large cost. In our example of childcare, parents would be forced to pay a €1 fee for every booking which would be paid to Planyo.

Planyo Free is free to install but has fees. Planyo Pro is £15.63 a month (dependant on conversion rates).

### - BookingWizz by CriticalGears [6]

BookingWizz is another booking system written in PHP. BookingWizz, unlike the others, focuses around events with multiple spaces. Events can be created allowing for a specific amount of people to attend. BookingWizz also allows for deposit and reservation payments to be processed. Like Planyo, BookingWizz can generate widgets so that portions of the system can be integrated to the user’s current website if needed.

BookingWizz is split into two parts, the customer portal and the administrator portal. The customer portal displays a calendar allowing specific dates to be booked. The administration portal allows for handling of events and services as well as a few reporting functions. BookingWizz boasts features such as:

- Responsive design
- Payment processing
- SMS notifications
- Printable schedules
- Reporting
- Time zone support
- Calendar exporting



When comparing the requirements, BookingWizz comes closest to providing a viable solution but still falls short in a number of areas. The system can only be used for one type of event. If an organisation wishes to offer more than one event, BookingWizz is not a viable option. When booking a space only one space is allowed per user meaning BookingWizz could not be used for room or seat allocation for multiple bookings.

BookingWizz is priced at \$38.

## 2.3 Methods & Tools

These existing projects focus on specific solutions rather than a more general approach to bookings. Many contain features that can be implemented in the final project but which must be expanded upon to better fit the requirements. One consistent feature among the existing projects is their choice of implementing PHP to achieve their result. PHP is a widely used language that comes preinstalled on the majority of web servers. For this reason, the project will be written in PHP and will communicate with a MySQL database. I have chosen to use PHP 5.5 as a minimum as this version of PHP contains the “password\_hash” function [7]. I wanted the ability to use this function as it currently supports the bcrypt algorithm [8], but is designed to change over time as new and stronger algorithms are added to PHP.

For the mobile applications I intend to release apps for both the iOS and Android platforms. To facilitate this, I wanted to find a tool or language that would allow me to develop both at the same time. I chose the Ionic Framework [9] which allows cross platform applications to be developed using HTML and CSS, with functionality provided by Angular JS. Ionic includes many features that are present in native mobile applications and translates them into HTML.

## 3 Specification & Design

### 3.1 System Requirements

This section will describe the functional and non-functional requirements of the system. This section will be divided in to three sections for each part of the system to ensure each requirement is applied to the relevant section.

#### 3.1.1 Online Booking System

Functional Requirements	Non-functional Requirements
An event must be able to be created or cancelled allowing a quantity of attendees, rooms or seats to be allocated.	The system will be universal allowing all browsers to access and display the content correctly.
The event creator must be able to access contact information of attendees who have reserved a place on their event.	The online booking system will have a responsive design to allow the user to access the system using any screen resolution from any device.
The system will be able to employ a best fit algorithm when allocating seats or rooms. This will allow the creator to maximum seat or room usage and keep wastage to a minimum.	The online booking system will be light, using as few images as possible. This will keep the download package size down, ensuring a quick download, as well as keeping the browser load time quick when accessing the system online.
The system will deliver email notifications for specific functions. For instance when a user has a received a message, or when an event is cancelled all attendees will be notified via email.	The design will be simplistic and clear allowing the user quick access to whatever function they require. The user will not have to navigate numerous screens to perform a simple task.
The system will be installed using an install function to promote ease of use.	The user's password will be securely stored using encryption so as not to be viewable in plaintext.

#### 3.1.2 Mobile Application

Functional Requirements	Non-functional Requirements
The user must be able to connect to their required booking system by entering the URL of the website on which it is hosted.	The mobile application will be cross-platform and function on both the iOS and Android operating systems.
The user must be able to log in to their account using the same details that they would use to log in to the web application.	The design will be simplistic and clear. The overall design will mimic the online booking system but will accommodate the iOS and Android recommended design guidelines.

The user will be able to accomplish all the tasks they are able to do via the web application.	The application will employ touch navigation feature such as swipe to reveal and pull to refresh.
The user will be able to log out, removing all data stored in local storage so that they may connect to another booking system.	The mobile application will save the connection and login details in to local storage so that the user does not have to retype them in every time they open the app.

### 3.1.3 Hosting Website

Functional Requirements	Non-functional Requirements
The hosting website must allow for the package to be easily downloadable.	The hosting website design will allow users to quickly obtain the information they require.

## 3.2 Overview

This project aims to deliver a complete package comprised of three parts, the online booking system, the mobile application and the website to host the downloadable package.

The online system will give the user the ability to create events with different booking options such as attendee spaces, seats and rooms. The user will then be able to monitor attendees reserving a position and be able to view their details. The system will also allow the user the option to cancel the event. The system itself will be customisable so that the user may alter its appearance to better fit their website for better integration. The online system will be navigated using a side menu. The online system will have three different types of users; a 'Super User', who is the user created during the installation process and has access to all functionality of the system, an 'Administrator', who can be selected by the 'Super User', has the ability to create events and monitor their events and a 'Client User', who has the ability to book in to events. The side menu reflects the user's access by only displaying links that are available to the specific user level.

The mobile application will give the user the ability to connect to their online booking system and perform the same functions that are accessible on the web application. The mobile application will save the user's connection details in device storage so that the user does not have to enter their booking system and login details every time the mobile application is opened. The mobile application will also be navigated using a side menu.

The website that hosts the downloadable package will clearly outline the booking system's functions and provide support documentation such as installation instructions and frequently asked questions.

### 3.2.1 Online Booking System

#### 3.1.1.1 Login & Registration Portal

The user will be presented with the options to either login or register. There will also be a link that will allow the user to reset their password via email validation if they have forgotten their password.

If logging in, the user must enter their username and password then submit. They will then be forwarded to the online booking system. If the user enters an incorrect username or password they will be notified. If the user has not activated their account via email they will also be notified.

If registering, the user will be asked to fill out some information required for registration. The user will then receive an email with an activation link that will give the user the ability to login. Like the login section there will be validation for some of the fields.

#### 3.2.1.2 Book Event

After login the user will be presented with the first page of the online booking system, the book event page. The book event page will display events that the current user is able to book in to. The event details will be displayed as well as the ability to book in to the event or message the host. If booking a seat or room reservation, the user will be asked the quantity they would like.

#### 3.2.1.3 Create Event

This page will only be available to users of 'Administrator' level and above. The user will be asked to enter an event name, an event description, choose a start and end time (using a calendar) and, finally, the amount of bookable spaces with a dropdown menu for attendees, seats or rooms. The forms will contain some validation such as whether the end time precedes the start time. If so, the user will be presented with an error.

#### 3.2.1.4 My Created Events

Like the create event page, this page will only be available to users of 'Administrator' level and above. This page will display events that the current user has created. The event details will be displayed as well as the option to view people attending or cancel the event. When choosing to view attendees all current attendees will be displayed on a new page so giving the user the ability to message the attendee or remove them from them event. If the event is cancelled, or the attendee is removed, they will be notified via email.

#### 3.2.1.5 My Booked Events

This page will be available to all users and will display events that the current user has booked in to. The event details will be displayed as well as the options to message the host or cancel booking.

#### 3.2.1.6 Messages

This page will also be available to all users and will display messages that the current user has received. If the user has a new message, the number of new messages will appear next to the messages link in the side menu. The page will display the message sent date, the message sender's name and the message itself. It will also provide the user with the option to reply or delete the message. When the user receives a message they will also receive an email notification.

#### 3.2.1.7 Accounts

The accounts page is only available to the 'Super User'. This page displays a list of all

users registered with the online booking system. The registered user's details will be displayed including whether they have activated the account or not. The ability to grant or revoke 'Administrator' access will be displayed dependant on the user's current level of access. There will also be the option to remove the user.

#### 3.2.1.8 Profile

This page is available to all users and will provide users with the ability to change some of their account details. The page will contain two forms; one for the user to modify basic details, such as name and phone number and another to change their password. The user will be required to fill in their current password before submission.

#### 3.2.1.9 Settings

This page is only accessible to the 'Super User' and will allow them to modify certain aspects of the booking system. This page will display a form allowing the user to change details such as the title, sub header and URL of the booking system as well as customising the look by changing the colours the system displays in.

### 3.2.2 Mobile Application

#### 3.2.2.1 URL Entry Screen

The first screen the mobile application user is presented with is the URL entry screen. The user will be presented with a field to enter the booking system's URL. After submitting, the application will verify an online booking system exists at that address then it will progress to the next screen. If the system does not exist at that address an error will be displayed.

#### 3.2.2.2 Login Screen

After progressing through the URL entry screen the user will be presented with the login screen. The user's login details would be the same as the online booking system. If the user enters an incorrect username or password an error will be displayed.

#### 3.2.2.3 Book Event

After successfully logging in the user will be presented with the book event screen. Much like the online system this is the first screen they will encounter. This screen will display events available for the user to book in to. The details of the event will be displayed but, unlike with the web application, the user must swipe to the left for the options to appear. When swiping to the left the user will be presented with the Message Host and Book Functions. If the chosen event is a room or seat reservation the user will be presented with a field to enter their chosen quantities before the booking completes.

#### 3.2.2.4 Create Event

Similar to the web application the create event screen will only be available to users with 'Administrator' level and above. The details required on this screen will be the same as for the web application, but displayed in a slightly different manner to accommodate the device from which it is running. The fields for event title, description and quantity

will remain the same but the selection for the choice of attendees, room or seats will be in the form of a checkbox rather than a dropdown menu. This is to ensure cross-platform compatibility. Once the required fields have been entered and submitted the user will be presented with a start time input in the form of a calendar. This screen will be repeated again for the end time prior to event submission. Much like the web application these fields will need to be validated.

#### 3.2.2.5 My Created Events

Like the create event screen this screen will only be available to users of 'Administrator' level and above. The screen will display event details for events that the current user created. When an event is swiped to the left they will be presented with the options to either tap Attendees or Cancel. When Cancel is pressed the event will be removed and all attendees will be notified via email of the cancellation. When Attendees is pressed the user will be presented with another screen displaying the details of all the users booked in to the event. This window will display the details of the user and when swiped left display options to either message the user or remove the user's booking. The user will receive an email notification for either action.

#### 3.2.2.6 My Booked Events

The My Booked Events screen will be available to all users and will display events that the current user is booked in to. The screen will display the event information and when the event is swiped to the left the user will have the option to message the host or cancel their booking.

#### 3.2.2.7 Messages

The messages screen will also be available to all users. This screen will display all messages the user has received. Much like the web application, the amount of new messages will display next to the messages link in the side menu. When a message is swiped to the left the user will be presented with the option to reply to the message or delete the message.

#### 3.2.2.8 Profile

The profile screen will also be available to all users. This screen will display two options, one to Update info, one to Change password. When Update Info is selected the user will be presented with a new screen containing fields to change their name or contact number. When Change Password is selected the user will be presented with a new screen containing fields to change their password. The user will be required to enter their current password to complete this function.

#### 3.2.2.9 Settings

This screen will only be accessible by the 'Super User'. This screen will present the user with fields to change specific settings relating to the web application. These settings will be the same as the web application settings page and will allow the user to change settings such as the system title, sub header, URL, admin email and numerous colour settings.

### 3.2.3 Hosting Website

#### 3.2.3.1 Homepage

The homepage of the hosting website will simply display a screenshot of the web application and underneath provide a link to download the system. This page will also act as the download page when the link is clicked.

#### 3.2.3.2 About Page

The about page of the hosting website will describe the web application and mobile application as well as list the system's features.

#### 3.2.3.3 Docs Page

The docs page of the hosting website will provide a brief set of instructions to install the system on the user's web server. These instructions will also be placed in a text file in the root directory of the downloadable package.

#### 3.2.3.4 Videos Page

The videos page of the hosting website will contain a video in the form of a screen recording that provides an example of how the web application can be installed on a web server.

#### 3.2.3.5 Frequently Asked Questions Page

The FAQ page of the hosting website will simply list of possible issues a user could encounter during the installation and use of the system and the steps to resolve them. This list would be populated from experiences during development and issues encountered during the testing phase.

## 3.3 Changes from Initial Plan

Two areas evolved during the progress of the project and now differ slightly from the original plan. Originally the scope for the completed system was to be restricted to offering events that only allowed a quantity of reservations to be dictated. These reservations would be limited to one per user. As the system evolved extra functions were included that allowed a user to create an event and offer different reservation types including seats and rooms. When creating an event, the user would now choose a quantity then select whether they wished to offer the quantity in attendees, rooms or seats. If rooms or seats were selected the person creating the reservation would be given the option of selecting how many seats or rooms they required. The system would then allocate seats or rooms based on a best-fit type algorithm described in Section 4.1.10.4. The system would work through the available seats or rooms and attempt to find a space that exactly fit the required quantity. If no such space could be found, the system would then scan through looking for the next best fit. If no bookings exist, the system will allocate the first available rooms or seats. The user would now be able to view attendees and the seats that they had been allocated for their created events. This method was decided upon to help promote minimal wastage of reservation spaces.

A second area, which did not become apparent until development, was time zones. A web hosting company that was being used during the development of the web application had issues with their MySQL servers which led to the time zone of the server being changed to GMT +7:00. This created issues as events would now not delete automatically upon ending as the server time was several hours behind. This expanded the scope of the outcome and added a requirement for time zone selection to allow for the user to select their current time zone.

## 3.4 Design Principles

The design for the overall package will follow a simplistic and modern plan. The package will always remain professional so not to limit the target audience. The design throughout the different sections of the project will match each other as closely as possible to promote continuity.

### 3.4.1 Online Booking System

The design for the online booking system will adhere to the simplistic and modern plan, having a clear flow allowing the user to quickly obtain the functionality they require. The design itself will be responsive to ensure compatibility with all screen sizes. This means that if a user does not wish to use the mobile application the system will accommodate their mobile screen size. The colours will be customisable so that the user can either choose their own colour scheme or utilise colours from their current website for integration purposes. The quantity of images will be kept to a minimum promote a small file size and load time.

### 3.4.2 Mobile Application

The design for the mobile application will match the design for the online booking system as closely as possible whilst adhering to the recommended guidelines set out for iOS [10] and Android [11] applications. Careful consideration will be taken when designing the flow of the mobile application to ensure compatibility with touch actions such as swipe to reveal.

### 3.4.3 Hosting Website

The design for the hosting website will closely resemble the overall layout of the online booking system. The design will be minimal and clearly display the information provided. The quantity of images will be kept to a minimum to promote a small file size and load time.

### 3.5 Visual Designs

The visual designs section will provide a wireframe view of how each section of the project will be displayed. Each wireframe will provide an overview of each screen that the user will encounter when using the system.

#### 3.5.1 Online Booking System



Figure 1 – Online Booking System Login View

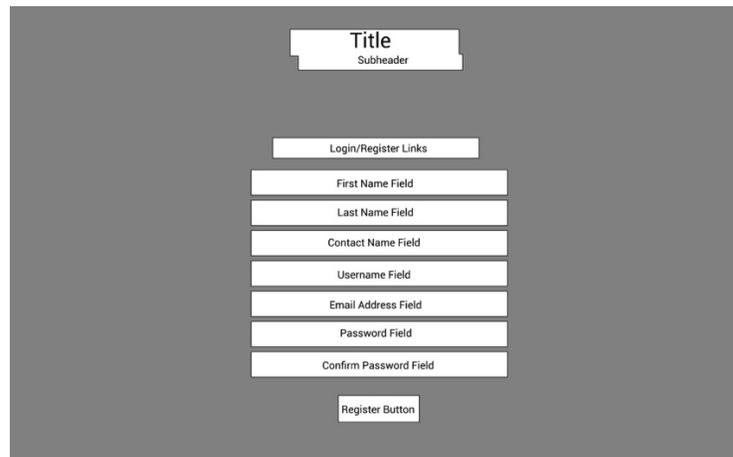


Figure 2 – Online Booking System Register View

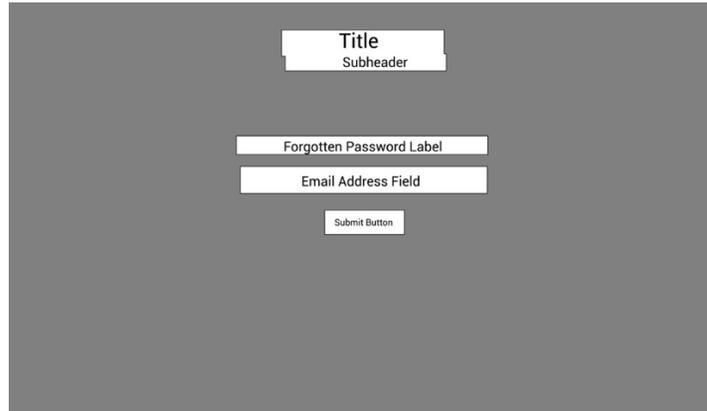


Figure 3 - Online Booking System Forgotten Password View

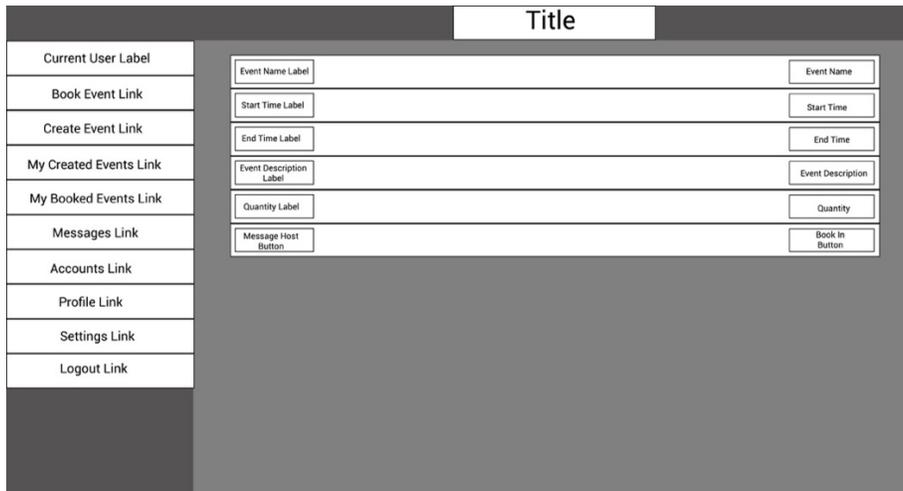


Figure 4 - Online Booking System Book Event View

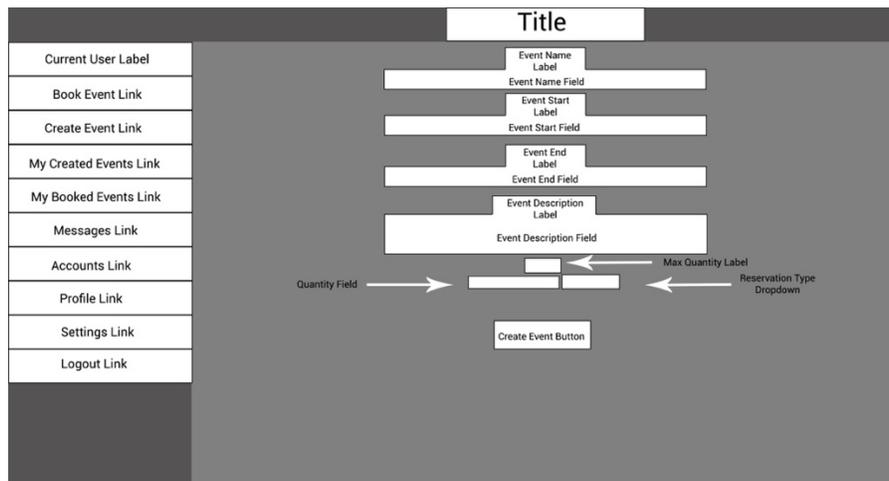


Figure 5 - Online Booking System Create Event View

Title	
Current User Label	Event Name Label <input type="text"/> Event Name
Book Event Link	Start Time Label <input type="text"/> Start Time
Create Event Link	End Time Label <input type="text"/> End Time
My Created Events Link	Event Description Label <input type="text"/> Event Description
My Booked Events Link	Quantity Label <input type="text"/> Quantity
Messages Link	Cancel Event Button <input type="button"/> View Attendees Button <input type="button"/>
Accounts Link	
Profile Link	
Settings Link	
Logout Link	

Figure 6 - Online Booking System My Created Events View

Title	
Current User Label	Event Name Label <input type="text"/> Event Name
Book Event Link	Start Time Label <input type="text"/> Start Time
Create Event Link	End Time Label <input type="text"/> End Time
My Created Events Link	Event Description Label <input type="text"/> Event Description
My Booked Events Link	Quantity Label <input type="text"/> Quantity
Messages Link	Message Host Button <input type="button"/> Cancel Booking Button <input type="button"/>
Accounts Link	
Profile Link	
Settings Link	
Logout Link	

Figure 7 - Online Booking System My Bookings View

Title	
Current User Label	From Label <input type="text"/> From
Book Event Link	Date Label <input type="text"/> Date
Create Event Link	Message Label <input type="text"/> Message
My Created Events Link	Reply Button <input type="button"/> Delete Button <input type="button"/>
My Booked Events Link	
Messages Link	
Accounts Link	
Profile Link	
Settings Link	
Logout Link	

Figure 8 - Online Booking System Messages View

Title	
Current User Label	User ID Label <input type="text"/> User ID
Book Event Link	First Name Label <input type="text"/> First Name
Create Event Link	Last Name Label <input type="text"/> Last Name
My Created Events Link	Username Label <input type="text"/> Username
My Booked Events Link	Email Label <input type="text"/> Email
Messages Link	Phone Label <input type="text"/> Phone
Accounts Link	Role Label <input type="text"/> Role
Profile Link	Active Label <input type="text"/> Active
Settings Link	Admin Button <input type="button" value="Delete Button"/>
Logout Link	

Figure 9 - Online Booking System Accounts View

Title	
Current User Label	First Name Label <input type="text"/>
Book Event Link	Last Name Label <input type="text"/>
Create Event Link	Phone Number Label <input type="text"/>
My Created Events Link	
My Booked Events Link	Update Info Button <input type="button" value="Update Info Button"/>
Messages Link	Change Password Label <input type="text"/>
Accounts Link	Current Password Field <input type="text"/>
Profile Link	New Password Field <input type="text"/>
Settings Link	Confirm New Password Field <input type="text"/>
Logout Link	Change Password Button <input type="button" value="Change Password Button"/>

Figure 10 - Online Booking System Profile View

Title	
Current User Label	Title Label <input type="text"/>
Book Event Link	Title Field <input type="text"/>
Create Event Link	Subheader Label <input type="text"/>
My Created Events Link	Subheader Field <input type="text"/>
My Booked Events Link	URL Label <input type="text"/>
Messages Link	URL Field <input type="text"/>
Accounts Link	Admin Email Label <input type="text"/>
Profile Link	Admin Email Field <input type="text"/>
Settings Link	Timezone Label <input type="text"/>
Logout Link	Timezone Dropdown Menu <input type="text"/>
	Background Colour Label <input type="text"/>
	Background Colour Field <input type="text"/>
	Title Colour Label <input type="text"/>
	Title Colour Field <input type="text"/>
	Login Text Colour Label <input type="text"/>
	Login Text Colour Field <input type="text"/>
	Submit Button Colour Label <input type="text"/>
	Submit Button Colour Field <input type="text"/>
	Hover Button Colour Label <input type="text"/>
	Hover Button Colour Field <input type="text"/>
	Update Settings Button <input type="button" value="Update Settings Button"/>

Figure 11 - Online Booking System Settings View

### 3.5.2 Mobile Application

The mobile application wireframes may differ slightly depending on which platform they are being displayed on due to some operating system defaults such as placements or fonts.



Figure 12 - Mobile Application URL Entry Screen

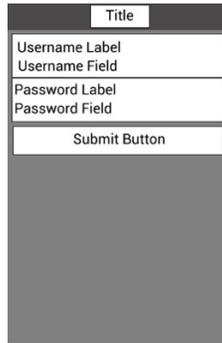


Figure 13 - Mobile Application Login Screen

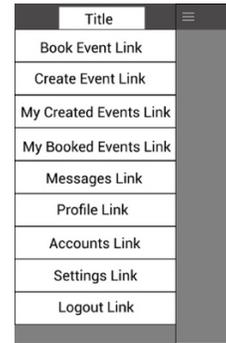


Figure 14 - Mobile Application Side Menu View



Figure 15 - Mobile Application Book Event View



Figure 16 - Mobile Application Create Event View

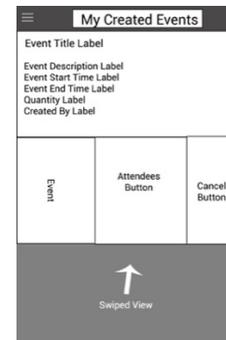


Figure 17 - Mobile Application My Created Events View

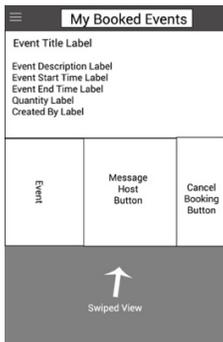


Figure 18 - Mobile Application My Booked Events View

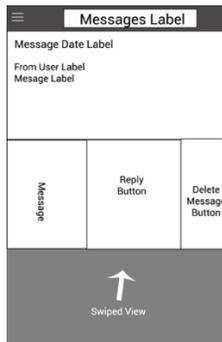


Figure 19 - Mobile Application Messages View



Figure 20 - Mobile Application Send Message View



Figure 21 - Mobile Application Profile View

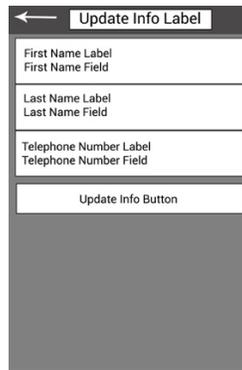


Figure 22 - Mobile Application Update Info View

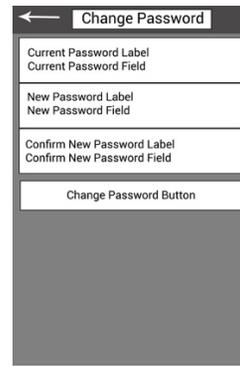


Figure 23 - Mobile Application Change Password View

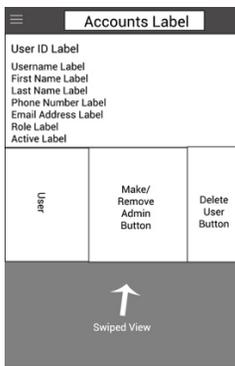


Figure 24 - Mobile Application Accounts View

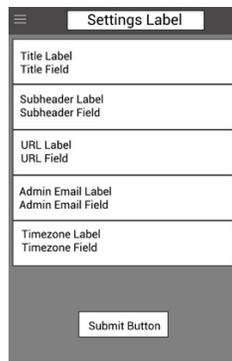


Figure 25 - Mobile Application Settings View

### 3.5.3 Hosting Website

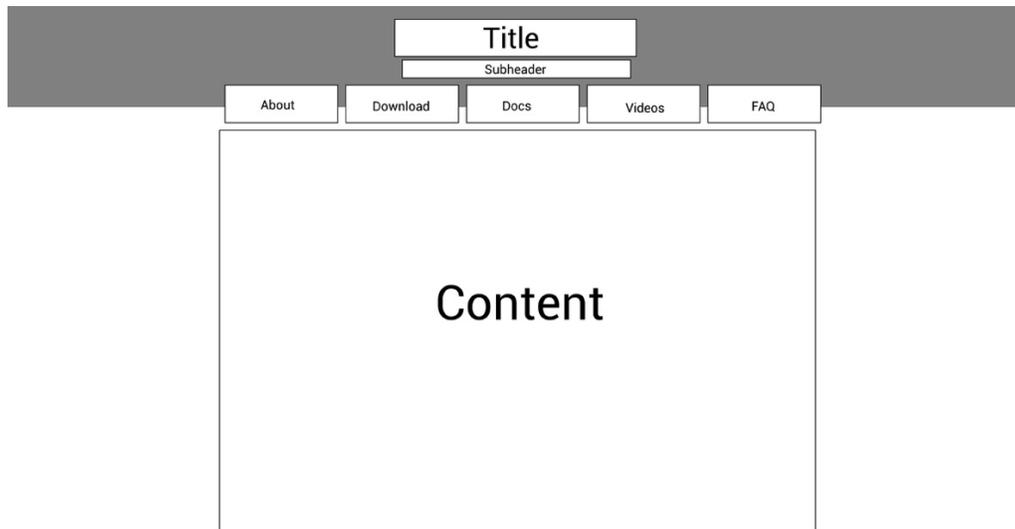


Figure 26 - Hosting Website Wireframe

## 3.6 Use Cases

This section will provide use cases for the functions that will be available in the finished project. This section will again be split in to three parts, corresponding to each part of the final package, and will describe the flow of each function.

### 3.6.1 Online Booking System

Use Case:	1. Installing the Online Booking System
Actors:	Client
Pre-Conditions:	Client must have a web server running PHP 5.5+ and access to a MySQL database. Client also has FTP access and knowledge of how to use it to upload files. Client has downloaded the package from the hosting website.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client extracts the package zip file.</li> <li>2. Client opens the book/functions/db.php file in a text editor and fills in their MySQL database settings.</li> <li>3. Client connects to there web server via FTP and places the book folder in the web server's root folder.</li> <li>4. Client navigates to there URL followed by /book/install.php e.g. <a href="http://client.url/book/install.php">http://client.url/book/install.php</a></li> <li>5. Client fills in the required fields and clicks submit.</li> <li>6. Client receives an installation success message.</li> <li>7. Client connects to the web server via FTP and removes the /book/install.php file.</li> </ol>

Post-Conditions:	Client has successfully installed the Online Booking System and is able to login using the Super User account details they created during installation.
------------------	---

Use Case:	2. Registering a new user
Actors:	New User
Pre-Conditions:	An Online Booking System has previously been installed.
Basic Flow:	<ol style="list-style-type: none"> <li>1. New User navigates to the Online Booking System's URL.</li> <li>2. New User clicks the register link.</li> <li>3. New User fills in the required fields and clicks submit.</li> <li>4. New User navigates to the email they have received from the Online Booking System.</li> <li>5. New User clicks the activation link.</li> <li>6. New User is presented with a message informing them of a successful registration.</li> </ol>
Post-Conditions:	New User has successfully registered and activated their account. New User is now able to login using the details they entered during registration.

Use Case:	3. Logging In
Actors:	Registered User
Pre-Conditions:	The user has previously registered and activated an account with the selected Online Booking System.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Registered User navigates to the Online Booking System's URL.</li> <li>2. Registered User fills in their username and password then clicks submit.</li> </ol>
Post-Conditions:	Registered User now has access to the Online Booking System.

Use Case:	4. Recovering a Lost Password
Actors:	Client
Pre-Conditions:	Client has registered an account. Account is active.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client navigates to the Online Booking System's URL.</li> <li>2. Client clicks on the Forgotten Password link.</li> <li>3. Client fills in the required information.</li> <li>4. Client navigates to the email containing the recover link and clicks it.</li> <li>5. Client fills out new password then submits.</li> </ol>
Post-Conditions:	Client has recovered their password. Client can now login to the system using their new details.

Use Case:	5. Booking an Event Offering Attendee Reservations
Actors:	Client
Pre-Conditions:	Client has logged in to the system and is on the Book Event page. Client is not already booked on to this current event.

Basic Flow:	<ol style="list-style-type: none"> <li>1. Client sees an event that offers attendee reservations that they would like to book in to.</li> <li>2. Client clicks “Book In” underneath the event details.</li> </ol>
Post-Conditions:	Client is now booked in to this event and will be redirected to the “My Booked Events” page which will display the event.

Use Case:	6. Booking an Event Offering Seat/Room Reservations
Actors:	Client
Pre-Conditions:	Client has logged in to the system and is on the Book Event page.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client sees an event that offers seat/room reservations that they would like to book in to.</li> <li>2. Client clicks “Book In” underneath the event details.</li> <li>3. Client selects the amount of seats/rooms they require then clicks book.</li> </ol>
Post-Conditions:	Client is now booked in to this event and will be redirected to the “My Booked Events” page which will display the event.

Use Case:	7. Creating an Event
Actors:	Host
Pre-Conditions:	Host has logged in to the system. Host has an access level of ‘Administrator’ or above.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host clicks the Create Event link on the side menu.</li> <li>2. Host fills in the event details and selects whether they would like to allow attendee, seat or room reservations.</li> <li>3. Host clicks Create Event.</li> </ol>
Post-Conditions:	Host has now created an event. Host is presented with an Event Created message.

Use Case:	8. Viewing Created Events
Actors:	Host
Pre-Conditions:	Host has logged in to the system. Host has created an event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host clicks the My Created Events in the side menu.</li> </ol>
Post-Conditions:	System displays all events created by the host.

Use Case:	9. Viewing Attendee Details
Actors:	Host
Pre-Conditions:	Host has logged in to the system. Host has created an event and attendees have booked in to the event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host clicks the My Created Events link on the side menu.</li> <li>2. Host clicks the View Attendees button under the relevant event’s details.</li> </ol>
Post-Conditions:	The system will display all attendee details that are booked in to that event.

Use Case:	10. Removing an Attendee
Actors:	Host
Pre-Conditions:	Host has logged in to the system. Host has created an event and attendees have booked in to the event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host clicks the My Created Events link on the side menu.</li> <li>2. Host clicks the View Attendees button under the relevant event's details.</li> <li>3. Host clicks the Remove Attendees button under the relevant attendee.</li> </ol>
Post-Conditions:	Host has removed attendee from the event. Host is presented with a success message.

Use Case:	11. Cancelling an Event
Actors:	Host
Pre-Conditions:	Host has logged in to the system. Host has created an event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host clicks the My Created Events link on the side menu.</li> <li>2. Host clicks the Cancel Event button under the relevant event's details.</li> </ol>
Post-Conditions:	Host has cancelled event. Host is presented with a success message.

Use Case:	12. Viewing Booked Events
Actors:	Client
Pre-Conditions:	Client has logged in to the system. Client has booked in to an event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client clicks My Booked Events in the side menu.</li> </ol>
Post-Conditions:	System displays all events that the client is booked in to.

Use Case:	13. Cancelling Booking
Actors:	Client
Pre-Conditions:	Client has logged in to the system. Client has booked in to an event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client clicks My Booked Events in the side menu.</li> <li>2. Client clicks Cancel Booking under the relevant event's details.</li> </ol>
Post-Conditions:	Client has removed their booking from the event. Client is presented with a success message.

Use Case:	14. View Messages
Actors:	Client
Pre-Conditions:	Client has logged in to the system. Client has messages.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client clicks Messages link in the side menu.</li> </ol>
Post-Conditions:	System displays all messages the client has received.

Use Case:	15. Message Host
-----------	------------------

Actors:	Client, Host
Pre-Conditions:	Client has logged in to the system. Client is either on the Book Event page or My Booked Events page. Client isn't Host.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client clicks Message Host under the relevant event's details.</li> <li>2. Client enters message then clicks send.</li> </ol>
Post-Conditions:	Client has sent message to Host. System displays a success message.

Use Case:	16. Message Attendee
Actors:	Host, Attendee
Pre-Conditions:	Host has logged in to the system. Host has created an event. Event has attendees. Host isn't attendee.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host clicks My Created Events link in the side menu.</li> <li>2. Host clicks View Attendees under the relevant event's details.</li> <li>3. Host clicks Message Attendee under the relevant Attendee's details.</li> <li>4. Host enters message then clicks send.</li> </ol>
Post-Conditions:	Host has sent message to Attendee. System displays a success message.

Use Case:	17. Replying to a Message
Actors:	Client
Pre-Conditions:	Client has logged in to the system. Client has messages.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client clicks the Messages link in the side menu.</li> <li>2. Client clicks Reply under the relevant message.</li> <li>3. Client enters message then clicks send.</li> </ol>
Post-Conditions:	Client has replied to a message. System displays a success message.

Use Case:	18. Deleting a Message
Actors:	Client
Pre-Conditions:	Client has logged in to the system. Client has messages.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client clicks the Messages link in the side menu.</li> <li>2. Client clicks Delete Message under the relevant message.</li> </ol>
Post-Conditions:	Client has deleted the message. System displays a success message.

Use Case:	19. Change Contact Information
Actors:	Client
Pre-Conditions:	Client has logged in to the system.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client clicks the Profile link in the side menu.</li> <li>2. Client fills out required information then presses the Update Profile button.</li> </ol>

Post-Conditions:	Client has changed their information. System displays a success message.
------------------	--

Use Case:	20. Change Password
Actors:	Client
Pre-Conditions:	Client has logged in to the system. Client knows current password.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client clicks the profile link in the side menu.</li> <li>2. Client fills out current password then new password and the new password again to confirm.</li> <li>3. Client clicks Update Password button.</li> </ol>
Post-Conditions:	Clients password has been changed. System displays a success image. Client can now login with the new password.

Use Case:	21. View Registered Users
Actors:	Super User
Pre-Conditions:	Super User must be logged in. Super User must have 'Super User' access rights.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Super User clicks Accounts link in side menu.</li> </ol>
Post-Conditions:	System displays all registered accounts.

Use Case:	22. Make User an Administrator
Actors:	Super User, User
Pre-Conditions:	Super User must be logged in. Super User must have 'Super User' access rights. User must have 'Client' access.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Super User clicks Accounts link in side menu.</li> <li>2. Super User clicks Make Admin button under relevant user's details.</li> </ol>
Post-Conditions:	Super User has made User an Administrator. System displays a success message.

Use Case:	23. Revoke Administrator Rights
Actors:	Super User, User
Pre-Conditions:	Super User must be logged in. Super User must have 'Super User' access rights. User must have "Admin" access.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Super User clicks Accounts link in side menu.</li> <li>2. Super User clicks Remove Admin button under relevant user's details.</li> </ol>
Post-Conditions:	Super User has made User a Client. System displays a success message.

Use Case:	24. Delete a User
Actors:	Super User, User
Pre-Conditions:	Super User must be logged in. Super User must have 'Super User' access rights. User must not have 'Super User' rights.

Basic Flow:	<ol style="list-style-type: none"> <li>1. Super User clicks Accounts link in side menu.</li> <li>2. Super User clicks Delete User button under relevant user's details.</li> </ol>
Post-Conditions:	Super User has removed User from the system. System displays a success message.

Use Case:	25. Change a System Setting
Actors:	Super User
Pre-Conditions:	Super User must be logged in. Super User must have 'Super User' access rights.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Super User clicks Settings link in side menu.</li> <li>2. Super User fills in required fields then clicks the Update Settings button.</li> </ol>
Post-Conditions:	Super User has changed the system settings. System displays a success message.

### 3.6.2 Mobile Application

Use Case:	1. Connecting via the Application
Actors:	Client
Pre-Conditions:	Client knows the URL of the Online Booking System they wish to connect to. Client is registered with the Online Booking System.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client opens the application and enters the URL of their chosen online booking system.</li> <li>2. Client taps submit.</li> <li>3. Client enters their login details.</li> </ol>
Post-Conditions:	Client is logged in to the online booking system. Client is presented with the Book Event screen.

Use Case:	2. Booking an Event Offering Attendee Reservations
Actors:	Client
Pre-Conditions:	Client is logged in. An applicable event is available.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Client taps the Book Event link in the side menu.</li> <li>3. Client finds an event they wish to book in to. Client swipes left on the event.</li> <li>4. Client taps the Book button.</li> </ol>
Post-Conditions:	Client is booked on to the event. Client is redirected to the My Booked Events screen.

Use Case:	3. Booking an Event Offering Seat/Room Reservations
Actors:	Client
Pre-Conditions:	Client is logged in. An applicable event is available.

Basic Flow:	<ol style="list-style-type: none"> <li>1. Client accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Client taps the Book Event link in the side menu.</li> <li>3. Client finds an event they wish to book in to. Client swipes left on the event.</li> <li>4. Client taps the Book button.</li> <li>5. Client enters the quantity they require then taps the Submit button.</li> </ol>
Post-Conditions:	Client is booked on to the event. Client is redirected to the My Booked Events screen.

Use Case:	5. Creating an Event
Actors:	Host
Pre-Conditions:	Host is logged in. Host has access level of 'Administrator' and above.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Host taps the Create Event button in the side menu.</li> <li>3. Host enters the required information.</li> <li>4. Host taps on the type of reservation they would like (attendees, rooms or seats).</li> <li>5. Host taps Next button.</li> <li>6. Host taps the Event Start field and selects a date and time using the calendar pop up. Host then taps the next button.</li> <li>7. Host taps the Event Stop field and selects a date and time using the calendar pop up. Host then taps the next button.</li> <li>8. Host confirms the event details are correct then taps the Create Event button.</li> </ol>
Post-Conditions:	Event is now created. Host is redirected to the My Created Events screen.

Use Case:	6. Viewing Created Events
Actors:	Host
Pre-Conditions:	Host is logged in. Host has access level of 'Administrator' and above. Host has created an event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Host taps My Created Events link in the side menu.</li> </ol>
Post-Conditions:	Application displays events the Host has created.

Use Case:	7. Viewing Attendees
Actors:	Host
Pre-Conditions:	Host is logged in. Host has access level of 'Administrator' and above. Host has created an event. Event has attendees.

Basic Flow:	<ol style="list-style-type: none"> <li>1. Host accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Host taps My Created Events link in the side menu.</li> <li>3. Host swipes to the left on the selected event and taps the Attendees button.</li> </ol>
Post-Conditions:	Application displays attendees booked on to the event.

Use Case:	8. Removing an Attendee
Actors:	Host
Pre-Conditions:	Host is logged in. Host has access level of 'Administrator' and above. Host has created an event. Event has attendees.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Host taps My Created Events link in the side menu.</li> <li>3. Host swipes to the left on the selected event and taps the Attendees button.</li> <li>4. Host swipes to the left on selected attendee and taps the Remove User button.</li> </ol>
Post-Conditions:	Attendee is removed from event. Screen will refresh to reflect changes.

Use Case:	9. Cancelling an Event
Actors:	Host
Pre-Conditions:	Host is logged in. Host has access level of 'Administrator' and above. Host has created an event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Host taps My Created Events link in the side menu.</li> <li>3. Host swipes to the left on the selected event.</li> <li>4. Host taps the Cancel button.</li> </ol>
Post-Conditions:	Host has cancelled event. Screen will refresh to reflect changes.

Use Case:	10. Viewing Booked Events
Actors:	Client
Pre-Conditions:	Client is logged in. Client is booked in to an event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Client taps My Booked Events link in the side menu.</li> </ol>
Post-Conditions:	Application displays all events Client is booked in to.

Use Case:	11. Cancelling a Booking
Actors:	Client

Pre-Conditions:	Client is logged in. Client is booked in to an event.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Client taps My Booked Events link in the side menu.</li> <li>3. Client swipes to the left on the selected event.</li> <li>4. Client taps the Cancel Booking button.</li> </ol>
Post-Conditions:	Client has cancelled their booking. Screen will refresh to reflect changes.

Use Case:	12. View Messages
Actors:	Client
Pre-Conditions:	Client is logged in. Client has messages.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Client taps Messages link in the side menu.</li> </ol>
Post-Conditions:	Application displays messages Client has received.

Use Case:	13. Message Host
Actors:	Client, Host
Pre-Conditions:	Client has logged in. Client is either on the Book Event screen or My Booked Events screen. Client isn't Host.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client swipes left on the selected event.</li> <li>2. Client taps the Message Host button.</li> <li>3. Client enters message then taps the Send button.</li> </ol>
Post-Conditions:	Client has sent Host a message. Application presents a success message.

Use Case:	14. Message Attendee
Actors:	Host, Attendee
Pre-Conditions:	Host has logged in. Host has created an event. Event has attendees. Host isn't attendee.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Host accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Host taps My Created Events link in the side menu.</li> <li>3. Host swipes left on the selected event.</li> <li>4. Host taps Attendees button.</li> <li>5. Host swipes left on the selected Attendee.</li> <li>6. Host taps Message User button.</li> <li>7. Host enters message then taps the send button.</li> </ol>
Post-Conditions:	Host has sent Attendee a message. Application presents a success message.

Use Case:	15. Replying to a Message
Actors:	Client

Pre-Conditions:	Client has logged in. Client has messages.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Client taps the Messages link in the side menu.</li> <li>3. Client swipes left on selected message.</li> <li>4. Client taps the Reply button.</li> <li>5. Client enters message then taps the send button.</li> </ol>
Post-Conditions:	Client has replied to a message. Application presents a success message.

Use Case:	16. Deleting a Message
Actors:	Client
Pre-Conditions:	Client has logged in. Client has messages.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Client taps the Messages link in the side menu.</li> <li>3. Client swipes left on selected message.</li> <li>4. Client taps the Delete button.</li> </ol>
Post-Conditions:	Client has deleted the message. Screen refreshes to reflect changes.

Use Case:	17. Change Contact Information
Actors:	Client
Pre-Conditions:	Client has logged in.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Client taps the Profile link in the side menu.</li> <li>3. Client taps the Update Info button.</li> <li>4. Client fills in the require information and taps the Submit button.</li> </ol>
Post-Conditions:	Client has changed contact information.

Use Case:	18. Change Password
Actors:	Client
Pre-Conditions:	Client has logged in.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Client accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Client taps the Profile link in the side menu.</li> <li>3. Client taps the Change Password button.</li> <li>4. Client enters current password and the new password followed by the new password again.</li> <li>5. Client taps the Submit button.</li> </ol>
Post-Conditions:	Client has changed password. Client can now log in using the new details.

Use Case:	19. View Registered Users
Actors:	Super User
Pre-Conditions:	Super User must be logged in. Super User must have 'Super User' access rights.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Super User accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Super User taps the Accounts link in the side menu.</li> </ol>
Post-Conditions:	Application displays registered accounts.

Use Case:	20. Grant/Remove User Administration Rights
Actors:	Super User, User
Pre-Conditions:	Super User must be logged in. Super User must have 'Super User' access rights. User must not be Super User
Basic Flow:	<ol style="list-style-type: none"> <li>1. Super User accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Super User taps the Accounts link in the side menu.</li> <li>3. Super User swipes left on User details.</li> <li>4. Super User taps Make/Remove Admin.</li> </ol>
Post-Conditions:	Super User has granted or revoked 'Administrator' access for User depending on their current access level. Screen will refresh to reflect changes.

Use Case:	21. Delete a User
Actors:	Super User
Pre-Conditions:	Super User must be logged in. Super User must have 'Super User' access rights. User must not be Super User
Basic Flow:	<ol style="list-style-type: none"> <li>1. Super User accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Super User taps the Accounts link in the side menu.</li> <li>3. Super User swipes left on User details.</li> <li>4. Super User taps Delete User button.</li> </ol>
Post-Conditions:	Super User has deleted user. Screen will refresh to reflect changes.

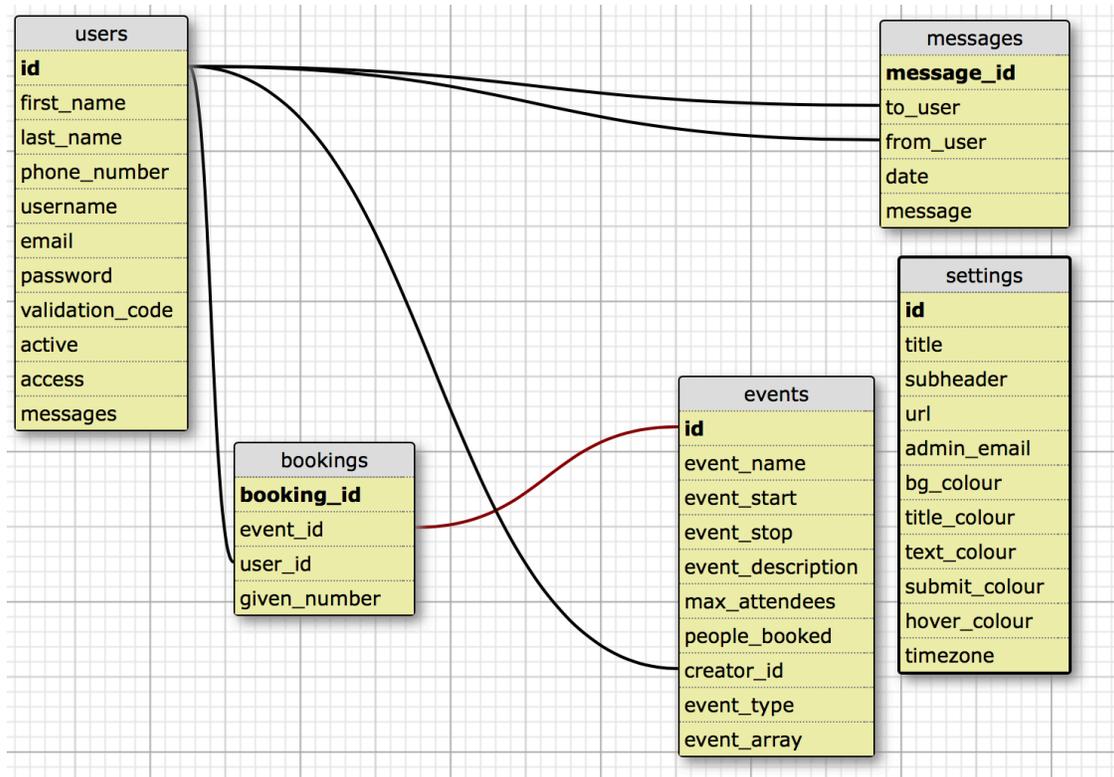
Use Case:	22. Change a System Setting
Actors:	Super User
Pre-Conditions:	Super User must be logged in. Super User must have 'Super User' access rights.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Super User accesses the side menu by either swiping to the right or tapping the button in the top left corner.</li> <li>2. Super User taps the Settings link in the side menu.</li> <li>3. Super User enters the required information.</li> <li>4. Super User taps the Submit button</li> </ol>

Post-Conditions:	Super User has changed the system settings. Application displays a success message.
------------------	---

## 3.7 Database Design

### 3.7.1 Overview

The database will be a MySQL database and shall follow this design:



### 3.7.2 Field Types

#### Users

- **id** – int(11)
- **first\_name** – varchar(255)
- **last\_name** – varchar(255)
- **phone\_number** – varchar(11)
- **username** – varchar(32)
- **email** – varchar(255)
- **password** – text (No limit, password will be encrypted)
- **validation\_code** – text (No limit, code is generated randomly)
- **active** – int(1)

- access – int(1)
- messages – int(11)

#### Bookings

- booking\_id – int(11)
- event\_id – int(11)
- user\_id – int(11)
- given\_number – longblob (Storing serialized array, see Section 4.1.10.4)

#### Events

- id – int(11)
- event\_name – varchar(255)
- event\_start – datetime
- event\_stop – datetime
- event\_description – varchar(255)
- max\_attendees – int(11)
- people\_booked – int(11)
- creator\_id – int(11)
- event\_type – int(1)
- event\_array – longblob (Storing serialized array, see Section 4.1.10.4)

#### Messages

- message\_id – int(11)
- to\_user – varchar(255)
- from\_user – varchar(255)
- date – datetime
- message – varchar(255)

#### Settings

- id – int(1)
- title – varchar(255)
- subheader – varchar(255)
- url – varchar(255)
- admin\_email – varchar(255)
- bg\_colour – varchar(255)
- title\_colour – varchar(255)
- submit\_colour – varchar(255)
- hover\_colour – varchar(255)
- timezone – varchar(10)

## 4 Implementation

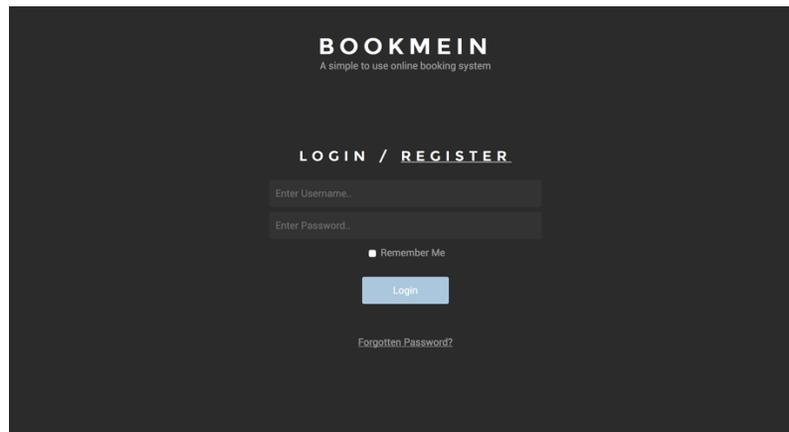
This section will describe the implementation of the system in finer detail. The section will again be broken down into the online booking system, mobile application and installation script. Each will describe the functionality of the relevant package and how this functionality was achieved. This section will not break down the source code but instead include relevant pieces of code that better describes the end result.

### 4.1 Online Booking System

#### 4.1.1 Login & Registration System

##### 4.1.1.1 Overview

I wanted to write the login and registration system code myself, rather than use a previously written example, because I wanted it to integrate perfectly with the rest of the system. I also wanted to be able to specifically choose what information was stored and retrieved in the database, as well as better understand the security behind encrypting passwords so that they do not appear in plaintext view.



The login and registration page will be the first page the user encounters when using the online booking system. When the user clicks the links for Login, Register or Forgotten Password the required fields will slide in and out of view. The title and sub header of the login and registration page are dynamic and are called from the 'settings' table in the database.

When logged in the user will be presented with the main view of the online booking system:



The title, much like the login and registration page, is dynamic and called from the ‘settings’ table in the database. The design is responsive and when the screen resolution becomes too small the side menu is hidden and can be revealed by clicking (or tapping) the icon in the top left corner. This causes the side menu to slide out. This action is implemented using “Bamboo.js” [20].



---

The side menu itself is also dynamic. It displays the current user’s first and last name at the top by checking the session username and using this to call the respective information from the ‘users’ table. The side menu will also only display links according to the user’s access. In a similar fashion the session username is used and checks the users access level from the ‘users’ table. The logout link simply forwards the user to ‘logout.php’ which deletes the session and clears the cookie. The user is then redirected back to the login and registration system.

#### 4.1.1.2 Registration

The registration function takes the information input into the required fields using ‘\$\_POST’[18] and first passes it through a validation function. The validation function first passes the information through the ‘htmlentities’ function [12]. This is done to prevent users from running html code and scripts that may be harmful inside the fields.

Once complete the registration function then passes the information through these validations:

- The username must have than three characters and less than twelve.
- The username does not already exist in the ‘users’ database table.
- The phone number equals eleven digits in length.
- The email address does not already exist in the ‘users’ table.
- The entered password fields match.

If any of these validations fail, a message is placed into the ‘errors’ array and printed at the end of the validation checks. If the ‘errors’ array is empty, the validation function passes the information to the register user function. The register user function first passes all the information through the ‘mysql\_escape\_string’ function [13], which ensures the information is safe to place in a MySQL query to prevent SQL injections [14]. The register function then encrypts the password using the password\_hash [7] function. A validation code, used to provide an activation link, is also generated using:

```
$validation = md5($username + microtime());
```

This code takes the entered username, adds the microtime [15] stamp, then produces the md5 value of the combination. This is done to produce a secure unique code.

This information is then placed in to the database:

- First name.
- Last name.
- Phone number.
- Username.
- Email address.
- The encrypted password.
- The unique validation code.

The database will also add a ‘0’ value to the ‘active’ and ‘access’ fields meaning that the user has not been activated and has ‘Client’ access. The register user function will then send the user an email, using the mail function [16], containing an activation link comprised of the URL of the system followed by:

```
$msg .= "activate.php?email=$email_address&code=$validation";
```

When the user clicks the link the activation receives the email address and validation code using the URL ‘\$\_GET’ [17] variable. This information is again cleaned, using ‘htmlentities’ then used in a query that searches for the email address in the ‘users’ table and compares the received validation code and the code in the table. If a match is found the selected user’s active field is changed from ‘0’ to ‘1’ and the validation code field is changed to ‘0’. The user is presented with a message informing them they are now active and can log in to the system.

#### 4.1.1.3 Login

The login function again takes the entered information and cleans it with ‘htmlentities’. It also checks to see if the “Remember Me?” checkbox has been selected and stores the value in a variable if so. The information is then passed through the following validation:

- Check the username has more than three characters but less than twelve.

- Uses a MySQL query to check whether the user has been activated or not.

If either fail a message will again be placed inside the ‘errors’ array, and displayed at the end of validation. If the array is empty the information is then passed to the login user function.

The login user function takes the information and creates a MySQL query that selects the ‘password’ field data in the ‘users’ table with the row that matches the given username. The function then uses the ‘password\_verify’ [19] function to check the received password matches the encrypted database password. If it does the login function then checks the ‘Remember Me?’ variable, and if true, creates a cookie set to expire 24 hours after the current time using:

```
setcookie('username', $username, time() + 86400);
```

If the variable is false, a session is created in a similar fashion using the username. The login function then forwards the user to the booking system.

#### 4.1.1.4 Forgotten Password

The forgotten password function will allow the user to reset their password via an email recovery link. The forgotten password function receives the entered email address and validates it by first using the ‘htmlentities’ function then using a MySQL query to verify that the user has activated their account. If the ‘errors’ array is empty, the function then creates a validation code similar to the register user function but this time using the user’s email address:

```
$validation_code = md5($email + microtime());
```

The function also sets a temporary cookie for the fifteen minutes within which the validation code must be used:

```
setcookie('temp_access_code', $validation_code, time()+900);
```

The validation code is then placed in the relevant field in the ‘users’ table and an email is sent to the user containing a password reset link comprising of their email and validation code:

```
$message .= "recover.php?email=$email&code=$validation_code";
```

When the user clicks the recovery link in the email the function receives the email and validation code using the ‘\$\_GET’ variable. The function first checks that the validation code received matches the validation code in cookie that was set. If successful, the new passwords are received via the “\$\_POST” variable and checked for a match. The new password is then encrypted using the ‘password\_hash’ function and a MySQL query is created to place the new password in the relevant field within the ‘users’ table where the email address and validation code match. The user may now login with the new password.

#### 4.1.1.5 Notes

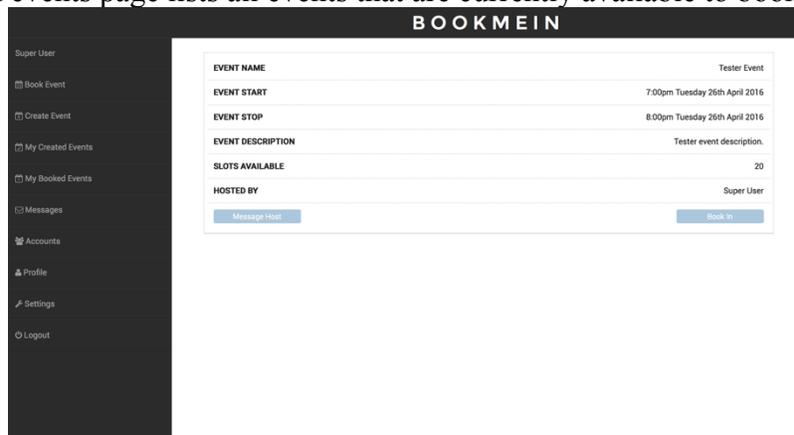
If the user is currently logged in, the Login and Registration system will not be shown. Instead the user will be forwarded directly to the first page of the online booking system. A ‘logged\_in()’ function was created that checks if a session or cookie already exists and

returns true or false based on the results. An IF statement is then placed at the top of the relevant pages that will redirect the user to the first page if ‘logged\_in()’ equals true.

## 4.1.2 Book Event Page

### 4.1.2.1 Overview

The book events page lists all events that are currently available to book.



The book event page first uses a MySQL query to select all information from the ‘events’ table. A while loop is then used to loop through all the events and check the event end time against the current time. If the event time has passed, the event is deleted. If the event is applicable it passes through a few functions:

- An ‘available\_slots’ variable is created by minusing the ‘people\_booked’ field from the ‘max\_attendees’ field.
- Variables for the host’s first and last name are created by retrieving the ‘creator\_id’ (which is in User ID format) and performing a query on the ‘users’ table to retrieve the relevant information.
- The dates are translated from ‘datetime’ format to a more user friendly format using the ‘date\_format’[21] function.
- The ‘event\_type’ field is retrieved and depending on the number given (0,1,2) the event will display Slots Available, Rooms Available or Seats Available.

The information is then displayed in table format:

EVENT NAME	Tester Event
EVENT START	7:00pm Tuesday 26th April 2016
EVENT STOP	8:00pm Tuesday 26th April 2016
EVENT DESCRIPTION	Tester event description.
SLOTS AVAILABLE	20
HOSTED BY	Super User

Message Host      Book In

The book in button in the bottom right corner is only displayed if the ‘available\_slots’ variable is greater than zero.

#### 4.1.2.2 Notes

This page is only viewable when the user is logged in. If the user was to navigate to the physical URL of the page without logging in, they would be redirected to the login screen. The page uses the same ‘logged\_in()’ function used in Section 4.1.1.5.

### 4.1.3 Create Event Page

#### 4.1.3.1 Overview

The create event page allows the user to create an event by entering the required information.

Upon clicking the ‘Start Time’ and ‘End Time’ fields, the user is presented with a calendar where they are able to select a date and time:

This feature was implemented by using ‘DateTimePicker’ a jQuery plugin by XDSOft [22].

#### 4.1.3.2 Notes

This page is only viewable by users of ‘Administrator’ level and above. An IF statement is used to first check whether the user is logged in using the ‘logged\_in()’ function used in Section 4.1.15 and then if the user is of access level ‘1’(Administrator) or ‘2’(Super User). If true the information is displayed, if false the user is redirected to the login page.

## 4.1.4 My Created Events Page

### 4.1.4.1 Overview

The My Created Events works in exactly the same way as the Book Events page reviewed in Section 4.1.2, apart from the first query. Instead of selecting all data from ‘events’ table the query only selects events where the ‘creator\_id’ field matches the user ID of the current user. Afterwards the information is validated and displayed in the same way.

EVENT NAME	Tester Event
EVENT START	7:00pm Wednesday 27th April 2016
EVENT STOP	8:00pm Wednesday 27th April 2016
EVENT DESCRIPTION	Tester event description.
SLOTS AVAILABLE	19
HOSTED BY	Super User
<a href="#">Cancel Event</a>	<a href="#">View Attendees</a>

The buttons displayed under the information differ and display Cancel Event and View Attendees. The View Attendees button is only displayed if the event field ‘people\_booked’ is greater than zero.

### 4.1.4.2 Notes

Again, this page is only visible to users of ‘Administrator’ level and above. This is verified using the same functionality described in Section 4.1.3.2.

## 4.1.5 My Booked Events Page

### 4.1.5.1 Overview

The My Booked Events page again uses the same functionality as the My Created Events page and the Book Events page. This time the query selects events from the ‘bookings’ table which match the user ID. It then takes these matches and queries the ‘events’ table for the specific event details. The results are again then displayed in a table.

<b>EVENT NAME</b>	Tester Event
<b>EVENT START</b>	7:00pm Wednesday 27th April 2016
<b>EVENT STOP</b>	8:00pm Wednesday 27th April 2016
<b>EVENT DESCRIPTION</b>	Tester event description.
<b>HOSTED BY</b>	Super User
<div style="display: flex; justify-content: space-between;"> <span>Message Host</span> <span>Cancel Booking</span> </div>	

The buttons under the event details on this page display Message Host and Cancel Booking. They are always available and not dependant on any conditions.

#### 4.1.5.2 Notes

The My Booked Events page is available to any user as long as they are logged in. This is Verified using the functionality described in Section 4.1.1.5.

### 4.1.6 Messages Page

#### 4.1.6.1 Overview

The Messages page displays all messages the user currently has. It works by using a MySQL query to search the 'to\_user' field in the 'messages' table for matches with the current user's ID. This information is passed through a few validations:

- The message creator's first and last name are called from the 'users' table and displayed.
- The sent date is translated in to a more readable format using the 'date\_format' function.

This information is then displayed in table format.

<b>FROM:</b>	Simon Herrick
<b>SENT:</b>	11:48am Friday 22nd April 2016
<b>MESSAGE:</b>	Tester message
<div style="display: flex; justify-content: space-between;"> <span>Reply</span> <span>Delete Message</span> </div>	

#### 4.1.6.2 Notes

The Messages page is available to any user as long as they are logged in. This is Verified using the functionality described in Section 4.1.1.5.

## 4.1.7 Accounts Page

### 4.1.7.1 Overview

BOOKMEIN	
Super User	
Book Event	
Create Event	
My Created Events	
My Booked Events	
Messages	
Accounts	
Profile	
Settings	
Logout	

ID	34
FIRST NAME	Super
LAST NAME	User
USERNAME	super
EMAIL	@@.com
PHONE NUMBER	07540834068
ROLE	Super User
ACTIVE	Yes
ID	35
FIRST NAME	Simon
LAST NAME	Herrick
USERNAME	sherrick
EMAIL	simon@simonherrick.com
PHONE NUMBER	07540834067
ROLE	Admin

The accounts page simply uses a MySQL query to select all data in the ‘users’ table. This information passes through two forms of validation:

- The user’s ‘active’ field is taken and translated (1 = Yes, else = No).
- The user’s ‘access’ field is taken and translated to a more readable format (0 = Client, 1 = Admin, 2 = Super User).

This information is then taken and displayed in a table.

ID	35
FIRST NAME	Simon
LAST NAME	Herrick
USERNAME	sherrick
EMAIL	simon@simonherrick.com
PHONE NUMBER	07540834067
ROLE	Admin
ACTIVE	Yes
<div style="display: flex; justify-content: space-between;"> <span>Remove Admin</span> <span>Delete User</span> </div>	

The buttons placed underneath the user’s details differ dependant on the user’s access level. If the user is the ‘Super User’, no buttons will be displayed underneath. If the user is an ‘Administrator’ the buttons will display Remove Admin and Delete User. If the user is a Client the buttons will display Make Admin and Delete User.

### 4.1.7.2 Notes

The accounts page is only accessible by the ‘Super User’. This is verified using the Functionality described in Section 4.1.3.2.

## 4.1.8 Profile Page

### 4.1.8.1 Overview

The profile page is split in to two parts. The user is able to update their contact information as well as change their password. The page uses a MySQL query to select the relevant information from the ‘users’ table using the current user’s ID.

### 4.1.8.2 Notes

The Profile page is available to any user as long as they are logged in. This is Verified using the functionality described in Section 4.1.1.5.

## 4.1.9 Settings Page

### 4.1.9.1 Overview

The settings page simply uses a MySQL query to select all data in the 'settings' table. This information is then set to each relevant field.

#### 4.1.9.2 Notes

The Settings page is only accessible by the 'Super User'. This is verified using the Functionality described in Section 4.1.3.2.

### 4.1.10 Book Function

#### 4.1.10.1 Overview

The book event function itself can be separated in to two functions. The first is used for events that allow attendee reservations, the second for events that allow room or seat reservations.

#### 4.1.10.2 Attendee Reservations

The attendee reservation book in function receives two pieces of information, the selected event ID and the current user ID. The function first checks that a booking does not already exist by selecting data from the 'bookings' table where the event ID and user ID match the received information. If no data is found the function can carry on, otherwise the user is presented with a message informing them they are already booked on to this event. If no booking is found a query is created to insert the event ID and user ID in to the 'bookings' table. Another query is then created to increment the 'people\_booked' field by one for the corresponding row in the 'events' table.

#### 4.1.10.3 Room or Seat Reservations

This function receives three pieces of information, the selected event ID, the current user ID and the required quantity. This information is then passed through the best fit allocation algorithm and if the quantity required is available the user will be allocated specific room or seat numbers.

#### 4.1.10.4 Best Fit Allocation Algorithm

The algorithm takes the information and first checks if a booking by that user already Exists in the same way described in Section 4.1.10.2. If a booking already exists, the 'given\_number' field data is selected using a MySQL query then assigned to an array variable using the 'unserialize()' [23] function. If a booking does not exist, the algorithm simply creates an empty array.

The algorithm then uses the given event ID and uses a query to retrieve the 'event\_array' field data from the 'events' table. This data is also then unserialized.

The algorithm then follows the next steps:

- If the event array is empty, it gives the required amount of seats beginning from 1, as long as the required quantity is available.
- If the event array isn't empty it then searches through the array trying to find a gap that is identical to the required quantity. If this cannot be found, the algorithm searches again for the next best fit (required quantity plus one).
- If no gap is present, e.g. 1,2,3,4 are booked, the algorithm will allocate the required quantity directly after.

Once the algorithm has found the required quantity. The given numbers are then inserted in to both the event array and the bookings array, serialized using the 'serialize()' [24] function then inserted in to their relevant tables. A query is then created to update the events field 'people\_booked' by the quantity requested.

#### 4.1.10.5 Applied In

The book function is implemented on the Book Event page described in Section 4.1.2. When the user clicks the Book In button, the information is either passed to the attendee reservation book function or they are required to choose a selected quantity prior to the room or seat reservation book function being implemented. This is dependent on the event type.

### 4.1.11 Create Event Function

#### 4.1.11.1 Overview

The create event function first receives the information via the '\$\_POST' variable and passes the information through the 'htmlentities' function. An empty array is created and then serialized. The information is then passed through some specific validation checks:

- The event name is longer than 3 characters and less than 255 characters.
- The event description is longer than 3 characters and less than 255 characters.
- The start time does not precede the current time.
- The end time does not precede the start time.
- The quantity is more than one.

If the validation passes, the information is then escaped for security and inserted into the 'events' table.

#### 4.1.11.2 Applied In

The create event function is implemented in the Create Event page described in Section 4.1.3. When the user enters the information in the required fields that information is then passed to the create event function.

## 4.1.12 Cancel Event Function

### 4.1.12.1 Overview

The cancel event function receives two pieces of information, the event ID and the current user ID. The function first checks that the current user ID matches the event's creator ID ensuring that only the event host can cancel the event. The function then creates a query to delete the row containing the event ID. The function then selects user ID's from the 'bookings' table that are linked to the event ID. These users are then emailed to inform them that the event has been cancelled. Finally, the function creates a query that deletes any rows that contain the event ID.

### 4.1.12.2 Applied In

The cancel event function is implemented on the My Created Events page discussed in Section 4.1.4. The function is called when the user clicks the Cancel Event button underneath the selected event's details.

## 4.1.13 Message Function

### 4.1.13.1 Overview

The message function receives three pieces of information the recipient's user ID, the current user's ID and the message. The function then checks the message length is between 3 and 255 characters long and records the current timestamp. The recipient's user ID, current user ID, timestamp and message are then inserted into the 'messages' table. The function then creates a query using the recipient's user ID and increments their 'messages' field by one. This means when the user logs in they will be able to see how many new messages they have in the side menu. Finally, the function sends an email to the recipient informing them they have a new message.

### 4.1.13.2 Applied In

The message function is implemented in many different sections of the online booking system:

- The Book Event page (Section 4.1.2) when the Message Host button is clicked.
- The My Created Events page (Section 4.1.4) when the Message Attendee button is clicked.
- The My Booked Events page (Section 4.1.5) when the Message Host button is clicked.
- The Messages page (Section 4.1.6) when the Reply button is clicked.

## 4.1.14 Cancel Booking Function

### 4.1.14.1 Overview

The Cancel Booking function, like the Book Event function (Section 4.1.10), is split in to two sections dependent on the selected event's type. The first is used for cancelling bookings that contained an attendee reservation, the second for cancelling bookings containing room or seat reservations.

### 4.1.14.2 Cancelling Attendee Booking

The function receives two pieces of information, the event ID and the current user's ID. A query is created that deletes the row from the 'bookings' table that matches the event ID and user ID. A query then decreases the 'people\_booked' field in the 'events' table for the relevant event by one.

### 4.1.14.3 Cancelling Seat or Room Booking

The function receives two pieces of information, the event ID and the current user's ID. A query is then created to select the row that matches the event ID and user ID. The 'given\_number' field (the field containing the allocated room or seat numbers) is unserialized. A query then also retrieves the event's 'event\_array' field (the field containing the events currently allocated room or seat numbers). This is also unserialized. The function then passes the event array and booking array through the "array\_diff()" [25] function. This removes all occurrences of the booking array allocations from the event's array. This means these room or seat numbers are available to be allocated again. The event array is serialized then placed back in the 'events' table. The function then removes the selected row from the 'bookings' table. Finally, a query is used to update the 'people\_booked' field for the selected event by subtracting the count of the booking array.

### 4.1.14.2 Applied In

This function is implemented on the My Booked Events page described in Section 4.1.5. The function is called when the user clicks the Cancel Booking button under the chosen booking's details.

## 4.1.15 Make/Remove Admin Function

### 4.1.15.1 Overview

This function simply uses a query to alter the selected user's access level. The 'access' field in the 'users' table is either replaced with a 1 (Make admin) or a 0 (Remove admin). A validation is passed to ensure the 'Super User' is not being edited.

#### 4.1.15.2 Applied In

This function is implemented on the Accounts page described in Section 4.1.7. Under the user's details will appear either a Make Admin button or a Remove Admin button dependent on the user's current access level. If the user is the 'Super User' no buttons will be displayed.

### 4.1.16 Remove Attendee Function

#### 4.1.16.1 Overview

This function receives two pieces of information, the event ID and selected user ID. The Function first checks that the user requesting the removal is the event host. A query is then created to remove the row from bookings that matches this information. An email is then sent to the attendee informing them of the action.

#### 4.1.16.2 Applied In

This function is implemented on the View Attendees page that is displayed when the View Attendees button is clicked on the My Created Events page. Once clicked the system will display an attendee remove success message.

### 4.1.17 Delete User Function

#### 4.1.17.1 Overview

The delete user function is a simple function that receives the selected user's ID then creates a query to remove that user from the 'users' table. The function first checks that the user requesting the deletion is the 'Super User'.

#### 4.1.17.2 Applied In

The delete user function is implemented in the Accounts page described in Section 4.1.7. A Delete User button will appear under the user's details.

### 4.1.18 Update Contact Info Function

#### 4.1.18.1 Overview

This function first checks the user ID of the current user then takes the information received via the '\$\_POST' variable and passes it through the 'htmlentities' function.

The phone number length is then validated, and if successful, a query is created to escape the information and insert it in to the user's row in the 'users' table.

#### 4.1.18.2 Applied In

The update contact info function is implemented on the Profile page described in Section 4.1.8. The user changes the required information then presses the Update Info button. This information is then passed to the function.

### 4.1.19 Change Password Function

#### 4.1.19.1 Overview

This function first takes the received information and passes it through the 'htmlentities' function. The new passwords are then checked to ensure they match. If successful a query is created to retrieve the current user's password from the 'users' table. This password is then checked against the entered current password using the 'password\_verify()' function. If a match is found the new password is encrypted using the 'password\_hash()' function and then inserted in to the user's 'password' field in the 'users' table.

#### 4.1.19.2 Applied In

The change password function is implemented on the Profile page described in Section 4.1.8. The user enters the current and new passwords. This information is then passed to the function.

### 4.1.20 Update Settings Function

#### 4.1.20.1 Overview

This function receives all the information via the '\$\_POST' variable and first passes it through the 'htmlentities' function. The information is then escaped to remove any code for security reasons. It is then inserted in to the 'settings' table.

#### 4.1.20.2 Applied In

The update settings function is implemented in the Settings page described in Section 4.1.9. The user fills in the information required and this information is passed to the function when the submit button is pressed.

## 4.2 Mobile Application

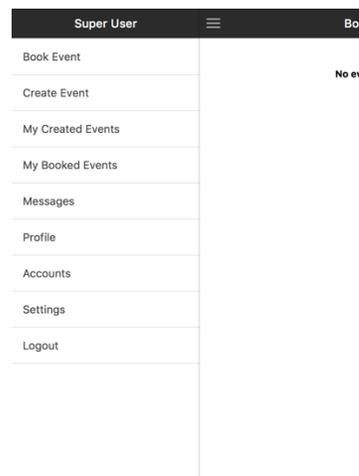
## 4.2.1 Overview

### 4.2.1.1 Overview

This section will describe the implementation of the mobile application. This section will also describe how the mobile application interacts with the online system. For this reason, this section will not describe the functions as they have been discussed in the previous section. Each screen will be described and will reference which functions they used from the previous section.

The mobile application itself was created using the Ionic Framework [9]. This allowed the finished application to be cross platform. Using Ionic also allowed easier integration with the online booking system as Ionic is built using AngularJS [26].

The overall design of the application follows the same design principles used in the online booking system.



Much like the online booking system the side menu displays some dynamic features. The name of the currently logged in user is displayed at the top of the side menu. The side menu only displays links available to the user logged in and is dependent on the user's access level. The side menu can be accessed by either swiping to the right or selecting the icon in the top left corner.

## 4.2.2 URL Entry Screen

### 4.2.2.1 Overview

The URL entry screen gives the user the ability to choose which online booking system they would like to connect to. The user is presented with a URL input field and a submit button. Upon submitting, the application verifies an online booking system is present at that address.



The screenshot shows a mobile application interface for 'BookMein'. At the top, there is a dark header with the text 'BookMein' in white. Below the header is a white form area. The form contains the text 'Enter URL' in a bold font, followed by a smaller line of text: 'Enter booking system url, e.g. http://bookmein.co'. At the bottom of the form is a blue button with the text 'Submit' in white.

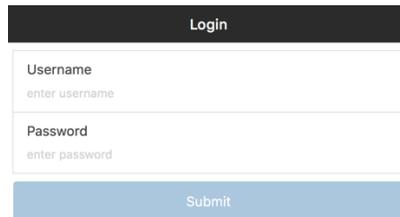
#### 4.2.2.2 Functions Used

Once submitted the screen controller receives the information. The controller then takes the URL variable and adds '/book/app/verify.html' to the end of the variable. The controller then uses the 'http.get' [27] service and looks for a response. If the expected response is given the URL is saved in to the devices local storage using the 'localStorage' item [28]. The controller then displays the login screen.

### 4.2.3 Login Screen

#### 4.2.3.1 Overview

The login screen allows the user to enter the details they would use to login to the online booking system. The user is presented with a text entry field and password entry field followed by a submit button.



The screenshot shows a mobile application interface for 'Login'. At the top, there is a dark header with the text 'Login' in white. Below the header is a white form area. The form contains two text entry fields. The first field is labeled 'Username' and has the placeholder text 'enter username'. The second field is labeled 'Password' and has the placeholder text 'enter password'. At the bottom of the form is a blue button with the text 'Submit' in white.

#### 4.2.3.2 Functions Used

Once submitted the screen controller receives the information. The controller then takes the username and password variables and posts them, using 'http.post' [29] to the URL saved in local storage appended by '/book/app/api.php'. This page on the online booking system then verifies the login using the login function described in Section 4.1.1.3. If this is successful a true response is received by the mobile application controller and the username and password are stored in local storage using 'localStorage'. The controller then displays the Book Event screen.

### 4.2.4 Book Event Screen

#### 4.2.4.1 Overview

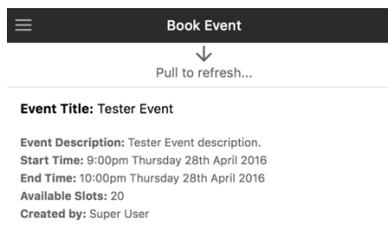
The book event screen displays all events currently bookable.



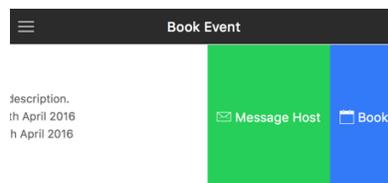
#### 4.2.4.2 Functions Used

The screen first loads events by using the screen controller to retrieve the URL in local storage and append it with `‘/book/app/book.php’`. The controller then uses `‘http.get’` to contact the page on the online booking system and request information. The page uses the function described in Section 4.1.2 to gather the events from the database. The page then converts this information to JSON using the `‘json_encode’` function [30]. The controller receives the JSON information and assigns it to a variable. The screen then displays each event’s information using the `‘ng-repeat’` directive [31].

This screen also implements the `‘ion-refresher’` function [32] which allows the user to swipe down on the screen to reload the events. Swiping down reloads the function described above and applies the modifications to the event list.



This screen also implements the swipe to reveal feature included in the `‘ion-list’` function [33]. This allows the user to swipe on the specific event and be presented with other new options.



If the event is an attendee type reservation, when `‘Book’` is pressed the event ID will be passed to the controller. The controller will then retrieve the username and URL from local storage, with the URL variable being appended by `‘/book/app/bookin.php’`. The information will then be posted to to this URL. This page will use the function described in Section 4.1.10.2 to book the user in to the event. The controller will then forward the

user to the My Booked Events screen.

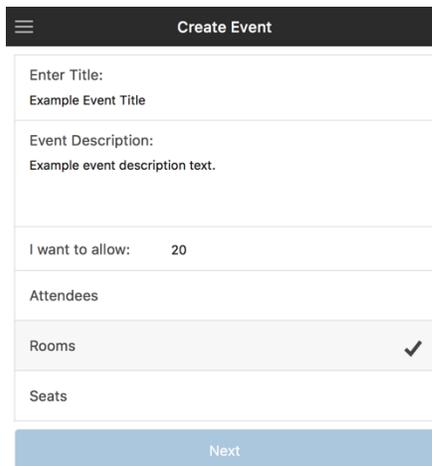
If the event is a room or seat type reservation the steps are similar but prior to posting the information the user is directed to another screen to select the quantity they require. This information is then posted and the page uses the function described in Section 4.1.10.3 to book the user in and allocated them their required quantity of bookings. The controller then forwards the user to the My Booked Events page.

When the user presses the Message Host button they are forwarded to the send message screen described in Section 4.2.8.2.

## 4.2.5 Create Event Screen

### 4.2.5.1 Overview

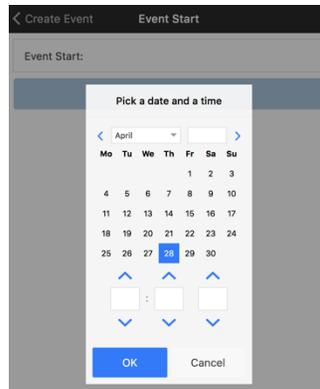
The create event screen displays fields for the required information to create an event. The screen also displays tick boxes for selection whether the reservation type is attendee, room or seat.



Create Event	
Enter Title:	Example Event Title
Event Description:	Example event description text.
I want to allow:	20
Attendees	<input type="checkbox"/>
Rooms	<input checked="" type="checkbox"/>
Seats	<input type="checkbox"/>
Next	

### 4.2.5.2 Functions Used

Once the next button has been pressed the controller receives the information and stores it in local storage. The controller then displays the Event Start screen which allows the user to select the start time using a calendar developed for Ionic called 'ion-datetime-picker' [34]:

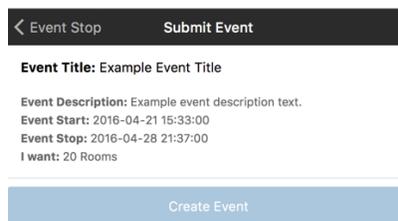


Once the date and time has been picked and the next button has been pressed the controller receives the input and converts it to a MySQL compatible timestamp:

```
var datetimeconv = $filter('date')($scope.datetimeValue, 'yyyy-MM-dd HH:mm:ss');
```

This variable is then placed in local storage and the controller forwards the user to the Event Stop page.

The Event Stop page repeats this process then forwards the user to an overview of the entered event information.

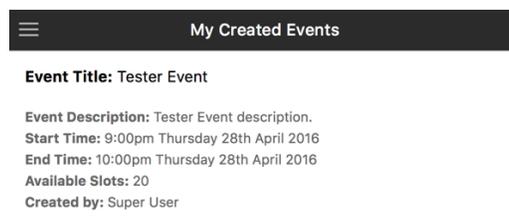


When the create event button is pressed this information is posted to the online booking system which processes it using the function described in Section 4.1.11. If the controller receives a true return, the user is forwarded to the My Created Events page where the event should be displayed.

## 4.2.6 My Created Events Screen

### 4.2.6.1 Overview

The My Created Events screen displays events that the current user has created.

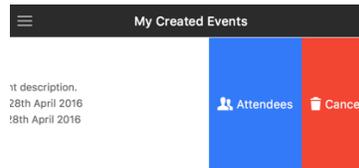


#### 4.2.6.2 Functions Used

This screen displays the information in a similar way as the Book Event page described in Section 4.2.4. The difference is that, in this instance, the controller uses the 'http.post' function to send the current user's username to the online booking system. The online booking system then processes this information using the functionality described in the My Created Events page in Section 4.1.4. The controller then receives the events and displays them using the 'ng-repeat' directive.

This screen also implements the 'ion-refresher' directive described in Section 4.2.4.2. This allows the events to be refreshed by swiping down on the screen.

This screen also implements the swipe to reveal feature described in Section 4.2.4.2. This allows the user to swipe left on the specific event to be presented with more options.



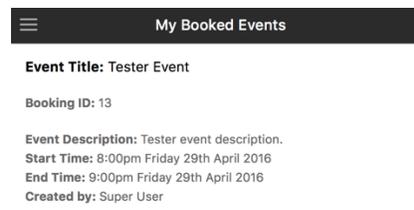
When the user taps the Attendees button, the event ID will be posted to the online booking system which will then send the attendees information back to the controller. The controller will then display the information on a new screen using the 'ng-repeat' Directive.

When the user taps the Cancel button, the event ID and currently logged in user is posted to the online booking system. The online booking system then implements the functionality described in Section 4.1.1.12 to cancel the event and notify the attendees.

### 4.2.7 My Booked Events Screen

#### 4.2.7.1 Overview

The My Booked Events screen displays events that the current user is booked in to.

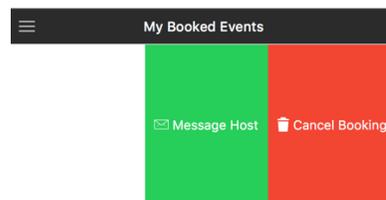


#### 4.2.7.2 Functions Used

This screen displays information in the same way as the My Created Events screen discussed in the section above. The controller posts the username to the online booking system which processes it using the functionality described in Section 4.1.5. The controller then receives the event information and displays it using the 'ng-repeat' directive.

This screen also implements the 'ion-refresher' directive described in Section 4.2.4.2. This allows the events to be refreshed by swiping down on the screen.

This screen also implements the swipe to reveal feature described in Section 4.2.4.2. This allows the user to swipe left on the specific event to be presented with more options.



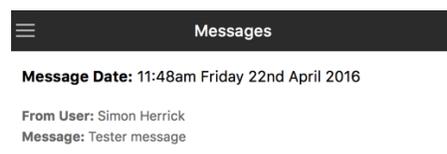
When the user presses the Cancel Booking button their username and the event ID are posted to the online booking system. The online booking system then uses the functionality described in Section 4.1.1.4 to remove the booking.

When the user presses the Message Host button they are forwarded to the send message screen described in Section 4.2.8.2.

### 4.2.8 Messages Screen

#### 4.2.8.1 Overview

The messages screen displays all messages the user currently has stored.



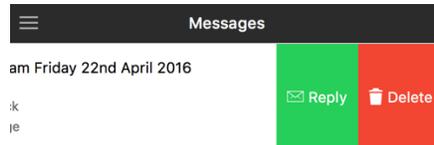
#### 4.2.8.2 Functions Used

This screen works in a similar way to the events display pages. The controller posts the current user's username to the online booking system. The online booking system then

implements the functionality described in Section 4.1.1.13. The controller receives the information and displays it using the ‘ng-repeat’ directive.

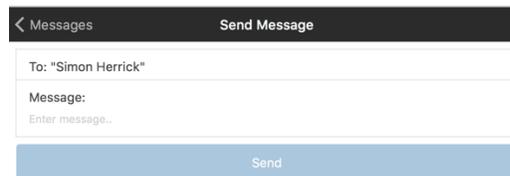
This screen also implements the ‘ion-refresher’ directive described in Section 4.2.4.2. This allows the messages to be refreshed by swiping down on the screen.

This screen also implements the swipe to reveal feature described in Section 4.2.4.2. This allows the user to swipe left on the specific message to be presented with more options.



When the user presses the Delete button the message ID and current user’s username are posted to the online booking system which processes and removes the message.

When the user clicks the Reply button the user is directed to a send message screen. This screen’s controller posts the recipients ID to the online booking system and sends back the recipients first name and last name.

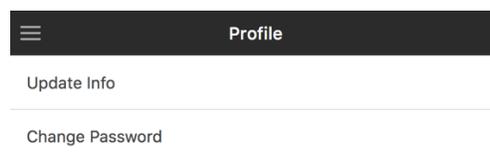


The user then fills in the message and presses send. The recipient ID, current user’s username and message are posted to the online booking system by the controller. The Online booking system then processes the information using the functionality described in 4.1.13.

## 4.2.9 Profile Screen

### 4.2.9.1 Overview

The profile screen simply displays two buttons to forward the user to the required action.



## 4.2.10 Update Info Screen

### 4.2.10.1 Overview

This screen displays the user's current contact details and allows them to edit them.

### 4.2.10.2 Functions Used

First the screen's controller posts the current user's username to the online booking system. The system then retrieves the user's information from the database. The controller receives the user's information and displays it in the given field. The user can now edit the required fields and click submit. The controller posts the new information to the online booking system, which updates into the database.

## 4.2.11 Change Password Screen

### 4.2.11.1 Overview

This screen allows the user to change their password. The user must enter their current password then two new passwords, which must match.

### 4.2.11.2 Functions Used

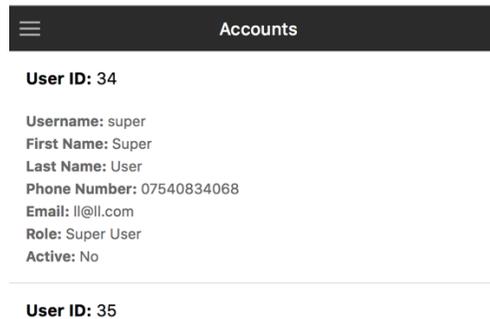
The screen's controller takes the information entered and posts it to the online booking system. The online booking system then passes it through the function described in

Section 4.1.19. If successful the controller will display a success message.

## 4.2.12 Accounts Screen

### 4.2.12.1 Overview

The accounts screen will display all user information for user's registered with the online booking system.

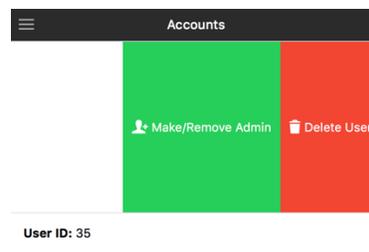


### 4.2.12.2 Functions Used

The controller uses 'http.get' to request the information from the online booking system. The online booking system uses retrieves the user information from the database, converts it to JSON then the controller receives it. The controller displays the information using the 'ng-repeat' directive.

This screen also implements the 'ion-refresher' directive described in Section 4.2.4.2. This allows the messages to be refreshed by swiping down on the screen.

This screen also implements the swipe to reveal feature described in Section 4.2.4.2. This allows the user to swipe left on the specific message to be presented with more options.



When the user selects the Make/Remove admin button the controller posts the selected user's ID to the online booking system. The online booking system then uses the functionality described in Section 4.1.15. The screen will refresh to reflect the changes.

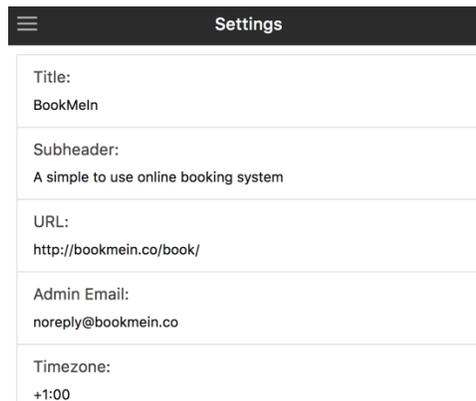
When the user selects the Delete User button the controller posts the selected user's ID to

the online booking system which uses a query to remove that user from the database. The controller then refreshes the screen to reflect the changes.

## 4.2.13 Settings Screen

### 4.2.13.1 Overview

The settings screen displays all the modifiable online booking system settings in editable fields.



Settings	
Title:	BookMeIn
Subheader:	A simple to use online booking system
URL:	http://bookmein.co/book/
Admin Email:	noreply@bookmein.co
Timezone:	+1:00

### 4.2.13.2 Functions Used

The settings screen controller uses the same functionality described in Section 4.2.10. The controller uses 'http.get' to retrieve the current settings from the online booking system. These settings are then displayed in the fields. The user can edit whichever settings they like then press submit. The controller takes the information as a whole and posts it to the online booking system. The online booking system then updates the database table.

## 4.3 Installation Script

### 4.3.1 Overview

The installation script, which is included in the package, allows the user to easily install the online booking system without having to create all the individual database tables. There are a few prerequisites to the script:

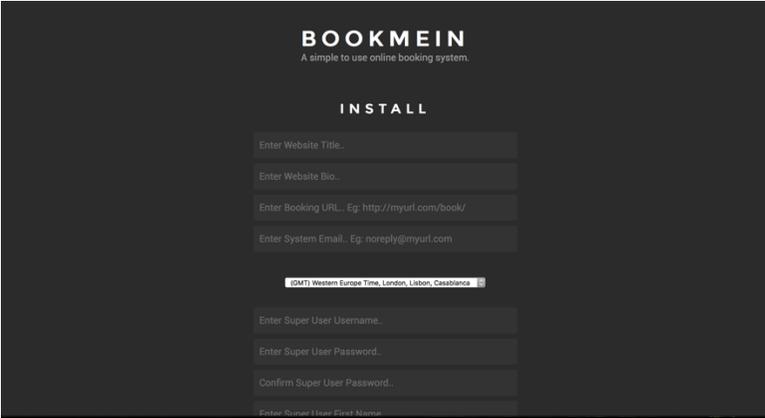
- The user must have a MySQL database created.
- The user must know the username and password for a database user attached to this database.
- This database user must have the appropriate privileges.

The user must edit the /functions/db.php with a text editor and modify it to include the

database name, database username and database password. The user must then upload the book folder in to the root of their webserver, then navigate to

<http://theirurl.com/book/install.php>

These installation instructions are included in the readme and on the download website.



The image shows a dark-themed web form for installing the BOOKMEIN system. At the top, it says "BOOKMEIN" and "A simple to use online booking system." Below that is the word "INSTALL" in all caps. The form contains several input fields: "Enter Website Title.", "Enter Website Bio.", "Enter Booking URL. Eg. http://myurl.com/book/", "Enter System Email. Eg. noreply@myurl.com", a time zone selector set to "(GMT) Western Europe Time, London, Lisbon, Casablanca", "Enter Super User Username.", "Enter Super User Password.", "Confirm Super User Password.", and "Enter Super User First Name".

### 4.3.2 Functionality

The user fills in the required information and presses submit. The installation script contains all the necessary MySQL queries to insert the correct tables in to the database. The script also dynamically uses the information supplied and inserts those in to the relevant fields. Once complete the user must delete the install.php file. They are now able to use the online booking system.

## 5 Testing

### 5.1 Overview

For the testing phase a selection of people with varying degrees of technical knowledge were chosen to perform the main functions and note down how hard they perceived the task to be. As some of this selection had no experience with web servers or of editing code they were given the option to bypass the installation. I did this because I wanted to gauge how easily someone who perceived themselves to have a low technical knowledge would be able to interact with the system and mobile applications.

## 5.2 User Testing

**Online Booking System Questionnaire**

Occupation: IT Technician

Age range: 16-20      21-25      26-35      36-45      45+

Deemed level of computer knowledge ..... Low 1 2 3 4 5 6 7 8 9 10 High

**Installing the Online Booking System**

Do you feel confident uploading files to a web server? (If no please move on to the next section)

Yes / No

How easy was it to enter the database information in db.php? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to use the install.php file? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

Overall, how easy was it to install the Online Booking System? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Using the Online Booking System**

How easy was it to create an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to book in to an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to message the host of the event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to view the attendees of an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel your booking? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to reply to a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to delete a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to make a user an Admin? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the first name of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the password of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the title of the system? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Mobile Application**

How easy was it to create an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to book in to an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to message the host of the event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to view the attendees of an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel your booking? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to reply to a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to delete a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to make a user an Admin? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the first name of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the password of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the title of the system? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Online Booking System Questionnaire**

Occupation: STUDENT  
Age range: 16-20    21-25    26-35    36-45    45+  
Deemed level of computer knowledge ..... *Low* 1 2 3 4 5 6 7 8 9 10 *High*

**Installing the Online Booking System**

Do you feel confident uploading files to a web server? (If no please move on to the next section)

Yes / No

How easy was it to enter the database information in db.php? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to use the install.php file? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
Overall, how easy was it to install the Online Booking System? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*

**Using the Online Booking System**

How easy was it to create an event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to book in to an event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to message the host of the event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to view the attendees of an event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to cancel an event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to cancel your booking? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to reply to a message? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to delete a message? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to make a user an Admin? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy is it to change the first name of the user? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy is it to change the password of the user? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy is it to change the title of the system? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*

**Mobile Application**

How easy was it to create an event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to book in to an event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to message the host of the event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to view the attendees of an event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to cancel an event? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to cancel your booking? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to reply to a message? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to delete a message? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy was it to make a user an Admin? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy is it to change the first name of the user? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy is it to change the password of the user? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*  
How easy is it to change the title of the system? ..... *Very Hard* 1 2 3 4 5 6 7 8 9 10 *Very Easy*

**Online Booking System Questionnaire**

Occupation: Student

Age range: 16-20      21-25      26-35      36-45      45+

Deemed level of computer knowledge ..... Low 1 2 3 4 5 6 7 8 9 10 High

**Installing the Online Booking System**

Do you feel confident uploading files to a web server? (If no please move on to the next section)

Yes/ No

How easy was it to enter the database information in db.php? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to use the install.php file? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

Overall, how easy was it to install the Online Booking System? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Using the Online Booking System**

How easy was it to create an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to book in to an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to message the host of the event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to view the attendees of an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel your booking? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to reply to a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to delete a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to make a user an Admin? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the first name of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the password of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the title of the system? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Mobile Application**

How easy was it to create an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to book in to an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to message the host of the event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to view the attendees of an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel your booking? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to reply to a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to delete a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to make a user an Admin? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the first name of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the password of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the title of the system? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

*Mobile app easy once swipe was found.*

**Online Booking System Questionnaire**

Occupation: student

Age range: 16-20 21-25 26-35 36-45 45+

Deemed level of computer knowledge ..... Low 1 2 3 4 5 6 8 9 10 High

**Installing the Online Booking System**

Do you feel confident uploading files to a web server? (If no please move on to the next section)

Yes  No

How easy was it to enter the database information in db.php? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to use the install.php file? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

Overall, how easy was it to install the Online Booking System? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Using the Online Booking System**

How easy was it to create an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to book in to an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to message the host of the event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to view the attendees of an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel your booking? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to reply to a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to delete a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to make a user an Admin? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the first name of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the password of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the title of the system? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Mobile Application**

How easy was it to create an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to book in to an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to message the host of the event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to view the attendees of an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel your booking? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to reply to a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to delete a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to make a user an Admin? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the first name of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the password of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the title of the system? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Online Booking System Questionnaire**

Occupation: Retiree

Age range: 16-20      21-25      26-35      36-45      46+

Deemed level of computer knowledge ..... Low 1 2 3 / 5 6 7 8 9 10 High

**Installing the Online Booking System**

Do you feel confident uploading files to a web server? (If no please move on to the next section)

Yes / No

How easy was it to enter the database information in db.php? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to use the install.php file? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

Overall, how easy was it to install the Online Booking System? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Using the Online Booking System**

How easy was it to create an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to book in to an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to message the host of the event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to view the attendees of an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel your booking? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to reply to a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to delete a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to make a user an Admin? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the first name of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the password of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the title of the system? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

**Mobile Application**

How easy was it to create an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to book in to an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to message the host of the event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to view the attendees of an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel an event? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to cancel your booking? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to reply to a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to delete a message? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy was it to make a user an Admin? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the first name of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the password of the user? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

How easy is it to change the title of the system? ..... Very Hard 1 2 3 4 5 6 7 8 9 10 Very Easy

### 5.3 Reflection on Testing Results

Upon review of the results there appears to be a clear pattern. The users appear to show a consistent level of rated difficulty until encountering the swipe to reveal feature in the mobile application. I possibly assumed that this feature would be obvious based on the way I would use a mobile application myself. A solution to this issue would be to show a quick ‘how to’ screen the first time the user uses the application. Overall, the testing went well. Users with little perceived technical knowledge were able to complete the given tasks with relative ease.

## 6 Future Work

There are a few ideas that I have identified as possible future work for the project. The project serves as a good base to build these ideas on top of and with minor tweaking these ideas could be easily implemented.

The first area I identified, which is present in some of the examples, is to accept payments through the online booking system. This would allow the hosts of the system to directly charge their user’s deposits or full payments directly through the online booking system or web application. This feature could quite easily be implemented, for instance companies like PayPal provide numerous SDKs [35] which provide support for the majority of the most used languages.

Another idea for future work was the ability to add reservation lists to the bookings. This would mean that if the event was fully booked a set amount of people would be able to add themselves to the reservation list. If someone who was booked in to the event cancelled, the users on the reservation list would be notified and given the opportunity to book in to the event.

Another feature that could be implemented is the upload of a ‘profile picture’ to the user’s profile. I believe this would add to the original scope of the after school childcare system as the parents would be able to upload a picture of the child for better recognition. This feature could also be implemented directly in to the mobile application to allow uploads from the mobile camera.

At current when customising the colour settings for the online booking system the modifications are only represented on the web application. The mobile application colours are static and cannot be changed. Adding the implementation of dynamic colour representation on the mobile application would provide a more integrated user experience. It would be simple to do as when the user enters the URL of the booking system the colours could be retrieved from the database and applied to the styling of the mobile application view.

Another feature that would be included on the future work list is email and mobile notifications. These notifications could be used for reminding users about upcoming appointments, new events or just new messages. The mobile notifications could be in the form of SMS notifications or push notifications.

Finally, it would be interesting to see the web and mobile applications implement some sort of location based feature. Possibly allowing hosts to specify more than one position for events allowing users to receive event recommendations based on the event’s location to the user’s.

## 7 Conclusion

The original scope proposed asked for a system to be created that would allow reservations to be taken for an after school child-care event. It stipulated that the system must allow the organisers to be able to view the details of the child booked in as well as contact details. The final project accommodates the original scope but has also taken a more universal approach offering the end user enhanced functionality in terms of not just allowing attendee reservations but including room and seat reservations, and intelligently allocating these reservations using a best-fit type algorithm to minimise wastage. The original scope of including the ability to view attendees has been completed and extra features added to make the system a powerful tool through giving the users the ability to message each other as well as modify the theme of the website for greater integration. The mobile applications, while not included in the original scope, provide the user with the ability to offer their clients increased access to their services as well as ease of application.

The timeline set out in the initial plan was adhered to up until the implementation of the mobile applications which proved to be a steep learning curve. I had not anticipated this when initially planning the project nor the time it took to develop and code all the functionality that I wanted to include. It meant that more hours had to be dedicated than initially planned so that the project stayed on track and the later sections were completed on time.

The Online Booking System itself is in a workable condition and could be implemented on a webserver and used in its current state. I am happy with the overall design of the system but perhaps would look to alter how some information is displayed. At current the events are listed down the page in chronological order. This is fine for users who do not host many events but for a user who hosts multiple events a day this may become an issue. This would mean that a user looking to select an event a month in advance would have to scroll through numerous events before finding the correct date. I was in two minds whether to implement a calendar selection function for the events so a user could simply click a day in the calendar to show events bookable for that date. I decided against it as in its current form the system would not be used in such a manner so information would more easily be displayed in its current format.

The mobile applications are also in a workable condition and could be implemented alongside an Online Booking System. I am very happy with design of the applications and how well it ties in to the web application. Similar to the online booking system some information could also be displayed in a different manner dependent on the target user.

Overall, I am pleased with the progression of the project and the final product. I believe the system has met the scope and aims of the project to a profession and high degree.

## Reflection

Prior to beginning this project I had some knowledge of web design but mainly focused around static web design. My only real prior experience of dynamic web implementation was a simple blogging project in my first year of University as well as executing already available PHP forum solutions such as phpBB. This experience with PHP forums gave me some knowledge of basic MySQL and management tools such as phpMyAdmin. I had created some very basic mobile applications for iOS using Objective C and Swift but these did not involve any forms of external communication and mainly focussed around basic mathematical equations.

From the offset I had decided I wanted to create applications for both the iOS and Android platforms and wanted to find a solution that would allow me to write both applications at the same time. Ionic very much fitted the brief and provided solutions for connecting to external servers through JSON. I had no previous experience with AngularJS so this is the area that created the steepest learning curve.

The time-frames I had set out in my initial plan was possibly over ambitious especially surrounding the time I had allocated to create the mobile applications. I found myself at this point fighting to keep up with time because I was determined to stick to the original time-frame. In hindsight I would have been better off allowing more time for researching the development of the mobile application gain a better understanding of the work involved to create a more realistic time plan.

Ultimately I am very happy with the project I have produced, as well as the time it was reached in. I believe I added another dimension to the original scope which now provides a much larger set of applications that the project is capable of achieving. I would now feel confident in programming in PHP while communicating with a MySQL database as well as programming in AngularJS as a front-end solution.

## References

1. Chromebook. Viewed 14/4/2016 At: <https://en.wikipedia.org/wiki/Chromebook>
2. Pingdom, A history of the dynamic web. Viewed 14/4/2016 At: <http://royal.pingdom.com/2007/12/07/a-history-of-the-dynamic-web/>
3. PHPJabbers, Appointment Scheduler. Viewed 14/4/2016 At: <https://www.phpjabbers.com/appointment-scheduler/>
4. Ole Jon Bjørkum, phpMyReservation. Viewed 14/4/2016 At: <https://github.com/olejon/phpmyreservation>
5. Planyo, Planyo FREE – Free online reservation system. Viewed 15/4/2016 At: <http://www.planyo.com/free-booking-system.php>
6. Critical Gears, BookingWizz. Viewed 15/4/2016 At: <http://codecanyon.net/item/booking-system/87919>
7. PHP, PHP: password\_hash. Viewed 16/4/2016 At: <http://php.net/manual/en/function.password-hash.php>
8. Bcrypt. Viewed 16/4/2016 At: <https://en.wikipedia.org/wiki/Bcrypt>
9. Ionic, Ionic: Advanced HTML5 Hybrid Mobile App Framework. Viewed 16/4/2016 At: <http://ionicframework.com/>
10. Apple, iOS Human Interface Guidelines: Designing for iOS. Viewed 17/4/2016 At: [https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html?utm\\_source=twitterfeed&utm\\_medium=twitter](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html?utm_source=twitterfeed&utm_medium=twitter)
11. Android, Design | Android Developers. Viewed 17/4/2016 At: <http://developer.android.com/design/index.html>
12. Tizag, PHP Tutorial – htmlentities. Viewed 17/4/2016 At: <http://www.tizag.com/phpT/php-entities.php>
13. PHP, PHP: mysql\_escape\_string – Manual. Viewed 17/4/2016 At: <http://php.net/manual/en/function.mysql-escape-string.php>
14. W3Schools, SQL Injection. Viewed 18/4/2016 At: [http://www.w3schools.com/sql/sql\\_injection.asp](http://www.w3schools.com/sql/sql_injection.asp)
15. PHP, PHP: microtime – Manual. Viewed 18/4/2016 At: <http://php.net/manual/en/function.microtime.php>
16. PHP, PHP: mail – Manual. Viewed 18/4/2016 At: <http://php.net/manual/en/function.mail.php>
17. PHP, PHP: \$\_GET – Manual. Viewed 18/4/2016 At: <http://php.net/manual/en/reserved.variables.get.php>
18. PHP, PHP \$\_POST – Manual. Viewed 18/4/2016 At: <http://php.net/manual/en/reserved.variables.post.php>
19. PHP, PHP \$\_GET – Manual. Viewed 18/4/2016 At: <http://php.net/manual/en/function.password-verify.php>
20. Andrew Greig, Bomboo.js – responsive sliding menu. Viewed 20/4/2016 At: <http://www.andrewgreig.com/617/>
21. PHP, PHP: date-format – Manual. Viewed 20/04/2016 At: <http://php.net/manual/en/function.date-format.php>

22. XDSOft, DateTimePicker. Viewed 20/04/2016 At:  
<http://xdsoft.net/jqplugins/datetimerpicker/>
23. PHP, PHP: unserialize – Manual. Viewed 21/04/2016 At:  
<http://php.net/manual/en/function.unserialize.php>
24. PHP, PHP: serialize – Manual. Viewed 21/04/2016 At:  
<http://php.net/manual/en/function.serialize.php>
25. PHP, PHP: array\_diff – Manual. Viewed 21/04/2016 At:  
<http://php.net/manual/en/function.array-diff.php>
26. AngularJS. Viewed 23/04/2016 At:  
<https://en.wikipedia.org/wiki/AngularJS>
27. AngularJS, AngularJS: API: \$http. Viewed 22/04/2016 At:  
[https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)
28. W3Schools, HTML5 Web Storage. Viewed 24/04/2016 At:  
[http://www.w3schools.com/html/html5\\_webstorage.asp](http://www.w3schools.com/html/html5_webstorage.asp)
29. AngularJS, AngularJS: API: \$http. Viewed 24/04/2016 At:  
[https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)
30. PHP. PHP: json\_encode – Manual. Viewed 24/04/2016 At:  
<http://php.net/manual/en/function.json-encode.php>
31. AngularJS, AngularJS: API: ngRepeat. Viewed 26/04/2015 At:  
<https://docs.angularjs.org/api/ng/directive/ngRepeat>
32. Ionic, ion-refresher. Viewed 27/04/2015 At:  
<http://ionicframework.com/docs/api/directive/ionRefresher/>
33. Ionic, ion-list. Viewed 27/04/2015. Viewed 27/04/2015 At:  
<http://ionicframework.com/docs/api/directive/ionList/>
34. Kate Miháliková, ion-datetime-picker. Viewed 30/04/2015 At:  
<https://github.com/katemihalikova/ion-datetime-picker>
35. PayPal, PayPal SDKs – PayPal Developer. Viewed 2/05/2015 At:  
<https://developer.paypal.com/docs/classic/lifecycle/sdks/>