ATMIYAUNIVERSITY

RAJKOT

A

Report On

HOTEL MANAGEMENT SYSTEM

Under subject of

PROJECT

B.TECH, Semester– VII

(Computer Engineering)

Submitted by:

1. PRINCE KHIMANI 201002014
2. SANYAM REHI 201002023

Prof. Nirali Borad

(Faculty Guide)

Prof. Tosal M.Bhalodia

(Head of the Department)

Academic Year

(2022-23)

CANDIDATE'SDECLARATION

We hereby declare that the work presented in this project entitled "**HOTEL MANAGEMENT SYSTEM"** submitted towards completion of project in **7th Semester** of B.Tech. (Computer Engineering) is an authentic record of our original work carried out under the guidance of "**Prof. Nirali Borad".**

We have not submitted the matter embodied in this project for the award of any other degree.

Semester:
7thPlace: Rajkot

Signature:

Prince Khimani (201002014)

Sanyam Rehi(201002023)

ATMIYA

UNIVERSITYRAJKOT

CERTIFICATE

Date:

This is to certify that the "**HOTEL MANAGEMENT SYSTEM**" has been carried out by **Prince M Khimani** under my guidance in fulfillment of the subject Project in COMPUTER ENGINEERING (7thSemester) of Atmiya University, Rajkot during the academic year 2022.

Prof. Nirali Borad — **(Project Guide)**

Prof.Tosal M.Bhalodia — **(Head of the Department)**

ATMIYA

UNIVERSITYRAJKOT

CERTIFICATE

Date:

This is to certify that the "**HOTEL MANAGEMENT SYSTEM**" has been carried out by **SANYAM REHI** under my guidance in fulfillment of the subject Project in COMPUTER ENGINEERING (7th Semester) of Atmiya University, Rajkot during the academic year 2022.

Prof. Nirali Borad — **(Project Guide)**

Prof.Tosal M.Bhalodia — **(Head of the Department)**

ACKNOWLEDGEMENT

We have taken many efforts toward this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to Prof. Nirali Borad for their guidance and constant supervision as well as for providing necessary information regarding the Major Project titled **"HOTEL MANAGEMENT SYSTEM".** We would like to express our gratitude towards staff members of the Computer Engineering Department, Atmiya University for their kind co-operation and encouragement which helped us in the completion of this project.

We even thank and appreciate our colleagues in developing the project and the people who have willingly helped us with their abilities.

Prince M Khimani(201002014)

Sanyam M Rehi(201002023)

ABSTRACT

The hotel management system provides quality service to the end user. This project aims at creating Hotel Management System which can be used by Admin and Customers. The admin advises/publish the availability of rooms in different hotels and customers are checking the availability of rooms in the quire hotel.

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1:

INTRODUCTION

1.1 Introduction

Hotel Management System is a system that provides us to reserve rooms, check whether the rooms are vacant or not, etc. by using online browsing. This system is very useful to all, especially for business people. Business people who don't have sufficient time for these then they can use these types of online Hotel Management Systems. Through this project, we will reduce the faults in bills of their expenditure and decrease the time of the delay to give the bills to the customers. We can also save the bills for the customer. In this project, we can also include all the taxes on the bills according to their expenditures. It has a scope to reduce the errors in making the bills. A computerized bill can be printed within a fraction of a second. Online ordering of Booking is possible by using this software. This Project is based on PHP. If anyone wants to book the room for a few days, then they can specify the specific number by seeing the types of rooms we have. The bill for this online booking is based on the type of room they can select is displayed. HOTEL MANAGEMENT SYSTEM is a hotel reservation site script where site users will be able to search for room availability with an online booking reservation system. Site users can also browse hotels, view room inventory, check availability, and book reservations in real time. Site users enter the check-in date and check-out date then search for availability and rates. After choosing the right room in the wanted hotel – all booking and reservation process is done on the site and an SMS is sent to confirm the booking.

1.2. PURPOSE

The purpose of the hotel booking system is to automate the existing manual system with the help of computerized equipment and full-fledged computer software, fulfilling their requirement so that their value or information can be stored for a longer period with easy accessing and manipulating of the same. The required software and hardware are easily available and easy to work with. This proposes that the efficiency of hotel organizations could be improved by integrating service-oriented operations service-oriented operations with project management principles. Such integration would instill innovation, proactive attitudes, and regulated risk-taking needed to pursue ongoing improvement and proactive response to change. By managing each change as a project, embedded in smoothly running operations, hotels would extend their life span by continuously reinventing themselves.

1.3. SCOPE

In this step, we provide a detailed description of the existing system and the problems faced by the existing system. At this stage there is no existing system previously; we are developing a new system. Till now no system is available with this type of features and facilities. This system is developed for all types of users with the highly flexible and configurable product is envisaged to ensure global marketing

1.4. FEASIBILITY STUDY

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts to the feasibility study.

1) Operational Feasibility
2) Technical Feasibility
3) Economical Feasibility

1.4.1 OPERATIONAL FEASIBILITY

Operational feasibility is the measure of how well a proposed system solves the problems and takes

advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility assessment focuses on the degree to which the proposed development projects fit in with the existing business environment and objectives regarding the development schedule, delivery date, corporate culture, and existing business processes. To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disability, sustainability, affordability, and others. These parameters are required to be considered at the early stages of the design if desired operational behaviors are to be realized. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

1.4.2 TECHNICAL FEASIBILITY

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on an outline design of system requirements in terms of input, processes, output, fields, programs, and procedures. This can be qualified in terms of volume of data, trends, and frequency of updating to give an introduction to the technical system. The application the fact has been developed on a windows XP platform and a high configuration of 1GB RAM on an Intel Pentium Dual core processor. This is technically feasible. The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

1.4.3 ECONOMICAL FEASIBILITY

Establishing the cost-effectiveness of the proposed system i.e. if the benefits do not outweigh the costs then it is not worth going ahead. In the fast-paced world today there is a great need for online social networking facilities. Thus the benefits of this project in the current scenario make it economically feasible. The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis.

1.5 ORGANISATION OF REPORT

1.5.1 INTRODUCTION

This section includes the overall view of the project i.e. the basic problem definition and the general overview of the problem which describes the problem in layman's terms. It also specifies the software used and the proposed solution strategy.

1.5.2SOFTWARE REQUIREMENTS SPECIFICATION

This section includes the Software and hardware requirements for the smooth running of the application.

1.5.3DESIGN & PLANNING

This section consists of the Software Development Life Cycle model. It also contains technical diagrams like the Data Flow Diagram and the Entity Relationship diagram.

1.5.4 RESULTS AND DISCUSSION

This section has screenshots of all the implementations i.e. user interface and their description.

1.5.5 SUMMARY AND CONCLUSION

This section has screenshots of all the implementations i.e. user interface and their description.

CHAPTER 2:

SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Hardware Requirements

Table 2.1.1 Hardware Requirements

Number	Description
1	PC with 250 GB or more Hard disk.
2	PC with 2 GB RAM
3	PC with Pentium 1 and Above.

2.2 Software Requirements

Table 2.2.1 Software Requirements

Number	Description	Type
1	Operating System	Windows XP / Windows
2	Language	PHP
3	Database	MySQL
4	IDE	Visual Code
5	Browser	Google Chrome

CHAPTER 3
DESIGN & PLANNING

3.1 Software Development Life Cycle Model

3.1.1 WATERFALL MODEL

The waterfall model was selected as the SDLC model due to the following reasons:

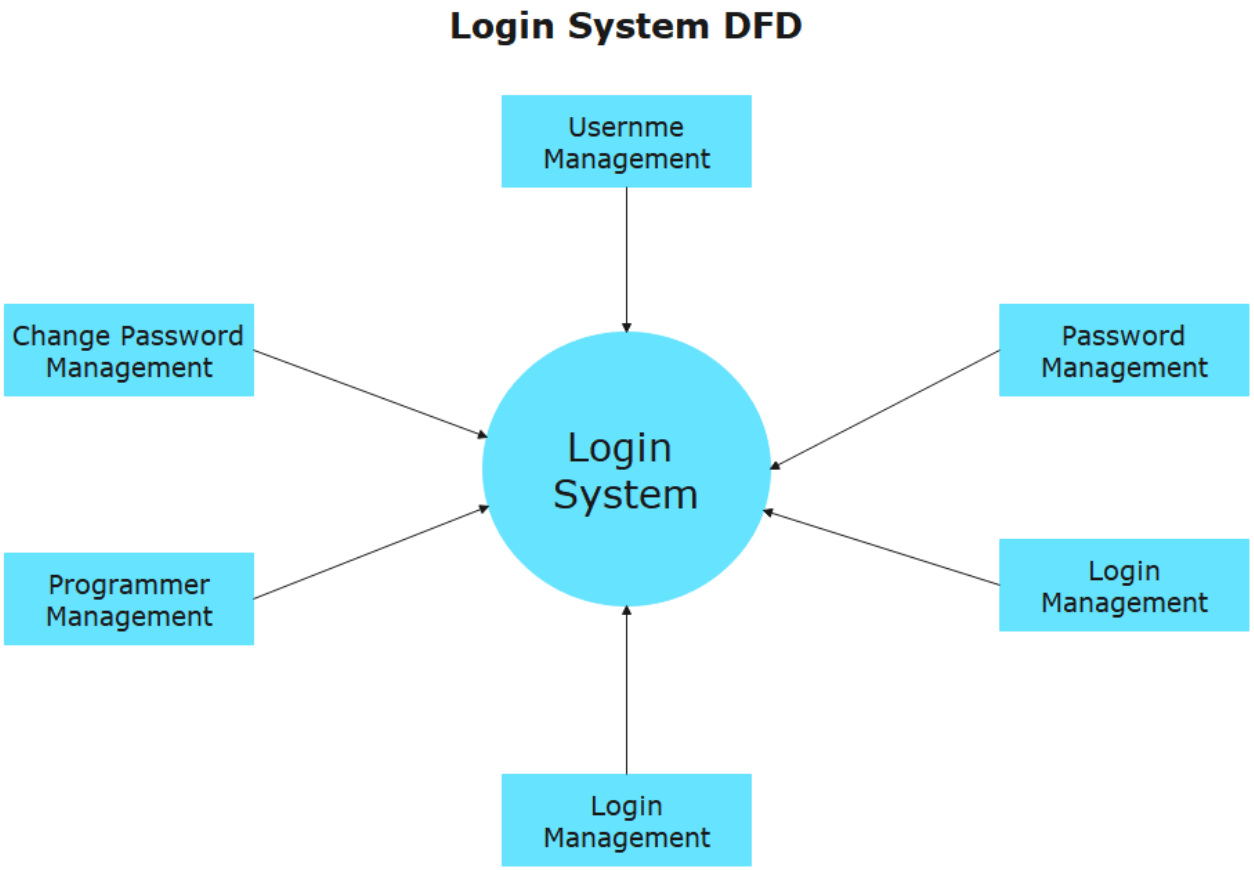
Requirements were very well documented, clear, and fixed. Technology was adequately understood. Simple and easy to understand and use. There were no ambiguous requirements.
Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process. Clearly defined stages. Well-understood milestones easy to arrange tasks.

3.2 GENERAL OVERVIEW

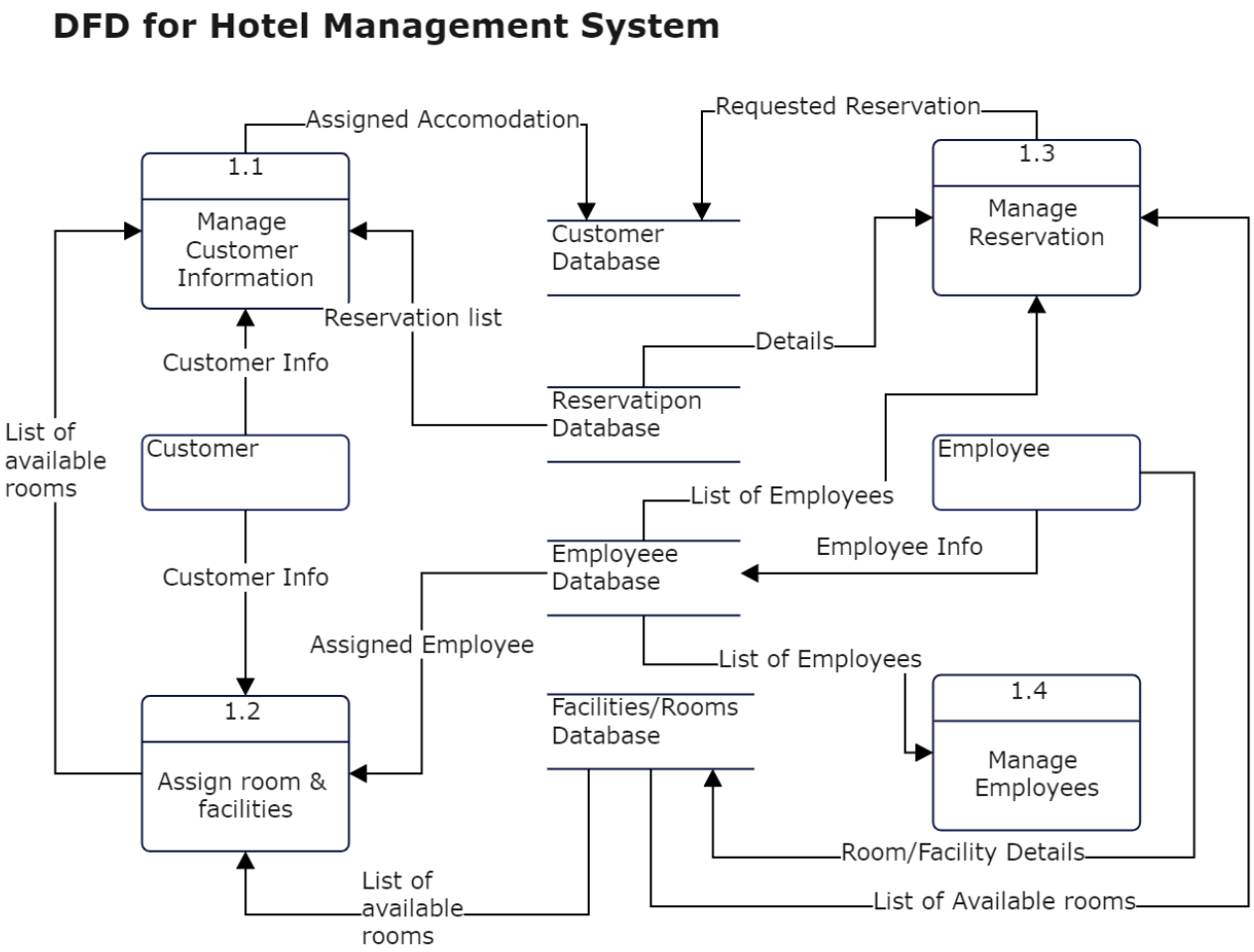
Hotel Management System is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to hospitals. More importantly, it is backed by reliable support. Hotel Management System is custom built to meet the specific requirement of mid and large-size hospitals across the globe. It has a counter that counts calories which will be accessed by users. It contains modules like booking appointments, managing reports, and medical history, User search, etc. It contains a section named contact us queries where you can ask for any doubts related to your appointment booking or rescheduling.

3.3 DATA FLOW DIAGRAM

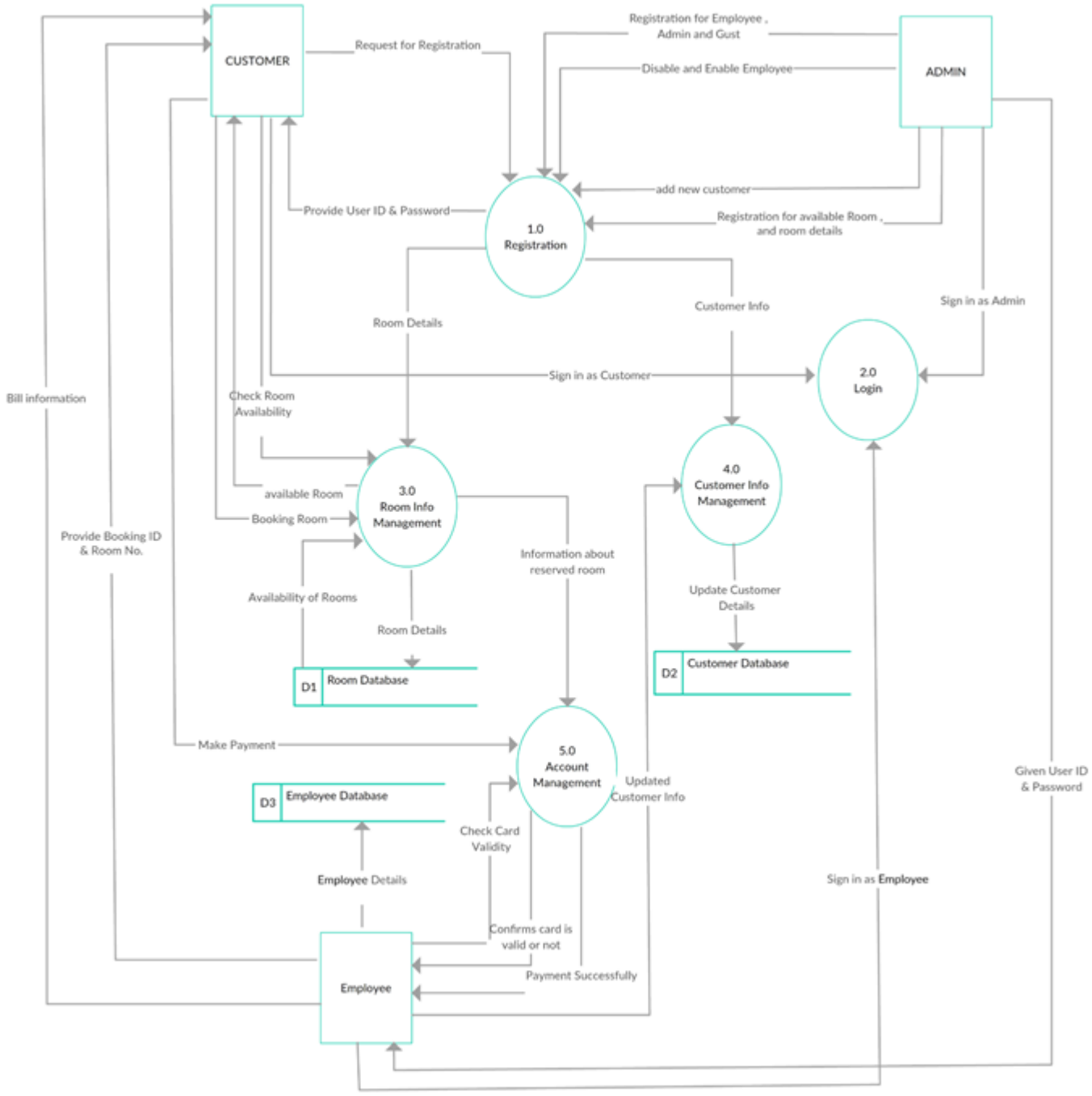
3.3.1 Level 0 DFD



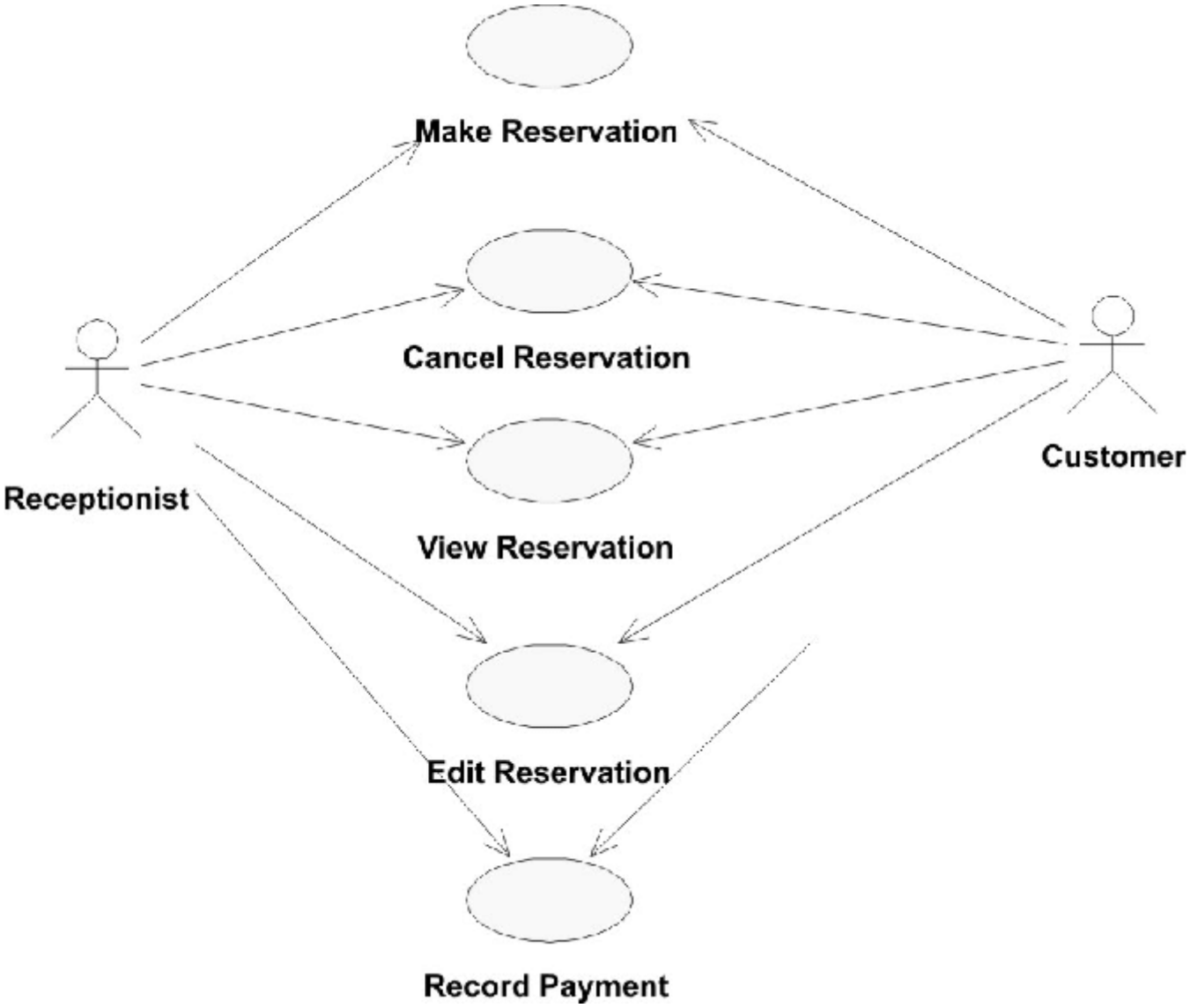
3.3.2 Level 1 DFD



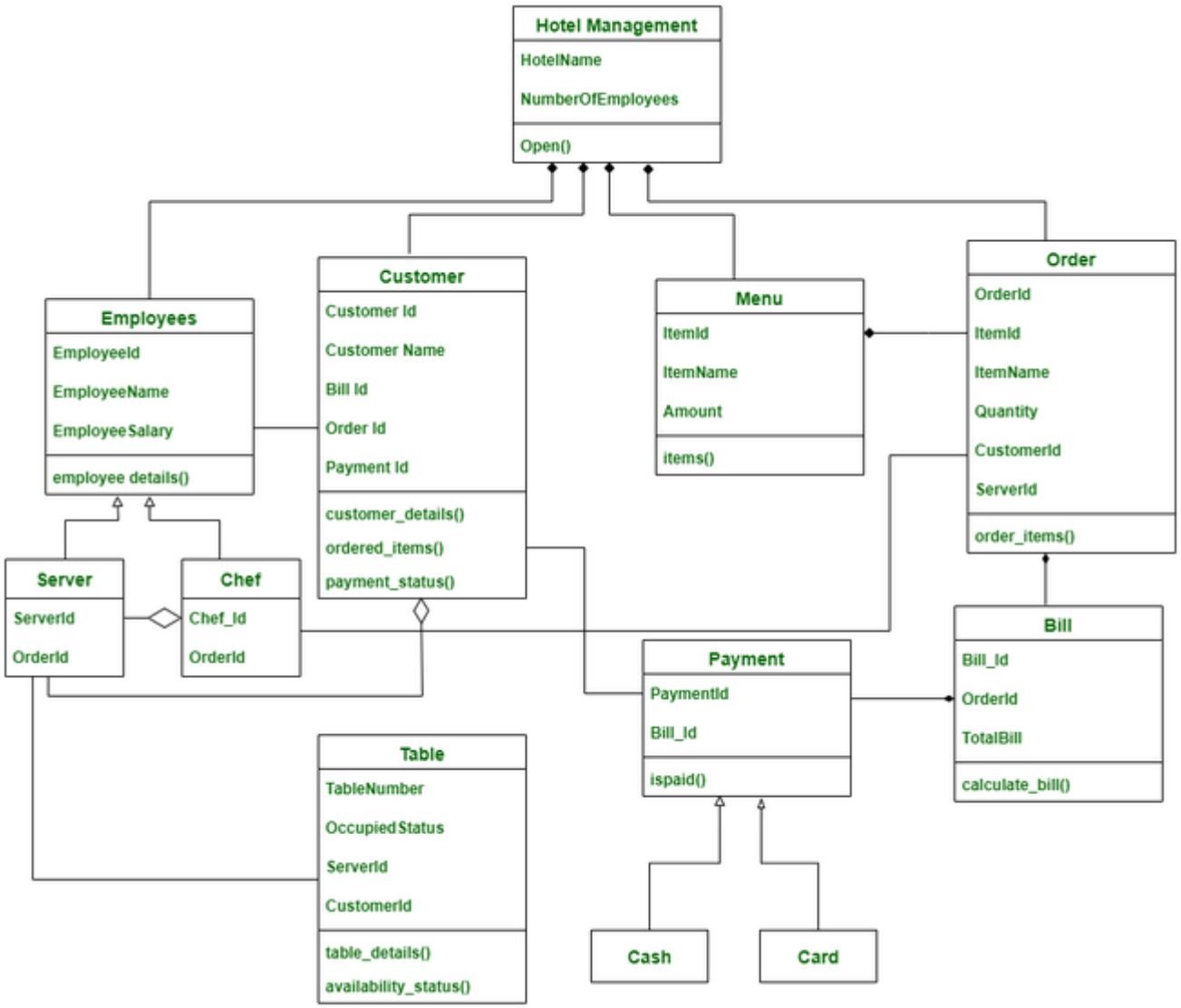
3.3.3 Level 2 DFD



3.4USE CASE DIAGRAM



3.5 CLASS DIAGRAM



3.6 Input /Output Interface

Fig.3.6.1Home Page

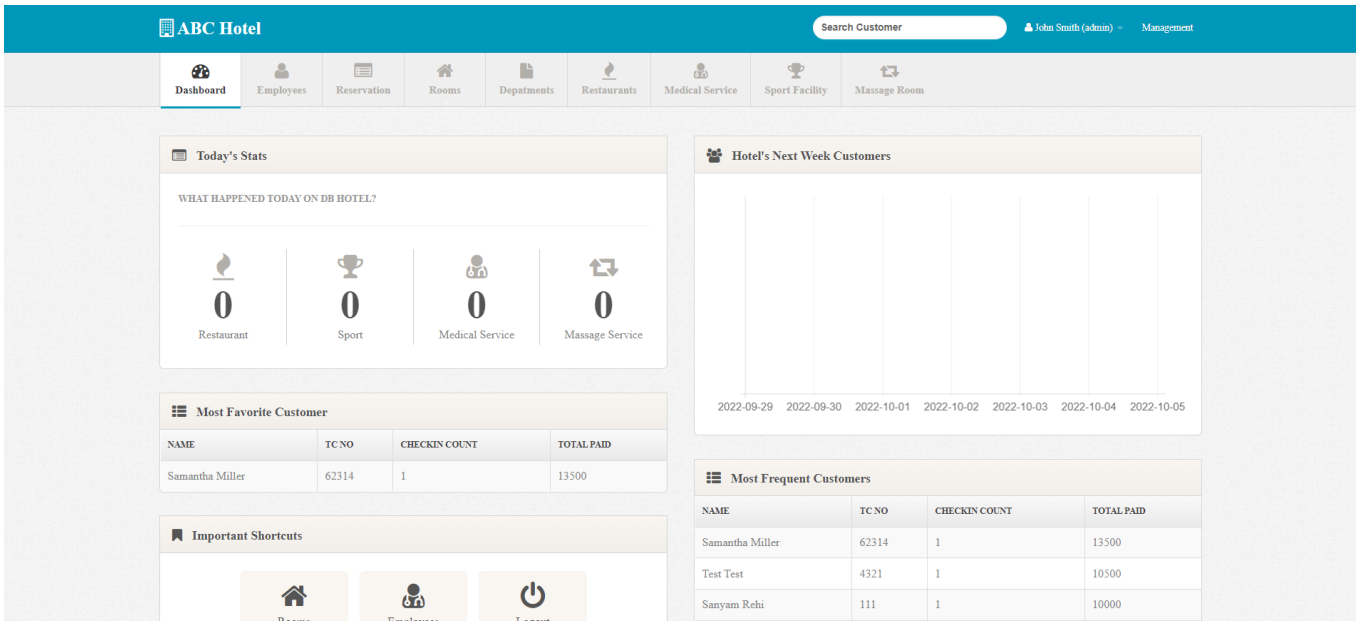


Fig.3.6.2 Employees Page

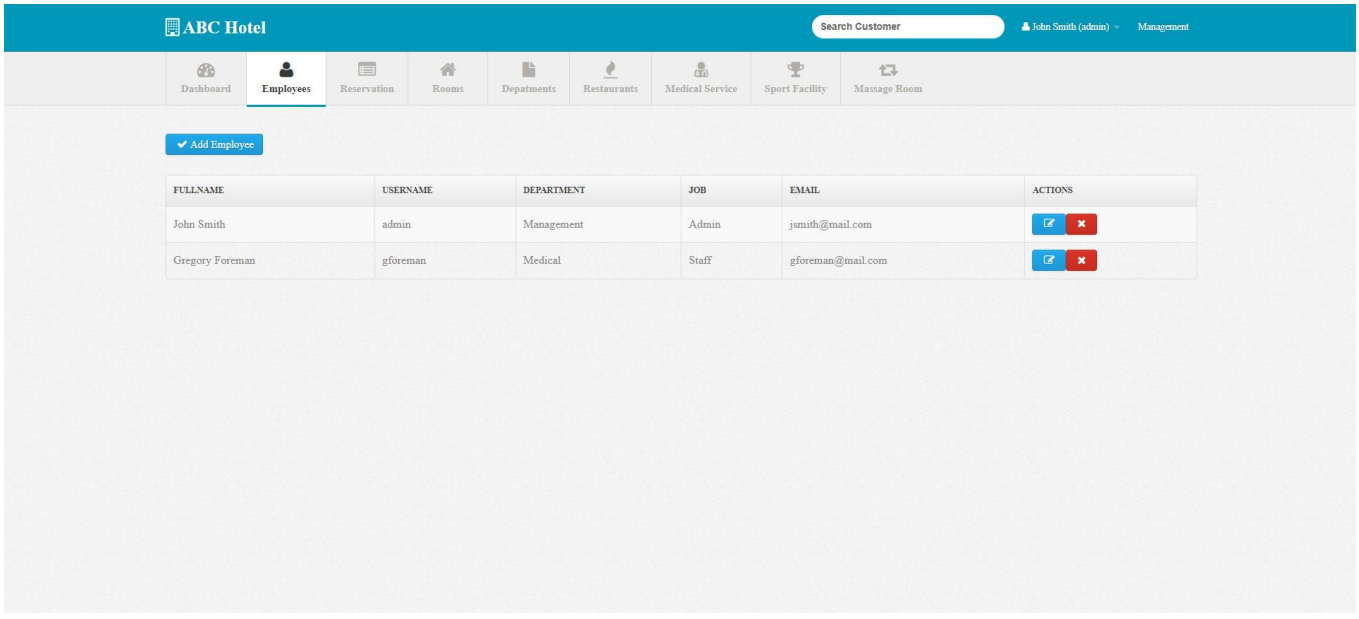


Fig.3.6.3 Hotel Reservation Page

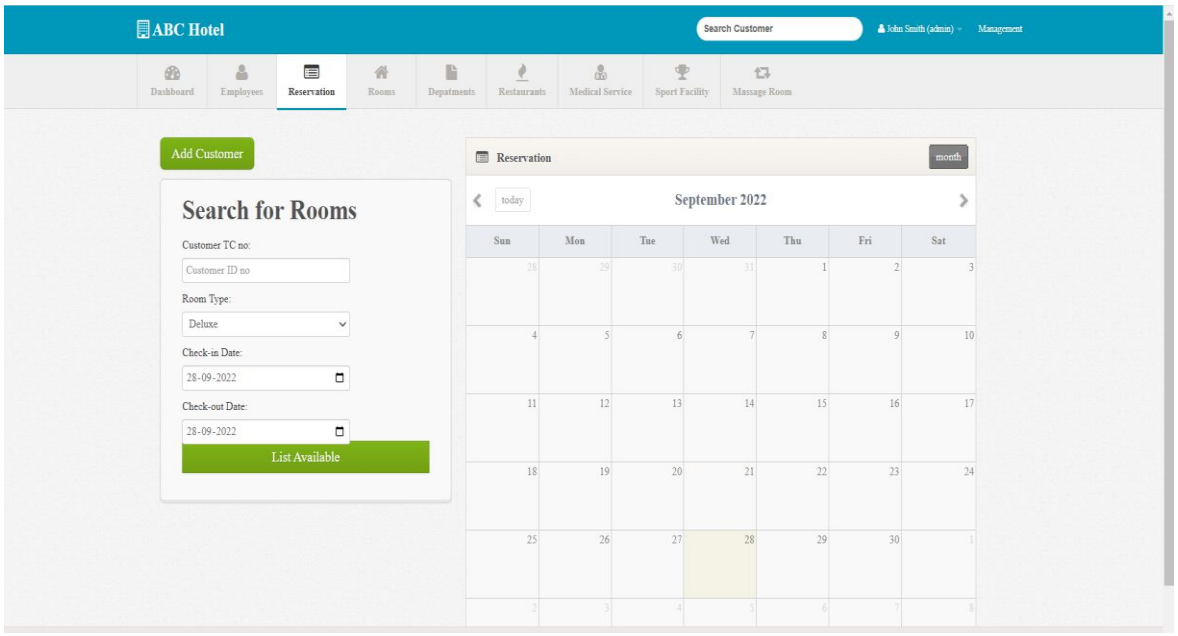


Fig.3.6.4 Employees Medical Page

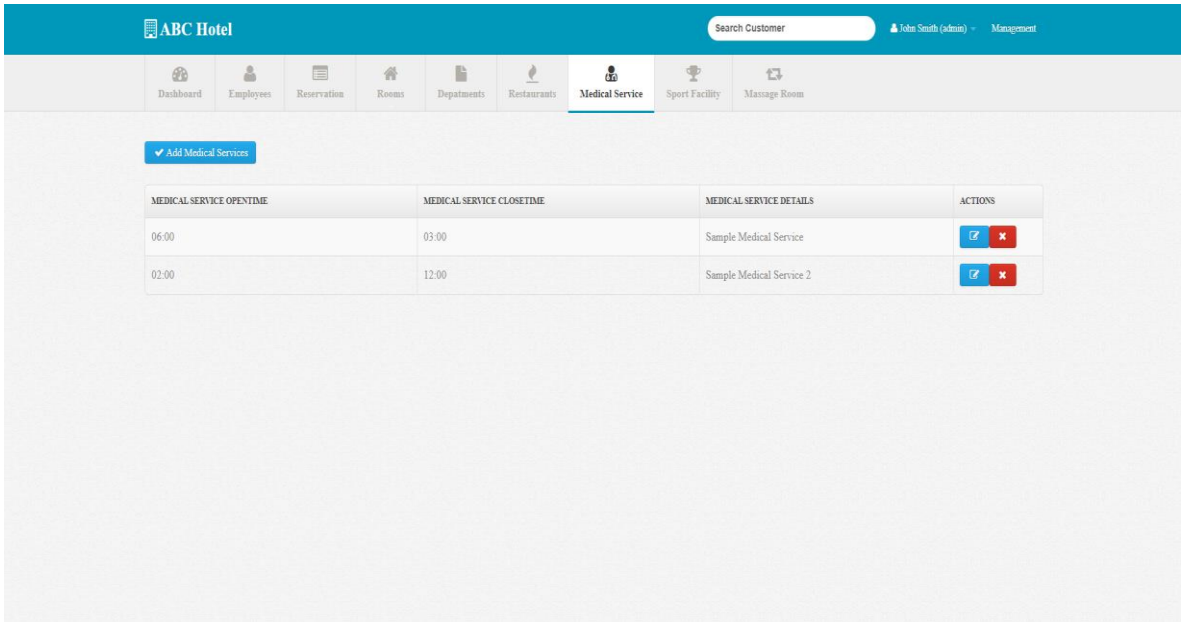


Fig.3.6.5 Employee Medical History Page

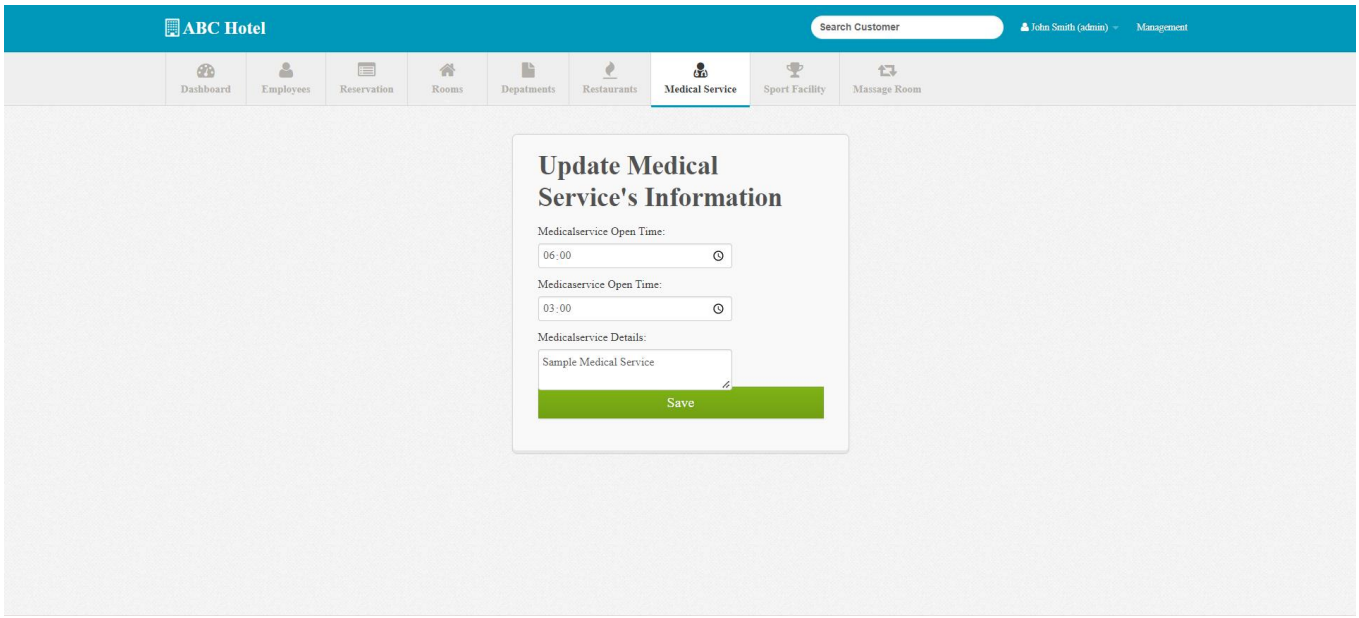


Fig.3.6.6 Admin Login Page

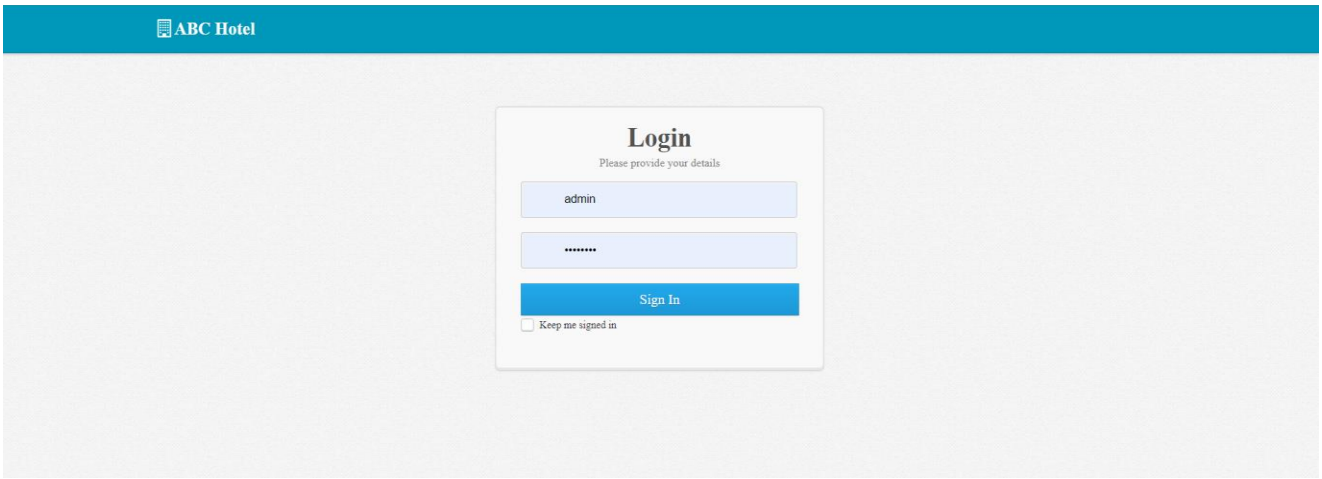
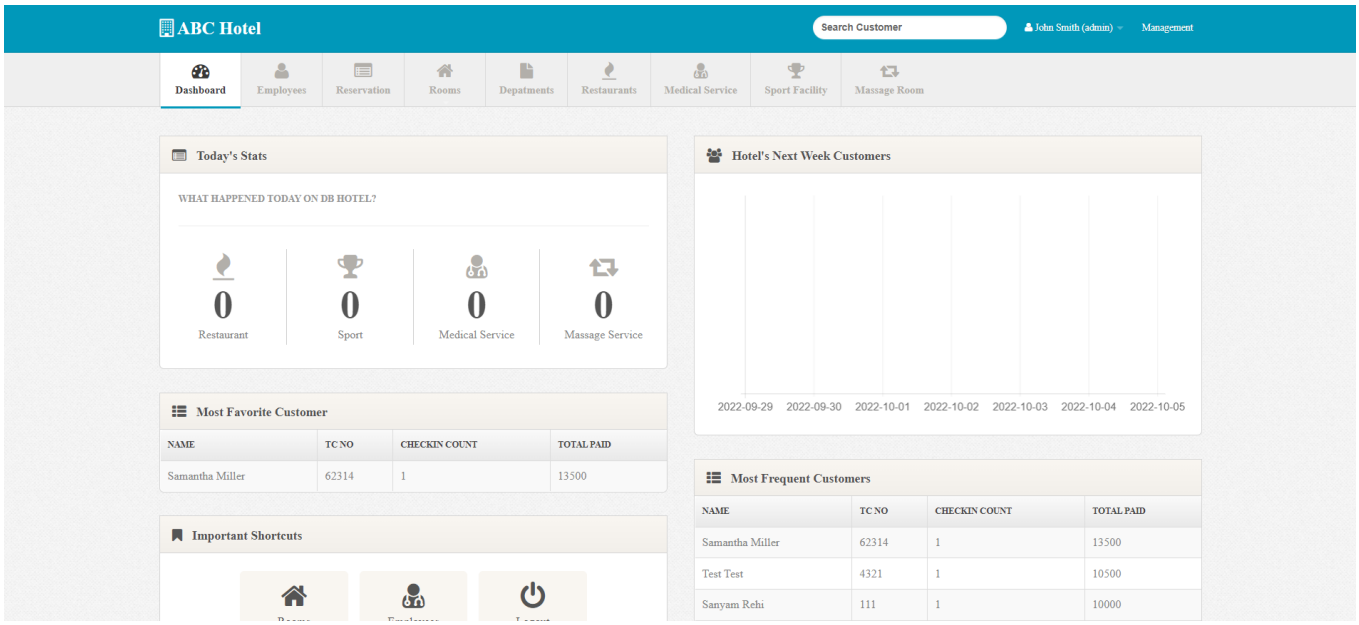


Fig.3.6.7 Admin Dashboard Page



CHAPTER 4

IMPLEMENTATION DETAILS

In this Section, we will do an Analysis of Technologies to use for implementing the project.

4.1 FRONT END

4.1.1 HTML

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), the former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

4.1.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation
characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file and reduce complexity and repetition in the structural content.

CSS information can be provided from various sources. These sources can be the web browser, the user, and the author. The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance, and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so authors can tailor the presentation appropriately for each medium. The style sheet with the highest priority controls the content display. Declarations not set in the highest priority source are passed on to a source of lower priority, such as the user agent style. The process is called cascading.

One of the goals of CSS is to allow users greater control over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet. Depending on the browser and the website, a user may choose from various style sheets provided by the designers, may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

4.2 BACK END

4.2.1 PHP

PHP is a server-side scripting language that is used to develop Static websites Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, which earlier stood for Personal Home Pages. PHP scripts can only be interpreted on a server that has PHP installed. The client computers accessing the PHP scripts require a web browser only. A PHP file contains PHP tags and ends with the extension ".php".

The term PHP is an acronym for PHP: Hypertext Preprocessor. PHP is a server-side scripting language designed specifically for web development. PHP can be easily embedded in HTML files and HTML codes can also be written in a PHP file. The thing that differentiates PHP from a client-side language like HTML is, PHP codes are executed on the server whereas HTML codes are directly rendered on the browser.

PHP: Hypertext Preprocessor (or simply PHP) is a general-purpose programming language originally designed for web development. It was created by Rasmus Lerdorf in 1994.PHP code may be executed with a command line interface (CLI), embedded into HTML code, or used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in a web server or as a Common Gateway Interface (CGI) executable. The web server outputs the results of the interpreted and executed PHP code, which may be any type of data, such as generated HTML code or binary image data. PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control.

4.2.2 MySQL

MySQL is an open-source relational database management system (RDBMS) based on Structured Query Language (SQL). It is one part of the very popular LAMP platform consisting of Linux, Apache, My SQL, and PHP. Currently, My SQL is owned by Oracle. My SQL database is available on the most important OS platforms. It runs on BSD Unix, Linux, Windows, or Mac OS. Wikipedia and YouTube use My SQL. These sites manage millions of queries each day. My SQL comes in two versions: My SQL server system and My SQL embedded system.

RDBMS TERMINOLOGY

Before we proceed to explain the MySQL database system, let's revise a few definitions related to databases.

Database: A database is a collection of tables, with related data.

Table: A table is a matrix with data. A table in a database looks like a simple spreadsheet.

Column: One column (data element) contains data of the same kind, for example, the column postcode.

Row: A row (= tuple, entry, or record) is a group of related data, for example, the data of one subscription.

Redundancy: Storing data twice, redundantly to make the system faster.

Primary Key: A primary key is unique. A key value cannot occur twice in one table. With a key, you can find at most one row.

Foreign Key: A foreign key is a linking pin between two tables.

Compound Key: A compound key (composite key) is a key that consists of multiple columns because one column is not sufficiently unique.

Index: An index in a database resembles an index at the back of a book.

Referential Integrity: Referential Integrity makes sure that a foreign key value always points to an existing row.

4.3 Coding Standards

Normally, good software development organization requires their programmers to adhere to some well-defined and standard style of coding called coding standard.

4.3.1 Variable Standards:

We have used a meaningful variable's name.

4.3.2 Comment Standards:

The comment should describe what is happening, how it is being done, what parameters mean, which global are used and which are modified, and any registration or bugs. The standards I have followed are:

- Every script should begin with a comment block, which describes the purpose of the script; any argument used (if applicable), return values (if applicable), inputs-outputs, and the name of the script.
- Comment may also be used in the body of the script to explain individual sections or lines of codes.
- It is also used to describe variable definitions or declarations.
- Inline comments should be made with the. //. Comment style and should be indented at the same level as the code described.
- For multiple line comments we write between /*….*/.

4.4 Program/Module Specification

- System GUI must be as simple and user-friendly as anyone can use it. On the front side, we implemented a registration form to access the system.
- Authentication is necessary to enter into the system only if one requires to start his/her project. This is required to prevent unauthorized access to the system.

- If someone steals the password of the administrator or any regular user then he can able to change the database or misuse the system and can enter a restricted area so for this purpose system will provide an encrypted password storage format in the database. Option to change the Password.
- A Session is maintained throughout the system when a particular user enters the system. The Session is regularly checked whenever it is required.

CHAPTER 5

TESTING AND IMPLEMENTATION

The term implementation has different meanings ranging from the conversation of a basic application to a complete replacement of a computer system. The procedures however are virtually the same. Implementation includes all those activities that take place to convert from the old system to the new one. The new system may be new replacing an existing manual or automated system or it may be a major modification to an existing system. The method of implementation and time scale to be adopted is found out initially. Proper implementation is essential to provide a reliable system to meet organizational requirements.

5.1 UNIT TESTING

5.1.1 Introduction

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent of the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist in testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

5.1.2 Benefits

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

1) Find problems early: Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. If the unit tests fail, it is considered to be a bug either in the changed code or the tests themselves. The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

2) Facilitates Change: Unit testing allows the programmer to refactor code or upgrade system libraries at a later date, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes that may break a design contract.

3) Simplifies Integration: Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

4) Documentation: Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviors that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

5.2 INTEGRATION TESTING

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

5.2.1 Purpose

The purpose of integration testing is to verify the functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black-box testing, success, and error cases being simulated via appropriate parameters and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages. Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. The cross-dependencies for software integration testing are: schedule for integration testing, strategy, and selection of the tools used for integration, define the cyclomatic complexity of the software and software architecture, reusability of modules and life-cycle, and versioning management. Some different types of integration testing are big-bang, top-down, and bottom-up, mixed (sandwich), and risky-hardest. Other Integration Patterns[2] are collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration, and high-frequency integration.

5.2.1.1 Big Bang

In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing. A type of big-bang integration testing is called "usage model testing" which can be used in both software and hardware integration testing. The basis behind this type

of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment. For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

5.2.1.2 Top-down and Bottom-up

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures, or functions are integrated and then tested. After the integration testing of lower-level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage. Top-down testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module. Sandwich testing is an approach to combining top-down testing with bottom-up testing.

5.3 SOFTWARE VERIFICATION AND VALIDATION

5.3.1 Introduction

In software project management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle. Validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements. This is done through dynamic testing and other forms of review. Verification and validation are not the same things, although they are often confused. Boehm succinctly expressed the difference between

Validation: Are we building the right product?
Verification: Are we building the product right?
According to the Capability Maturity Model (CMMI-SW v1.1)

Software Verification: The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Software Validation: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

In other words, software verification is ensuring that the product has been built according to the requirements and design specifications, while software validation ensures that the product meets the user's needs and that the specifications were correct in the first place. Software verification ensures that "you built it right". Software validation ensures that "you built the right thing". Software validation confirms that the product, as provided, will fulfill its intended use.

From Testing Perspective

Fault – wrong or missing function in the code.
Failure – the manifestation of a fault during execution.
Malfunction – according to its specification the system does not meet its specified functionality
Both verification and validation are related to the concepts of quality and of software quality assurance. By themselves, verification and validation do not guarantee software quality; planning, traceability, configuration management, and other aspects of software engineering are required. Within the modeling and simulation (M&S) community, the definitions of verification, validation, and accreditation are similar:

M&S Verification is the process of determining that a • computer model, simulation, or federation of models and simulations implementations and their associated data accurately represent the developer's conceptual description and specifications.
M&S Validation is the process of determining the degree to which a model, simulation, or federation of models and simulations, and their associated data are accurate representations of the real world from the perspective of the intended use(s).

5.3.2 Classification of Methods

In mission-critical software systems, where flawless performance is necessary, formal methods may be used to ensure the correct operation of a system. However, often for non-mission-critical software systems, formal methods prove to be very costly and an alternative method of software V&V must be sought out. In such cases, syntactic methods are often used.

5.3.3 Test Cases

A test case is a tool used in the process. Test cases may be prepared for software verification and software validation to determine if the product was built according to the requirements of the user. Other methods, such as reviews, may be used early in the life cycle to provide for software validation.

5.3.3.1 Test Suit

Admin login test:

Test Case	Test Data	Test Result	Test Report
Blank Username	Username	Invalid	Fill required detail
Invalid Username	Username: ADMIN	Invalid	Username Incorrect
Invalid Password	Password: user	Invalid	Password Incorrect
Valid Username and Password	Username: admin Password: admin	Valid	Login

Table 5.3.3.1.1 Admin Login Test

5.4 Black-Box Testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system, and acceptance. It typically comprises most if not all higher-level testing, but can also dominate unit testing as well.

5.4.1 Test Procedures

Specific knowledge of the application's code/internal structure and programming knowledge, in general, is not required. The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.

5.4.2 Test Cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements, and design parameters. Although the tests used are primarily functional in nature, non-functional tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output, often with the help of an oracle or a previous result that is known to be good, without any knowledge of the test object's internal structure.

5.5 White-Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests the internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). White-box testing can be applied at the unit, integration, and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

5.5.1 Levels

1) Unit testing: White-box testing is done during unit testing to ensure that the code is working as intended before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.

2) Integration testing: White-box testing at this level is written to test the interactions of each interface with the other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behavior in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.

3) Regression testing: White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

5.5.2 Procedures

White-box testing's basic procedures involve the tester having a deep level of understanding of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analyzed for test cases to be created. These are the three basic steps that white-box testing takes to create test cases:

Input involves different types of requirements, functional specifications, detailed designing of documents, proper source code, and security specifications. This is the preparation stage of white-box testing to lay out all of the basic information.
Processing involves performing risk analysis to guide the whole testing process, proper test plan, execute test cases and communicate results. This is the phase of building test cases to make sure they thoroughly test the application and the given results are recorded accordingly.
Output involves preparing a final report that encompasses all of the above preparations and results.

5.5.3 Advantages

White-box testing is one of the two biggest testing methodologies used today. It has several major advantages:

The side effects of knowing the source code are beneficial to thorough testing.

Optimization f code by revealing hidden errors and being able to remove these possible defects.
Gives the programmer introspection because developers carefully describe any new implementation.
Provides traceability of tests from the source, allowing future changes to the software to be easily captured in changes to the tests.
White box testing gives clear, engineering-based, rules for when to stop testing.

5.5.4 Disadvantages

Although white-box testing has great advantages, it is not perfect and contains some disadvantages:

White-box testing brings complexity to testing because the tester must know about the program, including being a programmer. White-box testing requires a programmer with a high level of knowledge due to the complexity of the level of testing that needs to be done.
On some occasions, it is not realistic to be able to test every single existing condition of the application and some conditions will be untested.
The tests focus on the software as it exists, and missing functionality may not be discovered.

5.6 SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

CHAPTER 6

LIMITATIONS

Though we tried our best in developing this system as limitations are mere parts of any system so are our system. Some limitations of the Hotel Management System are: -

- Online payment is not available at this version
- Data delete and edit system is not available for all section
- User account verification by mobile SMS is not available in this system
- Loss of data due to mismanagement.

CHAPTER 7

CONCLUSION

Hotel Management systems as an industry in an infant stage. It has a long way to go. It has a bright future. This industry is growing at a reasonably quick rate. There should also be a set of standards for the events that are being conducted. Shortly, the companies form network associations in different stages to expand the industry so that events can be held professionally.

CHAPTER 8

REFERENCES

Books

- **Software Engineering by Roger Pressman**
- **Full Stack Web Development with GraphQL and react**

Websites

- **www.github.com**
- **www.stackoverflow.com**
- **www.w3schools.com**