BEN LETTERY CONNEMARA HOSTEL
Development of a Management Information System

24th March 2014
I declare that the work described in this dissertation has been carried out in full compliance with the ethical research requirements of the School of Computer Science and Statistics.

Signed: ____________________________

Sarah Keane
24/03/2014
ABSTRACT

The aim of this project was to design and develop a management information system with a web interface for Ben Lettery Hostel based in Connemara. The management information system serves to provide a system for the use of staff in the hostel. The system was created in a clear and logical way maximising ease of use for the user. The system allows users to check bed availability, book in a guest, update a guest’s details, update a booking, cancel a booking and generate reports. The system consists of a technical side composing a web front end, developed using a combination of PHP, CSS, HTML, JavaScript and MySQL.
PREFACE

The client for this project is part of a voluntary group that run Ben Lettery Hostel. The hostel has recently reopened and the client, Joseph Vale is part of the group that made this happen. Joseph is the main client contact throughout the project, hereafter referred to as ‘the client’. The project was successful in surpassing the terms of reference and the client has expressed fulfilment with the system delivered.

The purpose of this project was to develop a management information system with a web interface to assist with organisation within the hostel. This involved providing a password protected system in which staff members can log-in. During the course of the project I developed proficiency in a number of software languages which deemed more challenging to grasp then was expected. This required learning how to develop high level systems using PHP and MySQL to a high standard in a very short time frame. I furthered my knowledge of software development and software engineering far beyond what I predicted at the outset. Five languages were used to build the application.

Although the user interface has not been put on the internet, it has been demonstrated to the client who has provided positive feedback. The next phase for the client will be to put the system online. This will be discussed further in the report.

I would like to give a special thanks to the client, Joseph Vale. Without his help and co-operation the project would not have been a success. He gave me as much of his time that was needed and his assistance guided me in the right direction.

Finally I would like to express my sincere gratitude to my project supervisor, Mr. Eamonn Mullins. His support and feedback throughout the project greatly contributed to its completion.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>NO.</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>INTRODUCTION AND SUMMARY</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>The Client</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Project Background</td>
<td>1</td>
</tr>
<tr>
<td>1.3</td>
<td>Terms of Reference</td>
<td>1</td>
</tr>
<tr>
<td>1.4</td>
<td>Achievement of Objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.5</td>
<td>Summary of remaining chapters</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>SYSTEM OVERVIEW</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Objectives of the System</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>Technical Environment</td>
<td>3</td>
</tr>
<tr>
<td>2.3</td>
<td>System Overview Diagram</td>
<td>6</td>
</tr>
<tr>
<td>3.</td>
<td>DESCRIPTION OF WORK DONE</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Requirements Phase</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Design Phase</td>
<td>10</td>
</tr>
<tr>
<td>3.3</td>
<td>Development Phase</td>
<td>12</td>
</tr>
<tr>
<td>3.4</td>
<td>Testing</td>
<td>16</td>
</tr>
<tr>
<td>3.4</td>
<td>Points Regarding Further Attention</td>
<td>17</td>
</tr>
<tr>
<td>4.</td>
<td>CONCLUSIONS AND RECOMMENDATIONS</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>Conclusions</td>
<td>19</td>
</tr>
<tr>
<td>4.2</td>
<td>Recommendations</td>
<td>20</td>
</tr>
</tbody>
</table>
# APPENDICES

<table>
<thead>
<tr>
<th>NO.</th>
<th>CONTENT</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Original Project Outline</td>
<td>A.1</td>
</tr>
<tr>
<td>B</td>
<td>Interim Report</td>
<td>B.1</td>
</tr>
<tr>
<td>C</td>
<td>Technical Report</td>
<td>C.1</td>
</tr>
<tr>
<td>D</td>
<td>Database Tables and Relationships</td>
<td>D.1</td>
</tr>
<tr>
<td>E</td>
<td>User Manuals</td>
<td>E.1</td>
</tr>
<tr>
<td>F</td>
<td>Testing Documentation</td>
<td>F.1</td>
</tr>
<tr>
<td>G</td>
<td>Sample Pages</td>
<td>G.1</td>
</tr>
<tr>
<td>H</td>
<td>Source Code</td>
<td>H.1</td>
</tr>
</tbody>
</table>

References
INTRODUCTION AND SUMMARY

The purpose of this chapter is to introduce the client, provide the project background and the terms of reference that have been agreed with the client. A summary of the remaining chapters is also included.

1.1. The Client

The client is Joseph Vale a volunteer at Ben Lettery Hostel. This hostel is one of several hostels around Ireland the client helps run. Ben Lettery Connemara Hostel is a unique Irish hostel, which first opened its doors on the slopes of the Twelve Bens in 1969. Unfortunately the hostel had to close its doors in 2006 but reopened in June 2013 with new community based volunteer management. Cyclists, hill walkers and backpackers favour the hostel’s idyllic location. The hostel is a hill walking landmark; known as the start or finish of one of Ireland’s most exhilarating hikes, the Glencoaghan Horseshoe, which crosses six of the Twelve Bens.

1.2. Project Background

In 2012 an eclectic group of volunteers came together to renovate and reopen Ben Lettery Hostel. The group ranged in age from students to active retirees, from architects to carpenters, and from hill walkers to artists. The hostel reopened its doors on June 1st 2013. The hostel has become a teahouse for some hill walking clubs, a base of operations for Galway Mountain Rescue, and a community resource.

The hostel manager is in need of a hostel management application to manage guest bookings, assign beds to guests and identify available beds. The purpose of the project is to develop a management information system which will record the availability of rooms, will assign guests to beds, will record arrival and departure dates. It will also record guest names, email addresses, source of booking, age and nationality. It will allow the user to update or cancel a reservation or a guests personal details. A simple interface assuming low user technical ability is required. The client would value a heat map of bookings and availability on any night in the future.

The hostel manager should have the ability to run reports such as guest profile by age group, nationality or source of booking. There are several hostels currently managed by the same volunteer group, which could be supported by the management information system in the future. User logins and permissions should be considered.

It is important that the management information system will be intuitive, easy to use and the client’s requirements are identified correctly.

1.3. Terms of Reference

The following are the final terms of reference for the project agreed with the client in December 2013.

- To develop a user-friendly web – based management information system for the use of hostel staff to make bookings.
- To ensure the website is simple to navigate using heat maps for bookings and availability.
• The system should have the ability to run reports such as guest profile by age group, nationality or source of booking.
• User access should be password projected.
• Review and recommend open source database and web development technologies.
• Review and recommend web hosting companies.

1.4. **Achievement of Objectives**

This project satisfies the terms of reference agreed with the client in December 2013. All the above requirements have been met and management information system allows the user to book in guests easily, run reports and check availability. It will save the user time, avoid confusion and allow them to track the bookings easily.

1.5. **Summary of remaining chapters**

• **Chapter 2: System Overview** provides an overview of the system including system objectives, the technical environment and a system overview diagram. It also shows how the management information system is structured using Entity Relationship and Data Flow Diagrams.
• **Chapter 3: Description of Work done** describes the work undertaken throughout the project. It will discuss requirements, design, implementation, testing, points regarding further attention. Instructing the client of the use of the management information system, and the maintenance and future of the system.
• **Chapter 4: Conclusions and Recommendations** contains conclusions reached and recommendations to the client for proper maintenance and future development of the system.
2. SYSTEM OVERVIEW

This chapter provides an overview of the system. It will begin by explaining the purpose of the system, the objective of the system, the technical environment, overview of the system and implementation.

2.1. Objectives of the System

The purpose of the system is closely linked to the terms of reference. This system is viewed as a management information system which has the main purpose of booking guests into the hostel. The system will provide a clear and precise layout to allow users to check availability and make a reservation, update a booking and cancel a booking. The current system involves taking bookings for the hostel with a pen and paper. The hostel is located on several hostel websites including Hostel World, Hostel International, Hostel Bookers and An Oige. When someone makes a booking on one of these websites it is then necessary for the booking company to ring Ben Lettery and make the booking. Details are taken and the requested dates are investigated by the user to check the availability in the hostel. This can take time and be the cause of confusion for the staff in the hostel. Especially if guests wish to change the dates of bookings or cancel bookings. It can sometimes result in hostel rooms being left empty or double booked.

The new systems heat maps will allow the staff to see if rooms are available, partially available or full. The system will only allow staff to book guests into rooms that are available. This will prevent any confusion and make the process of booking in guests much faster. It also allows guests bookings to be changed, details to be updated and bookings to be cancelled. As there is a log-in process only staff have access to the system. When a staff member is logged in the system will record which staff member made a booking, this will keep track of all reservations so if there is a problem the manager will know which staff member made the booking.

The client has specified certain reports that he would like included in the system. These include the age profile of guests, how many bookings per staff member, who booked which reservations and nationality profile of guests. The age profile can be seen in graphical format which makes it easier for the user to interpret, an example can be seen in Figure 3.3.2, page 16. Within the staffing report it displays which staff member took a specific booking and then how many booking each member took. All of the reports are based on dates that are entered by the user. Thus it allows the user to compare occupancy for different seasons or compare weekdays with weekends. This knowledge is very important to the client as it can be used to develop a strategy to increase occupancy levels. For example; the report shows spring has the lowest occupancy rates, therefore the client can increase their marketing and offer discounts during this season to their target market to encourage more bookings.

2.2. Technical Environment

Within this section the technical layout of the system will be summarised. The system is made up of a technical side, the bulk of the system is made up of a series of PHP pages which becomes the user interface. The system contains a MySQL database and a web-enabled management tool or application. The server – side components can be seen in Figure 2.2.1.
Figure 2.2.1: Server-side Components (XAMPP)

- Windows is a popular operating system.
- Apache (Apache HTTP server) is a web server.
- MySQL (My Structured Query Language) is the world’s second most widely used open-source relational database management system. MySQL offers great reliability and ease of use. MySQL is ideal for this system.
- PHP (Hypertext Pre-processor) is a server-side website development language. It is a simple programming language for building dynamic web pages.

The application runs locally on a free and open source cross platform web server solution stack package called XAMPP. This can be downloaded at http://www.apachefriends.org/download.html. XAMPP is composed of Apache HTTP Server, MySQL database and interpreters for scripts written in PHP. XAMPP is free and easy to use. Its components are open source. Open source software is built with its source code made publicly available, meaning that its growth and development is a public collaborative process.

The web application uses PHP as the scripting language. PHP was selected after much thought of other competing technologies which for many reasons were less suitable than PHP. One of the reasons for choosing it is due to its ability to connect to the database through SQL. The back end’s main components are PHP and a MySQL database. PHP and SQL are server-side embedded scripting languages, this means they can only be seen and interpreted by the computer that holds the website files, not by the user accessing it. As it is embedded this means that it can be written in the same place as HTML code. PHP takes the user’s input and uses the MySQL database to store it (Welling & Thompson, 2008). The user can view the contents of the database on the site view by running queries in MySQL. It is also possible to input data into the database from the front end.

An advantage of PHP is that its existence is entirely transparent to the end-user. With some websites, you can use the browser to view the document’s source code. You can view all the HTML that was used to create the page. If a PHP page is inspected only the HTML output would be shown. Only results of the PHP processing are sent to the browser, the code generated remains hidden and secure. (Nixon, 2009) Another advantage is that most web hosting providers support PHP for use by their clients.
MySQL is a relational database system that is used to store information. SQL (Structured Query Language) is used for managing data held in MySQL database. SQL statements are used to perform tasks on a database including; insert, delete, update and selecting data from the database. For the scope of this project it is perfect and with past experience using a MySQL database it was the best and most reliable option.

Figure 2.2.2 describes the system. On one end of the system you have a user opening a Web Browser expecting to see a typical HTML Web page. On the other end is the content of the site, which contains several tables in a MySQL database that only knows how to respond to SQL queries. Figure 2.2.2 demonstrates how PHP is the link that speaks both languages (Welling & Thompson, 2008). The PHP engine analyses the script. Within the script is an ‘include file’ with a command to connect to the database and execute a search query. PHP then opens a connection to the MySQL server and executes the query. For example displaying the guest’s details in the ‘Invoice’ page as seen in Appendix G.1.19, page G.8. The MySQL server accepts the request and returns the results the guests’ information to the PHP engine. The PHP engine stops running the script, which will involve formatting the query results in HTML. The HTML is returned to the web server. The user can then view the customer’s details on the browser as the web server has passed it back.

When creating the database the Normalisation Method was used. Normalisation is the process of efficiently organizing the data in a database. The two main features when applying the Normalisation Method are eliminating redundant data (for example when the same data is stored in more than one table) and ensuring data dependencies make sense. This means only storing related data in a table, for example for this project it would not make sense to store the arrival dates in the customers table, instead they are stored in the reservation table. (Anon., n.d.)
The management information system has been created using HTML, PHP pages and forms which are enhanced in some places using JavaScript and CSS (Cascading Style Sheets) (Duckett, 2010). Microsoft Web Expression 4 is the application used to write the code and it is run on a Windows laptop. PhpMyAdmin is a software tool which can handle the administration of a MySQL database. It comes in the XAMPP package. Microsoft Web Expression 4 is the tool that is used for writing the code, this will be discussed further in Section 3.3, page 12.

2.3. **System Overview Diagram**

Figure 2.3.1 displays a System Overview Diagram of the hostel management information system. “System Overview Diagrams represent all external entities that may interact with a system, the diagram pictures the system at the centre with no details of its interior structure, surrounded by all its interacting systems, environments and activities." (Kossiakoff, et al., 2011) The objective of the system overview diagram is to focus attention on the external factors and events that need to be looked at in developing a set of system requirements and constraints. Figure 2.3.1 describes the layout of the system and maps how the user will use it. In the Appendix Figure C.3.1, page C1 is a Use Case Diagram displaying the interactions between the user and the system. The Use Case Diagram and the System Overview Diagram can be viewed together to see how the system works from different perspectives.

The system is password protected. Only staff members will be given passwords. Once a staff member is added to the system after choosing a username and password it is important that it is not forgotten. When a password is created the user must come up with a unique username. If the name is already being used by another staff member the user will not be added to the system. If the credentials are forgotten the user will not be able to log into the system. It is necessary to contact the client or the manager of the hostel in order to be added to the system or if a password is lost or forgotten. Once the user is logged in they will be directed to the tailored homepage where there is a menu bar containing links to several different functions of the system. These functions are as follows:

- Making a Booking
- Cancelling a Booking
- Updating a Booking
- Viewing Reports
- Adding a Staff Member
- Log – Out

Within these functions contain more functions and methods, all of which are ensuring ease of use. For example the 'Home’ page displays a table with a list of guests that are due to arrive today. See Appendix G.1.2, page G.1.
The Data Flow Diagram (DFD) is shown in Appendix C.4.1, page C.6. A data flow diagram is a graphical representation of the “flow” of the data through the system. The DFD shows what sort of data will be input and output from the system including where the data will come from and go to, and where it will be stored.

In the Appendix D.1.1, page D.1 represents the Entity-Relationship Diagram. This is a systematic way of describing and defining the business process. It is a graphical representation of entities and their relationships to each other. A relationship is how the data is shared between entities. Figure D.1.1, page D.1 demonstrates how the tables (entities) are linked together. An example in this case is that when a user logs in they can proceed to make multiple reservations, therefore meaning there is a ‘one to many’ relationship. This will be explained further in Section 3.
Overall this chapter discusses the technical background to the system, a system overview and the objectives of the system. The next chapter will look into the methodology used and describes how the work got done.
3. DESCRIPTION OF WORK DONE

This chapter discusses in detail the work done throughout the project. The system was developed using the Incremental Model. The Incremental Model consists of many stages including requirements, design, development, testing and implementation. To allow modifications to be made to the system it is necessary to use this process. As the client continuously provided feedback over the duration of the project, some stages were repeated. These changes required the use of this iterative process.

![Diagram of Incremental Design Methodology](image)

Figure 3.0.1 – Incremental Design Methodology

3.1 Requirements Phase

On first meeting the client the first thoughts and ideas were discussed. The requirements phase was critical to the design phase as it was where the needs of the client were clearly defined. This was carried out over several meetings with the client concluding in the terms of reference as were described in Section 1.3, page 2. A current technical system does not exist in Ben Lettery Hostel. A notebook is used for booking in guests. This system works until the hostel is 75% full and then making bookings becomes challenging for the staff. If a booking needs to be cancelled it can create problems with guests being double booked or rooms being left empty. As the technical competency of the users is unknown a very simple and easy to use system was required. The requirements specified that the preferred system would involve a user interface with a heat map for the availability of rooms. This means that red would indicate that a room was fully booked, orange that a room was partially booked and green that all beds were available in the room. The client also wanted several reports which will be discussed later.

Some analysis of similar systems was carried out. It was difficult to find comparable systems as many examples allow the guest to make the booking themselves, for example [www.hostelworld.com](http://www.hostelworld.com). As many of these sites were user friendly it was possible to gather and use some ideas in the design phase.
The choice of technology will be summarised here but has been covered in greater detail in the Section 2.2, page 5. Choosing what technology to use was one of the main requirements (See Appendix A, page A1). The software package used was XAMPP. This runs on a Windows Operating System using Apache server, a MySQL database and runs on PHP. As it is free the client can download it on his computer without having to purchase expensive software packages.

The main advantages of PHP are that it works alongside HTML and MySQL databases. It is possible to use PHP right inside your already existing HTML content, this allows the PHP to add to HTML functions and abilities. PHP and MySQL complement each other and work in a way that neither language can do unaccompanied. PHP can collect the data while MySQL can store the information. (Welling & Thompson, 2008)

3.2. Design Phase

This phase requires the planning of how the system will actually work before beginning development of the system. The Incremental Model was used when designing the system which is shown in greater detail in Appendix C, page C1. This ensured that the client was satisfied with the design as the project developed and he could offer input and feedback over the duration of the project. The two main sections in the design phase will be as follows:

- Database Design
- User Interface Design

Database Design

The database was designed using ER (entity-relationship model) diagrams. An ER diagram helps design a database in an efficient way (See Appendix D, page D1). Before creating an ER diagram all the necessary entities were recorded, this includes all types of information that can be used in the system for example in this case, reservation number, customer name, customer ID etc. These data were then divided into separate tables with each entity be assigned to a given table. Primary keys were selected for each table. A Primary Key is a unique key belonging to a table, for example Customer ID is the primary key for the Customers table while Reservation Number is the primary key for the Reservations table(See Appendix D2, page D1). Forming relationships between the tables was the next step, this defined how the data in one table were related to the data in another. In this case customers joined the reservations table as a many to many relationship because a customers can have many reservations and more than one reservation can belong to a customer. Once these methods were followed it was then possible to produce an ER diagram. The data were then normalized which minimizes redundancy and dependency. The advantage of having highly normalized data schema is that information is stored in one place and one place only, reducing the possibility of inconsistent data. (For further explanation see Section 2.2, page 5) The ER diagram is shown in detail in Appendix D1, page D1.

User Interface Design

After several meetings with the client discussing different layout options one was eventually agreed upon. The client suggested the creation story boards which help the designer visualize the layout of the system. These are seen and described in further details in Appendix C.2, page C.2. The system overview diagram (See Section 2.3.1, page 7) shows
how the user will navigate through the system. One of the client requirements was that the system is password protected. On opening any page the user will be directed to the log-in page and must enter a username and password before been given access to the main system. Once the correct details are entered the ‘Home’ page will open. This page displays a menu bar containing many options, including booking a room, cancelling a booking, updating a booking, adding staff member, viewing reports, arrivals due to arrive today and a search customer box.

![Figure 3.2.1 – Screenshot of Home Page](image)

Figure 3.2.1 is a screenshot of the ‘Home’ page. The system is manageable and self-explanatory. A simple layout was designed allowing the users easily navigate through the user interface. The client’s main requirement regarding design was to use heat maps. There were some issues trying to decide on the best design for this page. It was eventually decided that on entering the booking dates, a form will appear containing images representing each room within the hostel. These rooms will be coloured either green, orange or red depending on the availability. A room will only be clickable if there is an available bed. This will then direct the user to the specific room containing the beds. Similar to a room in the ‘Rooms’ page a bed will be green if it is available and red if it is already booked. A screen shot of these features is shown in Figure 3.2.2.
A web template from web expression was used for the basic layout of the website. This included the side menu bar, the header which includes the staff member that is currently logged in.

Due to the large number of web pages, approximately 37, it was necessary to build the system as efficiently as possible. This will save the client time if he needs to change code during the lifetime of the system. One way of creating an efficient system is to ensure the code is written such that there is as little repetition as possible. The code is made reusable throughout the files, and commented consistently to ensure the person in charge of maintenance can understand the code and build on it in the future.

One method of reusing the code is using Server Side Include (SSI). This means that a file can be included as part of your document while still on the server before it is executed and before the data reaches the client (Nixon, 2009). This minimises repetitive code and allows important code like CSS (Cascading Style Sheets) to be included automatically in the file without having to copy and paste the code each time. It is useful when one small change is required, it is only necessary to make the change in the main file and it will make the changes in the whole application. For example if the colour of the background is changed the code will need to be modified in one file and this will make the changes to all the pages that are linked to this. The “include” file is used for the connection which is used to connect the database to the PHP. This file contains the username and password of the database. By using include here it ensures that the username and password are not displayed in each file, this adds security to the system.

3.3. Development Phase

This process involves actually building the systems including:

- Database Development
- Web Interface Development
- Feature Development
Database Development

A key part of my project was the development of a booking system database. After the relational database was designed in the Entity Relationship Diagram (See Appendix D1, page D1) it was transferred into tables in phpMyAdmin. The tables can be seen in Appendix D2, page D1. The most important table in this database is the reservations table. It is the link between the rooms, booking and customers table. The database consists of five tables in total, rooms, reservations, bookings, customers and login.

Web Interface Development

The development of the web interface for the management information system was done using Microsoft Web Expression 4. The program was chosen based on previous experience using it. Joomla! and Dreamweaver were also on the list of options. Joomla! is a content management system, one of its down sides is that it needs constant updating as new versions are released on a regular basis. (Anon., 2006) This may require the code to be slightly updated regularly. It would therefore not be the ideal application for this project. Dreamweaver offers a number of advantages when creating web pages, including the creation of dropdown boxes and lists which automatically connect to the database. The down-side of Dreamweaver is that it is very expensive; it is not provided by the College and is approximately €300 to buy. Microsoft Web Expression 4 was therefore the best option. It is free and can be downloaded at http://www.microsoft.com/en-IE/download/details.aspx?id=36179. This gave me the opportunity to further advance my coding skills as the application does not connect with the database automatically.

Once Web Expression 4 was installed it was then necessary to connect it to the database. Simple forms were created in the user interface allowing the user to enter their name into the database. It took a while to become familiar with the code and technologies. Once the information could be stored in the database and pulled back into the web interface the development could begin on the system. The first step was creating a reservation where the user could choose the details for the booking and enter in the customer details. These details would then go into the reservations, booking and customer tables respectively. Once this was completed the next step was to develop the heat map. This was quite difficult as, in the design, the plan was to be able to click on each room to show the beds; it resulted in having to have a different form for each room. The availability query was then created which involved a vast amount of testing which will be discussed in Section 3.4. On the completion of the availability query, a booking could be made without double booking the same room on any night.

The functions of the system are as follows:

1. **Login:** The first page that will be seen is the log in page. The user cannot pass this page without entering the correct username and password. If the password is forgotten is will be necessary to email the client who’s email will be given at the bottom of the page and a new password will be created.

2. **Home Page:** The home page will contain a list of guests due to arrive today or over the next few days. There are also boundaries that the user can set; they can specify
the dates in which they want to see customer arrivals. It also allows searching for a specific guest who is due to arrive. The query is written in a way that it will only allow viewing bookings that have yet to arrive. Every time a booking is completed the user will be directed back to the home page.

3. **Making a Booking:** This process involves several pages. When ‘Make a Booking’ is clicked it directs the user to a page where the dates for the booking are entered. These dates are used in the availability query. Once these are entered, the heat map appears (See Figure 3.2.2). The user may only click on rooms that are available. The number of beds available in each room is displayed so the user will know before clicking on the room. This then directs the user to the beds page where one or more beds must be selected in order to proceed.

4. **Completing Reservation:** This page allows the user to see the room and beds that have been selected. It gives the user the opportunity to change the dates if they wish. The user must enter the guest details in this section, including guest name, age, email, nationality and source of booking. There is a comment box which allows the user to enter any other details which may be important to the guest, for example ‘Pay on Arrival’. Once the ‘Book’ button is clicked the booking will be entered into the database, the beds booked will be made unavailable, and the customer will be added to the database.

5. **Invoice:** This page gives another overview confirming the booking. There are two options, one is to click ‘Finish’ which will direct the user to the home page and the other is ‘Add More Rooms’ this will be the case if more than one room is required. The user will be brought back to the ‘Search Dates’ page where the dates from the booking will already be entered in the search boxes. This will bring the user through the same format as before, except the guest details will not need to be entered. These are all stored as session variables. Session variables are functions within PHP and allow you to carry a variable from one PHP page to another.

6. **Cancel a Booking:** This page allows the user to cancel a booking. It ensures that the user can view bookings from today onwards, so this will not allow cancelling a booking that has already occurred. Once the name of the guest is picked it will give all their reservations details. The reservation number chosen will then be deleted from the database.

7. **Updating a Booking:** Updating a booking allows the user to update both the customer details and a specific booking. To change a guests details the user must select a guests name and their details will appear, the user can then change any details by simply changing what appears in the text-boxes, the only exception is the Customer ID. The user will be shown the details of each reservation and asked to choose which one they would like to update. This reservation then gets deleted from the database and the user is redirected to the ‘Search Dates’ page to make the booking again.

8. **Add Staff Member:** This allows a new staff member to be added to the system. Only when staff member is logged in can a new staff member be added.

9. **Reset Password:** This page is located within the ‘Add Staff Member’ page. It allows a user to reset their password once some authorised person with knowledge of the Master Password. The Master Password will only be given to certain staff members.
10. Reports: This page allows the user to view reports. There are currently four reports; age of guests, nationality of guests, total guests and staff allocation.

11. Logout: This allows the user to logout of the system.

Feature Development

The development of the system was improved through several visual design and layout techniques. They include:

- Google Charts Technology
- JavaScript
- Filters

Google Charts Technology: Google Chart API is a tool that allows you to create a chart from data and embed it in a web page. Google creates a PNG image of a chart from data and formatting parameters in an HTTP request (Anon., 2010). Google charts have been used in this system to display the distribution of customer ages for a date range specified by the user (See Figure 3.3.1). This was programmed in JavaScript which allows you to change many features in the graph to be changed, thus allowing simple user interpretation.

![Figure 3.3.1 – Age Profile within the Hostel](image)
JavaScript: JavaScript was used throughout the application. It was used for validation checks and for visually improving the user interface. When making the heat map, JavaScript was used to select the colours for the images. It is useful as PHP and HTML variables can be used within the JavaScript functions. It was also used to implement the validation rules. JavaScript is faster to download than other front-end technologies, it permits the improvement of the user interface without the need to be concerned about disturbing download speeds. JavaScript was used to create the slideshow on the home page. It was also used to enable printing the invoice. This ensures that when the user clicks the “Print” button only the page heading and body context get printed, and not the whole web page, in a layout that fits onto A4 sheets. This enhances the visuals of the system. “For business, good user interfaces can lead to benefits such as higher staff productivity, higher staff morale and higher job satisfaction. Computer systems that are easy to use and easy to understand require less training. Bad user interfaces may result in stress and unhappiness among staff leading to reduced productivity and if this case not using the system.” (Kossiakoff, et al., 2011)

Figure 3.3.2 - To the user, the interface is the computer system (Stone, et al., 2005)

Figure 3.3.2 demonstrates this. To the user, the interface is the computer system. As maintaining a user friendly system is so important to this project as it will ensure the users don’t stop using it and go back to the old system.

Filters: Filters have been used throughout the system. Their main use is in the reports section where the user can filter information based on chosen dates. There is also a filter on the ‘Home’ page where the user chooses a guest from the dropdown menu and the guest’s information appears. Filters are a fundamental part of the system; without them too much information would be displayed and the user could get confused. The second example of filtering involves the guests who are due to arrive; once the dates are entered the user can see who is due to arrive within those dates. This filter works in ascending order, so guests due to arrive first will appear first on the list.

3.4. Testing

As the Incremental Model was used, the system was tested throughout the development. At the completion of every stage, the system was tested by entering different values into the user interface and ensuring they were entering the database correctly. Once the whole system was completed several stages of testing were carried out both by myself, a third party and by the client.
This required entering numerous values into the database viewing and editing the data, deleting trials, to ensure that each function was operating as expected. Debugging the code was a manual and demanding procedure. Although Microsoft Web Expression 4 has some debugging functions a lot of manual debugging is still required. PHP has a function called “echo” so it prints the output of queries and PHP statements onto the screen. After all the SQL queries the "or else mysql_error()" function was used. If there is an error in the SQL query an error produced by the server will be displayed on the screen specifying the source of the error. Every query that is in the system was tested in MyPHPAdmin before it was put into the code of the system. Here, it was possible to see the results immediately before entering it into the code. When making a query it is necessary to extract the data in steps. The query should be gradually built up. For example when creating the query for displaying the guests due to arrive on the ‘Home’ page it was done in steps. First select all the guests, followed by joining the customers table to the reservation table, then select the guests that have an arrival date that is equal to today’s date. The query will return values, it is required to check that the values returned match the values that are in the tables of the database. If the values returned by the query match the values in the tables of the database then the query is successfully working and if not it is necessary to make changes to the query.

The client and some classmates got the opportunity to test the system several times and point out any issues that existed. It was helpful getting an independent perspective, as when building the system minor details may be overlooked.

3.5. Points Regarding Further Attention

The next step for the client is regarding the implementation of the management information system. To fully implement the system it will be necessary to make the user interface available on the internet. There are many web hosting companies that support the technologies used in this system. A web hosting service allows individuals and organizations to make their websites accessible to the World Wide Web. The web host will offer space on a server owned or leased for use by clients.

The Web Hosting Companies reviewed were the following:

- Blacknight Solutions
- GoDaddy
- Hosting Ireland

Blacknight Solutions is a 100% Irish based web hosting site. It can be found at [www.blacknight.com](http://www.blacknight.com). It provides a comprehensive range of Microsoft Windows based hosting plans and domain name registration. [www.benlettery.ie](http://www.benlettery.ie) is currently available. For €49.95 annually provides the client with 10GB disk space, 200MB monthly transfer, MySQL and Windows. This package is suitable for a system of this scale.

GoDaddy is an American company run in Arizona. They service approximately 45 million domain names, this figure is constantly increasing. For €3.99 a month the package comes with unlimited bandwidth, 100GB Disk Space and 10 MySQL Databases. This package is more than enough for the system created. The website can be found at [www.godaddy.com](http://www.godaddy.com).
Hosting Ireland has provided domain name registration and hosting services to both small, medium and large business in Ireland for 10 years. For €39.95 per year the client can domain name five websites, 100GB Disk Space and a MySQL database. The website can be found at www.hostingireland.ie.

Overall I would recommend the client to choose Blacknight solutions, having spoken to people with experience using it, it has a good reputation and is an Irish company. It can provide all the services the client would need for a competitive price.
4. CONCLUSIONS AND RECOMMENDATIONS

This section will discuss conclusions reached and provide recommendations to the client on the upkeep or maintenance of the management information system. Overall, this was a challenging and interesting project. The client set out demanding requirements. The technical aspect of developing the management information system was deemed to be the most challenging part of the project. It involved a vast amount of work at the early stages of the project in order to understand the languages that needed to be used. As Microsoft Web Expression 4 was chosen to build the system it meant the whole system was hard coded. This ensured that the skill level developed for the completion of the project was very high, whereas I began as an inexperienced programmer, I turned myself into a relatively high standard programmer over the course of the project. A huge variety of skills and methods were learned along the way. The end result is a fully-developed, custom built, complete booking system. Designing each function of the booking system was puzzling, but the resulting system has exceeded the clients’ requirements and expectations. With the system created the client is now able to progress to the next stage of their project.

4.1. Conclusions

This project involved designing and developing a management information system for a hostel to use to book in and manage guests. Currently no technical system is in place so this will become a fundamental part of managing the hostel. The implementation of this system will become a vital part of the hostel, saving time and effort on the part of the administrative staff.

The final system contains five programming languages; PHP, CSS, JavaScript, SQL and HTML. The system has met the client’s requirements and contains some added functions that were not in the initial requirements of the project. The system is password protected and can make a reservation, update a reservation, cancel a reservation, check availability, produce reports, add new staff members and reset passwords.

As set out in the terms of reference, the reliability and ease of use of the system was the main priority. In order to meet these requirements, the system has been tested by both the third party and the client. The system has been thoroughly tested eliminating any bugs. Validation rules are present in all 37 forms that have been created. This will ensure the user cannot make mistakes when using the system. A user manual has also been provided to ensure that the day to day users have no problems. (See separate document)

To ensure the client gains maximum benefit from the system it has been developed in the most intuitive and clear way possible. Without this, the system might not have been used by the staff in the hostel. From an academic side, the project incorporated a wide range of the material that is covered in Management Science and Information System Studies course, including Software Applications, Strategic Information Systems, Software Engineering and Introduction to Programming. It is felt the project will have benefited significantly from the addition of this material.
4.2 Recommendations

This entire custom-built management information system was designed and developed in a short amount of time by what began as an inexperienced developer. Therefore, there are additional functionality, validation and testing methods that could be performed to ensure the system is secure and reliable. The time necessary to thoroughly complete these tasks is far beyond the timescale of this project.

The system should be piloted in the hostel for a few weeks under supervision of the client. It could be run parallel with the manual system that currently stands in the hostel. This will ensure the system runs smoothly and no errors occur with the system when it is in constant use.

The heat maps in the system could be made available to the public. This would involve putting the system online, taking away the password protection and giving the public a selected view of the system. This could include people checking availability in the hostel and requesting a booking. Once the request was sent a member of staff could ensure the dates are available and email the customer back confirming the reservation. If this method of booking became popular the client could look into increasing the security measures allowing the public to make reservations online.

The system will collect all customer details including email addresses, age and nationality. It is recommended that the user examines this information carefully and identify quiet periods when promotions could be run, possibly targeting guests who have already stayed there. This could also be seen as a customer relationship management (CRM) system which would use the technology to develop relationships with past guests.

The client volunteers at several other hostels around Ireland, the system developed has the capabilities to be expanded to the other hostels. The system could be contained within one database giving the client the ability to run reports for all hostels in one form. It is also possible to create an app for the staff to use which would allow them check availability or make a booking on their phone or tablet.
BEN LETTERY CONNEMARA HOSTEL
Development of a Management Information System

Appendices
A. ORIGINAL PROJECT OUTLINE

Client: Ben Lettery Connemara Hostel

Project: Hostel Management Web Site

Location: Ballinafad, Connemara, Co. Galway (client contact is based in Dublin)

Client Contact: Joe Vale (ex Graduate Student, Statistics Department), 086 259 2713

Dept. Contact: Eamonn Mullins

Background to project:

Ben Lettery Connemara Hostel is a unique Irish hostel, which first opened its doors on the slopes of the Twelve Bens in 1969. Cyclists, hill walkers and backpackers favour the hostel’s idyllic location. The hostel is a hill walking landmark; known as the start or finish of one of Ireland’s most exhilarating hikes, the Glencoaghan Horseshoe, which crosses six of the Twelve Bens.

In 2012 an eclectic group of volunteers came together to renovate and reopen Ben Lettery Hostel. The group ranged in age from students to active retirees, from architects to carpenters, and from hill walkers to artists. The hostel reopened its doors on June 1st 2013 and has enjoyed a very successful first season staffed entirely by volunteers. The hostel has become a teahouse for some hill walking clubs, a base of operations for Galway Mountain Rescue, and a community resource.

The hostel manager is in need of a hostel management application to manage guest bookings, assign beds to guests and identify available beds. The purpose of the project is to develop a booking web site which would know the layout of rooms, assignment of guests to beds, record check in and out dates, record guest names and addresses, record the source of a booking, and allow guests to extend or cancel their stay. A simple interface assuming low user technical ability is required. The user would value a heat map of bookings and availability on any night in the past or future.

The hostel manager should have the ability to run reports such as guest profile by age group, nationality or source of booking; and review bookings to identify peak periods or days of the week; and perhaps identify quiet times in which to offer promotions.

There are three hostels currently managed by the same volunteer group, which could be supported by the web site in future. User logins and permissions should be considered.

Desired result of project:

- A live web site (e.g. www.benletteryhostel.com or www.volunteerhostels.ie).
- The ability to record guest bookings, cancellations and extensions.
- The ability to view a heat map of room / bed availability.
- The ability to run reports profiling guests and occupancy.
- The ability to create user logons and grant permissions to different types of users.
What project would involve:

- Modelling of hostel bookings in a scalable 3NF relational database.
- Designing an intuitive web site for use by users with various levels of technical ability, and using various devices including iPads, smart phones and laptops.
- Review and recommendation of open source database and web development technologies, with low bandwidth and ease of future maintenance goals.
- Review and recommendation of web hosting companies
B. INTERIM REPORT

Management Science and Information System Studies

Project - Hostel Management Web Site
Client - Ben Lettery Connemara Hostel
Student – Sarah Keane
Supervisor – Eamonn Mullins

Background to Project and Work to Date

Ben Lettery Connemara Hostel is a unique Irish hostel, which first opened its doors on the slopes of the Twelve Bens in 1969. Cyclists, hill walkers and backpackers favour the hostel's idyllic location. The hostel is a hill walking landmark; known as the start or finish of one of Ireland’s most exhilarating hikes, the Glencoaghan Horseshoe, which crosses six of the Twelve Bens.

The client is part of a voluntary group that are volunteering to run the hostel. The hostel manager is in need of a hostel management application to manage guest bookings, assign beds to guests and identify available beds. The purpose of the project is to develop a booking management information system which would know the layout of rooms, assignment of guests to beds, record check in and out dates, record guest names and addresses, record the source of a booking, and allow guests to extend or cancel their stay. A simple interface assuming low user technical ability is required. The user would value a heat map of bookings and availability on any night in the past or future.

Since the beginning of the project there have been several client and supervisor meetings in order to establish the work that will be involved. Research has been undertaken into open source database and web development technologies. An understanding of the project has been gained along with software that will be used to build it. The terms of reference have been agreed with the client.

Terms of Reference

- To develop a user-friendly web – based management information system for the use of hostel staff to make bookings;
- To ensure the website is simple to navigate using heat maps for bookings and availability;
- The system should have the ability to run reports such as guest profile by age group, nationality or source of booking;
- User access should be password projected;
- Review and recommend open source database and web development technologies;
- Review and recommend web hosting companies;
Further Work

- Develop story boards for management information system functionality including the creation of, changing and cancelling a single or group booking.
- Design of the database.
- Begin development of the database and website; using MySQL, PHP, JavaScript and HTML (technologies that have already been selected).
- Test the website to ensure everything is fully functional.
- Deliver the website.
- Produce comprehensive user documentation as specified by the client.
C. TECHNICAL REPORT

C.1. Structured Systems Analysis and Design Methodology

Before the delivery of the management information system, five fundamental phases were involved in the systems development life cycle of the project. The model used in the construction of this site was the Incremental Model. This method consists of the following stages:

- Requirements Analysis
- Design
- Development
- Implementation
- Testing

As the system advanced, adjustments were made, repeating the design and development stages in the next iteration.

This methodology lets both me as the analyst, and the eventual users of the system to design, implement and test incrementally (a little more is added each time) until the system is complete. The system is only defined as finished when it fulfils all of its requirements. This model combines with some of the fundamentals of the Waterfall Model with the iterative philosophy of prototyping. This process was ideal to this project as I was not a strong programmer at the beginning of the project so it allowed the system to be tested while the systems functions were being created. If any changes needed to be made it was ok to go back to the design phase and make changes.

It was decided to use the Incremental Model after investigating other methods including parallel, phased and prototyping methods. These were analysed bearing in mind the clarity
of user requirements, knowledge with technology, system complexity and reliability and the relative short time frame of the project.

C.2. System Design - Story Boards

One of the methods suggested by the client was the use of Storyboards Prototypes. “Storyboarding is an iterative, interaction design methodology that uses a series of sketches or pictures to demonstrate an end to end solution for a user scenario” (Anon., 2006). This is used to illustrate design concepts and obtain feedback early in the design process. A simple layout of the user interface was discussed in meetings with the client.

Storyboards for the main features of the user interface were created and are shown below.

![Storyboard - Room Heat Map](image)

Figure C.2.1 – Storyboard - Room Heat Map
Figure C.2.2 Storyboard – Bed Selection

The user can select the beds they would like to book.

Figure C.2.3 – Storyboard – Confirm Reservation

This will allow the user to enter the guests details and ensure the reservation details are correct.
These storyboards were created not long after the original plan was set out. They were originally created on paper by drawing sketches that would represent the design of the user interface. By using these Storyboards in the development phase it ensured the original design was replicated as much as possible. Throughout the development phase it was necessary to
make some changes to the design of the user interface and the differences can be seen when in Appendix G1, page G1 in the Sample Pages.

C.3. **Use Case Analysis**

Use Case diagrams are useful for modelling the interactions that exist between the user and the system. In this case there are two user groups, the Manager and Staff. There is only one main difference between the two. The only difference is that the Manager has access to the Master Password and can allow staff to create a new password if theirs is lost.

The main interactions are:

- Log into the system.
- View bookings due to arrive today.
- Cancel a booking.
- Update a guests details.
- Update a guests booking details.
- View reports.
- Search for a particular customers booking.
- Print of an invoice.
- Log out.
- Reset a password (Manager).

![Use Case Diagram]

Figure C.3.1 – UML Use Case Diagram of the System
C.4. **Data Flow Diagram**

![Data Flow Diagram](image)

Figure C.4.1 – Data Flow Diagram
C.5. **Standards and Conventions Used**

**Naming Standards**

The code and variables were named referring to their content to be as consistent as possible. This guarantees that the names of the code and variables can be simply interrupted. This will assist the future developer of the system.

To avoid any confusion when entering dates the format used throughout the system is dd-mm-yyyy As the American system puts the date first this help maximise ease of use. It also aids the use of date in the code functions.

**Coding Conventions**

Three coding conventions were used:

- Comments
- PHP TAGS
- Expressions

Comments - have been used throughout the code using the // symbol.

PHP TAGS - are used and represented by <?php insert code ?>. These have been used throughout the code as they are the most convenient way of including PHP code on different operating systems.

Expressions – Parentheses are used to resolve uncertainty, this was used to save time for the reader, as there is no need to spend time memorising precedence of operators.

**Design Standards**

Each web page within the user interface is the same size and layout. To ensure the user will not have to scroll vertically to read the contents of any page the page widths have been set to a standard size.

The ‘Invoice.php’ page has been designed to print the wanted information on the page. It doesn’t print the headers, footers or side menu.
D DATABASE TABLES AND RELATIONSHIP

D.1. Entity Relationship Diagram

![Entity Relationship Diagram for Ben Lettery Booking System](image)

Figure D.1.1 – Entity Relationship Diagram for Ben Lettery Booking System

D.2. Database Tables

Table D.2.1 – Design for Login Table

<table>
<thead>
<tr>
<th><strong>Table Name</strong></th>
<th>Login</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brief Description</strong></td>
<td>Contains staff details to give access to the system</td>
</tr>
<tr>
<td><strong>Related Tables</strong></td>
<td>Reservations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Data Name</strong></th>
<th><strong>Data Type</strong></th>
<th><strong>Description</strong></th>
<th><strong>Validation Rule</strong></th>
<th><strong>Linked To</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Staff_ID (Primary Key)</td>
<td>Int(11)</td>
<td>Auto Increment</td>
<td>No Duplicate Number</td>
<td>Reservations</td>
</tr>
<tr>
<td>Staff_Name</td>
<td>Varchar</td>
<td>Staff Name – Name Used when logging in.</td>
<td>Must be less than 20 characters</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>Varchar</td>
<td>Staff Password used when logging in.</td>
<td>Must be less than 11 characters</td>
<td></td>
</tr>
</tbody>
</table>
## Table D.2.2 – Design for Customers Table

**Application Design Worksheet Tables**

<table>
<thead>
<tr>
<th>Table Name:</th>
<th>Customers</th>
</tr>
</thead>
</table>

**Brief Description:**
Contains the customers details

### Related Tables

<table>
<thead>
<tr>
<th>Related Tables:</th>
<th>Reservations</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Data Name</th>
<th>Data Type</th>
<th>Description</th>
<th>Validation Rule</th>
<th>Linked To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cust_id (Primary Key)</td>
<td>Int(11)</td>
<td>Auto Increment. Unique ID given to every customer.</td>
<td>No Duplicate Number.</td>
<td>Reservations</td>
</tr>
<tr>
<td>Cust_name</td>
<td>Varchar (20)</td>
<td>Guests Name – Necessary for when the guest arrives</td>
<td>Must be less than 20 characters. Mandatory</td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>Int (3)</td>
<td>Guest’s age, used in reports.</td>
<td>Must be less than 3 characters. Not mandatory.</td>
<td></td>
</tr>
<tr>
<td>Nationality</td>
<td>Varchar (20)</td>
<td>Guest’s Nationality, used in reports.</td>
<td>Must be less than 20 characters. Not mandatory</td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td>Varchar (20)</td>
<td>Guest’s age, used in reports.</td>
<td>Must be less than 20 characters. Mandatory.</td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Varchar (20)</td>
<td>Where the guest heard about the hostel.</td>
<td>Must be less than 20 characters. Not mandatory</td>
<td></td>
</tr>
<tr>
<td>Comment</td>
<td>Text</td>
<td>Comments that the user may want to take note of, eg. Pay on arrival.</td>
<td>Can be any length of text.</td>
<td></td>
</tr>
</tbody>
</table>
Table D.2.3 – Design of Reservations Table

<table>
<thead>
<tr>
<th>Application Design Worksheet Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Name: Reservations</td>
</tr>
<tr>
<td>Brief Description: Contains the reservation details</td>
</tr>
<tr>
<td>Related Tables: Booking, Customers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Name</th>
<th>Data Type</th>
<th>Description</th>
<th>Validation Rule</th>
<th>Linked To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res_num</td>
<td>Int(11)</td>
<td>Auto Increment. Each reservation is given a unique number.</td>
<td>No Duplicate Number</td>
<td>Booking, Customers</td>
</tr>
<tr>
<td>(Primary Key)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arrival_date</td>
<td>date</td>
<td>Date the guest is due to arrive.</td>
<td>Must be after today's date when booking.</td>
<td></td>
</tr>
<tr>
<td>Departure_date</td>
<td>date</td>
<td>Date guest is due to leave.</td>
<td>Must be after today's date and after the arrival date.</td>
<td></td>
</tr>
<tr>
<td>Staff_id</td>
<td>Int(11)</td>
<td>ID of staff member who took the booking.</td>
<td>Staff must be logged in to make booking.</td>
<td>login</td>
</tr>
<tr>
<td></td>
<td>Foreign Key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_id</td>
<td>Int(11)</td>
<td>Customer ID – ID of the customer for who the booking belongs to.</td>
<td>Must be less than 20 characters. Mandatory.</td>
<td>Customers</td>
</tr>
</tbody>
</table>
### Table D.2.4 – Design of Rooms Table

<table>
<thead>
<tr>
<th>Data Name</th>
<th>Data Type</th>
<th>Description</th>
<th>Validation Rule</th>
<th>Linked To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room_id</td>
<td>Int(11)</td>
<td>Auto Increment.</td>
<td>No Duplicate Number</td>
<td>Booking</td>
</tr>
<tr>
<td>Bed_id</td>
<td>Int(11)</td>
<td>The number of beds in each room.</td>
<td>Must be over one.</td>
<td>Booking</td>
</tr>
</tbody>
</table>

**Table Name:** Rooms  
**Brief Description:** Contains the room numbers and number of beds in each room.  
**Related Tables:** Booking

### Table D.2.5 – Design of Bookings Table

<table>
<thead>
<tr>
<th>Data Name</th>
<th>Data Type</th>
<th>Description</th>
<th>Validation Rule</th>
<th>Linked To</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Int(11)</td>
<td>Auto Increment.</td>
<td>No Duplicate Number</td>
<td></td>
</tr>
<tr>
<td>Room_num</td>
<td>Int(11)</td>
<td>The room number which have been booked.</td>
<td>Must contain a room number</td>
<td>Rooms</td>
</tr>
<tr>
<td>Bed_num</td>
<td>Int (11)</td>
<td>The bed number/s which have been booked.</td>
<td>Must contain at least one bed</td>
<td></td>
</tr>
<tr>
<td>Res_num</td>
<td>Int(11)</td>
<td>Reservation number for which the booking belongs to.</td>
<td>Must have a reservation number</td>
<td>Reservations</td>
</tr>
</tbody>
</table>

**Table Name:** Booking  
**Brief Description:** Contains details of the rooms and beds for each reservation  
**Related Tables:** Reservations
E. USER MANUALS

Please see the User and Technical Manuals accompanying this report. The User Manual gives instructions on how to navigate the user interface. The Technical Manual describes how to set up the system for initial deployment.

Electronic copies of both can be found in the CD at the back.
F. TESTING DOCUMENTATION

Table F.1 – Staff Log-in

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log In</td>
<td>1.1</td>
<td>Check that someone with a username and password can log in</td>
<td>Check that all member saved in the database can be logged in. Add new staff members and check they can be logged in. Unique staff names are compulsory.</td>
<td>In order to gain access to the management information system the user must have the correct credentials. When the wrong details are entered the user will not have access.</td>
</tr>
</tbody>
</table>

Table F.2 – Adding Staff Member

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding Staff Member</td>
<td>1.2</td>
<td>Check that a new staff member has been added</td>
<td>Does the correct form appear? Does the system check that there is correct information in the fields required? Does the user get a confirmation? Unique staff names are compulsory.</td>
<td>A user must be logged in order add a staff member. The username must be unique. The password field must match the confirm password box. If they don’t match an error message will appear. The user will be informed when the action is complete</td>
</tr>
<tr>
<td>Function</td>
<td>Test</td>
<td>Objective</td>
<td>Test Case</td>
<td>Expected Result</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
<td>----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Choosing the Room</td>
<td>1.3</td>
<td>Checking that the correct rooms appear once dates have been entered.</td>
<td>Does the correct form appear? Do the correct rooms appear? Once a room has been booked will it appear again? If a bed in a room is booked will it turn red or orange? Can you click on the red rooms?</td>
<td>When ‘Search Dates’ is clicked in the ‘Dates Form’ the ‘Rooms Form’ appears. The correct rooms appear. When the user books a bed in a room for certain dates and you go back to see if the room is still available it isn’t. The rooms either turn red or orange depending if all the beds are gone or just some of them. The user will not be able to click on the rooms that are red. An error message will be displayed ‘This room is not available.’</td>
</tr>
</tbody>
</table>
### Table F.4 – Choosing Dates

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
<th>Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose Dates</td>
<td>1.4</td>
<td>Check that when the user enters dates the dates are valid</td>
<td>Does the correct form appear? Does a validation appear if the date entered has passed? Can arrival date be after departure date?</td>
<td>The user can enter any dates but when the submit button is pressed it will send an error message if the arrival date is after the departure date, if they are on the same day or if the date entered has already passed.</td>
<td></td>
</tr>
</tbody>
</table>

### Table F.5 – Change Dates

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
<th>Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Dates</td>
<td>1.5</td>
<td>Check that if the user clicks this proper form opens and the unwanted reservation won’t be saved</td>
<td>Does the correct form appear? What dates appear when the ‘Search Dates’ form opens?</td>
<td>When the ‘Change Dates’ button is clicked in the reservations form it redirects the user to the ‘Search Dates’ page. Here the same dates as entered before appear.</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Test</td>
<td>Objective</td>
<td>Test Case</td>
<td>Expected Result</td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>------</td>
<td>---------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Confirm Reservation/ Enter Customer Details</td>
<td>1.6</td>
<td>Check that the correct beds and room appear on the form. Check the guest’s details are entered.</td>
<td>Does the correct form appear? Does it contain the details from the correct rooms and beds that were chosen? Can the customer details be changed? If the new guest checkbox is ticked will it create a new guest?</td>
<td>When the beds are picked and the ‘Confirm Button’ is clicked the ‘Reservation Form’ is opened. This contains the information that has been chosen so far, including the correct arrival and departure dates, the room number, and the bed number/s and allows the user to enter that guest’s details. The user will click the new customer checkbox if the guest’s information doesn’t already appear. The guest’s name and email address will be mandatory. If they are not entered an error message will be displayed.</td>
<td></td>
</tr>
</tbody>
</table>
Table F.7 – Invoice Form

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
<th>Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice</td>
<td>1.7</td>
<td>Ensure that the correct details are displayed. Check if the ‘Print’ button works.</td>
<td>Does the correct form appear? Are the details that were chosen and confirmed the same as the details shown? Does the print button print the whole page or just a selection of the text?</td>
<td>Once the ‘Confirm Reservation’ button is clicked in the reservations page the invoice form appears. It contains the correct information from the last booking. When the ‘Print’ button is pressed it prints out the text within the form and removes the page headings.</td>
<td></td>
</tr>
</tbody>
</table>

Table F.8 – Adding More Rooms

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
<th>Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding More Rooms</td>
<td>1.8</td>
<td>Check that the user can add more rooms to a booking using the same guest’s details.</td>
<td>Does the correct form appear? Are the dates from the original booking the same? Do the guest’s details appear in the reservations form? Are these details saved under the same guest or creating a new guest?</td>
<td>When the user clicks the ‘Add More Rooms’ in the ‘Invoice’ page the user is directed to the ‘Search Dates’ page. The same dates used for the previous booking are displayed. When the ‘Reservation’ page is reached the guests details are displayed. If the user clicks the checkbox for new customer a new customer will be added with the same credentials as the previous.</td>
<td></td>
</tr>
</tbody>
</table>
### Table F.9 – Updating a Password

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updating a Password</td>
<td>1.9</td>
<td>Check if the user's password is updated.</td>
<td>Is the user able to log in on the new password? Are the security setting working to allow user to get a new password?</td>
<td>The user must enter a new password and confirm the new password. A master password must also be entered to ensure that not everyone can change the password. The user can then log in once the password is changed.</td>
</tr>
</tbody>
</table>

### Table F.10 – Reports

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reports</td>
<td>1.10</td>
<td>Check if the correct reports are being displayed</td>
<td>Does the correct form appear? Will the correct reports be displayed when they are supposed to be? Are the date ranges working?</td>
<td>The user will be directed to the ‘Reports’ page when the ‘Reports’ link is pressed. The reports available will be displayed. The user must enter the date ranges after clicking on the chosen report. Only after a date range is entered, will the report be displayed. The figures in the report were tested from the figures in the database.</td>
</tr>
</tbody>
</table>
Table F.11 – Updating a Booking

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updating a Booking</td>
<td>1.11</td>
<td>Will the guest’s details get updated in the database? Can a guest’s booking be updated?</td>
<td>Does the correct form appear? Will the guest’s updated details be displayed throughout the system? When the booking is updated will the old one be deleted?</td>
<td>The user will be directed to the ‘Update Booking’ page when the ‘Update a Booking link is pressed’. The user will be required to pick the name of the guest they want to update. Once the user updates the details the details will appear throughout the system when the user details are displayed. When updating the dates the original booking will be deleted and the user will not be required to enter the customer details again.</td>
</tr>
</tbody>
</table>
Table F.12 – Cancelling a Booking

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancelling a Booking</td>
<td>1.12</td>
<td>Check that the reservation will be deleted.</td>
<td>Does the correct form appear? Will the reservation be deleted? Will the correct reservation</td>
<td>The user will be directed to the ‘Cancel Booking’ page when the ‘Cancel a Booking’ link is pressed. It is necessary to select the name of the guest who wants to cancel a booking. The guests detail will be displayed which will be deleted. This then makes the room/beds available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>details appear? Will the room/beds become available?</td>
<td></td>
</tr>
</tbody>
</table>

Table F.13 - Logout

<table>
<thead>
<tr>
<th>Function</th>
<th>Test</th>
<th>Objective</th>
<th>Test Case</th>
<th>Expected Result</th>
<th>Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logout</td>
<td>1.13</td>
<td>Check that the user can log out of the system.</td>
<td>Does the user get logged out? Does the user get directed to the log in page?</td>
<td>The user gets logged out and brought back to the log in page.</td>
<td></td>
</tr>
</tbody>
</table>
G. SAMPLE PAGES

Figure G.1.1 – Staff Log-in Page

Figure G.1.2 – Home Page
Figure G.1.3 – Search Dates

Figure G.1.4 – Room Layout/Heat Map
Figure G.1.5 – Bed Layout

Figure G.1.6 – Bed Layout
Figure G.1.7 – Entering Customer Details/Confirming Reservation

Figure G.1.8 - Invoice
Figure G.1.9 – Selecting a Guest for Booking Cancelation

Figure G.1.10 – Ensuring Selection of Correct Guest

Figure G.1.11 – Confirming Cancelation by Choosing Reservation Number
Figure G.1.12 – Selecting a Guest to Update

Figure G.1.13 – Ensuring Correct Guest Selection

Figure G.1.14 – Updating Guests Details
**Update a Booking**

Gerry Fallon details have been updated.

Click the 'Update Dates' button if you would like change the dates for this guest. But first select the reservation number you wish to change.

If you have made all the necessary changes then click Finish.

**Figure G.1.15 – Updating a Reservation**

<table>
<thead>
<tr>
<th>Res Num</th>
<th>Arrival Date</th>
<th>Departure Date</th>
<th>Room Num</th>
<th>Bed Num</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2014-03-22</td>
<td>2014-03-23</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>2014-03-22</td>
<td>2014-03-23</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Add Staff Member**

Staff Name
Password
Confirm Password

**Figure G.1.16 – Adding a Staff Member**
Figure G.1.17 – Resetting a Password

Figure G.1.18 - Reports

Figure G.1.19 - Invoice
Figure G.1.20 – Report – Age Profile of Guests

Figure G.1.21 – Report - Nationality Profile of Guests
### Figure G.1.22 – Reports – Staff Allocations

![Staff Allocations](image1)

**Staff Allocations**

Please enter the dates you would like to search between:

- **03/03/2014**
- **20/05/2014**

Below shows which staff took which booking:

<table>
<thead>
<tr>
<th>Staff Name</th>
<th>Reservation Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sarah</td>
<td>1</td>
</tr>
<tr>
<td>Sarah</td>
<td>4</td>
</tr>
<tr>
<td>Sarah</td>
<td>5</td>
</tr>
<tr>
<td>Sarah</td>
<td>6</td>
</tr>
<tr>
<td>Sarah</td>
<td>7</td>
</tr>
<tr>
<td>Sarah</td>
<td>9</td>
</tr>
</tbody>
</table>

Number of Booking per Staff Member:

<table>
<thead>
<tr>
<th>Staff Name</th>
<th>Reservation Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sarah</td>
<td>6</td>
</tr>
</tbody>
</table>

![Staff Details](image2)

**Guest Details**

Please enter the dates you would like to search between:

- **12/03/2014**
- **22/03/2014**

The table below shows all the guests:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Nationality</th>
<th>Email</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Conor Newman</td>
<td>56</td>
<td>Ireland</td>
<td><a href="mailto:conor@gmail.com">conor@gmail.com</a></td>
<td>Walk-in</td>
</tr>
<tr>
<td>5</td>
<td>Niamh Fanning</td>
<td>23</td>
<td>Ireland</td>
<td><a href="mailto:niamh@tcd.ie">niamh@tcd.ie</a></td>
<td>An Oige</td>
</tr>
<tr>
<td>6</td>
<td>Annemarie Murphy</td>
<td>23</td>
<td>Ireland</td>
<td><a href="mailto:murphy@tcd.ie">murphy@tcd.ie</a></td>
<td>Walk-in</td>
</tr>
<tr>
<td>8</td>
<td>Gerry Fallon</td>
<td>33</td>
<td>Ireland</td>
<td><a href="mailto:gerry@hotmail.com">gerry@hotmail.com</a></td>
<td>Walk-in</td>
</tr>
</tbody>
</table>
Source Code

Due to the large amount of source code written for the application the source code can be found on the CD attached to this document. However there are two samples of code shown here.

Home Page – default.php

```php
<?php
require '../Connection1.php';
include 'styles/stylesheet.php';
//turning of error for variables that aren't being used
ini_set('display_errors', 'Off');

session_start();
$staff_name = $_SESSION['staff_name'];

//ensuring someone is logged in
if($staff_name == ''){
    ?><script> location.href = "../mysite/login/default.php";</script> <?php
}

$today = date("Y-m-d");
$tomorrow = date("Y-m-d", time()+86400);

// unsetting session variables
if(isset($_SESSION['cust_name_new']))
    unset($_SESSION['cust_name_new']);
if(isset($_SESSION['nationality_new']))
    unset($_SESSION['nationality_new']);
if(isset($_SESSION['email_new']))
    unset($_SESSION['email_new']);
if(isset($_SESSION['comment_new']))
    unset($_SESSION['comment_new']);
if(isset($_SESSION['age_new']))
    unset($_SESSION['age_new']);
if(isset($_SESSION['cust_id_new']))
    unset($_SESSION['cust_id_new']);
if(isset($_SESSION['arrival_date']))
    unset($_SESSION['arrival_date']);
if(isset($_SESSION['departure_date']))
    unset($_SESSION['departure_date']);
if(isset($_SESSION['cust_id']))
    unset($_SESSION['cust_id']);
if(isset($_SESSION['cust_name']))
    unset($_SESSION['cust_name']);
if(isset($_SESSION['cust_age']))
    unset($_SESSION['cust_age']);
if(isset($_SESSION['cust_email']))
    unset($_SESSION['cust_email']);
if(isset($_SESSION['cust_nat']))
    unset($_SESSION['cust_nat']);
if(isset($_SESSION['cust_source']))
```
unset($_SESSION['cust_source']);
if(isset($_SESSION['custComment']))
    unset($_SESSION['custComment']);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html dir="ltr" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>Home</title>
<link href="styles/style1.css" media="screen" rel="stylesheet" title="CSS" type="text/css" />
</head>
<body>
<!-- Begin Container -->
<div id="container">
    <!-- Begin Masthead -->
    <div id="masthead">
        <img alt="" height="66" src="logo2.png" width="150" />
        <p>
            Logged In - <?php echo $staff_name; ?>
        </p>
        <a href="logout.php">Logout</a>
    </div>
    <!-- End Masthead -->
    <!-- Begin Page Content -->
    <div id="page_content">
        <!-- Begin Sidebar -->
        <div id="sidebar">
            <ul>
                <li><a href="default.php">Home</a></li>
                <li><a href="dates/default.php">Make a Booking</a></li>
                <li><a href="CancelBooking/cancel.php">Cancel Booking</a></li>
                <li><a href="UpdateBooking/default.php">Update a Booking</a></li>
                <li><a href="Add Staff Member/default.php">Add Staff Member</a></li>
                <li><a href="Reports/reports.php">Reports</a></li>
                <li><a href="login/default.php">Logout</a></li>
            </ul>
        </div>
        <!-- End Sidebar -->
        <!-- Begin Content -->
        <div id="content">
            <!-- Begin Container -->
            <div id="container">
                <!-- Begin Masthead -->
                <div id="masthead">
                    <img alt="" height="66" src="logo2.png" width="150" />
                    <p>
                        Logged In - <?php echo $staff_name; ?>
                    </p>
                    <a href="logout.php">Logout</a>
                </div>
                <!-- End Masthead -->
                <!-- Begin Page Content -->
                <div id="page_content">
                    <!-- Begin Sidebar -->
                    <div id="sidebar">
                        <ul>
                            <li><a href="default.php">Home</a></li>
                            <li><a href="dates/default.php">Make a Booking</a></li>
                            <li><a href="CancelBooking/cancel.php">Cancel Booking</a></li>
                            <li><a href="UpdateBooking/default.php">Update a Booking</a></li>
                            <li><a href="Add Staff Member/default.php">Add Staff Member</a></li>
                            <li><a href="Reports/reports.php">Reports</a></li>
                            <li><a href="login/default.php">Logout</a></li>
                        </ul>
                    </div>
                    <!-- End Sidebar -->
                    <!-- Begin Content -->
                    <div id="content">
                        <!-- Begin Container -->
                        <div id="container">
                            <!-- Begin Masthead -->
                            <div id="masthead">
                                <img alt="" height="66" src="logo2.png" width="150" />
                                <p>
                                    Logged In - <?php echo $staff_name; ?>
                                </p>
                                <a href="logout.php">Logout</a>
                            </div>
                            <!-- End Masthead -->
                            <!-- Begin Page Content -->
                            <div id="page_content">
                                <!-- Begin Sidebar -->
                                <div id="sidebar">
                                    <ul>
                                        <li><a href="default.php">Home</a></li>
                                        <li><a href="dates/default.php">Make a Booking</a></li>
                                        <li><a href="CancelBooking/cancel.php">Cancel Booking</a></li>
                                        <li><a href="UpdateBooking/default.php">Update a Booking</a></li>
                                        <li><a href="Add Staff Member/default.php">Add Staff Member</a></li>
                                        <li><a href="Reports/reports.php">Reports</a></li>
                                        <li><a href="login/default.php">Logout</a></li>
                                    </ul>
                                </div>
                                <!-- End Sidebar -->
                                <!-- Begin Content -->
                                <div id="content">
                                    <!-- Begin Container -->
                                    <div id="container">
                                        <!-- Begin Masthead -->
                                        <div id="masthead">
                                            <img alt="" height="66" src="logo2.png" width="150" />
                                            <p>
                                                Logged In - <?php echo $staff_name; ?>
                                            </p>
                                            <a href="logout.php">Logout</a>
                                        </div>
                                        <!-- End Masthead -->
                                        <!-- Begin Page Content -->
                                        <div id="page_content">
                                            <!-- Begin Sidebar -->
                                            <div id="sidebar">
                                                <ul>
                                                    <li><a href="default.php">Home</a></li>
                                                    <li><a href="dates/default.php">Make a Booking</a></li>
                                                    <li><a href="CancelBooking/cancel.php">Cancel Booking</a></li>
                                                    <li><a href="UpdateBooking/default.php">Update a Booking</a></li>
                                                    <li><a href="Add Staff Member/default.php">Add Staff Member</a></li>
                                                    <li><a href="Reports/reports.php">Reports</a></li>
                                                    <li><a href="login/default.php">Logout</a></li>
                                                </ul>
                                            </div>
                                            <!-- End Sidebar -->
                                            <!-- Begin Content -->
                                            <div id="content">
                                                <!-- Begin Container -->
                                                <div id="container">
                                                    <!-- Begin Masthead -->
                                                    <div id="masthead">
//-->
</script>
</head>
<body>
<img src="CoverPhoto1.png" name="slide" width="450" height="300" />
<script>
//-->
//variable that will increment through the images
var step=1
function slideit(){
//if browser does not support the image object, exit.
if (!document.images)
return
document.images.slide.src=eval("image"+step+".src")
if (step<3)
step++
else
step=1
//call function "slideit()" every 2.5 seconds
setTimeout("slideit()",2500)
}
slideit()
//-->
</script>
</body>
</html>

<h2 id="heading">Todays Bookings</h2>
<form method="post" action="default.php">
<input name="arr_date" type="date" value="<?php echo $today?>" id ="text" style="width: 148px" />
<input id="text" name="dep_date" type="date" value="<?php echo $tomorrow ?>" style="width: 148px" />
<input id ="button"name="Submit1" type="submit" value="Search Dates" style="left: 4px; top: -2px; width: 113px" />
</form>

<form method="post" action="Invoice/invoice.php">
<select id="text" name="cust_details">
<?php
// selecting customer details which are displayed on home page
$query2 = "select (customers.cust_name), customers.nationality, COUNT(booking.bed_num) from customers join reservations ON customers.cust_id = reservations.cust_id join booking ON booking.res_num = reservations.res_num where reservations.arrival_date >= '$arr_date' group by customers.cust_id";
$query_run2 = mysql_query($query2);
while($qr = mysql_fetch_array($query_run2)){
 echo "<option>$qr[cust_name]</option">;
}
?>
</select>
</form>
<input id="button" name="Submit1" type="submit" value="Find Customer Details" />
</form>

<label id="smalltext">Customer due to arrive today and over the next few days.</label>

<?php
$arr_date = $_POST['arr_date'];
$dep_date = $_POST['dep_date'];

   //selecting customers due to arrive today
   $queryTodayBooking = "select (customers.cust_name), customers.nationality, COUNT(booking.bed_num) from customers join reservations ON customers.cust_id = reservations.cust_id join booking ON booking.res_num = reservations.res_num where reservations.arrival_date >= '$arr_date' and reservations.arrival_date >= '$today' group by customers.cust_id order by reservations.arrival_date ";
   $result = mysql_query($queryTodayBooking);

   echo "<table border='1' id = 'customers'>
<tr>
<th>Name</th>
<th>Nationality</th>
<th>Num of Beds</th>
</tr>

while ($query_row = mysql_fetch_assoc($result)){
   echo "<tr>
   echo "<td>" . $query_row['cust_name'] . "</td>";
   echo "<td>" . $query_row['nationality'] . "</td>";
   echo "<td>" . $query_row['COUNT(booking.bed_num)'] . "</td>";
   echo "</tr>";
}
   echo "</table>";

?>
<p>&nbsp;</p>
</li>
</ul>
</div>
</div>

<!-- End Content -->
<!-- End Page Content -->
<!-- Begin Footer -->
<div id="footer">
..</div>
<!-- End Footer -->
<!-- End Container -->
<?php
require './Connection1.php';
include './style/stylesheet.php';

session_start();
ini_set('display_errors', 'Off');

$staff_name = $_SESSION['staff_name'];

//ensuring someone is logged in
if($staff_name == ''){
    ?> <script> location.href = './login/default.php'; </script> <?php
}

$_SESSION['arrival_date'] = $_POST['arrival_date'];
$_SESSION['departure_date'] = $_POST['departure_date'];
$staff_name =$_SESSION['staff_name'];
$password = $_SESSION['password'];

$dateA = $_POST['arrival_date'];
$dateD = $_POST['departure_date'];

//checking availability
$query1 = "SELECT COUNT(room_num) from booking join reservations on reservations.res_num = booking.res_num where (booking.room_num = 1) AND ('$dateA' <= reservations.arrival_date) AND ('$dateD'> reservations.arrival_date)OR (booking.room_num =1) AND ('$dateA' < reservations.departure_date) AND ('$dateD'> reservations.arrival_date)OR (booking.room_num = 1) AND ('$dateA' < reservations.departure_date) AND ('$dateD' > reservations.departure_date)";
$result1 = mysql_query($query1);
$tot_beds1 = mysql_result($result1,0);

//How many beds are in room 1
$query11 = "select bed_id from rooms where room_id = 1";
$result11 = mysql_query($query11);
$total_beds1 = mysql_result($result11,0);
$freebedR1 = $total_beds1 - $tot_beds1;

//checking availability
$query2 = "SELECT COUNT(room_num) from booking join reservations on reservations.res_num = booking.res_num where (booking.room_num = 2) AND ('$dateA' <= reservations.arrival_date) AND ('$dateD'> reservations.arrival_date)OR (booking.room_num =2) AND ('$dateA' < reservations.departure_date) AND ('$dateD'> reservations.arrival_date)OR (booking.room_num = 2) AND ('$dateA' < reservations.departure_date) AND ('$dateD' > reservations.departure_date)";
$result2 = mysql_query($query2);
$tot_beds2 = mysql_result($result2,0);

//How many beds are in room 2
$query12 = "select bed_id from rooms where room_id = 2";
$result12 = mysql_query($query12);
$$\text{$total\_beds2 = mysql\_result($result12,0);}$$

$$\text{$freebedR2 = $total\_beds2 - $tot\_beds2;}$$

//checking availability
$$\text{$query3 = "SELECT COUNT(room\_num) from booking join reservations on reservations.res\_num = booking.res\_num where (booking.room\_num = 3) AND ($dateA <= reservations.arrival\_date) AND ($dateD >= reservations.departure\_date) OR (booking.room\_num = 3) AND ($dateA < reservations.departure\_date) OR (booking.room\_num = 3) AND ($dateD > reservations.arrival\_date) OR (booking.room\_num = 3) AND ($dateA < reservations.departure\_date) AND ($dateD > reservations.departure\_date)";}$$

$$\text{$result3 = mysql\_query($query3);}$

$$\text{$tot\_beds3 = mysql\_result($result3,0);}$

//How many beds are in room 3
$$\text{$query13 = "select bed\_id from rooms where room\_id = 3";}$$

$$\text{$result13 = mysql\_query($query13);}$

$$\text{$total\_beds3 = mysql\_result($result13,0);}$

$$\text{$freebedR3 = $total\_beds3 - $tot\_beds3;}$$

//checking availability
$$\text{$query4 = "SELECT COUNT(room\_num) from booking join reservations on reservations.res\_num = booking.res\_num where (booking.room\_num = 4) AND ($dateA <= reservations.arrival\_date) AND ($dateD >= reservations.departure\_date) OR (booking.room\_num = 4) AND ($dateA < reservations.departure\_date) OR (booking.room\_num = 4) AND ($dateD > reservations.arrival\_date) OR (booking.room\_num = 4) AND ($dateA < reservations.departure\_date) AND ($dateD > reservations.departure\_date)";}$$

$$\text{$result4 = mysql\_query($query4);}$

$$\text{$tot\_beds4 = mysql\_result($result4,0);}$

//How many beds are in room 4
$$\text{$query14 = "select bed\_id from rooms where room\_id = 4";}$$

$$\text{$result14 = mysql\_query($query14);}$

$$\text{$total\_beds4 = mysql\_result($result14,0);}$

$$\text{$freebedR4 = $total\_beds4 - $tot\_beds4;}$$

//checking availability
$$\text{$query5 = "SELECT COUNT(room\_num) from booking join reservations on reservations.res\_num = booking.res\_num where (booking.room\_num = 5) AND ($dateA <= reservations.arrival\_date) AND ($dateD >= reservations.departure\_date) OR (booking.room\_num = 5) AND ($dateA < reservations.departure\_date) OR (booking.room\_num = 5) AND ($dateD > reservations.arrival\_date) OR (booking.room\_num = 5) AND ($dateA < reservations.departure\_date) AND ($dateD > reservations.departure\_date)";}$$

$$\text{$result5 = mysql\_query($query5);}$

$$\text{$tot\_beds5 = mysql\_result($result5,0);}$

//How many beds are in room 5
$$\text{$query15 = "select bed\_id from rooms where room\_id = 5";}$$

$$\text{$result15 = mysql\_query($query15);}$

$$\text{$total\_beds5 = mysql\_result($result15,0);}$

$$\text{$freebedR5 = $total\_beds5 - $tot\_beds5;}$$

//checking availability
$query6 = "SELECT COUNT(room_num) from booking join reservations on reservations.res_num = booking.res_num where (booking.room_num = 6) AND ('$dateA' <= reservations.arrival_date) AND ('$dateD' > reservations.arrival_date)OR (booking.room_num = 6) AND ('$dateA' < reservations.departure_date) AND ('$dateD' > reservations.arrival_date)OR (booking.room_num = 6) AND ('$dateA' < reservations.departure_date) AND ('$dateD' > reservations.departure_date)";
$result6 = mysql_query($query6);
$tot_beds6 = mysql_result($result6,0);

//How many beds are in room 6
$query16 = "select bed_id from rooms where room_id = 6";
$result16 = mysql_query($query16);
$total_beds6 = mysql_result($result16,0);

$freebedR6 = $total_beds6 - $tot_beds6;

//checking availability
$query7 = "SELECT COUNT(room_num) from booking join reservations on reservations.res_num = booking.res_num where (booking.room_num = 7) AND ('$dateA' <= reservations.arrival_date) AND ('$dateD' > reservations.arrival_date)OR (booking.room_num = 7) AND ('$dateA' < reservations.departure_date) AND ('$dateD' > reservations.departure_date)OR (booking.room_num = 7) AND ('$dateA' < reservations.departure_date) AND ('$dateD' > reservations.departure_date)";
$result7 = mysql_query($query7);
$tot_beds7 = mysql_result($result7,0);

//How many beds are in room 7
$query17 = "select bed_id from rooms where room_id = 7";
$result17 = mysql_query($query17);
$total_beds7 = mysql_result($result17,0);

$freebedR7 = $total_beds7 - $tot_beds7;

//checking availability
$query8 = "SELECT COUNT(room_num) from booking join reservations on reservations.res_num = booking.res_num where (booking.room_num = 8) AND ('$dateA' <= reservations.arrival_date) AND ('$dateD' > reservations.arrival_date)OR (booking.room_num = 8) AND ('$dateA' < reservations.departure_date) AND ('$dateD' > reservations.departure_date)OR (booking.room_num = 8) AND ('$dateA' < reservations.departure_date) AND ('$dateD' > reservations.departure_date)";
$result8 = mysql_query($query8);
$tot_beds8 = mysql_result($result8,0);

//How many beds are in room 8
$query18 = "select bed_id from rooms where room_id = 8";
$result18 = mysql_query($query18);
$total_beds8 = mysql_result($result18,0);

$freebedR8 = $total_beds8 - $tot_beds8;
?>

<!-- The change image function are checking to see if the number of free beds is 0, between 0 and the room capacity or full -->
<!-- The submit function is a validation rule stopping the user clicking on a room with no free beds -->
<script type="text/javascript">
function changeImageR1()
{
var freebed1 = "<?php echo $freebedR1; ?>";
if (freebed1 == 2){
document.getElementById("img1").src = "./GreenR1.png";
}
else if ((freebed1 < 2) && (freebed1 > 0))
{
document.getElementById("img1").src = "./OrangeR1.png";
}
else {
document.getElementById("img1").src = "./RedR1.png";
}
}
</script>

<script type="text/javascript">
function room1submit()
{
var freebed1 = "<?php echo $freebedR1; ?>";
if ((freebed1 <= 2) && (freebed1 > 0)){
document.getElementById("form_R1").submit();
}
else{
alert("There is no beds available in this room");
}
}
</script>

<script type="text/javascript">
function changeImageR2()
{
var freebed2 = "<?php echo $freebedR2; ?>";
if (freebed2 == 2){
document.getElementById("img2").src = "./GreenR2.png";
}
else if ((freebed2 < 2) && (freebed2 > 0))
{
document.getElementById("img2").src = "./OrangeR2.png";
}
else {
document.getElementById("img2").src = "./RedR2.png";
}
}
</script>

<script type="text/javascript">
function room2submit()
{
var freebed2 = "<?php echo $freebedR2; ?>";
if ((freebed2 <= 2) && (freebed2 > 0)){
document.getElementById("form_R2").submit();
} else{
alert("There is no beds available in this room");
}
</script>
<script type="text/javascript">
function changeImageR3() {
 var freebed3 = "<?php echo $freebedR3; ?>";
 if (freebed3 == 4) {
 document.getElementById("img3").src = ".../GreenR3.png";
 } else if ((freebed3 < 4) && (freebed3 > 0)) {
 document.getElementById("img3").src = ".../OrangeR3.png";
 } else {
 document.getElementById("img3").src = ".../RedR3.png";
 }
}
</script>

<script type="text/javascript">
function room3submit() {
 var freebed3 = "<?php echo $freebedR3; ?>";
 if ((freebed3 <= 4) && (freebed3 > 0)) {
 document.getElementById("form_R3").submit();
 } else {
 alert("There is no beds available in this room");
 }
}
</script>

<script type="text/javascript">
function changeImageR4() {
 var freebed4 = "<?php echo $freebedR4; ?>";
 if (freebed4 == 4) {
 document.getElementById("img4").src = ".../GreenR4.png";
 } else if ((freebed4 < 4) && (freebed4 > 0)) {
 document.getElementById("img4").src = ".../OrangeR4.png";
 } else {
 document.getElementById("img4").src = ".../RedR4.png";
 }
}
</script>

<script type="text/javascript">
function room4submit() {
 var freebed4 = "<?php echo $freebedR4; ?>";
 if ((freebed4 <= 4) && (freebed4 > 0)) {
 document.getElementById("form_R4").submit();
 }
</script>
else{
    alert("There is no beds available in this room");
}
</script>

<script type="text/javascript"
function changeImageR5()
{
    var freebed5 = "<?php echo $freebedR5; ?>";
    if (freebed5 == 6){
        document.getElementById("img5").src = ".../GreenR5.png";
    }
    else if ((freebed5 < 6) && (freebed5 > 0))
    {
        document.getElementById("img5").src = ".../OrangeR5.png";
    }
    else {
        document.getElementById("img5").src = ".../RedR5.png";
    }
}</script>

<script type="text/javascript"
function room5submit()
{
    var freebed5 = "<?php echo $freebedR5; ?>";
    if ((freebed5 <= 6) && (freebed5 >0 )){
        document.getElementById("form_R5").submit();
    }
    else{
        alert("There is no beds available in this room");
    }
}</script>

<script type="text/javascript"
function changeImageR6()
{
    var freebed6 = "<?php echo $freebedR6; ?>";
    if (freebed6 == 6){
        document.getElementById("img6").src = ".../GreenR6.png";
    }
    else if ((freebed6 < 6) && (freebed6 > 0))
    {
        document.getElementById("img6").src = ".../OrangeR6.png";
    }
    else {
        document.getElementById("img6").src = ".../RedR6.png";
    }
}</script>

<script type="text/javascript"
function room6submit()
{
    var freebed6 = "<?php echo $freebedR6; ?>";
    if ((freebed6 <= 6) && (freebed6 >0 )){
        document.getElementById("form_R6").submit();
    }
    else{
        alert("There is no beds available in this room");
    }
}</script>
var freebed6 = "<?php echo $freebedR6; ?>";
if ((freebed6 <= 6) && (freebed6 > 0)){
document.getElementById("form_R6").submit();
} else{
alert("There is no beds available in this room");
}
</script>

<script type="text/javascript">
function changImageR7()
{
    var freebed7 = "<?php echo $freebedR7; ?>";
    if (freebed7 == 8){
document.getElementById("img7").src = "./GreenR7.png";
    } else if ((freebed7 < 8) && (freebed7 > 0))
    {
document.getElementById("img7").src = "./OrangeR7.png";
    } else {
    document.getElementById("img7").src = "./RedR7.png";
    }
}</script>

<script type="text/javascript">
function room7Submit()
{
    var freebed7 = "<?php echo $freebedR7; ?>";
    if ((freebed7 <= 8) && (freebed7 > 0)){
document.getElementById("form_R7").submit();
    } else{
    alert("There is no beds available in this room");
    }
}</script>

<script type="text/javascript">
function changImageR8()
{
    var freebed8 = "<?php echo $freebedR8; ?>";
    if (freebed8 == 8){
document.getElementById("img8").src = "./GreenR8.png";
    } else if ((freebed8 < 8) && (freebed8 > 0))
    {
document.getElementById("img8").src = "./OrangeR8.png";
    } else {
    document.getElementById("img8").src = "./RedR8.png";
    }
}
<script type="text/javascript">
function room8submit(){
  var freebed8 = "<?php echo $freebed8; ?>";
  if ((freebed8 <= 8) && (freebed8 >0 )){
    document.getElementById("form_R8").submit();
  }
  else{
    alert("There is no beds available in this room");
  }
}
</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html dir="ltr" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title></title>
<style type="text/css">
  .auto-style1 {
    margin-left: 36px;
  }
</style>
<link href="../styles/style1.css" media="screen" rel="stylesheet" title="CSS" type="text/css" />
</head>
<body>
<!-- Begin Container -->
<div id="container">
  <!-- Begin Masthead -->
  <div id="masthead">
    <img alt="" height="66" src="../logo2.png" width="150" /><p>
      <h8>Logged In - <?php echo $staff_name; ?>; </h8>
      <a href="../logout.php">Logout</a>
    </div>
  <!-- End Masthead -->
  <!-- Begin Page Content -->
  <div id="page_content">
    <!-- Begin Sidebar -->
    <div id="sidebar">
      <ul>
        <li><a href="../default.php">Home</a></li>
        <li><a href="../dates/default.php">Make a Booking</a></li>
        <li><a href="../CancelBooking/cancel.php">Cancel Booking</a></li>
        <li><a href="../UpdateBooking/default.php">Update a Booking</a></li>
        <li><a href="../Add%20Staff%20Member/default.php">Add Staff Member</a></li>
        <li><a href="../Reports/reports.php">Reports</a></li>
        <li><a href="../login/default.php">Logout</a></li>
      </ul>
    </div>
  <!-- End Page Content -->
</div>
</body>
</html>
<div id="content">
<title>Ben Lettery</title>
<html dir="ltr" xmlns="http://www.w3.org/1999/xhtml">
<body>
<h2 id="heading">Room Map</h2>
<p id="textstyle">Green are empty rooms, Orange are partially empty and Red are full.</p>
<table style="width:500px" >
<tr><td>
<form id="form_R1" method="post" action="Room1.php" style="width: 165px">
<img name="Room_1" value="1" src="../GreenR1.png" id="img1" onload="changelmageR1()" onclick="room1submit()" height="99" width="162" />
<input name="RoomNum" type="hidden" value="1" />
&nbsp;&nbsp;&nbsp;<label id="smalltext"> <?php if($freebedR1 > 0) echo $freebedR1. " Free Beds"; else echo "0 Free Beds" ?></label>
</form>
</td><td>
<form id="form_R2" method="post" action="Room2.php" style="width: 153px">
<img name="Room_2" value="2" src="../GreenR2.png" id="img2" onload="changelmageR2()" onclick="room2submit()" height="99" width="162" />
<input name="RoomNum" type="hidden" value="2" />
&nbsp;&nbsp;&nbsp;<label id="smalltext"> <?php if($freebedR2 > 0) echo $freebedR2. " Free Beds"; else echo "0 Free Beds" ?></label>
</form>
</td></tr>
<tr><td>
<form id="form_R3" method="post" action="Room3.php" style="width: 152px">
<img name="Room_3" value="3" src="../GreenR3.png" id="img3" onload="changelmageR3()" onclick="room3submit()" height="99" width="162" />
<input name="RoomNum" type="hidden" value="3" />
&nbsp;&nbsp;&nbsp;<label id="smalltext"> <?php if($freebedR3 > 0) echo $freebedR3. " Free Beds"; else echo "0 Free Beds" ?></label>
</form>
</td><td>
<form id="form_R4" method="post" action="Room4.php" style="width: 153px">
<img name="Room_4" value="4" src="../GreenR4.png" id="img4" onload="changelmageR4()" onclick="room4submit()" height="99" width="162" />
<input name="RoomNum" type="hidden" value="4" />
&nbsp;&nbsp;&nbsp;<label id="smalltext"> <?php if($freebedR4 > 0) echo $freebedR4. " Free Beds"; else echo "0 Free Beds" ?></label>
</form>
</td></tr>
</table>
</body>
</html>
</div>
<table>
<thead>
<tr>
<th>Room</th>
<th>Free Beds</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>
References


BEN LETTERY CONNEMARA HOSTEL
Development of a Management Information System

Technical Manual
<table>
<thead>
<tr>
<th>NO.</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>About</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>Technical Environment</td>
<td>2</td>
</tr>
<tr>
<td>3.</td>
<td>Deployment</td>
<td>3</td>
</tr>
<tr>
<td>3.1</td>
<td>Database Set-Up</td>
<td>3</td>
</tr>
<tr>
<td>3.2</td>
<td>PHP Set-Up</td>
<td>4</td>
</tr>
<tr>
<td>4.</td>
<td>Testing</td>
<td>6</td>
</tr>
<tr>
<td>5.</td>
<td>Aesthetics</td>
<td>7</td>
</tr>
<tr>
<td>6.</td>
<td>System Back Up</td>
<td>8</td>
</tr>
</tbody>
</table>
1. OVERVIEW

1.1. Introduction

The user manual details how to navigate the user interface. This manual offers instructions for primary set up, alterations and extensions to the more complex back end of the system.

1.2. About

This application was produced by Sarah Keane for Ben Lettery Connemara Hostel.
2. **TECHNICAL ENVIRONMENT**

The bulk of this system was developed using Microsoft Expression Web 4. This is a free software available to download at [http://www.microsoft.com/en-IE/download/details.aspx?id=36179](http://www.microsoft.com/en-IE/download/details.aspx?id=36179).

This application is made up of four different languages:

- PHP, for building the web pages.
- SQL, for querying the database.
- JavaScript, for improving the appearance and validation rules.
- HTML, for building the web pages.
- CSS, for the aesthetics of the user interface.
3. **DEPLOYMENT:**

In order for the client to run the system correctly the server needs to be set up correctly. A XAMPP (Windows, Apache, MySQL, PHP) software bundle is what was used in this project for developing the system and it would be recommended to the client to ensure correct installation of the system. It downloads and sets up all the features that are necessary in just one step.

- Windows is the operating system.
- Apache is the web server.
- MySQL is a relational database management system.
- PHP is a server-side website development language.
- PhpMyAdmin is intended to handle the administration of MySQL with the use of a web browser.

3.1 **Database Set-Up**

PhpMyAdmin where the database will be created. The following tables should be created in the database. It is necessary to use the exact name which are used below. If they are not used a large amount of modifications will be necessary.

**Table Name: Reservations**

<table>
<thead>
<tr>
<th>res_num</th>
<th>arrival_date</th>
<th>departure_date</th>
<th>staff_ID</th>
<th>cust_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int(11) AUTO INCREMENT</td>
<td>date</td>
<td>date</td>
<td>Int(11)</td>
<td>Int(11)</td>
</tr>
</tbody>
</table>

**Table Name: Booking**

<table>
<thead>
<tr>
<th>id</th>
<th>room_num</th>
<th>bed_num</th>
<th>res_num</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int(11) AUTO INCREMENT</td>
<td>Int(11)</td>
<td>Int(11)</td>
<td>Int(11)</td>
</tr>
</tbody>
</table>

**Table Name: customers**

<table>
<thead>
<tr>
<th>cust_id</th>
<th>cust_name</th>
<th>age</th>
<th>nationality</th>
<th>email</th>
<th>source</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int(11) AUTO INCREMENT</td>
<td>Varchar(50)</td>
<td>Int(11)</td>
<td>Varchar(20)</td>
<td>Varchar(20)</td>
<td>Varchar(20)</td>
<td>text</td>
</tr>
</tbody>
</table>

**Table Name: rooms**

<table>
<thead>
<tr>
<th>room_id</th>
<th>bed_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int(11)</td>
<td>Int(11)</td>
</tr>
</tbody>
</table>
Once these have been fully set up, several modifications need to be made to the Connection1.php page, this will include the security details of the new database.

- Connection1.php must be updated to mirror the new database, its host, username, database name and password.

```php
<?php
    $mysql_host = 'localhost';
    $mysql_user = 'root';
    $mysql_pass = '';
    $mysql_db = 'fyp';
    $test_db = 'true';
    if(mysql_connect($mysql_host,$mysql_user,$mysql_pass) && @mysql_select_db($mysql_db)){
        $test_db = 'true';
    }else{
        die('Database connection error. ');
    }
?>
```

### 3.2. PHP Set-Up

Select all the files and place all of them on the server. Click on where the XAMPP folder is stored.

There are two options:

- Place the files in the `htdocs` folder.
- Open the `htdocs` folder and then the `mysite` folder and place the files in here.

The second option will keep the folder neater, especially if the user has more than one website. From here testing must now be carried out.

#### Table Name: login

<table>
<thead>
<tr>
<th>Staff ID</th>
<th>Staff Name</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int(11) AUTO INCREMENT</td>
<td>Varchar(11)</td>
<td>Varchar(11)</td>
</tr>
</tbody>
</table>
4. TESTING

Testing must be carried out on the system to ensure that everything is working together.

Before testing can begin it will be necessary for the user to make themselves a staff member in the database, allowing them to log in. To begin open the XAMPP application and click the start button on Apache and MySQL. (See Figure D.1.1)

![Diagram of XAMPP](image)

Figure 4.1 – Diagram of XAMPP

Then the user must open phpMyAdmin and go into the database that is being used. Within this database click on the SQL tab at the top of the page. Into this write the following query:

```
INSERT INTO `database_name`.`login` (`Staff_ID`, `Staff_Name`, `Password`) VALUES (NULL, Joe, '1234');
```

This will give the user Joe entry to the system with the password 1234.

Test 1: To Determine Whether PHP and MySQL are Communicating

- Open the webpage on Google Chrome. http://mywebsite.localdomain/mysite/login/default.php If you followed option two in the set-up PHP page then you can use this web address otherwise leave out ‘mysite’.
- Enter staff name and password. Click login. You should now be successfully logged into the system.
- Click on the ‘Make a Booking’ link. Go through the process and create a booking.
- When directed back to the home page see if the booking is on the page. It should be on the bottom of the page within ‘Customer due to Arrive’ section.
- Cancel the booking.
Once there is a connection between the database and the user interface the system can be thoroughly tested. In the code there following line:

```php
ini_set('display_errors', 'Off');
```

If the user feels that there are errors this line can be commented out by putting '// in front of it and the error will be displayed. This line is to stop error appearing on the screen for variables that are not being used in a page.
5. AESTHETICS

CSS (Cascading Style Sheets) were used to create the site template. Using an application like Microsoft Expression Web 4 will make any adjustments easier. Any class (‘X’) or ID (‘.’) has various style options available, altering font, colour, images, alignment, borders and much more.

All the pages of the site are linked to an external style sheets, these sheets are located in the style folder. If any modifications are needed then the changes only need to be made in one simple step to change the layout of the entire system.

For example the heading on each page is defined by the code below. The colour, font, and layout of the heading is under the #heading section.

```css
#heading
{
  margin-left: 10px;
  color: blue;
  font-size: 24px;
  font-style: oblique;
  text-align: left;
}
```

For example you could change the font size by changing:

```
font-size: 36px;
```

Similarly for the sidebar the code is shown below, modifications can be made including the background colour, in this case the colour is defined by a code, but it also will allow you to put in words, for example the background colour could be changed to green by changing:

```
Background-color: #Green;
```

```css
#sidebar {
  float: left;
  width: 150px;
  background-color: #d3d3d3;
  position: relative;
  clear: both;
  margin-left: 10px;
  display: inline;
}
```
7. SYSTEM BACK UP

Backing up the system from PhpMyAdmin is quick and easy. The following steps are necessary:

- Open PHP and select the *Export* tab.
- Now select the tables that are to be backed up, or select the whole database.
- Choose the favoured format for the output from the list provided. (MS Excel is the most recognised.)
- Select the checkbox for ‘Put columns names in the first row’.
- Label the back up using the ‘Save As’ and press the ‘Go’ button.
TRINITY COLLEGE DUBLIN
Management Science and Information Systems Studies
Project Report

BEN LETTERY CONNEMARA HOSTEL
Development of a Management Information System

User Manual
<table>
<thead>
<tr>
<th>NO.</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>About</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>Logging into a System</td>
<td>2</td>
</tr>
<tr>
<td>3.</td>
<td>Making a Booking</td>
<td>3</td>
</tr>
<tr>
<td>4.</td>
<td>Cancelling a Booking</td>
<td>6</td>
</tr>
<tr>
<td>5.</td>
<td>Updating a Booking</td>
<td>8</td>
</tr>
<tr>
<td>6.</td>
<td>Adding a Staff Member</td>
<td>10</td>
</tr>
<tr>
<td>7.</td>
<td>Reports</td>
<td>11</td>
</tr>
</tbody>
</table>
OVERVIEW

1.1. Introduction

This user Manual accompanies the hostel management information system developed on behalf of the staff in Ben Lettery Hostel. Instructions on how to use each type of the systems functions, from a user perspective, are provided in this manual.

1.2. About

This application was produced by Sarah Keane for Ben Lettery Hostel Connemara.
2. LOGGING INTO THE SYSTEM

The user will be directed to the 'Login' page when the application is opened.

![Login Page]

On the login page enter the username and password. If either the username or password is incorrect an error message will be displayed and the user will have to contact the administrator by emailing Joseph Vale at joseph.anthony.vale@citi.com.

The 'Home' page then opens. On the home page a list of the customers due to arrive appears in a table on the bottom. There are two date input boxes on the page. Here the user can enter dates and click the 'Search Dates' button. In the table below a list of guests, nationality and the number of beds booked will appear. If no dates are entered into the input boxes the table below will display the guests who are due to arrive today.
In Figure 2.2 the ‘Home’ page can be seen. The side menu will be displayed on every page throughout the user interface. On the top of the page the staff member who is logged in should see their own name displayed. To logout of the system it is necessary to click the ‘logout’ button on the top right hand side of the page or the logout on the side menu. Both of these options will be available on every page.
3. **MAKING A BOOKING**

Click on the 'Make a Booking' tab in the menu bar. This will direct the customer to the 'Search Dates' page (See Figure 3.1). On this page the user will enter the arrival and departure dates for the guests to be booked in.

![Figure 3.1 – Search Dates](image)

Once the dates are entered press the ‘Search Dates’ button and the ‘Rooms Page’ containing a heat map of the rooms will appear.

![Figure 3.2 – Heat Map of Rooms](image)

The heat map is shown is Figure 3.2. When the room is red it means there are no beds available in that room. The amount of beds left in each room is displayed under each room. In this example, the user will not be able to click on ‘Room 1’ instead an error message will pop up displaying ‘There are no available beds in this room’. The user can click on any other...
rooms depending on how many guests they want book in. When a Room is clicked on it will open the page belonging to that specific room.

In this example if ‘Room 2’ is clicked on it will open up what can be seen in Figure 3.3. As seen in Figure 3.2 Room 2 has only one bed available. So when it is opened there are two beds with only one green bed meaning that this is available. The checkbox at Bed 2 can be selected. Whereas if the checkbox at Bed 1 is selected it will be immediately unselected because the room is already taken. When you have selected the beds click the ‘Complete Booking’ button.

![Room 2 Heat Map](image)

**Figure 3.4 – Heat Map of Beds in Room**
This will open the ‘Reservation Confirm’ page. On this page check if the reservation details are correct including arrival and departure date, room number, bed number and staff member.

![Image](image.png)

**Figure 3.5 – Confirming Reservation/Enter Guest Details**

Then click the ‘New Customer’ checkbox. The only time you don’t click this checkbox is when the guest’s details are already displayed. The mandatory fields are customer name and email. The other values can be skipped if the guest does not wish to give these details. The comment box gives the user the opportunity to enter in any further information about the customer for example, if the customer will pay on arrival. The user has two options to press ‘Book’ which will book the guest into the hostel for those dates.
When the user clicks ‘Book’ the ‘Invoice’ form will be displayed as shown in Figure 3.6. In this form the user has three options. Clicking ‘Finish’ will direct the user back to the ‘Home’ page. Clicking ‘Print’ will also allow the user to print out the page. To add more rooms it will be required to press the ‘Add More Rooms’ button. This will direct the user back to the ‘Search Dates’ page.

### 4. CANCELLING A BOOKING

To cancel a booking the first step is to click the ‘Cancel a Booking’ tab on the side menu bar.

![Figure 4.1 – Select Guest for Booking Cancellation](image)

Figure 4.1 displays the page which will be displayed after clicking the ‘Cancel a Booking’ tab. Click on the drop-down box and select the customer whose booking will be cancelled. To proceed click the ‘Search’ button.

Once the ‘Search’ button is pressed a page opens containing the names of customers with the name that was chosen (See Figure 4.2). This is for when there are two people with the same name. A guest id must be chosen from the dropdown menu. When the ‘Search’ button is pressed on this page it opens what can be seen on Figure 4.3.
On this page the guest details are displayed, looking at Figure 4.2 it is clear that Gerry has one room booked for the night. When the user knows which booking to cancel they can select the reservation number from the dropdown box. There are two options, one to ‘Confirm Cancellation’ which will cancel the booking and one to ‘Cancel’ the cancellation where the user will be directed to the ‘Home’ page.
5. UPDATING A BOOKING

Similar to cancelling a booking, you click on the ‘Update a Booking’ tab. Here use the dropdown box to select the customer whose details you want to update. See Figure 5.1, press the ‘Find Customers Details’

![Figure 5.1 - Select a Guest](image)

This directs you to a page containing guests that may contain the same name. From Figure 5.2 it shows there is only one Gerry Fallon, but if there were two it would be necessary you choose the guest ID in the dropdown box based on the ID in the table.

![Figure 5.2 – Select a Guest](image)

The next step is to update the customer’s personal details. The options can be seen in Figure 5.3. All the text boxes in this can be updated except the Guest ID. To update a booking it is necessary to click the ‘Update Details’ button. To go back press the ‘Cancel’ button. This will not make any changes and direct back to the ‘Home’ page.
To update the booking details for a guest, you choose the reservation number from the dropdown box. This reservation number is linked to the number seen in the table (See Figure 5.4). There are two options here, one to ‘Update Dates’ which will update the reservation and direct back to the ‘Search Dates’ page. The other is to click the ‘Finish’ button, this will direct you back to the ‘Home’ page making no changes.

Figure 5.4 – Updating a Booking

Gerry Fallon details have been updated.

Click the ‘Update Dates’ button if you would like change the dates for this guest. But first select the reservation number you wish to change.

If you have made all the necessary changes then click Finish.

Figure 5.4 – Updating a Booking
6. **ADDING A STAFF MEMBER**

Adding a staff member involved using the user entering their name or if there name already exists a unique name. If the name already exists the user will be notified by an error message. Then enter a password that will be used to log into the system from now on. In the confirm password box it is necessary to enter the same password as before. If this is not correct the password will not be created.

![Figure 6.1 – Adding a Staff Member](image)

As can be seen in Figure 6.1 there is a ‘Reset Password’ button. This can only be used by managers with access to the Master Password. It is for when someone forgets their password.

![Figure 6.2](image)

The user must enter the user name they have been using and confirm their new password by entering it once in the new password box and once in the confirm password box, the manager must then give his authorization using the Master Password.
7. **REPORTS**

This section displays reports based on age, nationality, staff and overall guests. To view a report you simply click one of the buttons as shown in Figure 7.1.

![Figure 7.1 - Reports](image)

The Age Profiles report is seen in Figure 7.2. The user must enter the date ranges in which they want to view the report for. A bar graph is displayed showing the number of guests from 20/03/2014 to 06/03/2014. If you hover over the graph it will display the number of guests in each age range.
The other reports work in the same way so once they are clicked on the dates need to be entered and then press the ‘Search’ button. There are also tables displaying the people staying in the hostel.

Figure 7.2 – Age Profile of Guests