Easy Booking: A Web Application for Booking Sports Facilities on Campus

Chris Redahan
B.A. (Mod.) Business and Computing
Final Year Project, April 2015
Supervisor: Mary Sharp

School of Computer Science and Statistics

O'Reilly Institute, Trinity College, Dublin 2, Ireland
Declaration

I hereby declare that this project is entirely my work and that it has not been submitted as an exercise for this or any other university.

___________________________  ______________________
Name                                      Date
Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Mary Sharp for her constant guidance and advice over the course of this project. I would also like to thank my fellow classmates, friends and family for their continued support during the year.
Abstract

This report aims to outline the research, design and implementation of a web application for booking sports facilities in a University setting. While Trinity College Dublin’s Sports Centre allows users to telephone, walk in or email their requested time, this can be a long and arduous task when your preferred time slot is not available. Easy Booking is an online system that will allow the user to clearly see what pitches are available and at what times, and then book the time slot they choose.

Easy Booking is a web application that can be opened on a desktop browser, tablet or mobile phone, which allows the user to carry out their tasks remotely without the need to go into reception, or start an email or phone dialogue, to negotiate a time slot or booking.

A business model has been developed to maximise the potential of this web application, and to highlight the potential opportunities from its development. Finally, future work on this project will be discussed as well as an evaluation of the project’s development.
# Table of Contents

Declaration ........................................................................................................... 2  
Acknowledgements ............................................................................................. 3  
Abstract .............................................................................................................. 4  

List of Figures ....................................................................................................... 7  

1 Introduction ....................................................................................................... 8  
1.1 Aims............................................................................................................... 9  
1.2 Personal Goals............................................................................................. 9  
1.3 Motivations.................................................................................................. 10  
1.4 Reader’s Guide............................................................................................ 11  

2 Background ....................................................................................................... 13  
2.1 Chapter Introduction.................................................................................. 13  
2.2 Web Applications and Accessibility......................................................... 13  
2.3 The Current TCD Sports Centre System................................................. 14  
2.4 Alternative Booking Software................................................................. 14  
2.4.1 Club Manager.ie.................................................................................... 14  
2.4.2 BookingBug.com................................................................................. 15  
2.5 Chapter Conclusion................................................................................... 16  

3 Design ............................................................................................................... 17  
3.1 Design Introduction.................................................................................. 17  
3.2 Problem Overview................................................................................... 17  
3.3 The Incremental Software Development Model...................................... 17  
3.4 Requirements........................................................................................... 18  
3.4.1 Functional Requirements..................................................................... 19  
3.4.2 Non-functional Requirements............................................................. 20  
3.5 Technical Approach.................................................................................. 23  
3.6 Database development............................................................................ 24  
3.6.1 Database Tables and Attributes Naming Schema.............................. 24  
3.7 User Interface.......................................................................................... 25  

4 Technical Implementation ............................................................................... 27  
4.1 Implementation introduction.................................................................. 27  
4.2 The Front-end......................................................................................... 27  
4.3 The Database......................................................................................... 28  
4.3.1 Database Structure............................................................................... 28  
4.4 Sign Up.................................................................................................... 31  
4.5 Display Available Time Slots & Making a Booking............................... 33  
4.5.1 Displaying the Available Time Slots.................................................. 33  
4.5.2 Make a Booking.................................................................................. 36  
4.6 Login/Logout............................................................................................ 37  
4.7 View ‘My Bookings’ & Delete a Booking.................................................... 38  
4.7.1 Display Upcoming and Previous Bookings.......................................... 39  
4.7.2 Delete Booking.................................................................................... 39  
4.8 Chapter Conclusion.................................................................................. 41  

5 Chapter 5 – Evaluation ..................................................................................... 42  
5.1 Evaluation Introduction........................................................................... 42  
5.2 Testing....................................................................................................... 42  
5.2.1 The Author’s Testing......................................................................... 42
List of Figures

Figure 3.1 From paper prototype to a mock-up of the displaying available timeslots table................................................................. 25
Figure 4.1: The database booking_system_db and its tables ......................................................... 29
Figure 4.2: The table 'bookings' and its attributes........................................................................... 30
Figure 4.3: The relational view of the bookings table ................................................................. 31
Figure 4.4: SQL query to insert user's details into the database ..................................................... 32
Figure 4.5: The current display table for available bookings...................................................... 34
Figure 4.6: SQL query to select only the available timeslots for the display ........................... 35
Figure 4.7: Snippet of PHP code from test2.php, which sends the variables slot_id and pitch_id to confirmbooking.php where the booking is made ............................. 36
Figure 4.8: PHP $_GET method to extract data from the URL..................................................... 36
Figure 4.9: SQL query to insert booking details into the database........................................... 37
Figure 4.10: Using PHP sessions to get the user's id................................................................. 37
Figure 4.11: If statement determines whether 'Login' or 'Logout' is shown ....................... 38
Figure 4.12: SQL query to display upcoming and previous bookings ........................................ 39
Figure 4.13: JavaScript function confirmDelete() ...................................................................... 40
Figure 4.14: SQL query to delete a booking.............................................................................. 40
Figure 5.1: The SUS results stored in an excel document......................................................... 44
Figure 5.2: SUS results scale to display results ............................................................................ 44
Figure 5.3: New nav bar with name displayed ............................................................................ 46
Figure 5.4: PHP code snippet, displays name if logged in......................................................... 46
Figure 5.5: The old and new sign up forms compared .............................................................. 48
Figure 5.6: Date selection and calendar drop down menu.................................................... 49
Figure 7.1 Ash Maurya’s Lean Canvas Business Model ............................................................... 59
1 **Introduction**

University sports facilities yield huge benefits to the organisation that possesses them, not to mention their members and users. However, these facilities are hugely expensive and are often underutilized and, as a result, they fail to capitalize on their full potential as revenue generators. If we take our very own Trinity College Dublin as a case example of this, we see how they recently (within the last six years) had a new sports facility built on campus, yet there still exists no online system for members to book sports facilities. This results in the TCD Sports Centre missing out on potential revenues for bookings. Currently, members are required to either telephone, email or visit reception in person.

The problem with telephoning and emailing is the customer has no visualized information of what pitch is available. Because of this, the user must begin a dialogue with and question the receptionist until a time that suits both the user and their friends is available on the system. This can be a long and arduous task for both the customer and the receptionist, which often results in the customer needing to retreat in order to discuss a new time with their friends, classmates, or work colleagues. This can often lead to the failure of potential bookings due to the barriers that exist when a customer tries to book an available timeslot.

Easy Booking is a web application that aims to allow users to log in and book a certain sports facility or space, be it for five-a-side football, squash, tennis or basketball. By granting the user the ability to go online and book their chosen space, at their chosen time, Easy Booking will be saving both the user and the receptionists time, whilst simultaneously generating increased revenue for the University Sports Centre.
1.1 Aims

The aim of this project was to establish a solid piece of foundation software that could be taken further towards developing a fully functional booking system for Universities. The initial aim was to set up a booking system based on Trinity College Dublin's five-a-side football pitches.

The web application attempts to achieve the aforementioned aim by means of the following:

1. Create a database unique to Trinity College Dublin's five-a-side football facilities
2. Store all necessary data in the booking_system_db database
3. Allow the user to search the availability of on chosen dates and book a pitch they choose
4. Clear presentation of the time table and pitch availability in order to increase the usability of the web application

1.2 Personal Goals

Along with the aforementioned aims, the author had goals of his own for this project. Having grown up in a society obsessed with technology and software applications, choosing this degree was a deliberate move by the author towards being able to understand the technology that surrounds us and to hopefully one day develop some software of his own. These goals are as follows:

I. Learn how to create a database
II. Learn how to design a web application
III. Learn the necessary languages and technologies that are required to build this online booking system

Thankfully, these goals were achieved. The author learned how to build a relational database, at least as far as this project required. He also learned sufficient amounts of the necessary languages (PHP, SQL, HTML, CSS and some
JavaScript) and technologies (e.g. MAMP) that are required to build and design a web application.

1.3 Motivations

The motivation for this project came from the difficulty that was experienced, by the author and his Business and Computing classmates, when trying to book a pitch each week. As a group, the author and some classmates enjoys playing five-a-side football once or twice a week but regularly come into trouble when trying to book a pitch. The group discusses what time it suits them to play using a Whatsapp group. One person then goes to the TCD Sports Centre to book the time. This works fine when the pitch is available at that time, but if it is not, then the process starts again. People have to go back to their calendars and see when they have lectures, tutorials, work etc. The end result is that the group usually only plays one game every two weeks. For the sports centre this is a loss of between two and six bookings a month on just one group.

The problem with the current medium for booking these pitches is that the person going to book a pitch is given very little information. They ask a question such as; ‘Is the botany bay football pitch available for 11am on Friday morning?’ and usually get a boolean response of yes or no. This means the person has to ask again, and again until they find a pitch that is available except this time they do not know if their friends are available to play at this time. Whether this is carried out in person, over the phone or by email this can be a long and arduous task for both parties involved.

Easy Booking aims to create an online web application that will clearly display the timetable and available time slots, and will allow the user to book their chosen facilities in an easier and more time efficient manner.
1.4 Reader’s Guide

Chapter 2 – Background

This chapter examines the author's decision to use a web application design. It also looks at the current methods for booking sports facilities on campus and the other companies that provide a similar service to Easy Booking.

Chapter 3 – Design

This chapter covers the design that was created for this project and how the author went about creating said design. It looks at everything from the software development model that was used and the initial requirements specification, through to the mock up design for the user interface and the database design.

Chapter 4 – Technical Implementation

This chapter will outline how the core functionalities were implemented and the process that the author went through. In this chapter, each functionality will be carefully explained using snippets of code.

Chapter 5 – Evaluation

This chapter will discuss the various methods that were used to evaluate the project and the system in its current state. The methods used were firstly my own testing, then user testing for all the functionalities, the system usability scale (hereby referred to as SUS) and a meeting and demonstration with Dr. Sonja Hermann who is a lecturer of Human Factors in software design in Trinity College Dublin.

Chapter 6 – Future Work

This chapter will discuss the future work that the author plans to complete for this project.

Chapter 7 – Business Opportunities
This chapter will look at the business opportunities for Easy Booking going forward and why The Lean Canvas Business Model was chosen for this project. The Lean Canvas that was created for Easy Booking can be seen in appendix.

Chapter 8 – Conclusion
2 Background

2.1 Chapter Introduction

This chapter will discuss the current system available in Trinity College Dublin’s Sports Centre and why this can and should be upgraded to an online system. This chapter will also look at the alternative technologies available, their pros and their cons.

2.2 Web Applications and Accessibility

A web application, or web app, is any software that uses a web browser as a client, that is, it is software that runs in the browser. This means that web apps require internet access in contrast to native applications that do not necessarily require internet access. Although, native apps such as mobile apps, do not need to internet to be opened and used, many of the functions that they are used for do in fact require internet access. The project was originally designed to be a mobile application built with PhoneGap and was going to be deployed for both Android and iOS. However, the reason this project was switched from being developed on PhoneGap to being a web app was two-fold.

Firstly, the developer was having trouble accessing the database in PhoneGap as it does not interpret .php files. This was a problem, as the developer uses MAMP as his local server and database, which is built for use with PHP. Secondly, it was after this that the developer realised that the reason he chose PhoneGap was not a technical one, but rather a personal one. The developer began researching web applications, and ultimately opted to go for this design as it granted the end-user more accessibility.

The users should be able to access Easy Booking on a laptop, desktop, tablet or mobile phone to check the timetable and make their bookings. Building a web app allows the user to do this.
2.3 **The Current TCD Sports Centre System**

Currently, Trinity’s system for booking sports facilities, from tennis courts to five-a-side football pitches, is carried out in three different ways. Members can email, call reception or walk in and stand in the queue to talk to a receptionist. The problem with these three ways is that the user is never given more information than they ask for. This means that if the time the user initially wanted is not available they will have to ask about other available times, report this information back to their friends, wait for nine responses, and then try to book the pitch again. This can be quite a drawn out and frustrating task, which is simply unnecessary considering the technology available today.

2.4 **Alternative Booking Software**

Currently, there exists online software that is being sold by companies both in Ireland, the UK and further afield, that all aim to create an easy online booking system for organisations and customers. This section will discuss two of the more relevant websites, as they are similar to Easy Booking’s model.

2.4.1 **Club Manager.ie**

This is an Irish system that aims to provide a similar service to Easy Booking. This software is a very similar to Easy Booking in that it allows an organisation to pay for a service that allows for online booking of their sports systems.

One drawback of this system is that it offers deals on up to 1000 members and then charges €5 a month for every additional 100 members. In the 2013/2014 academic year Trinity College Dublin had 2,868 (Full-Time Equivalent/FTE) staff and 16,729 registered students. This makes a total of 19,697 potential members before adding the number of public members that join the sports centre. This would mean there would be an additional fee of approximately €980 per month on top of the standard clubmanager.ie fee of €30 a month (for five sports), totaling €1,010 a month or €12,120 per annum.
This is before any additional and unexpected costs are incurred. And more significantly, this is before we adjust for the number of public members which could be anywhere from 500 to 1,500. If we add the cost of the 1,000 (€5 x [1000/100]) public members together with the cost of the members we’ve previously accounted for we get a figure of €1,030 a month. And that is just the additional cost for the number of members.

Clubmanager.ie is a good piece of software, however it seems it is not designed for an organisation the size of Trinity. It is also worth noting that the users of clubmanager.ie cannot pay for their bookings online, which is something that Easy Booking plans to implement in the future (see future plans 6.2.1).

2.4.2 BookingBug.com

This is the closest online booking system to Easy Booking that the author came across. It is also the most professional and suitable for an organisation such as Trinity College Dublin. BookingBug.com offers an online scheduling system for both staff and customers and can be embedded in an organisation’s website. On their website they do not specify a price for an organisation the size of Trinity but they do state that such organisation would require a contract in order to avail of the service.

BookingBug.com is a fully functioning piece of software that is also well funded1, making it a strong competitor in the market for online booking platforms. However, as stated previously in the motivations section, this project is not aiming to build a better product than BookingBug. This project aims to build a foundation system that Trinity could implement and, thus, make it easier for people to book their desired sports facilities and reduce the time that receptionists spend dealing with said bookings.

---

1 BookingBug raised a further $1 million in funding in 2014, and in their seed round of funding in 2011 they raised $350,000 (O’Hear, 2014).
BookingBug.com uses PayPal to allow their users to pay for their bookings online. The developer also plans to implement a payment system for Easy Booking, however in contrast to BookingBug the developer has chosen to use the Stripe API. The reasons for this will be discussed in depth in chapter six (see 6.2.1). One significant reason for choosing Stripe over PayPal is security and the handling of the user’s sensitive credit card details.

2.5 Chapter Conclusion

Firstly, this chapter discussed the developer’s reason for choosing the web application design over a PhoneGap app or even a native mobile app. It looked at the rise of web applications and accessibility that a web app offers the user in comparison to solely developing a mobile app. Secondly, it looked at the problem of obesity in Ireland and why systems like Easy Booking need to be considered for implementation in organisations such as Trinity College Dublin, as they make it easier for people to engage in physical activity that is both sociable and enjoyable. It then discussed the current methods for booking pitches and facilities in Trinity and the current alternatives to Easy Booking that exist today. Although similar systems exist such as ClubManager.ie and BookingBug.com, neither of these systems are designed in precisely the same way as Easy Booking. ClubManager.ie does not allow users to pay for their bookings online, and BookingBug.com uses PayPal for their payments, rather than the developer’s favoured Stripe.
3 Design

3.1 Design Introduction

This chapter will cover the design that was created for this project and how the author went about creating said design. It will cover everything from the software development model that was used and the initial requirements specification, through to the mock up design for the user interface and the database design.

3.2 Problem Overview

When starting out with this project it was apparent there were a few obvious problems both large and small, that would have to be overcome. For example one such problem was the steep learning curve that was involved in order to develop this project, and the overall sense of the unknown that lay ahead for the author. Along with this, there were things such as; database development, connection to the front-end, user log in, storing user’s details, and displaying the available times in a clear and concise manner, which were all design and/or implementation problems that were known to the developer from the start.

There were also some unforeseen problems that were encountered along the way. One of these problems was the aforementioned problem with PhoneGap and PHP. As previously mentioned, late into the project the author decided PhoneGap was not the right choice for this type of project and instead opted for a web application software structure and design. This took some reformatting, redesign and some testing. In the end it was worth it, as the current product is a much-improved version of what PhoneGap was allowing the author to produce.

3.3 The Incremental Software Development Model

In the beginning of this project it was decided that a software development model should be chosen in an effort to focus the development
process. The waterfall development method was researched but ultimately it was rejected as the author felt that, as this was his first solo-programming project, he would need more flexibility with the requirements and design than the waterfall method would allow. In Winston W. Royce's original waterfall model there were seven phases in developing the software: requirements specification, design resulting in the software architecture, construction, integration, testing and debugging, installation and maintenance (Royce, 1970). The waterfall method states a developer should only move onto the next phase of development when the last phase has been finished, reviewed and verified. Thus, beginning a one-way process or flow, hence the name. This inflexibility was what led to further research and ultimately a different model being chosen.

The model that was chosen was the incremental model as this would allow the author to develop the web app in stages by planning, designing, coding and implementing on a recurring basis, rather than strictly moving ‘downstream’. This gave the author the flexibility to redesign the system and to add and take-away features as the project moved along.

Ultimately, the development of this project was done in five sprints. The first sprint was the front-end design. This was done to give the author a better idea of the systems design and requirements. The second sprint was the creation of the relational database and the various tables that were needed. The third sprint was where the sign up function. The fourth sprint focused on Easy Booking’s core function, which is to display available time and to allow the user to make a booking. The fifth and final sprint included implementing the functionalities of log in, log out, view my bookings, delete a booking and other added pages.

3.4 Requirements

There was no 3rd part client for this project, which meant that the author was the developer and also responsible for drawing up the requirements document in phase one. The requirements document specified both functional
and non-functional requirements for this project. These will both, in turn, be discussed in this section.

### 3.4.1 Functional Requirements

The requirements set out in the beginning were as follows: to allow users to sign up, log in, log out, make a booking, view their bookings, delete a booking and pay for their booking online. With the exception of the online payment, all of these requirements were met.

#### 3.4.1.1 User Sign Up

The first functional requirement was a user ‘Sign Up’ function. As a user of multiple websites and web apps such as Soundcloud, Spotify, facebook, IMDb, and YouTube to name a few, the author favors the idea of having a profile on sites that can keep track of what the user is using the site for. Using profiles allows for a more valuable, beneficial and enjoyable user experience. It was decided that the database should not only record when pitches were being booked, but also who was making the booking. This would then allow for the ‘My Bookings’ section to be designed. The profiles would also allow for the collection of valuable sales/bookings data that could be analysed in the future to produce sales forecasts. This will be discussed in chapter six (see section 6.2.5).

#### 3.4.1.2 Log In and Log Out

The second functional requirement was a user log in system. The user would enter their email address, which had to be unique upon sign up, and would then enter their password. Once signed in they would remain signed in until they clicked the log out button, which was present on each page. The aforementioned ‘Logout’ button would be the same button as the ‘Login’ button. This would dynamically change depending on whether the user had logged in yet, or not.
3.4.1.3 *Make a Booking*

The third functional requirement, and the core of the whole project, was to implement a booking function for football pitches on campus. This would require displaying the available pitches and other information such as date and time, and allowing the user to easily view and absorb this information before making their booking. This functionality was by far the hardest to implement and will be discussed in depth in chapter four Technical Implementation.

3.4.1.4 *View ‘My Bookings’*

The fourth functional requirement was a ‘My Bookings’ section where the user could view their previous and upcoming bookings. This feature would require all three of the previous section to work before this could be implemented. It would also add to the user’s experience and is another benefit of using user profiles with Easy Booking.

3.4.1.5 *Delete a Booking*

The fifth and final functional requirement of the project was to allow the user to delete a booking they had previously made. Again, this feature required the previous four functions to be completed prior to its implementation. It was decided that the delete section would fit the best, in the ‘My Bookings’ part of the web app. The idea was that a user could view their bookings and, for upcoming bookings only, they could click a ‘delete’ button. The non-functional requirement worth mentioning here is usability.

3.4.2 *Non-functional Requirements*

As mentioned previously, several non-functional requirements were outlined in the beginning of the project.

3.4.2.1 *Reliability Requirements*

Reliability is the ability of a system to perform its required functions under stated conditions for a specific period of time. This can be considered
under the headings of availability, failure-rate and dependability (Cremers, 2006).

First and foremost, Easy Booking should be available when requested, reliable and dependable. There are multiple reasons for this, for example if Easy Booking is regularly not available or not functioning then its reputation will be ruined and the users would simply not use the system. It could also stop someone logging in and deleting a booking that they had made. This would mean the slot could go unused and would result in a loss of potential earnings for the sports centre.

3.4.2.2 Performance Requirements

Performance requirements concern the speed of operation of a system. Within performance there are different types of measurements such as the response time and the throughput time, which measure how quickly a system reacts to user input and how much the system can accomplish within a specified time, respectively (Cremers, 2006).

The author realised the project should perform its task quickly and efficiently in order to increase the usability and overall user experience that the system would offer.

3.4.2.3 Usability

Usability, in this case, is the ease with which the user can learn to use, navigate and operate the system (Cremers, 2006). For this, the following requirements were set out, well-structured user interfaces, informative error messages and help facilities that would help the user get to grips with how the system works. Other metrics outlined here were learning requirements, which denoted time needed to learn how to use the system, handling requirements and likeability.

3.4.2.4 Supportability

Supportability requirements are concerned with the ease of changes to the system after deployment (Cremers, 2006). The two main supportability
requirements Easy booking should go by were adaptability, maintainability. And tailorability

3.4.2.4.1 Adaptability

The ability to change the system to deal with additional application domain concepts.

3.4.2.4.2 Maintainability

The ability to change the system, to deal with new technology or to fix defects. This was particularly relevant to Easy Booking as there would regularly be additional features added, which will be discussed in chapter six, and new technologies that could improve the system.

3.4.2.4.3 Tailorability

Totter and Stary (1999) write about how “tailorability is traditionally described as a feature of interactive software that allows the change of certain aspects of the software in order to meet different user characteristics and requirements.”

This is especially relevant to Easy Booking, as the software will have to be changed slightly from client to client. Tables in the database will have to be specifically designed for the customer so as to accommodate the different sports they wish to have on the Easy Booking web app, the number of pitches/courts, and the times that these facilities are available for booking. As well as the database changes affecting the user interface, there will also be other client specific changes. For example, if Easy Booking were to be embedded into a client’s website then they might want the colour scheme to match, or they might want the users’ data to be passed by the website’s log in and not Easy Booking’s own log in page.
3.5 Technical Approach

This project was originally being developed for iOS using PhoneGap however late into the project it was decided Easy Booking should be designed for more than just phones. This decision was made after coming across some major difficulties using PHP with SQL in PhoneGap. Although these difficulties were being dealt with overtime, it was at this stage the author spoke to a number of classmates and began to question why he had chosen PhoneGap in the beginning. The reason, quite simply, was that the author had grown up through the app explosion, which began in 2008 (after the App Store launch on July 10th, 2008), and had always wanted to build his own app.

The author, after realising the main reason choosing PhoneGap was because of a personal ambition rather than a technical decision, changed tact and opted for the web application design. There were also a few technical reasons for this. Firstly, the end-user should be able to open Easy Booking on their laptop or desktop in the library, in class, in the lab or in work and easily make a booking. They should not be restricted to using their phone to make a booking. Secondly, using PhoneGap presents some technical difficulties of its own. For example, PhoneGap does not allow you to use PHP files, as quoted from their website “a PhoneGap application may only use HTML, CSS, and JavaScript” (PhoneGap, 2012). This was a problem as the project relied heavily on communicating with a database and dynamic data to be loaded to the page, which back-end scripting is useful for, e.g. telling a user they're logged in, displaying available times, confirming a booking was deleted. It was for these reasons that PhoneGap was put aside and the author opted for a web application design.

There was a steep learning curve involved in developing this project as the author had little experience with the various technologies needed. Although the author had some experience with HTML and CSS, the author had no experience with PHP, JavaScript, SQL, MAMP and database design, which was a
significant part of the project. The author made use of sites such as Stackoverflow.com, W3CSchools.com and PHP.net, to name a few.

3.6 Database development

Prior to this project, the author had no previous experience with database development, MySQL or relational databases. It is because of this that the author regrets not taking the Information Management II module that is available to Junior Sophister students. In order to learn about relational database development and design the database correctly, it was necessary for the author to obtain past lecture slides and notes from the aforementioned module.

An entity relationship diagram was prepared before the actual booking_sys.db was created. This can be seen in appendix 9.4.

3.6.1 Database Tables and Attributes Naming Schema

For this project, after some initial database work it was decided that a naming schema would be needed in order to structure that database accordingly. The following naming schemas were followed:

1. Tables were named as plurals e.g. bookings rather than booking, this was because the table would contain several instances of a booking. Each instance of a booking is an individual record.
2. All attributes or columns were named as singular and never plural.
3. If a name contained two or more words (e.g. user_id) then an underscore would be used to separate the two names.
4. Each table would have a primary key.
5. Each primary key would be an auto incremented attribute.
6. All characters would be lowercase.
3.7 User Interface

The core function of Easy Booking is to allow users to make a booking. To allow a user to make a booking quickly they must be able to easily absorb and interpret the times and pitches that are available on a specified date. The author imagined a very user friendly and concise way of displaying this information. This design went from a paper-sketch to a front-end prototype (see figure 3.1) before finally being changed for the beta version that was ultimately demonstrated on April 1st. This will be discussed further in Chapter 4: Technical Implementation.

The mock-up had a nice table structure with the available pitches being displayed as either blue or red, if available, blue representing outside (colder temperature) and red representing the indoor pitches. If a pitch was booked out at a certain time and not available it could be displayed as a grey and black icon and would not be a clickable button.

Figure 3.1 From paper prototype to a mock-up of the displaying available timeslots table

*Note the grey and black icons where a pitch is not available to be booked. In this particular screenshot only four hour long slots are being displayed, in contrast to the full 12-hour day that the system operates on.*
4 Technical Implementation

4.1 Implementation introduction

As mentioned previously in the Design chapter (Chapter 2) when developing this project the author opted for the incremental development methodology rather than the waterfall method. This chapter will outline how the aforementioned functionalities were implemented and the process that the author went through. This project had a steep learning curve attached to it and this directly affected the development process that the author undertook. If this project was being undertaken again the author would have a much clearer understanding of what was involved and would therefore be able to map out exactly what was needed, when and why. However, for this project the steep learning curve often dictated what was being developed, when and how.

In this chapter, each functionality will be carefully explained using snippets of code.

4.2 The Front-end

To start this project, the author decided to develop a mock front-end in an effort to better understand what the system would look like and what was required (see figure 3.1).

After developing the mock front-end this was then built upon and developed into a more usable web app front-end. Although the author had some previous experience with HTML and CSS this was not sufficient to build the front-end that was required.

It was decided that the front-end would have a navigation bar with a minimum number of tabs. The tabs that were decided upon were as follows; ‘Home’, ‘Booking’, ‘Sign Up’, ‘Info’ and ‘Login/Logout’. The final tab would display login or logout depending on whether a user was currently logged in or not. This will be discussed in the upcoming section ‘4.6 Login/Logout’. Some of
these tabs then had sub-tabs, for example the 'Booking' tab had the sub-tabs of 'Make a Booking' and 'View My Bookings'.

As discussed in the 'User Interface' section in the Design chapter, the background to the front-end was a darker image with lighter coloured text. This was done to increase the readability of the text to a user. The opposite would also work, that is choosing a lighter background and darker text. The reason for opting for the former and not the latter was down to the author's personal preference for a darker background.

4.3 The Database

Creating a relational database was crucial in the implementation of this project. It was because of this that the author studied lecture slides from Declan O'Sullivan's Junior Sophister module 'Information Management II', on designing a relational database.

For testing and demonstration purposes, it was decided to develop the database on a local server. The software that was decided upon was MAMP. MAMP is very similar to LAMP and WAMP except it is designed for Apple Macintosh computers. MAMP stands for Macintosh, Apache, MySQL and PHP.

As this was the author's first time creating any sort of database it was decided that a mock-up database be developed in order to grasp the concepts of a relational database such as primary keys and inter-table attribute relationships. After the mock-up database was developed and the author felt familiar with the concepts of a relational database, the 'booking_system_db' was created.

4.3.1 Database Structure

From the offset, the database structure had to be clear and concise due to the data it would be handling. The Easy Booking database should have
multiple tables that would be linked or related to each other. The tables would be as follows: bookings, memberships, payments, pitches, teams, timeslots, and users (see figure 4.1 below). The bookings table was the most 'linked' table of all the tables in the booking_system_db, and is therefore a good starting point for explain the structure of the database.

![Server: localhost:8889 - Database: booking_system_db](image)

**Figure 4.1**: The database booking_system_db and its tables

Bookings has the following attributes; booking_id, user_id, team_id, time_slot, payment_id, date and pitch_id (see figure 4.2 below). The primary key in this table is the booking_id attribute, which is set to automatically increment every time a new booking is created. The booking table is designed to store the relevant information about each unique instance of a booking.
Figure 4.2: The table ‘bookings’ and its attributes

Note the primary key ‘booking_id’ is underlined, as it is the primary key in the bookings table.

As mentioned previously, this is a relational database where the tables are linked to other tables by certain attributes. As can be seen in figure 4.3, the attributes user_id, time_slot, payment_id and pitch_id are all linked to their respective tables. If we look at user_id we can see how it is linked to the table ‘users’ and the attribute in that table of the same name.
Figure 4.3: The relational view of the bookings table

Having a relational database structured in this way made storing and handling the necessary data easier later on. For the beta version that was demonstrated on April 1st the author had to sever the link between payment_id in bookings and payment_id in payments. The reason for this was that the payment system was not ready in time to demonstrate and having this link prevented the bookings from being inserted into the database because the payment_id column was not being filled with any data.

4.4 Sign Up

Although the sign up functionality is not at the very core of this project it was decided to implement this before the core booking function. The reason for this was to learn how PHP and SQL are used together to connect to the database. The sign up form was built using HTML and CSS and was inspired by a tutorial found online at formget.com (2014).
This particular method of submitting user input to a database uses AJAX and jQuery as well as PHP and SQL. AJAX (Asynchronous JavaScript and XML) is the art of exchanging data with a server, and updating parts of a web page – without reloading the whole page (FormGet.com, 2014). jQuery is a fast, small, and feature-rich JavaScript library, which makes things like HTML document traversal and manipulation, event handling, animation, and AJAX much simpler with an easy-to-use API that works across a multitude of browsers (jQuery.com, 2015).

The reason for choosing this design for the sign up form was largely due to the speed that it handled the data and reacted to the user's actions. This decision was made with the performance requirements outlined in the Design chapter in mind.

Three files were created for this to work. A .js file, a .html file and a .php file as well as some CSS added to the index.css file. The HTML file would build the form with CSS and, once the user enters their details and clicks submit, would send the user's input to the .js file. This file would contain the jQuery and AJAX code that would present and alert window if the form was no filled in correctly. However, if the form were filled in correctly these details would be posted (using PHP's POST) to the .php file in a datastring. The details would then be inserted (using SQL's INSERT) into the database. The .php file had four steps to carry out in order to perform its task. These were as follows; first it would establish connection with server, then the database is selected (in this case we are selecting booking_system_db), next the SQL INSERT query is executed (see figure 4.4), before finally closing the connection with the server.

```php
//Insert query
$sql = "INSERT into users(student_number, first_name, second_name, email, password) values (".$student_number2", "$first_name2", "$second_name2","$email2","$password2")";
$query = mysqli_query($conn, $sql);
```

Figure 4.4: SQL query to insert user's details into the database
4.5 Display Available Time Slots & Making a Booking

Displaying the available time slots and pitches was by far the hardest problem that was encountered in this project. This is after all, the core functionality of Easy Booking. This section will discuss the problem, the failed attempts and how it was eventually solved.

4.5.1 Displaying the Available Time Slots

The first part to this functionality was that the available time slots had to be displayed in an effective manner for the user to easily absorb and interpret before they made their decision and made a booking. As was discussed in Chapter 2: Design, the original design for display table included red, blue and grey buttons and was a much more concise and visually pleasing display (see figure 3.1 in chapter 3). However, although this was the author’s ideal visual display and was relatively easy to code in HTML and CSS it proved too difficult to make a reality when it came to the beta implementation. The author failed to devise a way to query the database for available time and to dynamically display these times in a table of this style, in time for demonstration. It is for this reason this has been added to the future work section as the author believes this to be an important feature in the Easy Booking web app. Usability is one of the non-functional requirements that was outlined for this project and the author believes that displaying the available time slots in a clear and concise manner will increase Easy Booking’s usability as an online booking system.

The settled upon display table (see figure 4.5) was more of a list of available time slots. As you can see, the list starts with the first hour slot (8am-9am) and goes through each pitch from one to three. In this list, a time slot and pitch is only displayed if it is available to be booked. This is different to the mock-up where the pitches were displayed in grey when they were not available to be booked.
**Figure 4.5: The current display table for available bookings**

*Note how timeslot number 2 is not being displayed, as it has already been booked. And note how the button for time slot 1 is shaded slightly red at the bottom, this is the colour of the button when the cursor hovers over the button.*

As mentioned previously, this was the hardest part of the project to implement, as it required careful data handling using PHP and SQL. The way it works is that the user selects the date they are interested and are then shown all the available times for that day. They can then scroll through the times before they find the time they want. At this time they can see what pitches are available. They then click the button (which is the pitch number) and the booking is made.
When the user selects the date they want that date is posted (using PHP’s $_POST method) from booking.php to test2.php. In test2.php the date is reformatted and then used in an SQL query (see figure 4.6) which requests only the available dates from the database. These dates are passed to test2.php and then displayed in the list format that can be seen in figure 4.5. After this, the user can simply click the green and blue button with the pitch number on it and the booking will be made.

As previously mentioned, the database design was crucial to the author’s ability to implement a functional front-end. This was especially evident when the problem of trying to display the available times arose. Originally, the database had a table called ‘timeslots’ which in it had twelve hour long slots stored in it. This was to be traversed three times, once for each pitch, using an SQL query. The idea was that from the one table of 12 slots, and the pitches table that had three records in it, the author should be able to display a table with all the times on it. This proved more difficult than anticipated. After many failed attempts with different SQL queries, the author finally realised the timeslots table was not what he needed. Instead, he designed a new table ‘availables’ which had 36 slots, one slot for every pitch at each hour. After this redesign was made to the database, the SQL query (see figure 4.6) was easier to design and that is how the current display table was developed.

```php
$date1 = $_POST["date"]; $date2 = date('Y-m-d', strtotime(str_replace('/', '-', ($POST["date"]))));
$sql = "SELECT slot_id, start_time, finish_time, pitch_id FROM availables WHERE slot_id NOT IN (SELECT slot_id FROM bookings WHERE bookings.booking_date = ".$date2."); $result = $conn->query($sql);
```

Figure 4.6: SQL query to select only the available timeslots for the display
The SQL query here uses the ‘SELECT FROM’ and ‘NOT IN’ SQL syntax in order to request only the timeslots from the ‘availables’ table that are not already booked. Note how the date variable had to be converted from the user input format of dd/mm/yyyy to the database date format of yyyy-mm-dd.

4.5.2 Make a Booking

When the user clicks the button to make a booking they will be redirected to the confirmbooking.php page. The necessary parameters are sent with the URL using this PHP code:

```
<button class="btn3">([$row("pitch_id")]–2).</button></a></td>:
```

Figure 4.7: Snippet of PHP code from test2.php, which sends the variables slot_id and pitch_id to confirmbooking.php where the booking is made.

The snippet of code above shows the variables being passed to the page confirmbooking.php. This is where the PHP method $_GET is used to extract the values (see figure 4.8) from the URL and use them in the SQL query to make the booking (see figure 4.9).

```
$user_id1 = $_SESSION["user_in_id"];
$time_slot1=$_GET["time_slot"];
$date1=$_GET["date"];
```

Figure 4.8: PHP $_GET method to extract data from the URL

The values for ‘time_slot’ and ‘date’ are extracted from the URL using the PHP $_GET method. Note the user’s id is being taken from the PHP session. The author acknowledges that using the $_POST method here would be more secure and plans to implement this in the near future.
The values necessary to make a booking are inserted into the database using this SQL query. Note how the $sql variable contains the string which is then executed in the mysqli_query. $conn is the server and database connection that was set up at the start of the PHP file.

### 4.6 Login/Logout

The login and logout functionalities were implemented using PHP sessions. PHP sessions are used to store details about a user that is currently using the web application. The details are set with the PHP global variable $SESSION, which acts like an array storing values under different variable names. For example, in my code the user’s ID is passed to the relevant pages using the line of code shown below (see figure 4.10). Prior to this, at the very start of the PHP code of each page the session needs to be started by calling the session_start() function.

In the navigation bar there is a tab for logging in or logging out. This is a dynamic tab, which uses sessions to check if a user is currently logged in or not. When a user is logged out there session is cleared. This enabled the author to the PHP isset() method (see figure 4.11) to decided whether to display the ‘Login’ or ‘Logout’ button.
Figure 4.11: If statement determines whether 'Login' or 'Logout' is shown

This is the if-statement used to determine whether the user is logged in or not, and thus to show the 'Login' or 'Logout' tab.

The author also fixed the problem of users not being logged in after they had signed in. This was a simple fix; the author used the PHP session code that is in the login.php page. This now means once a new user is signed up they are automatically signed in.

4.7 View ‘My Bookings’ & Delete a Booking

The view ‘My Bookings’ section was also implemented using PHP sessions and the user’s ID. Just like the login and logout functionalities, the ‘My Bookings’ page calls on the PHP global variable $_SESSION and from this it is able to access the current user’s name and id number. These are then included in the SQL queries that request the bookings from the database (see figure 4.12).
The SQL queries that call for the upcoming bookings and the previous bookings. Note how the current date ($curr_date) is used to determine which bookings are in the future and which have already past.

4.7.1 Display Upcoming and Previous Bookings

The current date is stored in a variable named $curr_date and this is used to determine which bookings are in the future and which are in the past (see figure 4.12). These bookings are then displayed in a similar way to the available bookings table in test2.php.

4.7.2 Delete Booking

The delete-a-booking functionality was added in on the recommendation of a user after the first round of user testing was carried out. It was during this time that the author realised he could not simply put a delete button on all bookings and would have to only allow the user to delete upcoming bookings and not bookings that had already taken place. This was for data and payment reasons. A user should not be able to delete their booking history as it will directly tamper with the databases records and affect payments.
When the delete button is clicked a JavaScript function confirmDelete() is called (see figure 4.13). This function first displays a confirmation window to the user to ensure that they want to delete this booking.

```
function confirmDelete(b_id) {
  var x;
  if (confirm("Are you sure you want to delete this booking?") == true) {
    x = "You pressed OK! Your booking was deleted."
    var y = s_time;
    window.location.href = "deletebooking.php?booking_id=\"+b_id;\"\/+\"+s_time；
  } else {
    x = "You pressed Cancel! Your booking was not deleted."
  }
  document.getElementById("demo").innerHTML = x;
}
```

Figure 4.13: JavaScript function confirmDelete()

The JavaScript function confirmDelete() is called when the user clicks a delete button. Note the confirmation window, and how the booking id is sent in the URL to the deletebooking.php page.

If the user confirms that they do want to delete it then the booking id is sent with the URL to deletebooking.php where an SQL query is used to delete the specific booking that was chosen (see figure 4.14).

```
$sql_delete = "DELETE
FROM bookings
WHERE booking_id = \\
"$booking_id1\";"
```

Figure 4.14: SQL query to delete a booking

SQL query to delete a booking record from the bookings table. Note that after this booking is deleted it will now be displayed when a user goes to search available time slots.
4.8 Chapter Conclusion

This chapter looked at the technical implementation of this project from the front-end to the back-end and all the functionalities that were developed over the past six months.

Snippets of code were included in order to give the reader an idea of how the author went about implementing these functionalities and overcoming any problems that arose.

Any technologies that were used in the development of this project such as the chosen local server and database software MAMP, and all the necessary languages both for the front-end and the back-end have been discussed and explained in this chapter.
5 Chapter 5 – Evaluation

5.1 Evaluation Introduction

This chapter will discuss the various methods that were used to evaluate the project and the system in its current state. The methods used were firstly my own testing, then user testing for all the functionalities, the system usability scale (hereby referred to as SUS) and a meeting and demonstration with Dr. Sonja Hermann who is a lecturer of Human Factors in software design in Trinity College Dublin.

5.2 Testing

5.2.1 The Author’s Testing

The first testing to take place was that of the author. This testing includes everything that the user needed to check before the web app was ready to be tested by other users. This testing ranged from the more basic front-end visual testing to the more technical back-end and SQL query testing. For example, testing the navigation bar for the front-end was more of a trial and error process whereby the author was trying to get the alignment of HTML divs just right. An example of the more technical testing was the testing that had to be done to check that the SQL queries were calling the right available time slots from the database. This took some trial and error, and ultimately would not have been possible if it wasn’t for the book ‘A Guide to SQL’ by Pratt and Last (2009), kindly lent to the author by Prof. Mary Sharp.

5.2.2 User Testing For All Functionalities

After receiving ethical approval for research from Trinity’s School of Computer Science and Statistics Ethics Committee, the author began to test my web app prototype with users (see the ethics approval documents in appendix 9.2). The users that were involved in the testing were a combination of classmates, friends and family.
Each user was asked to use the web app on a specified date. These users would almost always go to the ‘login’ or ‘make a booking’ pages and were then successfully directed to the sign up page. User suggestions, struggles, problems and successes were noted and recorded as the testing was carried out. As mentioned previously, it was during this user testing that one user suggested that they should be able to delete a booking. This was incorporated into the current version of the software and is proof as to the author of the value that user testing provides.

5.3 System Usability Scale (SUS)

After each user testing session they were asked to anonymously fill out the SUS questionnaire. The SUS (system usability scale) is a list of ten statements where the user is asked to state on a scale of one to five if how much they agree with each statement. One means the user ‘strongly disagrees’ with the statement and five means they ‘strongly agree’ with the statement. These answers were recorded in an excel document on the author's laptop (see figure 5.1). With the SUS statements the odd statements are the ones where the developer would rather see a four or a five showing the user agrees. For example, number three states “I thought the system was easy to use”. In contrast from this, for the even numbered statements the developer would rather see users disagree and answer with more one’s and two’s. For example, number two states “I found this system unnecessarily complex”. At present the author has only tested the web application with 21 users and, as a result, only has 21 responses to the SUS survey. The results from this survey can be seen in figure 5.2. The results table in figure 5.2 uses two different tools to display the results in an easily absorbable manner. Firstly, the green shading in the cells gets darker as the number of responses that are recorded in this cell increases. Secondly, there is a blue bar that represents the number of users that selected that answer as a percentage of the whole.
Figure 5.1: The SUS results stored in an excel document

<table>
<thead>
<tr>
<th>SUS</th>
<th>Answers</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>I think I would like to use this system frequently</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Q2</td>
<td>I found the system unnecessarily complex</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Q3</td>
<td>I thought the system was easy to use</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Q4</td>
<td>I thought that I would need the support of a technical person to be able to use the system</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Q5</td>
<td>I found the various functions of this system were well integrated</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Q6</td>
<td>I thought there was too much inconsistency in this system</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Q7</td>
<td>I would imagine that most people would learn to use this system very quickly</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Q8</td>
<td>I found the system very cumbersome to use</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Q9</td>
<td>I feel very confident using the system</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Q10</td>
<td>I needed to learn a lot of things before I could get going with this system</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Test Subject:</td>
<td>70</td>
<td>62.5</td>
<td>67.5</td>
<td>72.5</td>
<td>65</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.2: SUS results scale to display results

The results here have been made into a table with the answer scale on the top (horizontal x-axis) and the statement number on the vertical y-axis. Note how the green shading in the cells gets darker as the number of answers there increases, as well as a blue bar-scale showing the percentage of all users that choose this answer.

5.4 Human Factors Testing with Dr. Sonja Hermann

On Thursday the 3rd of April 2015, the author met with Dr. Sonja Hermann, a lecturer of Trinity College Dublin. The author is a past pupil of Sonja’s, taking her Human Factors module from October to December 2014.
The meeting was set up to see what advice Dr. Hermann could offer after using the Easy Booking web application and testing all of its functionalities. It is important to note here that Dr. Hermann's testing and advice was specifically aimed at the front-end design and she did not look into the database design.

Dr. Hermann gave the author some useful constructive criticism and advice, which will be discussed now. The seven main things that Dr. Hermann noticed were as follow:

1. It is not evident what sports you can actually book
2. On the landing page make it so that you can click on the image to go to booking
3. Visually not clear that you are logged in
4. There should be an alternative route to booking, not just via the navigation bar
5. The sign up form layout is confusing
6. The date selection for making a booking is too small
7. Should be able to search by pitch number and/or date

Some of these changes were implemented by the author, however some required deeper structural change that the author did not have time to do before April 7th. It is because of this that the author is disappointed that he did not have a working prototype finished sooner as he could have met with Dr. Hermann in time to implement all the recommended changes.

The first thing Dr. Hermann pointed out was that it was not clear what sports could be booked using Easy Booking. The author explained that the future aim is to be able to book for a range of sports and possible study rooms on campus, but at present the prototype was designed for booking football pitches only.

The second suggestion from Dr. Hermann was that the image on the landing page be turned into a clickable link to another part of the web app. The
The author has since implemented this change by linking the image to the ‘make a booking’ page and has also linked the welcome text to the login page.

The third suggestion was that it was not visually apparent to the user if they were logged in or not. The author has since changed this and the user can now see that they are logged in because their name will be in the navigation bar (see figure 5.3). The code for this used PHP sessions and the isset() method (see figure 5.4).

The navigation bar now displays the user’s name when they are logged in. Note how the ‘log out’ tab is now a sub-tab under the user’s name. The user’s name is now a link to their ‘my bookings’ page.

The navigation bar has a tab that is dynamically built depending on if the user id logged in or not.
Dr. Hermann suggested that there should be more than one pathway to get to the bookings page, not just the tab in the navigation bar. Although this was already partially implemented this has been reviewed and further implemented. The user can now get to the bookings page by clicking on buttons that have been inserted into several different pages.

The fifth point Dr. Hermann mentioned was that the sign up form was poorly laid out, as it was not evident which details were to be entered into each text box. The problem here was spacing. The original layout of the sign up form (see figure 5.5) had the text “First Name” closer to the input field for “Student Number”. This not only confused Dr. Hermann but several of the users who tested the web app and has since been changed (see figure 5.6). Dr. Hermann also recommended that the term “Surname” or “Last Name” be used in place of “Second Name” as this was ambiguous and may cause users problems if English is not there first language. Dr. Hermann noted that in English language schools the term “surname” is often taught and not “second name”. This change has since been implemented.
Figure 5.5: The old and new sign up forms compared

Note: The original sign up form is poorly laid out and not spaced correctly. The new and more spread out version is clearer.

The sixth point Dr. Hermann noted was that the date selection box was too small and she also said she found it confusing to use (see figure 5.7). This is a date selection box from the HTML language and the author intends to find or design a better more usable date-selection box for the web app. This change has been taken on board but has not yet been implemented.
The date-selection box has a drop down arrow, which opens a calendar where the user can pick a date.

The seventh and final recommendation from Dr. Hermann was one that the author had admittedly not thought of before. Dr. Hermann pointed out that at present the user could only filter the available bookings by date and not pitch number. From this the author has made future plans to allow for users to search by, not just date, but pitch, time, date and pitch, date and time, and pitch and time. This would reduce the time spent searching through the available bookings list and would increase the usability of the system. This could have implications when it comes to the author’s plans to implement the available bookings table shown in figure 3.1.

5.5 Project Successes

There were multiple successes throughout this project, however the author is most proud of completing the ‘Make a Booking’ functionality. This was after all, the core functionality of the Easy Booking web app. As mentioned previously in chapter four, before the make a booking function could be implemented, the displaying available bookings function would have to be developed. This was by far the most difficult function of the entire web app.
Overall this project can be considered a success. The five core requirements that were outlined in chapter three were all achieved, and the web app was also available to be opened in a mobile web browser (see appendix 9.1). Along with this the author achieved his personal goals that he outlined in the beginning, which were largely based on the learning that he hoped to obtain from developing this project.

5.6 Difficulties Encountered

Just like any software project there difficulties that were encountered along the way. One of these difficulties that were mentioned above is that of displaying the available times in an easily readable and usable manner so that the user could absorb the information and make a booking that suits them. Thankfully this challenge was overcome. Largely due to trial and error but also due to the aforementioned redesigning of the database. The author realised that the current database structure was correctly tailored to the systems needs. After creating a new table named ‘availables’ which contained 36 available slots (12 hour slots per football pitch), this problem was quickly overcome.

Another significant difficulty that was overcome was when the author decided to cease developing Easy Booking as a PhoneGap mobile app. This decision was not taken lightly due to the late stage that it was made at. The author was struggling to pass data from the database through to the web app and vice versa. It was at this time that the author realised that PhoneGap was causing this problem, as it is not suited to the programming language PHP. PHP however, was a requirement as it was an integral part of the local server MAMP (Mac, Apache, MySQL, PHP) and therefore could not be left out of the project. The author also realised that his reason for choosing PhoneGap was not a technical one but rather, a personal choice. After speaking with the head of an Irish software company and web hosting company, Wupav Hosting, the author realised he should be building a web application. By building a web application the author was automatically increasing the accessibility of the app, as is discussed in chapter two.
5.7 Chapter Conclusion

This chapter looked at the different methods the author used to evaluate the project including user testing, the SUS survey, and testing with Dr. Sonja Hermann. The seven suggestions made by Dr. Hermann and whether they have since been implemented, were carefully discussed in this chapter.

This chapter also evaluated the success of the project and the difficulties that were encountered over the six months, and how the author dealt with these difficulties.
6 Chapter 6 – Future Work

6.1 Chapter Introduction

This chapter will discuss the future work that the author plans to complete for this project.

6.2 Future Work & Features to be implemented

As discussed in chapter two there were certain core requirements that the author outlined in the beginning of this project. Thankfully these five core requirements were implemented in time for the demonstration on April 1st. However, along the way there were other functionalities that the author would have liked to implement but could not do so due to time restrictions. These functionalities will be discussed in this section.

6.2.1 Stripe API for Payments

When a user goes to book a football pitch in reception they have to pay €16.00 to the sports centre. If Easy Booking were to move Trinity's booking system online then an online payment system would need to be included. The developer looked at both PayPal and Stripe under various key areas such as security, the current API, and data portability.

6.2.1.1 Security

When it comes to security, both companies take security seriously. However, one key difference between the two is how Stripe intrinsically encourages good security. Stripe use Stripe.js, which means that when a customer enters their credit card details into the payment form on your site the details don’t get sent to or dealt with on your server at all. The details are sent directly to Stripe. This means that you never have to handle the customers’ sensitive credit card details and are therefore, automatically PCI compliant.
Another reason this is more secure is because a breach of your servers won’t result in any stolen credit card data (Memberful, 2014).

In contrast to Stripe, PayPal will either store the credit card details on your server on they will store them in their vault but pass them through your server beforehand. This is a big security burden, and it is for this reason the developer favours Stipe’s system design (Memberful, 2014).

6.2.1.2 Current API

The Stripe API is clean, easy and well documented. PayPal’s API has become significantly better in recent years since they modeled their API documentation on Stripe’s. Having said that, PayPal is rushing to catch up and currently it is Stripe that is leading the API race.

6.2.1.3 Data Portability

This is a key difference between the two competing companies. PayPal has a policy of not allowing you to take your customers’ data when you leave them. This is their way of forcing you to stay. If you were to switch from PayPal to Stripe all your customers would have to give their credit card details again.

In stark contrast to this, Stripe will actually help you migrate your credit card data in a secure and PCI-compliant way. This is because Stripe values data portability. If a company chooses to use Stripe instead of PayPal they won’t be locked in forever and can easily migrate their credit card details if they choose to do so (Memberful, 2014).

6.2.1.4 Decision to use Stripe

Ultimately, after looking at the aforementioned key areas it was decided that Stripe was the better option for Easy Booking.

6.2.2 Teams

Another feature the developer would like to implement is the option for members to join ‘teams’ and book a pitch/court as a team (or group). This could be done when the user initially signs up to Easy Booking, or the user could decide to ‘join a team’ at a later stage.
6.2.3 PAT – Pay As Teams

After implementing payments and teams the developer would like to allow users to pay for bookings as a team rather than leaving just one individual to pay. The inspiration for this feature comes from the developers experience in booking and paying for football pitches in Trinity. The price for one booking is €16.00 however this is usually paid for by one person. Although ten people could bring in €1.60 each this rarely happens and as a result certain people are often left with the burden of paying more than others.

The PAT functionality would be an option where a booking is made and each member who agrees to play (and use the booking) will pay his or her share of the price. This will increase the usability, a non-functional requirement of this project, of Easy Booking and could also lead to more bookings because the payment is being spread over more users rather than burdening the same people each week.

6.2.4 Promotions via Easy Booking

The sports centre could post promotions to their page and, using a RSS feed, Easy Booking could publish these promotions on the TCD page. These promotions could be special offers on pitch or court bookings that aimed at times when there are fewer bookings expected than normal.

6.2.5 Sales Forecasting & Smoothing

Easy Booking could keep track of who is booking, when and where. This data could be analysed to create sales or booking forecasts. Customers could be grouped into different segments based on their demographics. These different segments could be targeted with different promotions and special offers at different times of the year. These offers would hopefully create some sales or booking smoothing, whereby the user traffic is spread out more evenly rather than being concentrated on a few key points in the year. For example, it is
common knowledge that October is a busy month for bookings in college because people are back at college and want to see their friends and catch up. In October, the number of football, squash and tennis booking are higher than December. In December, the weather is colder, assignments are due and the students break for Christmas. These factors lead to a significant reduction in bookings. Using sales forecasting and smoothing techniques Easy Booking would aim to increase the bookings in November and December and create a steadier stream of user bookings throughout the year.

6.2.6 Increase User Interaction & Communication

The author would also like to increase the interaction between the user and the TCD Sports System by enabling automatic communications to be sent out to the user.

6.2.6.1 Email Confirmation

The system would email the user after each booking is made and after a booking is deleted. This email would be a confirmation email to ensure the author has details of the booking or deleted booking off the system. This feature would be implemented using the PHP mail() function.

6.2.6.2 Text Message Reminders

This feature would send a text message when their booking is in 24 hours as a reminder. Currently, it is expected that Twilio will be used for this feature (www.twilio.com). Twilio is an API that allows developers to programmatically send, receive and track SMS text worldwide in web applications.

6.2.7 Booking More than just Sports Facilities

Towards the end of the development of the project the author realised that Easy Booking could be used as more than just a sports facilities booking system. It could be used to make bookings for all sorts of facilities not only on campus but in other organisations as well. For example, in Trinity there are computer labs and group rooms that can be booked or reserved by groups who
are working on an assignment or a project, or even just for studying in. Easy Booking could be developed further to incorporate these types of facilities as well as the sports facilities on campus. This could also be used to broaden the potential customer base, as some organisations might want to allow users to book more than just a football pitch.

6.2.8 Redesign of the Front-end

The author would also like to redesign the front-end in order to produce a more aesthetically pleasing and visually superior user interface. The suggestions and comments made by Dr. Sonja Hermann will be taken into account here along with the comments recorded during user testing.

Easy Booking can be opened in a mobile browser. In appendix 9.1 there are screen shots of Easy Booking being opened in a mobile web browser using an iPhone 5. For this to happen the code was put onto an online server and not just the MAMP local server used for testing. This was done using Wupav Hosting who kindly gave the author an web address where he could view and use the web application. There navigation bar is not finished being scaled for mobile phones. This is something the author plans to address in the near future.

6.2.8.1 Twitter Bootstrap

In 2011, a small group of Twitter developers released the first version of Bootstrap, a free and open-source front-end framework for creating websites and web applications. Today, version 3.3.4 is available and on its website it describes itself as being “the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web” (Bootstrap, 2015). With this in mind, the developer plans to use Bootstrap in the redesigning of the front-end.

6.2.8.2 Redesign of Displaying Available Time Slots Table

In chapter two the developer’s ideal available bookings display was discussed (see figure 3.1). The developer plans to work towards making this
display a reality and not just a mock-up. The developer is confident that with a little more SQL and PHP knowledge this table view can be displayed.

### 6.2.8.3 Design System to Handle Simultaneous Bookings

The problem did arise of how the system would handle two users who are logged in on different computers and are trying to book the same slot. One method to do this is to use a transactional database so that there are no clashes. With this the system could use a tentative allocation of the seat that expires after some length of time (e.g., 10 minutes). This gives the user enough time to pay for their booking. If the transaction falls through, or times out, the seat allocation can be released back into the pool of available bookings. Airlines use a similar system for booking seats online, although their system is much more complex than this.

The author plans to change the database in order to solve this problem. Before this would be carried out, the author plans to meet with members of Trinity’s management team to discuss the possibility of this system being taken further.

### 6.3 Chapter Conclusion

This chapter discussed some of the main functions the author wished to implement with this project in the near future.

---

2 A transactional database is a database management system, where transactions on the database can be rolled back if they are not completed properly.
7 Business Opportunities

7.1 Chapter Introduction

This chapter will look at the business opportunities for Easy Booking going forward and why The Lean Canvas Business Model was chosen for this project. The Lean Canvas that was created for Easy Booking can be seen in appendix.

7.2 The Business Model Canvas

Alex Osterwalder created The Business Model Canvas as a strategic management tool for entrepreneurs and businesses around the world. The business model describes “the rationale of how an organisation creates, delivers, and captures value” (Osterwalder, 2004). Osterwalder created a common language that enables the business model to be shared and understood by all individuals and organisations. This common language contains nine building blocks that incorporate the four main areas of business: customers, offers, infrastructure, and financial viability. The Business Model resembles a blueprint for strategy to be implemented through organisational structures, processes and systems (Osterwalder, 2004).

The nine building blocks:

1. Customer Segments
2. Value Proposition
3. Channels
4. Customer Relationships
5. Revenue Streams
6. Key Resources
7. Key Activities
8. Key Partnerships
9. Cost Structure

This business model is more suited to new businesses. Having said that, it is not a particularly good fit for technology projects and start-ups. That is why
the author has chosen the Lean Canvas Business Model that was created by Ash Maurya in 2010 to be suited for both current and new businesses, especially web applications (Maurya, 2010).

### 7.3 The Lean Canvas Business Model

The most important change to Maurya’s version of the Business Model Canvas is the additional division between those building blocks that affect the product and those that affect the market (see figure 7.1).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
<th>Unique Value Proposition</th>
<th>Unfair Advantage</th>
<th>Customer Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 3 problems</td>
<td>Top 3 features</td>
<td>Single, clear, compelling message that states why you are different and worth paying attention</td>
<td>Can’t be easily copied or bought</td>
<td>Target customers</td>
</tr>
<tr>
<td>Key Metrics</td>
<td>Key activities you measure</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cost Structure</th>
<th>Revenue Streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Acquisition Costs</td>
<td>Revenue Model</td>
</tr>
<tr>
<td>Distribution Costs</td>
<td>Life Time Value</td>
</tr>
<tr>
<td>Hosting</td>
<td>Revenue</td>
</tr>
<tr>
<td>People, etc.</td>
<td>Gross Margin</td>
</tr>
</tbody>
</table>

**Figure 7.1 Ash Maurya’s Lean Canvas Business Model**

The following sub-sections in this chapter will go through the lean canvas business model for Easy Booking (see appendix 9.3) and will explain each of the nine headings for the web application.
7.3.1 **Problem/Solution**

The main problem that the developer aimed to solve when creating Easy Booking was to make it easier for users to make a booking in the Trinity Sports Centre. Along with this was the problem of time wasting for both the user and the receptionist, as the current system was inefficient and tedious. Easy Booking aims to solve these problems by moving the current booking system online, rather than making the user walk in, phone, or email to make a booking.

7.3.2 **Customer Segments**

As Easy Booking is a B2B (business to business) web application the customers would be organisations that require an online booking system. Their members however, would be our targeted users but not our customers, as they would not be paying us for the service.

7.3.3 **Unique Value Proposition**

The unique value of Easy Booking would be the ease with which a user could log on make a booking and eventually pay for their booking online. This type of system has not been implemented in Trinity College Dublin to date. If it were to work in Trinity then the aim would be to look at implementing Easy Booking in other university campuses in Dublin and Ireland.

7.3.4 **Key Metrics**

The key metrics here will relate to tracking the key activities that users can engage in on the Easy Booking web app. The key activities will include bookings made and bookings deleted. The aim is to track the number of bookings made each week or month, and compare these to previous weeks and months in order to give the organisation, such as Trinity, an idea of their performance. The metric ‘percentage of total utilisation’ will also be used as an indication of how many bookings are being made compared to the maximum (full capacity).

7.3.5 **Channels**

The channels that will be used to reach the users will be online using the organisation’s website e.g. [www.tcd.ie/sport](http://www.tcd.ie/sport). And the channels that will be
used to reach other organisations outside of Trinity that might be interested in implementing Easy Booking for their campus, would be done by email, phone call and face-to-face meetings.

7.3.6 Cost Structure

The cost structure for Easy Booking would be a mix of the server space required and the engineers needed for repairs, maintenance and future development.

7.3.7 Revenue Streams

The revenue streams for Easy Booking would be split in two. The larger organisations would be on a contract-based payment. The smaller organisations would be on a monthly subscription.

7.3.8 Unfair Advantage

Jason Cohen of the blog ‘A Smart bear’ describes unfair advantage as “something that cannot be copied or bought” (Cohen, 2010). If Easy Booking were to be implemented in Trinity College Dublin, the unfair advantage Easy booking has over the competition is that it would be a first mover. The developer of Easy Booking saw a problem in something he experienced every week and went about fixing it. This is a problem that the competition most likely has not come across yet and this is how Easy Booking has first mover status.

7.4 Chapter Conclusion

This chapter described the elements of the Lean Canvas Business Model for Easy Booking.
8 Conclusion

Overall the project was a success. The five main requirements that were outlined in the requirements document discussed in chapter two were achieved. This included the two core functions of the Easy Booking web app, which are displaying the available timeslots in an easily and absorbable way, and making a booking online.

In hindsight, the author would do things differently. For example, the author started this project with very little software project management experience and this affected his planning of this project. If the author were to start again he would aim to have a smaller list of requirements and focus purely on developing a simple and basic prototype before elaborating and extending on this prototype at a later stage. The author started by developing a database that would accommodate not only the core functions but also the payments and teams functions. In hindsight this was a poor decision as it took up a significant amount of time, and added no real functionality to the current beta version.

The author also considers the project a success, as he was able to achieve his personal goal of gaining a sufficient understanding of the technologies and multiple languages that are required to make a web application. There was a steep learning curve involved, however in the end it paid off as the web app is functioning appropriately.
9 Appendices

9.1 Screenshots of Easy Booking in a mobile browser
9.2 The Ethics Approval Documents

TRINITY COLLEGE DUBLIN
INFORMED CONSENT FORM

LEAD RESEARCHER: Chris Redahan

LEAD RESEARCHER BACKGROUND OF RESEARCH: My degree is in Business and Computing (TR082), class of 2015. The relevance of this research is to help me in finalising the design and development stages of my app for my final year project. My final year project is a web app that will allow the user to book various sports facilities online.

PROCEDURES OF THIS STUDY: I will ask you to either (1) use my app, unguided, and give an opinion at the end, or (2) use the app while performing certain tasks that will be asked my myself. These tasks will include clicking around on the web application and testing various functionalities, such as, making and deleting a booking, signing up, login and logout.

During the study I may ask you some questions about how you found the usability of the app. Examples of these questions include, ‘did you find any pages took an abnormal amount of time to load?’ and ‘did you have any problems navigating your way around the web app?’.

Answering these questions is completely voluntary. You can cease participation in the study at any time you wish, without penalty. The duration will be between 10 and 15 minutes. There are no risks to you as the participant.

PUBLICATION: Results will not include any participant personal information. Some direct quotes may be used, only with your consent. The final project will be submitted for marking.

Individual results may be aggregated anonymously and research reported on aggregate results.

DECLARATION:

• I am 18 years or older and am competent to provide consent.
• I have read, or had read to me, a document providing information about this research and this consent form. I have had the opportunity to ask questions and all my questions have been answered to my satisfaction and understand the description of the research that is being provided to me.
• I agree that my data is used for scientific purposes and I have no objection that my data is published in scientific publications in a way that does not reveal my identity.
• I understand that if I make illicit activities known, these will be reported to appropriate authorities.
• I freely and voluntarily agree to be part of this research study, though without prejudice to my legal and ethical rights.
• I understand that I may refuse to answer any question and that I may withdraw at any time without penalty.
• I understand that my participation is fully anonymous and that no personal details about me will be recorded.
• If you, or anyone in your family, have epilepsy then you are proceeding at your own risk.
• I have received a copy of this agreement.

PARTICIPANT’S NAME:

PARTICIPANT’S SIGNATURE:

Date:

Statement of Investigator’s responsibility: I have explained the nature and purpose of this research study, the procedures to be undertaken and any risks that may be involved. I have offered to answer any questions and fully answered such questions. I believe that the participant understands my explanation and has freely given informed consent.

RESEARCHER’S CONTACT DETAILS: redahanc@tcd.ie

INVESTIGATOR’S SIGNATURE:

Date: 6th March 2015

SCSS Research Ethics Application Form August 2014
TRINITY COLLEGE DUBLIN

INFORMATION SHEET FOR PROSPECTIVE PARTICIPANTS

The relevance of this research is to help me in finalising the design and development stages of my app for my final year project. My degree is in Business and Computing (TR082). For some participants in this testing, I will require you to use my app, unguided, and to give me your opinion at the end. For others, I will require you to use the app while performing certain tasks that will be specified by myself.

My objective is to have some of my friends, family and classmates test my app in order to give my feedback which I can then use to improve my project. I plan on having approximately five to ten participants take part in my research and testing. As I am taking advantage of existing relationships to further my research, possible conflicts of interest may arise. Due to these existing relationships, I intend to start every session by declaring that the research and testing are being carried out on a completely voluntary basis and that every question is optional to answer. All opinions are welcome, negative or positive. You have the right to withdraw from the testing or refuse to answer certain questions at all stages of testing, without penalty.

I plan on using the System Usability Scale (SUS) for testing usability. This is a reliable usability scale that helps test a system's usability. (http://www.usability.gov/how-to-and-tools/resources/templates/system-usability-scale-sus.html) Questions here will be asked by myself, and notes will be taken based on your answers.

Each question is optional. You can feel free to not respond to any question without penalty. However, I would be grateful if all questions were responded to. The duration will be between 10 and 15 minutes. In the extremely unlikely event that illicit activity is reported I will be obliged to report it to appropriate authorities.

There are no anticipated risks that will occur to you while you are completing testing. Your answers and any/all observations made will be completely confidential. Results will not include any participant personal information. Some direct quotes may be used, only with consent. Participants of my testing will be made aware of multiple facts when participating in my testing. The following will be how the participants were selected:

- can be male or female
- must be over the age of 18
- can have experience with iOS apps or not
- can have experience booking sports systems (e.g. football pitches) or not
- can have extensive experience of using a smart phone or not
- The participants have received a copy of this agreement.
TRINITY COLLEGE DUBLIN

Research Project Proposal

Project Title: An app for finding and booking available sports pitches in a University setting.

Purpose of Project: The purpose of this project is to complete my degree. My degree is in Business and Computing (TR082). The relevance of this research is to help me in finalising the design and development stages of my app for my final year project.

Methods and Measurements to be used:
I plan on using the System Usability Scale (SUS) for testing usability. The SUS is a reliable usability scale to test a systems usability.

The SUS gives a scale from one to five with one being strongly disagree, and five being strongly agree. This will give participants a scale to consider which will hopefully make their thinking more clear.

The questions I ask during the interview will comprise of both opened questions and more direct questions. All questions will be asked by myself, and I will take appropriate notes on the responses. For example, some questions may be: Do you find this system easy to navigate? Is the system aesthetically appealing to you as a user? Could you find booking section for your sport with relative ease? On a scale of one to five how likely would you be to use this app were it to be implemented in the TCD Sports Centre?

Participants:
My objective is to have some of my friends, family and classmates test my app in order to give my feedback which I can then use to improve my project. I plan on having approximately five to ten participants take part in my research and testing. As I am taking advantage of existing relationships to further my research, possible conflicts of interest may arise. Due to these existing relationships, I intend to start every session by declaring that the research and testing are being carried out on a completely voluntary basis and that every question is optional to answer. All opinions are welcome, negative or positive. You have the right to withdraw from the testing or refuse to answer certain questions at all stages of testing, without penalty.

I intend to recruit the participants by informally asking them if they are willing to take part in my research and test my app. For some participants I will want them to use my app and give their opinion, I may ask some open ended questions when seeking their opinion but this will only be to facilitate them to comment on the app and it will not be arranged for me to get the feedback that I would like to hear. Other participants, I will have specific questions and tasks for them to complete. As mentioned, I plan on having approximately five to ten participants. These will be picked at random from my pool of friends, family and classmates.

All data collected will be stored on my personal laptop. All data will be kept for the duration of this project and then handed over to my supervisor, Mary Sharp, to comply with the colleges policy. All collection and storage will comply with the Data Protection Act.

None of my participants will be under the age of 18. They can be male or female.

Debriefing Arrangements:
I will make it clear to all participants after taking part in research what it is their results will be helping with. I will also make it clear that they will not be publicly named in any documents. Made clear will also be the fact that I will not be keeping any data after the completion of this project.

Ethical Considerations:
The anonymity of participants is the most important factor at play here. I will protect participants data and delete said data when I have completed my project.

System Usability Scale

SCSS Research Ethics Application Form August 2014
1. I think that I would like to use this system frequently  

2. I found the system unnecessarily complex  

3. I thought the system was easy to use  

4. I think that I would need the support of a technical person to be able to use this system  

5. I found the various functions in this system were well integrated  

6. I thought there was too much inconsistency in this system  

7. I would imagine that most people would learn to use this system very quickly  

8. I found the system very cumbersome to use  

9. I felt very confident using the system  

10. I needed to learn a lot of things before I could get going with this system
9.3 The Lean Canvas Business Model

Easy Booking - Users

<table>
<thead>
<tr>
<th>PROBLEM</th>
<th>SOLUTION</th>
<th>UNIQUE VALUE PROPOSITION</th>
<th>UNFAIR ADVANTAGE</th>
<th>CUSTOMER SEGMENTS</th>
<th>CHANNELS</th>
<th>EARLY ADOPTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Five Problems:</td>
<td>1. Clearly display the pitches and facilities that are available and what times they are available</td>
<td>Easy Booking is an online booking system designed for a University campus, such as Trinity College Dublin, that will allow users to book sports facilities online rather than making them walk in to reception or start an arduous phone or email dialogue</td>
<td>If Easy Booking was to be implemented by Trinity College Dublin, it's unique advantage would be it's early mover status. Being ahead of the competition by getting the users and organisation used to the Easy Booking system would create a barrier to entry for other similar software providers</td>
<td>Trinity College Dublin's Sports Centre Members</td>
<td>Online using:</td>
<td>- Sports enthusiasts</td>
</tr>
<tr>
<td>1. Not being able to see for themselves what pitches and facilities are available at what dates and times</td>
<td>2. All bookings can be made online on a laptop, desktop, tablet or mobile phone</td>
<td></td>
<td>- Students</td>
<td>- tcd.ie/sport</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Having to either phone, email or walk into reception to make a booking</td>
<td>3. The user and their friends/colleagues can all log in and view the available times and agree on a time before booking</td>
<td>Promote on Social Media:</td>
<td>- Staff</td>
<td>- facebook</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. When the first requested time is not available the customer must then retreat to discuss a potential new time with their friends. This can result in bookings being skipped</td>
<td>4. Receptionists' time is being spent on negotiating these bookings</td>
<td>- twitter</td>
<td>- Public Members</td>
<td>Inform all current members, students and staff of the new system via email</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Receptionists' time is being spent on negotiating these bookings</td>
<td>5. Increase the usage of these sports facilities on campus and thus increase the revenue for the University</td>
<td></td>
<td>- Sports Clubs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. The sports facilities are not being fully utilised as revenue generating assets due to skipped bookings</td>
<td>KEY METRICS</td>
<td></td>
<td>- Societies</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXISTING ALTERNATIVES</td>
<td>In the beginning:</td>
<td>If you are interested in becoming an early adopter of Easy Booking, please contact us.</td>
<td>After this, Easy Booking could be quite easily implemented in other Universities both inside and outside of Ireland that have not yet implemented such a system</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In Trinity, to make a booking customers must either walk into reception or start a dialogue using email or one or more phone calls</td>
<td>- track the number of bookings each week/month/quarter and compare them to the equivalent week/month/quarter in previous years</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Later, we will be comparing against our own figures, as well as using the metric ‘% of max utilisation’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This metric will compare the facilities current usage against facilities being fully booked all the time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HIGH-LEVEL CONCEPT

Easy Booking is the Google Calendar for Sports Facility online bookings

COST STRUCTURE

Fixed Costs:
- Server space (depends on what the organisation already has)
- Software engineer's fee for system testing and updates

Variable Costs:
- Unforeseeable problems and system failures that require additional human resources to work on the system
- Initial software development costs

REVENUE STREAMS

The university will pay an initial installation fee, and after this there will be a monthly fee for server space and system checks by a software engineer

Lean Canvas is adapted from The Business Model Canvas (BusinessModelCanvas.com) and is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported License.
9.4 Database Entity Relationship Diagram

booking_system_db
Entity Relationship Diagram

Users
- user_id
- student_number
- first_name
- second_name
- email_address
- password

Availables
- slot_id
- start_time
- finish_time
- pitch_id

Memberships
- membership_id
- team_id
- user_id

Payments
- payment_id
- amount

Pitches
- pitch_id
- sport
- location

Teams
- team_id
- team_name

Bookings
- booking_id
- user_id
- team_id
- time_slot
- payment_id
- date
- pitch_id
10 References


