****

**MASENO UNIVERSITY**

**SCHOOL OF COMPUTING AND INFOMATICS**

**DEPATMENT OF COMPUTER SCIENCE**

**CCS 304: GROUP PROJECT II**

**PROJECT TITLE: DELUXE ONLINE HOTEL MANAGEMENT SYSTEM**

**REGISTRATION NUMBER NAME**

**CI/00013/014 SOLOMON ALEKA MURENYI**

**CI/00025/014 EVELYN ONGIRI**

**NAME OF SUPERVISOR: MICHAEL ADONGO**

**A project report submitted in partial fulfillment of the requirement for the Bachelor of Science Degree (BSc.) in Computer Science**

# **DECLARATION**

**We Solomon Aleka and Evelyn Ongiri** do hereby declare that this project report is our original work and to the best of our knowledge, it has not been presented to any other examination body.

Signature: ……………………... Date: ……………………

Signature: …………………...... Date: ……………………

This project report is hereby presented for examination with the approval of the project supervisor.

**Name: Mr. Michael Adongo**

**Signature: ----------------------------- Date: ------------------------**

# **ABSTRACT**

Deluxe Online Hotel Management System is a web based hotel management system that manages a number of hotels automatically ensuring that clients are able to reserve rooms remotely and duly updated on the relevant details about the hotels they wish to reserve rooms from. System was to designed in such a manner to ensure a user must create an account before he/she can reserve a room using this web based application. Once a user has account, he/she logs in to the system by supplying correct username and password. Upon authentication, the user is directed to the homepage where click the menu item pertaining room booking. The user must view the available rooms before reserving a room. All the booked rooms are not available to the clients until later availed by the system administrator. Booking involves submitting a fully filled form with all the relevant booking details. Upon successful booking, the user will be directed to a page that explicitly shows the reservation details as well as concise information on how payment is to be done. After the user makes payment, he/she receives confirmation from the concerned hotel either through a Short Message Service (SMS) or a phone call. All the operations in the system are well coordinated to increase efficiency and effectiveness.

# **ACKNOWLEDGEMENT**

First and foremost, we would like to thank our Group Project supervisor Mr. Michael Adongo. He has been of great help in completing this report. He was very keen and promptly corrected us whenever we did something wrongly or failed to comply by the set standards. Apart from that he consistently shed light on new functionality we should include in our system and also the improvements we should make to the existing functionalities.

Secondly, we would like to thank our parents. They were key in handling the financial aspect to see that the project is completed on time. Moreover, they provided their support and conducive environment apt for serious working during the holidays.

Thirdly, we would also like to thank the Faculty of Computing and Informatics for providing such a conducive environment for learning and conduction of adequate research. The readily available wireless connection (eduroam) simplified the research process.

Thirdly, we would also like to thank the experts who provided their advice on various aspects of the report. We remain greatly indebted to them. Their contributions in the project report have been of great value to us.

**TABLE OF CONTENTS**

[**DECLARATION** ii](#_Toc485896508)

[**ABSTRACT** ii](#_Toc485896509)

[**ACKNOWLEDGEMENT** ii](#_Toc485896510)

[**CHAPTER ONE: INTRODUCTION** 2](#_Toc485896511)

[1.1 Background of the study 2](#_Toc485896512)

[1.2 Problem statement 2](#_Toc485896513)

[1.3 Objectives of the study 2](#_Toc485896514)

[1.4 Significance of the study 2](#_Toc485896515)

[1.5 Scope and limitation of the study 2](#_Toc485896516)

[1.6 Basic assumptions 2](#_Toc485896517)

[**CHAPTER TWO: LITERATURE REVIEW** 2](#_Toc485896518)

[2.1 Introduction to literature review 2](#_Toc485896519)

[2.2 Hotel revenue management 2](#_Toc485896520)

[**CHAPTER THREE: METHODOLOGY** 2](#_Toc485896521)

[3.1 Introduction to methodology 2](#_Toc485896522)

[3.2 Database management systems 2](#_Toc485896523)

[3.3 Programming languages 2](#_Toc485896524)

[3.3.1 Hypertext Preprocessor (PHP) 2](#_Toc485896525)

[3.3.2 Hypertext Markup Language(HTML5) 2](#_Toc485896526)

[3.3.3 Cascading Style Sheets (CSS) 2](#_Toc485896527)

[3.3.4 JavaScript 2](#_Toc485896528)

[3.3.5 Ajax 2](#_Toc485896529)

[**CHAPTER FOUR: ANALYSIS AND DESIGN** 2](#_Toc485896530)

[4.1 Entity Relationship Diagram (ERD) 2](#_Toc485896531)

[4.2 Data Flow Diagram 2](#_Toc485896532)

[4.3 Use Case Diagram 2](#_Toc485896533)

[4.4 Activity Diagram 2](#_Toc485896534)

[4.5 System Flow Chart 2](#_Toc485896535)

[4.6 Database 2](#_Toc485896536)

[4.6.1 Database purpose 2](#_Toc485896537)

[4.6.2 Database format 2](#_Toc485896538)

[4.6.3 Database design 2](#_Toc485896539)

[**CHAPTER FIVE: IMPLEMENTATION** 2](#_Toc485896540)

[5.1 Testing 2](#_Toc485896541)

[5.1.1 Purpose of testing 2](#_Toc485896542)

[5.1.2 Types of testing applied 2](#_Toc485896543)

[5.1.3 Test plan and results tables 2](#_Toc485896544)

[5.2 Coding and debugging 2](#_Toc485896545)

[5.3 Graphical User Interfaces 2](#_Toc485896546)

[5.3.1 Front-End GUI 2](#_Toc485896547)

[5.3.2 Back-End GUI 2](#_Toc485896548)

[**CHAPTER SIX: CONCLUSIONS AND RECOMMENDATIONS** 2](#_Toc485896549)

[6.1 Overall conclusion 2](#_Toc485896550)

[6.2 Recommendations 2](#_Toc485896551)

[6.3 References 2](#_Toc485896552)

# **CHAPTER ONE: INTRODUCTION**

## 1.1 Background of the study

Most hotels are managed manually. This manual management of hotels is often coupled with a lot of problems some of them being lack of proper records pertaining the available rooms or already booked rooms at any given moment, redundancy of data which brings about replication, time wastage in that a client who intends to reserve a room in a given hotel is forced to travel to the actual location for him/her to make the reservation. Apart from the listed shortcomings, hotels are not able to properly publicize themselves among the general public and this consequently reduces the amount of income the concerned hotels are able to generate. Deluxe Online Hotel Management System was able to address these problems to a great extent. Using the system, clients were able to reserve rooms as long as they are connected to an internet connection thus evading time wastage and ensuring economical use of resources like minimizing the travelling expenses the client has to incur to and from the hotel just to reserve room(s). Moreover, there was proper record keeping using the system. All data was stored in a well-defined relational database. Each table in the database had a primary key thus ensuring there is no data redundancy. Manual records often required replacing which increased cost of running operations. Databases can store huge amounts of data and still run operations efficiently.

## Problem statement

In light of the problems associated with manual management of hotels, there was need to automate the whole process of room reservation. Deluxe Online Hotel management system is a system that enabled users to check availability of rooms through online browsing and then afterwards reserve rooms. This system was very useful to all especially time time-conscious clients. For instance, business people don’t have sufficient or ample time for manual room booking systems and therefore they comfortably relied on Online Hotel Management Systems. The factor that sparked the development of the system was the need to address the issue of users located far from the hotel being unable to reserve rooms at the comfort of their homes. Instead they are forced to travel to the actual locations of the desired hotels. This was solved using Deluxe Online Hotel Management System.

## Objectives of the study

The main objective of the project was to come up with an Online Hotel Management System that will automate the processes involved in room reservations.

Some of the specific objectives of the project were:

* To reduce the amount of time taken by clients while reserving rooms from desired hotels. This was possible because clients were able to reserve rooms without having to physically travel to the location of desired hotel.
* To increase efficiency in storage of all data pertaining to room reservation thus evading redundancy and incorrect records. Data in such a system was characterized by consistency.
* To lower the cost of room reservation. The fact that clients were not incurring travelling expenses meant a decline in the cost of booking a room.
* To develop a user-friendly GUI that will enable clients to perform activities such as such as login to the system, creating accounts, record updating, maintaining and searching.
* To enhance security in data handling. This was possible in that before a user was granted access to the system he/she had to be authenticated and the credentials validated to ensure they are genuine. Moreover, clients were not granted full privileges to ensure they won’t be able to alter crucial booking data. Only the system administrator has full privileges.

## 1.4 Significance of the study

We designed the Hotel Management System so as to the automate operations of a hotel. This system was useful to those entrusted or charged with the task of advertising a given hotel. The management was able to update details about hotel packages, update room details, update the list of available rooms, handle booking date among other operations crucial in running a hotel. We can boldly state that the final software was of great significance. Convenience was very key here. A client was in a position of booking a room at the comfort of his/her home as compared to going to the actual location of the hotel to book. This was possible as long as there was internet connectivity. The following points give a detailed description on why the Online Hotel Management was significant:

* Performance. For the existing manual management, records were supposed to be manually handled for all activities. The manual handling of the records was time consuming and also susceptible to errors. To improve organization and performance of Hotel Management, the computerized Online Hotel Management System was to be developed and consequently deployed. The developed was fully computerized and user friendly such that any duly registered client can view the relevant data about the various hotels managed by the system.
* Efficiency. The basic reason for developing the Hotel Management System was to increase efficiency. The system was efficient such that whenever a client submitted his/her details the relevant table(s) in the database was updated automatically.
* Control and Security. The complete control of the Hotel Management System was under the hands of the system administrator. He/she was able to update the available rooms, add new hotels, deregister clients depending on the circumstance among other essential operations. Privileges granted to clients was limited to prevent them from altering crucial system data or information. Generally, clients were limited to viewing the available data, submitting their requests and modifying their records. Illegal access was strongly prohibited.

## 1.5 Scope and limitation of the study

The Hotel Management System was a web based application meaning it was only functional online. No manual services were offered. The Hotel Management System managed a chain of hotels which were situated in different geographical but bore more or less the same features. In short, the application is not limited to a single hotel. Some of the limitations of the system were:

* A client was able to reserve only a single room at a time.
* A client was able to use the system only when connected to the internet.
* The system never specified the maximum number of occupants allowed per room.
* The system managed only a few selected hotels in Kenya. But later the scope of the system can be expanded to manage more hotels within the East Africa region.

## Basic assumptions

Some of the basic assumptions that were made while designing the system were:

* A user who checked in and checked out on the same day paid an amount equivalent to reserving a room for a single day.
* Users who booked rooms using the system had Safaricom sim cards to facilitate payment via M-pesa.
* A user who booked a room and failed to confirm within a period of 24 hours, then the administrator automatically availed the room for booking.

# **CHAPTER TWO: LITERATURE REVIEW**

## 2.1 Introduction to literature review

Yield management, a method for profitably managing capacity, has gained widespread acceptance in the hotel industry. Yield management is a method which helps hotels sell the right room to the right type of customer, at the right time and for the right price.1 Yield management guides the decision of how to allocate rooms to available demand in such a way as to maximize profit or revenue. The problem then becomes one of determining how much to sell at what price and to which market sector. The application of yield management has been most effective when applied to industries with relatively fixed capacity, predictable demand, perishable inventory, an appropriate cost and pricing structure and time-variable demand. 1'2 Yield management providers and users commonly tout revenue improvements of 2-7%.

Hotel yield management research has focused on room sales to individuals (transient business), and has used the group forecast as given. Yield management systems rely on accurate transient and group forecasts as inputs. Transient forecasts are used to determine the rate and avail- ability restrictions, while group forecasts are used to calculate the number of rooms available to sell to transient guests. If the group forecast is inaccurate, the number of rooms available for sale will be inaccurate, and the recommendations of the yield management system may lead to poor decisions. The intent of this research was to examine the extent of group forecast error and to identify factors associated with accurate group forecasts.

## 2.2 Hotel revenue management

While revenue management is very well developed both as a theoretical framework and a business practice in the hotel industry, it has not received enough attention as way of increasing efficiency. Research in hotel revenue management, in particular, is fragmented and lags significantly behind the RM practice in the field. In this regard, the aim of current paper is to critically evaluate contemporary hotel RM research, to identify the gaps in literature and provide directions for future research. The review is structured around the elements of hotel’s RM system and the stages of the RM process. It is based on publications (articles in academic journal, books and monographs) published predominantly in the last 10 years. The practical issues of RM remain beyond the scope of the paper, although it should be noted that the RM practice in the major hotel chains is sometimes better developed that the respective academic literature. When the customer places a booking request, it is registered by the hotel’s RM system. The latter consists of four structural elements (data and information, hotel revenue centres, RM software and RM tools), the RM process and the RM team. The operational results from the RM process are the specific booking elements of the particular booking request – e.g. booking status (confirmed/rejected), number of rooms, types and category of rooms, duration of stay, price, cancellation and amendment terms and conditions, etc. The booking details and the operation of the whole RM system influence customer’s perceptions of the fairness of hotel’s RM system and his/her intentions for future bookings with the same hotel/hotel chain. The RM system experiences the constant influences of the external (macro- and micro-) and internal environmental factors in which the hotel operates (e.g. company’s

goals, its financial situation, legislation, competition, changes in demand, destination’s image, or force majeure events among others) and revenue manager’s decisions have to take all these into considerations.

# **CHAPTER THREE: METHODOLOGY**

## 3.1 Introduction to methodology

The Hotel Management System was designed using several case tools. They include Hypertext Markup Language (HTML5), a markup language which was essentially used in designing the components of the application, Cascading Style Sheets (CSS) which was used in controlling the style of the system’s web pages in a simple and easy way, Hypertext Preprocessor (PHP), a server side scripting language and very powerful for making dynamic and interactive web pages, MySQL (Structured Query Language), the most popular Open Source Relational SQL database management system, JavaScript and Ajax.

## 3.2 Database management systems

As a group, we settled on MySQL as the database management system. MySQL is the most popular Open Source Relational SQL database management system. MySQL is one of the best RDBMS being used for developing web-based software applications. A database is a separate application that stores a collection of data. Each database has one or more distinct APIs (Application Programming Interface) for creating, accessing, managing, searching and replicating the data it holds. MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. Some of the reasons that made us choose MySQL over other database management systems are:

* MySQL is released under an open-source license. So, we had to pay nothing use to use it.
* MySQL is a very powerful program in its own right. It handled a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA. As a team we were more interested in PHP as the server side scripting language.
* MySQL works very quickly and works well even with large data sets. Response time of MySQL while fetching records or executing queries was faster when compared to other database management systems.
* MySQL was very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL was customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments and needs.

The MySQL we are relied on was not a standalone software application but rather an application embedded in the Wampserver version 2.4. The components found in this version of Wampserver were:

* Apache version 2.4.4
* PHP version 5.4.16
* MySQL version 5.6.12

## 3.3 Programming languages

## 3.3.1 Hypertext Preprocessor (PHP)

PHP is an HTML embedded scripting language. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The goal of the language is to allow web developers to write dynamically generated pages quickly." This is generally a good definition of PHP. However, it does contain a lot of terms you may not be used to. Another way to think of PHP is a powerful, behind the scenes scripting language that your visitors won't see!

When someone visits your PHP webpage, your web server processes the PHP code. It then sees which parts it needs to show to visitors (content and pictures) and hides the other stuff (file operations, math calculations, etc.) then translates your PHP into HTML. After the translation into HTML, it sends the webpage to your visitor’s web browser. PHP can perform the following:

* Reduce the time taken to create large websites.
* Create a customized user experience for visitors based on information that you have gathered from them
* Open up thousands of possibilities for online tools
* Allow creation of shopping carts for e-commerce websites.

PHP worked very closely with our HTML forms.

## 3.3.2 Hypertext Markup Language(HTML5)

A language that uses a set of codes that tells web browsers how to display pages when they visit a site.HTML is a markup language and not a programming language. An HTML file must have a htm or html file extensions.HTML is used in implementing the static component of a web based application. The component that the user or the site visitor can freely interact with either through navigating or supplying required data. HTML5 is a core technology markup language of the Internet used for structuring and presenting content for the World Wide Web. As of October 2014, this is the final and complete fifth revision of the HTML standard of the World Wide Web Consortium (W3C). The previous version, HTML 4, was standardized in 1997. Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices (web browsers, parsers, etc.). HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

HTML5 introduces a number of new elements and attributes that helps in building a modern website. Following are great features introduced in HTML5.

* New Semantic Elements − These are like <header>, <footer>, and <section>.
* Forms 2.0 − Improvements to HTML web forms where new attributes have been introduced for <input> tag.
* Persistent Local Storage − To achieve without resorting to third-party plugins.
* Web Socket – A next-generation bidirectional communication technology for web applications.
* Server-Sent Events − HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
* Canvas − This supports a two-dimensional drawing surface that you can program with JavaScript.
* Audio & Video − You can embed audio or video on your web pages without resorting to third-party plugins.
* Geolocation − Now visitors can choose to share their physical location with your web application.
* Microdata − This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
* Drag and drop − Drag and drop the items from one location to another location on the same webpage.

## 3.3.3 Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. Although most often used to change the style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.CSS is designed primarily to enable the separation of document content from document presentation, including elements such

as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content, such as semantically insignificant tables that were widely used to format pages before consistent CSS rendering was available in all major browsers. CSS makes it possible to separate presentation instructions from the HTML content in a separate file or style section of the HTML file. Advantages of CSS are:

* CSS saves time− You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
* Pages load faster − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
* Easy maintenance − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
* Superior styles to HTML − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
* Multiple Device Compatibility − Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
* Global web standards − Now HTML attributes are being deprecated and it is being recommended to use CSS. So it’s a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.
* Offline Browsing − CSS can store web applications locally with the help of an offline cache. Using this, we can view offline websites. The cache also ensures faster loading and better overall performance of the website.
* Platform Independence − The Script offer consistent platform independence and can support latest browsers as well.

## 3.3.4 JavaScript

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform. Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser. It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content. The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field. The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server. JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly. Advantages of using JavaScript are listed below:

* Less server interaction − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* Immediate feedback to the visitors − They don't have to wait for a page reload to see if they have forgotten to enter something.
* Increased interactivity − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
* Richer interfaces − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of using JavaScript are:

* Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
* JavaScript cannot be used for networking applications because there is no such support available.
* JavaScript doesn't have any multithreading or multiprocessor capabilities.

## 3.3.5 Ajax

AJAX, is a web development technique for creating interactive web applications. AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script. Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display. Conventional web applications transmit information to and from the server using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server. With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.XML is commonly used as the format for receiving server data, although any format, including plain text, can be used. AJAX is a web browser technology independent of web server software. A user can continue to use the application while the client program requests information from the server in the background. Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger. Data-driven as opposed to page-driven.

# **CHAPTER FOUR: ANALYSIS AND DESIGN**

The principles of system analysis and design are:

* Understanding the problem before you start creating the analysis model.
* Developing prototypes that will enable users understand how human machine interactions will occur.
* Recording the origin of and the reason for every requirement.
* Using multiple views of requirements like building data, functions and behavioral models.
* Working to eliminate ambiguity.

## 4.1 Entity Relationship Diagram (ERD)

This depicts the relationship between data objects. The attribute of each data object noted in the entity-relationship diagram can be described using a data object description. An entity-relationship diagram (ERD)is a model that shows the logical relationships and interaction among system entities. An ERD provides an overall view of the system and a blueprint for creating the physical data structures. The entity-relationship diagrams essentially serve two purposes:

* To provide an indication of how data is transformed as it moves through the system.
* To depict the functions that transform the data as it flows.

**Data objects:** A data object is a representation of almost any composite information that must be understood by the software. By composite information, we mean something that has a number of different properties or attributes. A data object encapsulates data but there is no reference within a data object to operations that act on the data.

**Attributes:** Attributes define the properties of a data object and take on one of three different characteristics. They can be used to; name an instance of data object, describe the instance and make reference to another instance in another table.

**Relationships:** Data objects are connected to one another in a variety of different ways. We can define a set of object relationship pairs that define the relevant relationships.

**Cardinality:** The data model must be capable of representing the number of occurrences of objects in a given relationship. The cardinality of an object relationship pair is:

One-To-One (1:1). An occurrence of object ‘A’ can relate to one and only one occurrence of object ‘B’ and vice versa.

One-To-Many (1: N). One occurrence of object ‘A’ can relate to one or may occurrences of object ‘B’ but an occurrence of object ‘B’ can relate to only one occurrence of object ‘A’.

Many-To-Many (N: N). Several occurrences of ‘A’ can relate to several occurrences of ‘B’ and vice versa.

**Modality:** The modality of a relationship is zero if there is no explicit need forthe relationship to occur or the relationship is optional. The Modality is one if the occurrence of the relationship is mandatory.

Hotel

Customer

Room

Information

Check In

Room

Figure 1: Deluxe Hotel

Hotel

Room Record

Hotel

Figure 2: One-to-One Relationship

Check Out

Room

Customer

Bill Generation

Hotel

Figure 3: Preview of Hotel Operations

## 4.2 Data Flow Diagram

A data flow diagram (DFD) shows how data moves through an information system but does not show program logic or processing steps. A set of DFDs provide a logical model that shows *what* the system does, not *how* it does it. That distinction is important because focusing on implementation issues at this point would restrict your search for the most effective system design. DFDs use four basic symbols that represent processes, data flows, data stores, and entities. Several different versions of DFD symbols exist, but they all serve the same purpose. DFD examples in this report use the **Gane and Sarson** symbol set. Another popular symbol set is the **Yourdon** symbol set. Symbols are referenced by using all capital letters for the symbol name.

ROOM

CUSTOMER

CUSTOMER CHECK OUT

PRINT DEVICE

Figure 4: Hotel Management System

USER

**F** FILE

Figure 6: Record Modification

## 4.3 Use Case Diagram

Use case diagrams are a set of use cases, actors and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. So use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So to model the entire system numbers of use case diagrams are used. So in brief, the purpose of use case diagrams can be as follows:

* Used to gather requirements of a system.
* Used to get an outside view of a system.
* Identify external and internal factors influencing the system.
* Show the interacting among the requirements are actors.

Figure 7: Hotel Management Admin Side

Figure 8: Hotel Management Client Side

Figure 9: Hotel Management Visitor Side

## 4.4 Activity Diagram

Activity diagram describes the flow of control in a system. So it consists of activities and links. The flow can be sequential, concurrent or branched. Activities are nothing but the functions of a system. A number of activity diagrams are prepared to capture the entire flow in a system. Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed. Activity diagrams capture the dynamic behavior of a system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques.

Login

No

Valid Credentials

Yes

Manage Feedback

Manage Client

Manage Cancellation

Manage Rooms

Manage Packages

Manage Advertiser

Manage Reports

Logout

Figure 10: Activity for Admin

Registered

No

Registration

Yes

Login

No

Valid Credentials

Yes

Give Feedback

Select Package

View Hotel Information

Payment

Cancellation

Room Booking

Manage Profile

Logout

Figure 11: Activity for Client

## 4.5 System Flow Chart

Start

View Site

Is Registered

Registration

Login

Is Admin

Manage Room

View Hotel Information

Manage Cancellation

Select Package

Manage Clients

Payment

Manage Feedback

Give Feedback

Manage Packages

Manage Profile

Room Booking

Manage Advertiser

Manage Reports

Cancellation

Logout

Figure 12: System Flow Chart

Stop

## 4.6 Database

## 4.6.1 Database purpose

Every organization has data that needs to be collected, managed, and analyzed. A relational database fulfills these needs. Along with the powerful features of a relational database come requirements for developing and maintaining the database. Data analysts, database designers, and database administrators (DBAs) need to be able to translate the data in a database into useful information for both day-to-day operations and long-term planning.

The relational model is the basis for any relational database management system(RDBMS). A relational model has three core components: a collection of objects or relations, operators that act on the objects or relations, and data integrity methods. In other words, it has a place to store the data, a way to create and retrieve the data, and a way to make sure that the data is logically consistent. The database is the knowledge base of the system and contains all the information about the various hotels available, the various rooms available, details of all the booked rooms, details of location of various hotels, charges for various rooms across the different types of hotels, collection of all comments made by members of the system and a record of all the users who at one point in time successfully logged into the system. In any future commercial developments, the database would be fully populated, but for now we are choosing a small number of hotels and rooms just to test the functionality. The database has been designed to work as efficiently as possible with the PHP code used in the application.

## 4.6.2 Database format

It was decided that it was in the best interest of the project to use MySQL as the database management system. The database did not need to be extremely complicated and by using an advanced alternative such as Postgress, problems arising from the database would have been more likely. MySQL is efficient and copes well with increasing amount of data.

## 4.6.3 Database design

The system comprised of a total of 9 tables each storing specific data. The various tables in the database are shown below.

Table 1: Admin\_Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Data Type | Constraint | Description |
| 1 | Admin\_id | Smallint(5) | Primary Key | Stores admin\_id |
| 2 | Username | Varchar(25) | Not Null | Stores username |
| 3 | Password | Varchar(25) | Not Null | Stores password |
| 4 | User\_type | Varchar(5) | Not Null | Stores user\_type |

Table 2: Booking\_Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Data Type | Constraint | Description |
| 1 | Booking\_id | Smallint(5) | Primary Key | Stores booking\_id |
| 2 | First\_name | Varchar(25) | Not null | Stores first\_name |
| 3 | Second\_name | Varchar(25) | Not null | Stores second\_name |
| 4 | Hotel\_name | Varchar(50) | Not null | Stores hotel\_name |
| 5 | Room\_type | Varchar(30) | Not null | Stores room\_type |
| 6 | Fixed\_charge | Decimal(6,2) | Not null | Stores fixed\_charge |
| 7 | Room\_number | Smallint(6) | Not null | Stores room\_number |
| 8 | Time\_of\_booking | Timestamp | Not null | Stores time\_of\_booking |
| 9 | Check\_in\_date | Text | Not null | Stores check\_in\_date |
| 10 | Check\_out\_date | Text | Not null | Stores check\_out\_date |

Table 3: City\_Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Data Type | Constraint | Description |
| 1 | City\_id | Smallint(6) | Primary Key | Stores city\_id |
| 2 | City\_name | Varchar(30) | Not Null | Stores city\_name |

Table 4: Contact\_Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Data Type | Constraint | Description |
| 1 | Contact\_id | Smallint(5) | Primary Key | Stores contact\_id |
| 2 | First\_name | Varchar(30) | Not Null | Stores first\_name |
| 3 | Email\_address | Varchar(50) | Not Null | Stores email\_address |
| 4 | Phone\_number | Int(10) | Not Null | Stores phone\_number |
| 5 | message | Varchar(200) | Not Null | Stores message |

Table 5: Commission\_Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Data Type | Constraint | Description |
| 1 | Commission\_id | Smallint(6) | Primary Key | Stores commission\_id |
| 2 | Hotel\_name | Varchar(50) | Not Null | Stores hotel\_name |
| 3 | Standard\_single | Decimal(6,2) | Not Null | Stores standard\_single |
| 4 | Superior\_single | Decimal(6,2) | Not Null | Stores superior\_single |
| 5 | Standard\_double | Decimal(6,2) | Not Null | Stores standard\_double |
| 6 | Superior\_double | Decimal(6,2) | Not Null | Stores superior\_double |
| 7 | Family\_room | Decimal(6,2) | Not Null | Stores family\_room |
| 8 | Vip\_room | Decimal(6,2) | Not Null | Stores vip\_room |

Table 6: Hotel\_Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Data Type | Constraint | Description |
| 1 | Hotel\_id | Smallint(6) | Primary Key | Stores hotel\_id |
| 2 | City\_id | Smallint(6) | Foreign Key | Stores city\_id |
| 3 | Hotel\_name | Varchar(50) | Not Null | Stores hotel\_name |
| 4 | Email\_address | Varchar(50) | Not Null | Stores email\_address |

Table 7: Room\_Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Data Type | Constraint | Description |
| 1 | Room\_id | Smallint(6) | Primary Key | Stores room\_id |
| 2 | Hotel\_name | Varchar(30) | Not Null | Stores hotel\_name |
| 3 | Room\_type | Varchar(30 | Not Null | Stores room\_type |
| 4 | Room\_number | Smallint(6) | Not Null | Stores room\_number |

Table 8: User\_Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Data Type | Constraint | Description |
| 1 | Login\_id | Smallint(5) | Primary Key | Stores login\_id |
| 2 | Username | Varchar(25) | Not Null | Stores username |
| 3 | Login\_time | time | Not Null | Stores login\_time |
| 4 | Login\_date | date | Not Null | Stores login\_date |

Table 9: User\_Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Data Type | Constraint | Description |
| 1 | User\_id | Smallint(5) | Primary Key | Stores user\_id |
| 2 | First\_name | Varchar(25) | Not Null | Stores first\_name |
| 3 | Second\_name | Varchar(25) | Not Null | Stores second\_name |
| 4 | Phone\_no | Int(11) | Not Null | Stores phone\_no |
| 5 | Email\_address | Varchar(50) | Not Null | Stores email\_address |
| 6 | Gender | Varchar(6) | Not Null | Stores gender |
| 7 | Username | Varchar(25) | Not Null | Stores username |
| 8 | password | Varchar(25) | Not Null | Stores password |

# **CHAPTER FIVE: IMPLEMENTATION**

## 5.1 Testing

## 5.1.1 Purpose of testing

Software testing is the process used to help identify the Correctness, Completeness, Security and Quality of developed computer software. Testing software is essential in removing bugs and errors from applications.

## 5.1.2 Types of testing applied

We used three different types of testing for our system. These were:

* Visual User Testing
* Automated Unit Test Cases
* End User Testing

Visual user testing meant a member of the group would actively work through the application and follow the set test cases to see if the expected results actually occurred. Any differences throughout any of the steps were noted down, documented and then processed for changing within the program.

The Automated test cases that were created executed certain parts of the system’s

code that to us seemed likely to be prone to errors. These include the spell checking system, and the ordering of illnesses ranked by severity. These test cases were run throughout the creation of the program so that when new code had been implemented we knew that our working code was never affected.

End user testing is an extremely essential part of the testing cycle, and it is here that feedback on our system from real users can be acquired. By testing the system with the type of people who would be using the system in real life, we can get an idea of how our system would be used in the real world. Testing by non-computer literate users also shows how user friendly the system is and will point out improvements for errors that may have initially been passed over.

With all testing carried out it is then necessary to review the results and then act on them. This means carrying out extensive bug fixing on parts that failed, or looking into aspects that haven’t failed but have been of inconvenience. An example of this is slow processing between screens.

## 5.1.3 Test plan and results tables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test module** | **What is tested** | **Test data** | **Expected Results** | **Actual Results** |
| Login Module | -User login | -Supply valid username and password  -Supply invalid username and password | -Display homepage  -Error message | -Homepage displayed  -Request for valid credentials |
| Booking Module | -User Booking  -Modify Booking Data  -Deleting Booking Data | -Fill the form and submit  -Leave any field blank  -Click edit and save  -Click delete and then click ok button | -Success message pops up  -Form cannot be submitted  -Booking modified  -Booking data deleted | -Booking successful  -Alert notifying on unfilled field  -Updated record  -Record no longer available |
| Room Module | -Add Room  -Modify Room  -Delete Room | -Click add a new room and save  -Click edit and save  -Click delete and click ok | -Addition of a room  -Modification of room  -Deletion of a room | -New room added  -Room changed  -Room removed |
| Hotel Module | -Add hotel  -Modify hotel  -Delete hotel | -Click add a new hotel and save  -Click edit and save  -Click delete and click ok | -Addition of a new hotel  -Modification of hotel  -Deletion of a hotel | -New room added  -Room changed  -Room removed |
| Comment Module | -Add user comment  -Delete user comment  -Edit user comment | -Add comment and submit  -Click delete and save  -Click edit and fill form | -Addition of comment  -Deletion of a comment  -Comment edited | -comment added  -Comment removed  -comment updated |
| User Profile | -Registration  -Edit profile  -Delete profile | - Fill form to create account  -Click edit and fill form  -Click delete and then ok | -New account created  -user profile edited  -User profile deleted | -New member created  -User profile updated  -User profile deleted |

## 5.2 Coding and debugging

To read and understand code easily, code must be written consistently to

conform to the known naming conventions. Readers familiar with the naming conventions for coding can easily understand. All the php code must be enclosed within the bounds of <? php---------- *code goes here*-----?> or <? ------*code goes here*----------?>. Each variable in php should be preceded by the dollar sign. Based on that note examples of valid variables are $age, $vehicle. PHP files can contain text, HTML, CSS, JavaScript, and PHP code. PHP code are executed on the server, and the result is returned to the browser as plain HTML. PHP files have extension “. php". In PHP, a user defined function declaration starts with the word "function". A function name can start with a letter or underscore (not a number). Give the function a name that reflects what the function does. Function names are NOT case-sensitive. To let a function, return a value, use the return statement. The proper format should be used to show the various types of css code implementation. External stylesheets are stored in CSS files. With an external style sheet, you can change the look of an entire website by changing just one file! An internal style sheet may be used if one single page has a unique style. Internal styles are defined within the <style> element, inside the <head> section of an HTML page. An inline style may be used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant element. HTML tags are **keywords** (tag names) surrounded by angle brackets. All HTML documents must start with a type declaration: **<! Doctype html>**. The HTML document itself begins with **<html>** and ends with **</html>**. The visible part of the HTML document is between **<body>** and **</body>**. HTML images are defined with the **<img>** tag. There are numerous other tags used in implementing the project.

## 5.3 Graphical User Interfaces

## 5.3.1 Front-End GUI

One aspect of Deluxe Online Hotel Management System’s visual functions is the front-end GUI which the users will be interacting with while performing various operations to the system. The front end was designed using HTML and styled using CSS. The GUI is organised into independent components which can all be accessed from the menu. Styling of all the interfaces has been maintained relatively constant to increase consistency and boost the user’s learnability. Some components are common across all interfaces. They include: header, menu and the footer. A vivid description of the interfaces in the front-end is done below.

**The welcome panel**

This interface is the first panel the user is presented with when the application is started. It contains the logo of our program and information about what the software does. It is merely aesthetic and does not contain any valuable methods, but is a definite requirement for our product to welcome new users as well as those who have already created accounts.

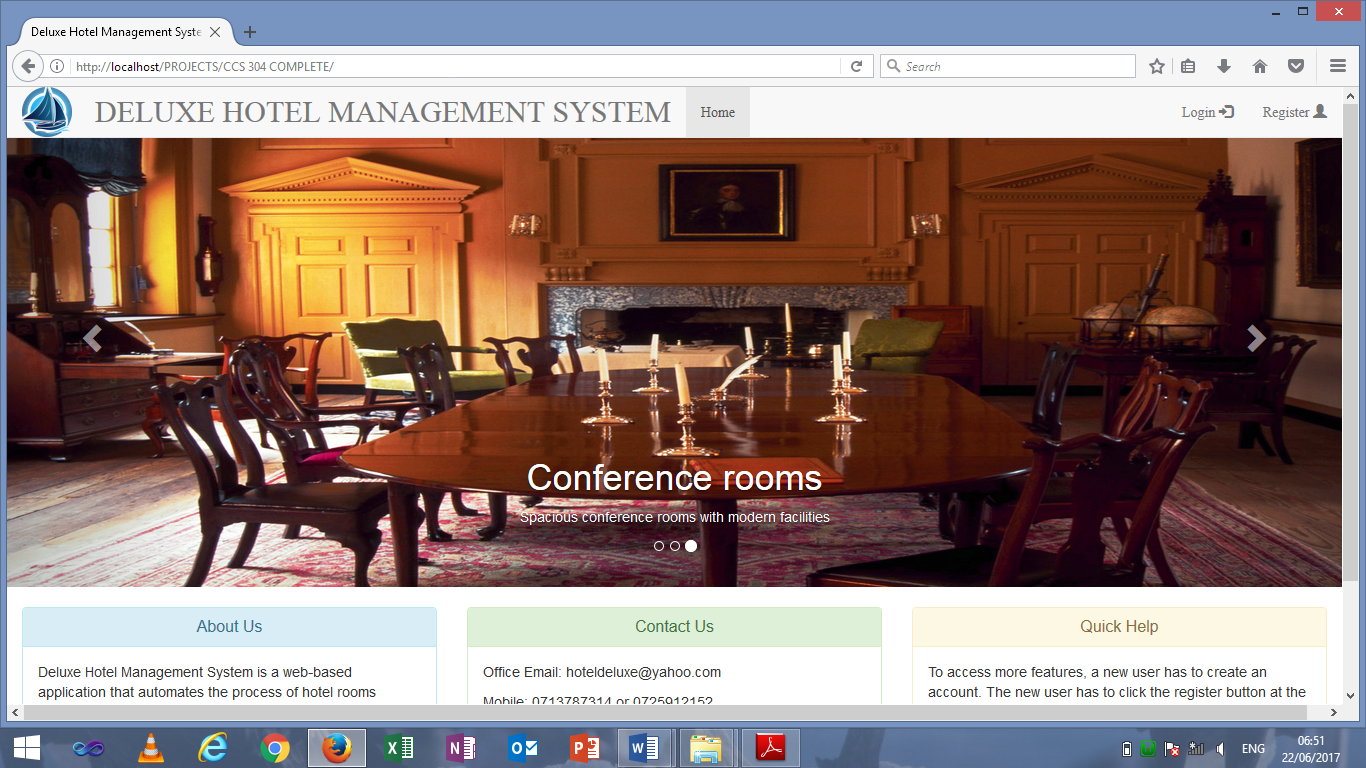


Figure 13: The Welcome Panel

**The login panel**

This interface is basically a form with two input fields and a submit button. The user is expected to enter correct username and password if he/she is a registered user. Upon successful login, the user will be directed to the user homepage.

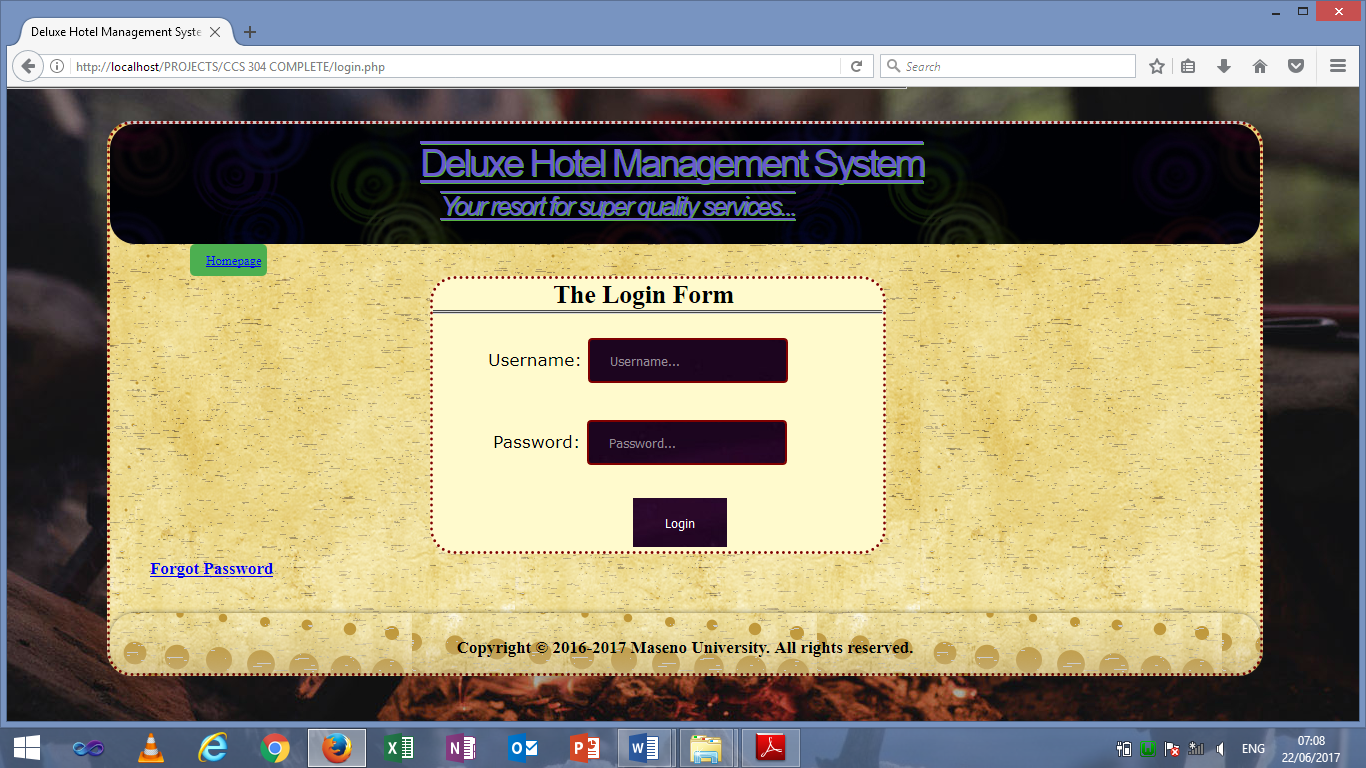


Figure 14: The Login Panel

**The registration panel**

A new user to the system is supposed to fill this form for him/her to create account. The form has several input fields that collect user personal details.



Figure 15: The Registration Panel

**The user homepage panel**

The first page that a user interacts with upon successfully login.

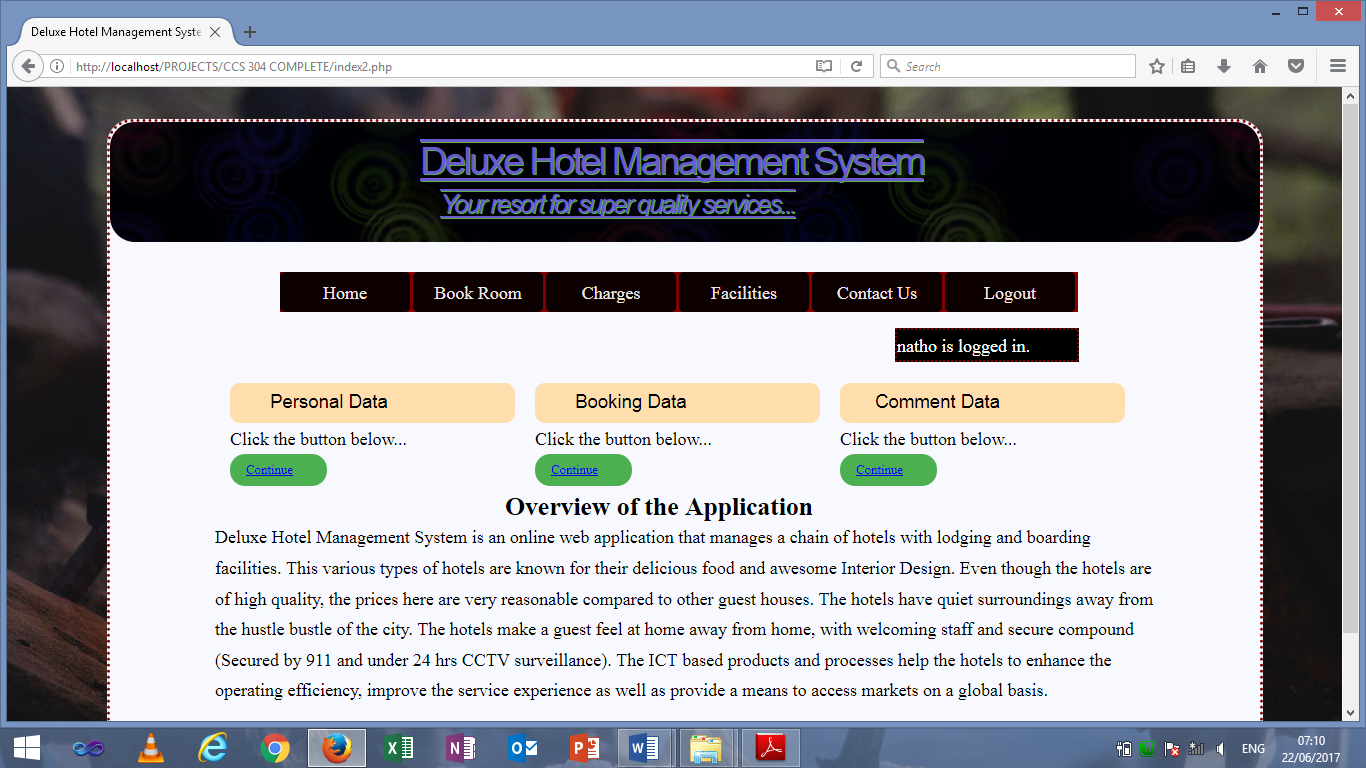


Figure 16: The User Homepage Panel

**The room overview panel**

This interface provides a visual illustration of some of the types of rooms found in the hotel. No detailed information is provided.

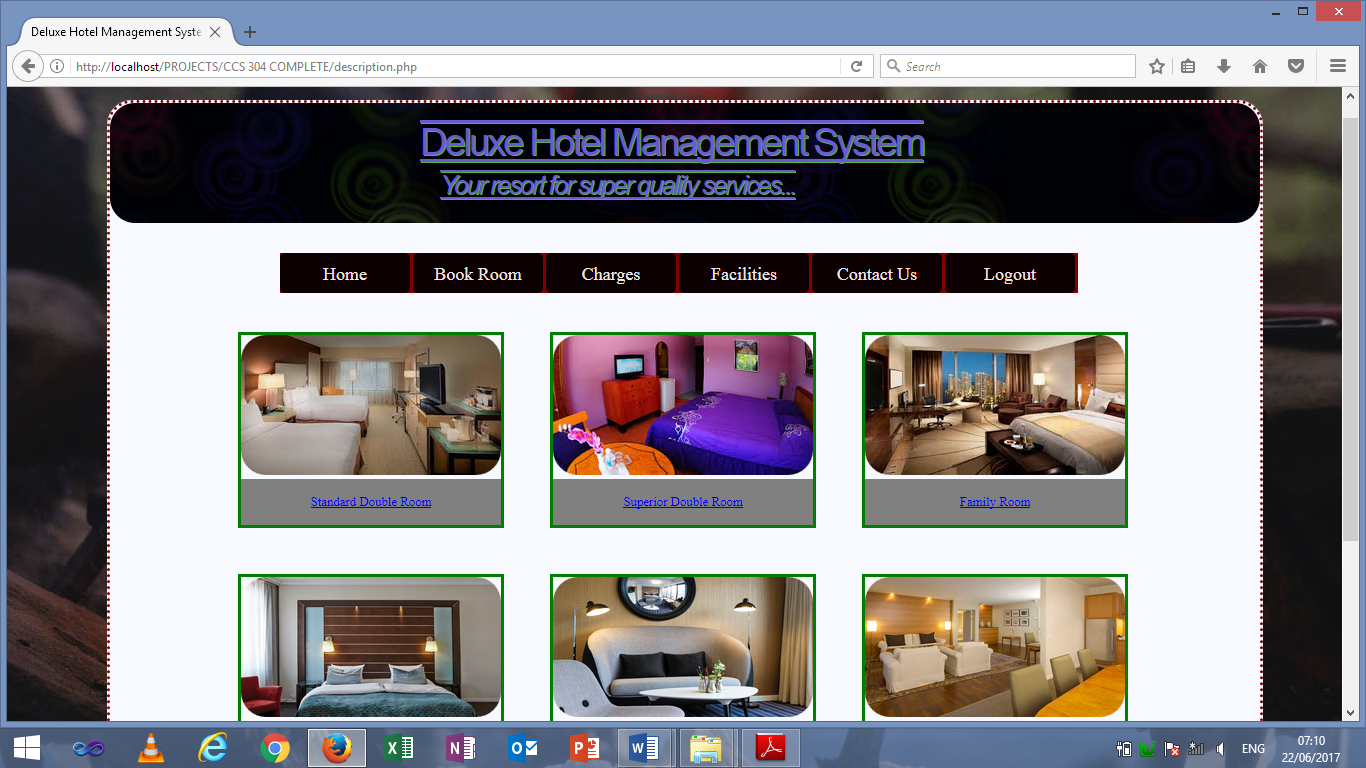


Figure 17: The Room Overview Panel

**The booking form panel**

A user is expected to fill this form by supplying booking details.

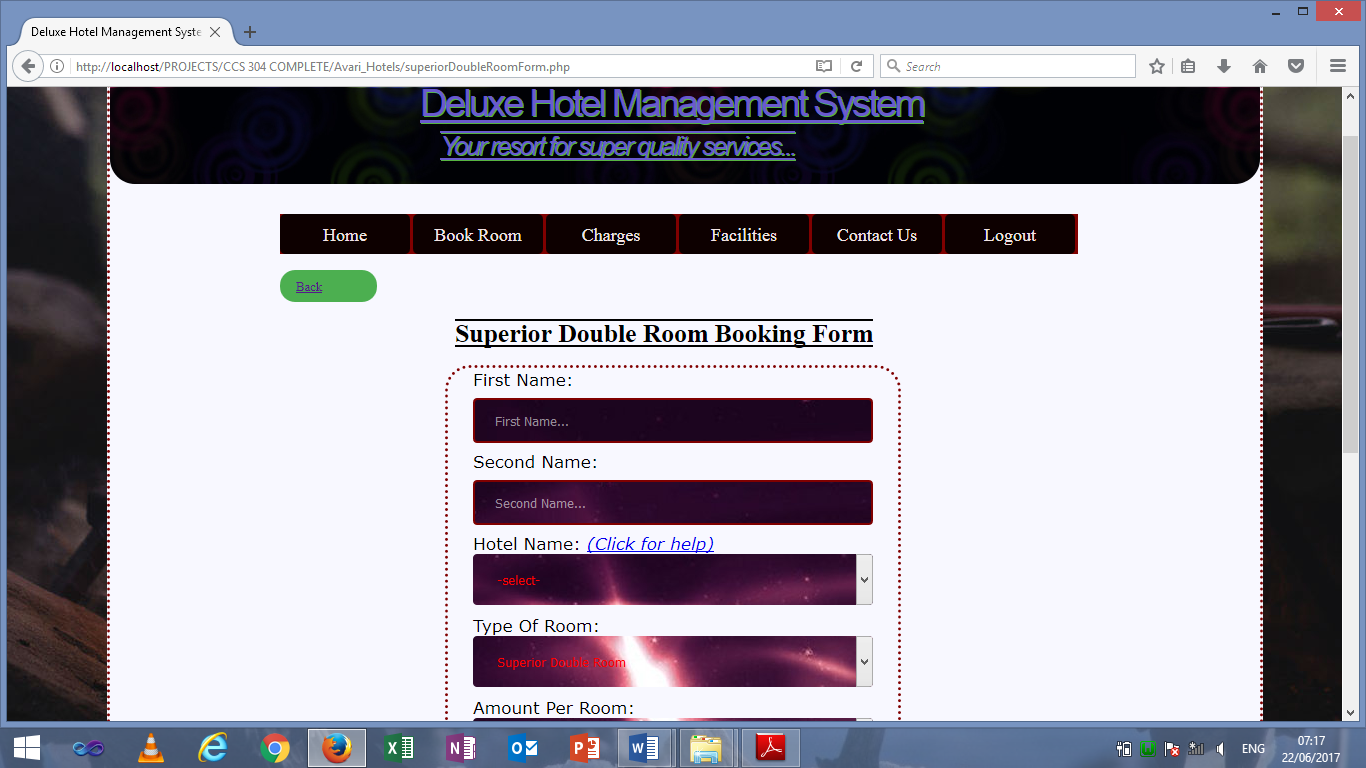


Figure 18: The Booking Form Panel

**The comment panel**

The user is expected to fill this form while adding a comment.

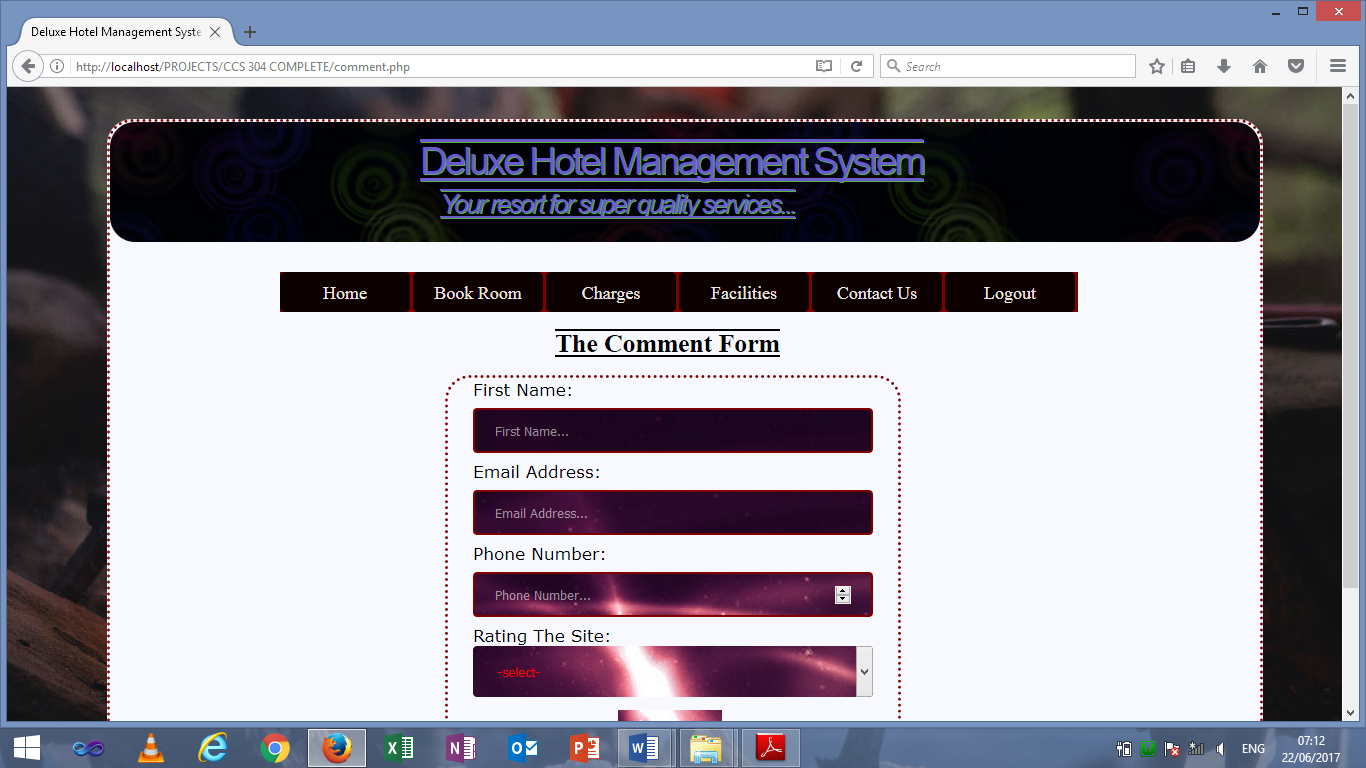


Figure 19: The Comment Panel

## 5.3.2 Back-End GUI

The system administrator uses the back-end while performing various operations. Like the front-end, was also designed using HTML and styled using CSS. The GUI is organised into independent components which can all be accessed from the menu. Styling of all the interfaces has been maintained relatively constant to increase consistency and boost the user’s learnability. Some components are common across all interfaces. They include: header, menu and the footer. A vivid description of the interfaces in the back-end is done below.

**Adding hotel panel**

The system fills all the fields in this form while adding a new hotel into the database specifically the Hotel\_Details table.

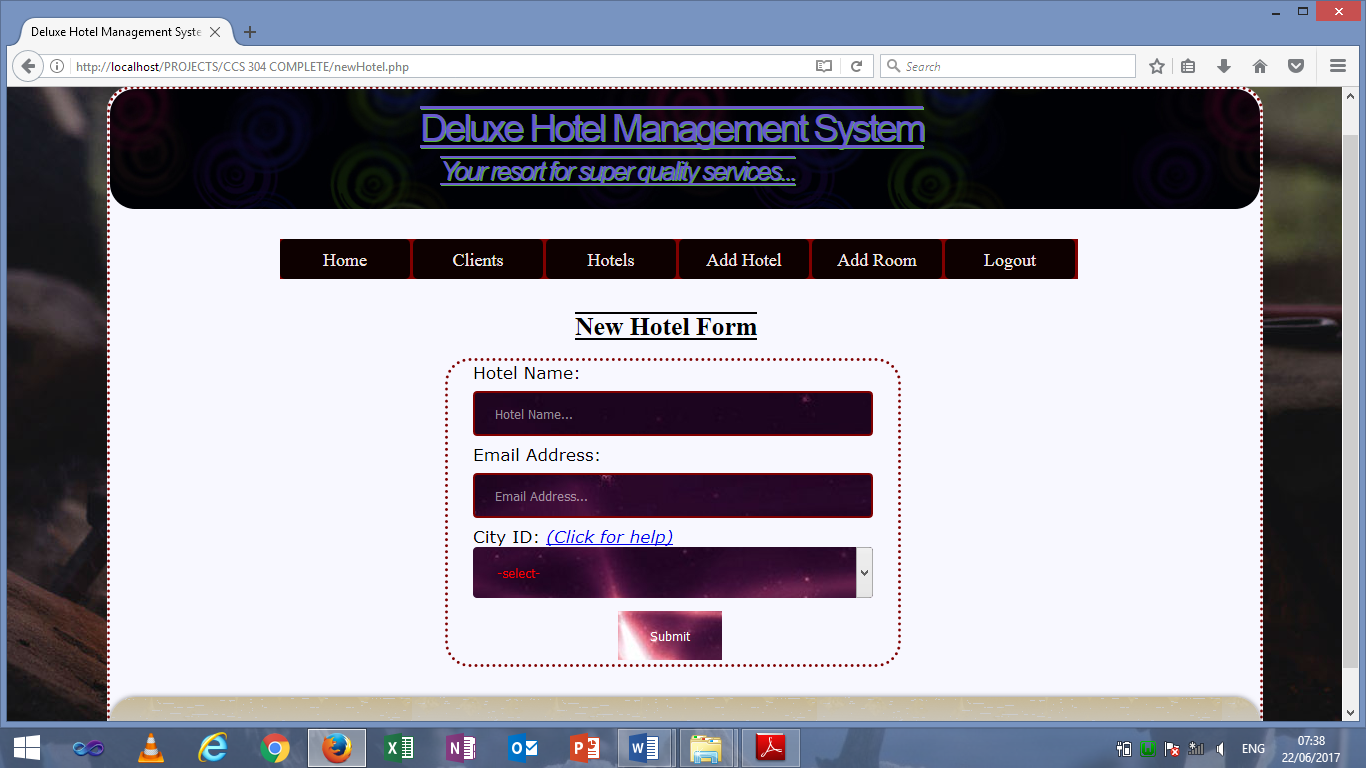


Figure 20: Adding Hotel Panel

**Adding room panel**

The system fills all the fields in this form while adding a new room into the database specifically the Room\_Details table.

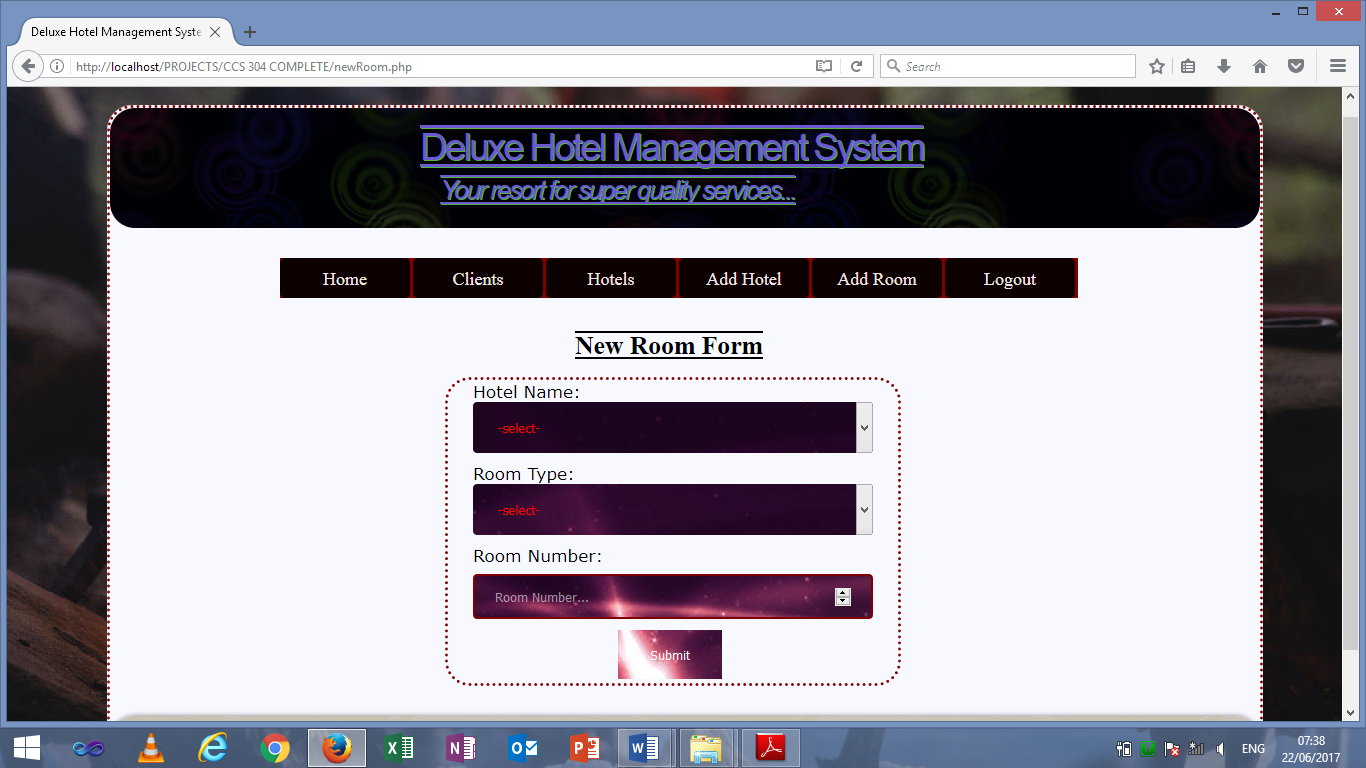


Figure 21: Adding Room Panel

**Booked rooms panel**

A list of all the booked rooms is displayed here. The records are fetched from the Booking\_Details table.



Figure 22: Booked Rooms Panel

**Registered users panel**

A list of all the registered users is displayed here. The records are fetched from the User\_Details table.



Figure 23: Registered Users Panel

# **CHAPTER SIX: CONCLUSIONS AND RECOMMENDATIONS**

## 6.1 Overall conclusion

Looking back on the overall development of the project, we believe we have succeeded in designing and creating a viable product that has demonstrated our wide range of computing skills gained in our years at university and from our year in industry. Our project does not only have potential commercial benefits, but is something that can benefit society as it was designed to provide a solution to a problem currently being experienced in the Kenya. Given more time and resources we believe we would have been able to develop this project further to fulfil the complete aims and be able to present it to a commercial investor for deployment into the public environment.

## 6.2 Recommendations

There are a number of ways through which the Hotel Management System can be improved to increase its efficiency and service delivery.

* Incorporation of an emailing capability. As soon as the user registers successfully, an email is sent to the user notifying him/her of the registration status. A user will be expected to click the link in the email so as to finalize the registration.
* Capability to book several types of rooms at the same time. As at now the system allows the user to book only one type of room. Later on this can be modified to enable user to book several rooms of different types. This functionality could not be implemented as it requires a lot of validation and time was a limit factor.
* Incorporation of the sms API (Application Programming Interface). As soon as a user books room(s) successfully instead of him/her being directed to a page bearing the room reservation details, an sms capability will be utilised so that a user receives a message via the mobile phone.
* Incorporation of more forms of payment. The system currently is making use of M-pesa as the only mode of payment. This mode of payment is dependent on ownership of a Safaricom sim card meaning that some users have to make adjustments for them to be in a position to make payment. In future, universal forms of payment will be incorporated into the system. This will favour all kinds of users.

## 6.3 References

**Related links:**

1. [www.tutorialspoint.com/mysql/index.html-](http://www.tutorialspoint.com/mysql/index.html-) 07/04/2017
2. [www.tutorialspoint.com/php/index.html](http://www.tutorialspoint.com/php/index.html) -15/04/2017
3. <http://www.studymode.com/essays/> - 22/04/2017
4. <http://www.slideshare.net/thissagamage1/> - 20/05/2017

**Books:**

1. Bob Bryla, Oracle Database Foundations, Oracle9i DBA JumpStart. Sybex. London, 2004.
2. C.R. Kothari, Research Methodology-Methods and Techniques, Second Edition, New Age International (p) Ltd Publishers, India, 2004.
3. Shelly, Rosenblatt, Systems Analysis and Design, Ninth Edition, Shelly Cashman Series, USA-Boston, 2012.