VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM-590014

An Internship Project Report
On

"HOTEL MANAGEMENT SYSTEM"

A project report submitted in partial fulfillment of the requirements for the 57h semester of ***Bachelor of Engineering in Computer Science and Engineering*** *of Visvesvaraya Technological University, Belgaum*

Submitted by:

ABHAY G DESHPANDE	**1RN15CS004**
MOHIT K.	**1RN15CS061**
NAMRATHA BHAT U	**1RN15CS065**
MEGHANA H.S	**1RN15CS129**

Under the Guidance of:

Mr. Sunil.D.Shashidhara
CEO of Valley Boot Camp

Department of Computer Science and Engineering
RNS Institute of Technology
Channasandra, Uttarahalli-Kengeri Main Road, Bangalore-560 098

2017-2018

RNS Institute of Technology
Channasandra, Uttarahalli-Kengeri Main Road,
Bangalore-560 061

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that the project work entitled **"Hotel Management Sytem"** has been successfully carried out by **Abhay G Deshpande** bearing USN **1RN15CS004** , **Mohit K** bearing USN **1RN15CS061**, **Namratha Bhat U** bearing USN 1RN15CS065 and **Meghana H S** bearing USN **1RN15CS129** bonafide students of **RNS Institute of Technology** in partial fulfillment of the requirements for the 7**th semester** of **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University**, Belgaum, during academic year 2017-2018. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the laboratory requirements of 5th semester BE, CSE.

Mr.Sunil.D.Shashidhara
CEO of Valley Boot Camp

Dr. G T Raju
Prof. and Head
Dept of CSE

External Viva:

Name of the Examiners **Signature with Date**

1.

2.

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project work. We would like to take this opportunity to thank them all.

We would like to thank **Dr. H N Shivashankar**, Director, RNSIT, Bangalore, for his moral support towards completing my project.

We are grateful to **Dr. M K Venkatesha,** Principal, RNSIT, Bangalore, for his support towards completing my project.

We would like to thank **Dr. G T Raju**, Dean of Engg., Prof & Head , Department of Computer Science & Engineering, RNSIT, Bangalore, for his valuable suggestions and expert advice.

We deeply express my sincere gratitude to my guide Mr.Sunil D Shashidhara,,CEO of Valley Boot Camp Bangalore, for their able guidance, regular source of encouragement and assistance throughout this project.

We would like to thank all the teaching and non-teaching staff of Department of Computer Science & Engineering RNSIT, Bangalore for their constant support and encouragement.

Date :
Place : Bangalore

Abhay G Deshpande 1RN15CS004
Mohith K 1RN15CS061
Namratha Bhat U 1RN15CS065
Meghana H S 1RN15CS129

ABSTRACT

In our project, on “Hotel Management System” we have tried to show how the data/information in hotels is managed. This is just an overview of management in hotels. This has been achieved by dividing the projects into various modules. Manager is provided with different services like checking in and checking out. If the manager wants, he or she can cancel the booking, manage and add employee etc.

Enquiry about any Employee can be made by Employee name. Enquiry about rooms available can also be made.

We build a chatbot with iterative content exploration that leads a user through a personalized knowledge aquisition session. The chatbot is designed as an automated manager support.

This project has been implemented using HTML, CSS, Bootstrap and JavaScript for front-end, flask and SQLite for back-end. The Machine learning part is implemented using IBM Watson and telegram bot.

CONTENTS

CHAPTER 1

INTRODUCTION TO DATABASE MANAGEMENT SYSTEMS

1.1 Introduction

Databases and database technology have a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, engineering, medicine, genetics, law, education, and library science. The word database is so commonly used that we must begin by defining what a database is. Our initial definition is quite general. A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database. The preceding definition of database is quite general; for example, we may consider the collection of words that make up this page of text to be related data and hence to constitute a database. However, the common use of the term database is usually more restricted.

A database has the following implicit properties:

- A database represents some aspect of the real world, sometimes called the mini world or the universe of discourse (UoD). Changes to the mini world are reflected in the database.
- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.

- A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications. Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database. The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called meta-data. Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS. Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the mini world, and generating reports from the data. Sharing a database allows multiple users and programs to access the database simultaneously.

1.2 History Of DBMS

A Database Management System allows a person to organize, store, and retrieve data from a computer. It is a way of communicating with a computer's "stored memory." In the very early years of computers, "punch cards" were used for input, output, and data storage. Punch cards offered a fast way to enter data, and to retrieve it. Herman Hollerith is given credit for adapting the punch cards used for weaving looms to act as the memory for a mechanical tabulating machine, in 1890. Much later, databases came along.

Databases (or DBs) have played a very important part in the recent evolution of computers. The first computer programs were developed in the early 1950s, and focused almost completely on coding languages and algorithms. At the time, computers were basically giant calculators and data (names, phone numbers) was considered the leftovers of processing information.

Computers were just starting to become commercially available, and when business people started using them for real-world purposes, this leftover data suddenly became important.

Enter the Database Management System (DBMS). A database, as a collection of information, can be organized so a Database Management System can access and pull specific information. In 1960, Charles W. Bachman designed the Integrated Database System, the "first" DBMS. IBM, not wanting to be left out, created a database system of their own, known as IMS. Both database systems are described as the forerunners of navigational database.
By the mid-1960s, as computers developed speed and flexibility, and started becoming popular, many kinds of general use database systems became available. As a result, customers demanded a standard be developed, in turn leading to Bachman forming the Database Task Group. This group took responsibility for the design and standardization of a language called Common Business Oriented Language (COBOL). The Database Task Group presented this standard in 1971, which also came to be known as the "CODASYL approach."

The CODASYL approach was a very complicated system and required substantial training. It depended on a "manual" navigation technique using a linked data set, which formed a large network. Searching for records could be accomplished by one of three techniques:

- Using the primary key (also known as the CALC key)
- Moving relationships (also called sets) to one record from another
- Scanning all records in sequential order

Eventually, the CODASYL approach lost its popularity as simpler, easier-to-work-with systems came on the market.

Edgar Codd worked for IBM in the development of hard disk systems, and he was not happy with the lack of a search engine in the CODASYL approach, and the IMS model. He wrote a series of papers, in 1970, outlining novel ways to construct databases. His ideas eventually evolved into a paper titled, A Relational Model of Data for Large Shared Data Banks, which

described new method for storing data and processing large databases. Records would not be stored in a free-form list of linked records, as in CODASYL navigational model, but instead used a “table with fixed-length records.”

IBM had invested heavily in the IMS model, and wasn’t terribly interested in Codd’s ideas. Fortunately, some people who didn’t work for IBM “were” interested. In 1973, Michael Stonebraker and Eugene Wong (both then at UC Berkeley) made the decision to research relational database systems. The project was called INGRES (*Interactive Graphics and Retrieval System*), and successfully demonstrated a relational model could be efficient and practical. INGRES worked with a query language known as QUEL, in turn, pressuring IBM to develop SQL in 1974, which was more advanced (SQL became ANSI and OSI standards in 1986 1nd 1987). SQL quickly replaced QUEL as the more functional query language.

RDBM Systems were an efficient way to store and process structured data. Then, processing speeds got faster, and “unstructured” data (art, photographs, music, etc.) became much more common place. Unstructured data is both non-relational and schema-less, and Relational Database Management Systems simply were not designed to handle this kind of data.

1.3 Characteristics of Database Approach

A number of characteristics distinguish the database approach from the much older approach of programming with files. In traditional file processing, each user defines and implements the files needed for a specific software application as part of programming the application. For example, one user, the grade reporting office, may keep files on students and their grades. Programs to print a student’s transcript and to enter new grades are implemented as part of the application. A second user, the accounting office, may keep track of students’ fees and their payments. Although both users are interested in data about students, each user maintains separate files— and programs to manipulate these files—because each requires some data not avail able from the other user’s files. This redundancy in defining and storing

data results in wasted storage space and in redundant efforts to maintain common up-to-date data. In the database approach, a single repository maintains data that is defined once and then accessed by various users. In file systems, each application is free to name data elements independently. In contrast, in a database, the names or labels of data are defined once, and used repeatedly by queries, transactions, and applications. The main characteristics of the database approach versus the file-processing approach are the following:

- Self-describing nature of a database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing.

1.3.1 Self-Describing Nature of a Database System

A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints. This definition is stored in the DBMS catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data. The information stored in the catalog is called meta-data, and it describes the structure of the primary database. The catalog is used by the DBMS software and also by database users who need information about the database structure. A general-purpose DBMS software package is not written for a specific database application. Therefore, it must refer to the catalog to know the structure of the files in a specific database, such as the type and format of data it will access. The DBMS software must work equally well with any number of database applications—for example, a university database, a banking database, or a company database—as long as the database definition is stored in the catalog. In traditional file processing, data definition is typically part of the application programs themselves.

Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs. For example, an application program written in C++ may have structure or class declarations, and a COBOL program has data division statements to define its files. Whereas file-processing software can access only specific databases, DBMS software can access diverse databases by extracting the database definitions from the catalog and using these definitions.

1.3.2 Insulation between Programs and Data, and Data Abstraction

In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file. By contrast, DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalog separately from the access programs. We call this property program-data independence. In some types of database systems, such as object-oriented and object-relational systems users can define operations on data as part of the database definitions. An operation (also called a function or method) is specified in two parts. The interface (or signature) of an operation includes the operation name and the data types of its arguments (or parameters).The implementation (or method) of the operation is specified separately and can be changed without affecting the interface. User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented. This may be termed program-operation independence. The characteristic that allows program-data independence and program-operation independence is called data abstraction. A DBMS provides users with a conceptual representation of data that does not include many of the details of how the data is stored or how the operations are implemented. Informally, a data model is a type of data abstraction that is used to provide this conceptual representation. The data model uses logical concepts, such as objects, their properties, and their interrelationships, that

may be easier for most users to understand than computer storage concepts. Hence, the data model hides storage and implementation details that are not of interest to most database users.

1.3.3 Support of Multiple Views of the Data

A database typically has many users, each of whom may require a different perspective or view of the database. A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored. Some users may not need to be aware of whether the data they refer to is stored or derived. A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

1.3.4 Sharing of Data and Multiuser Transaction Processing

A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct. For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger. These types of applications are generally called online transaction processing (OLTP) applications. A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently. The concept of a transaction has become central to many database applications. A transaction is an executing program or process that includes one or more database accesses, such as reading or updating of database records. Each transaction is supposed to execute a logically correct database access if executed in its entirety without interference from other transactions. The DBMS must enforce several transaction properties. The isolation

property ensures that each transaction appears to execute in isolation from other transactions, even though hundreds of transactions may be executing concurrently. The atomicity property ensures that either all the database operations in a transaction are executed or none are.

1.4 Applications Of DBMS

Applications where we use Database Management Systems are:

- **Telecom**: There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
- **Industry**: Where it is a manufacturing unit, warehouse or distribution center, each one needs a database to keep the records of ins and outs. For example distribution center should keep a track of the product units that supplied into the center as well as the products that got delivered out from the distribution center on each day; this is where DBMS comes into picture.
- **Banking System**: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
- **Education sector**: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.
- **Online shopping**: You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

1.5 Problem Description/Statement

Nowadays the hotel business is very tough and in order to compete it is necessary for staff especially managers to manage the working of the hotel. Our application deals with this issue by providing a mechanism for the managers to manage the rooms, employees , booking etc.

Our application also comes with machine learning component which allows easy access for the staff and for the managers. This enables them to handle the functioning of the hotel in a swift manner, this is essential in these times as customers are very particular about service time and quality. A hotel with bad service will get less stars on any hotel review application such as yelp, ibibo etc.

CHAPTER 2
INTRODUCTION TO MACHINE LEARNING

2.1 Introduction

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on explanatory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioral profiles for various entities and then used to find meaningful anomalies.

2.2 History

The name machine learning was coined in 1959 by Arthur Samuel. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data– such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders or malicious insiders working towards data breach and optical character recognition, learning to rank and computer vision.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 Hardware Requirements

The Hardware requirements are very minimal and the program can be run on most of the machines.

Processor	:	Intel Pentium 4 processor
Processor Speed	:	2.4 GHz
RAM	:	1 GB
Storage Space	:	40 GB
Monitor Resolution	:	1024*768 or 1336*768 or 1280*1024

3.2 Software Requirements

Operating System : Windows XP, Windows 7, Windows 8, Windows 8.1 or Windows 10, Ubuntu

IDE : Anaconda/

3.3 Functional Requirements

3.3.1 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render them 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. It 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 /> introduce content into the page directly. Others such as ...

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 affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

3.3.2 CSS & Bootstrap

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. 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 presentation and content, including aspects 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.

Bootstrap is a free and open-source front-end web framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

Bootstrap is modular and consists of a series of less stylesheets that implement the various components of the toolkit. These stylesheets are generally compiled into a bundle and included in web pages, but individual

components can be included or removed. Bootstrap provides a number of configuration variables that control things such as color and padding of various components

Grid system and responsive design comes standard with an 1170 pixel wide grid layout. Alternatively, the developer can use a variable-width layout. For both cases, the toolkit has four variations to make use of different resolutions and types of devices: mobile phones, portrait and landscape, tablets and PCs with low and high resolution. Each variation adjusts the width of the columns.

Bootstrap provides a set of stylesheets that provide basic style definitions for all key HTML components. These provide a uniform, modern appearance for formatting text, tables and form elements

3.3.3 JavaScript

JavaScript (sometimes abbreviated JS) is a prototype-based scripting language that is dynamic, weakly typed, general purpose programming language and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

JavaScript was formalized in the ECMA Script language standard and is primarily used in the form of client-side JavaScript, implemented as part of a Web browser in order to provide enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment.

JavaScript's use in applications outside Web pages for example in PDF documents, site-specific browsers, and desktop widgets is also significant.

In this application, JavaScript is used for validation purpose like text box validation, email validation, phone number validation. JavaScript is the good tool for validating the web-applications.

3.3.4 What is SQLite?

- SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.
- SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen()
- SQLite is a compact library. With all features enabled, the library size can be less than 500KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used, SQLite can be faster than direct filesystem I/O

3.3.5 Python and Flask

Flask is a micro web framework written in Python It is classified as a microframe work because it does not require particular tools or libraries.[4]It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.[5]Flask is commonly used with MongoDB which allows it more control over databases and history.

Applications that use the Flask framework include Pinterest, LinkedIn and the community web page for Flask itself

3.3.6 Amazon Web Server

Amazon Web Services (**AWS**) is a subsidiary of Amazon.com that provides on demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis. The technology allows subscribers to have at their disposal a virtual cluster of computers, available all the time, through the Internet. AWS's version of virtual computers emulate most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, hard-disk/SSD storage); a choice of operating systems; networking; and pre-loaded application software such as Web Servers, databases, crms etc. Each AWS system also virtualizes its console I/O (keyboard, display, and mouse), allowing AWS subscribers to connect to their AWS system using a modern browser. The browser acts as a window into the virtual computer, letting subscribers login, configure and use their virtual systems just as they would a real physical computer. They can choose to deploy their AWS systems to provide internet-based services for themselves and their customers.

CHAPTER 4

IMPLEMENTATION

4.1 Database Connectivity

Code for connection

```
try:
        with sql.connect("sqldb.db") as con:
                cur = con.cursor()
                cur.execute("sql statement")
                con.commit()
except:
        con.rollback()
finally:
        con.close()
```

Architecture used (4-TIER architecture)

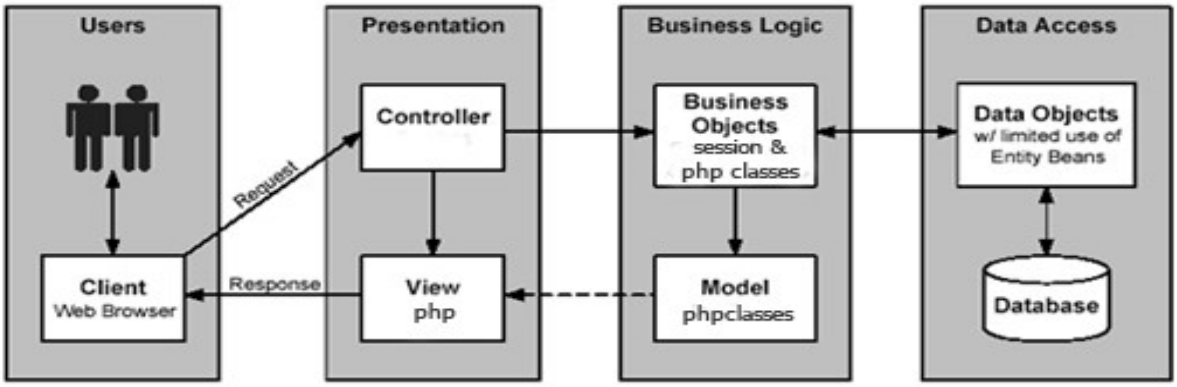

Fig: Architecture used

Four Tier architecture is a client–server architecture in which presentation, application processing, and data management functions are physically separated. Four-tier application architecture provides a model by which developers can create flexible and reusable applications. By

segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application.

Presentation layer

This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. In simple terms, it is a layer which users can access directly (such as a web page, or an operating system's GUI).

Business layer

Business layer or domain logic is the part of the program that encodes the real-world business rules that determine how data can be created, stored, and changed. It is contrasted with the remainder of the software that might be concerned with lower-level details of managing a database or displaying the user interface, system infrastructure, or generally connecting various parts of the program.

Data access layer

A Data Access Layer (DAL) in computer software, is a layer of a computer program which provides simplified access to data stored in persistent storage.

For example, the DAL might return a reference to an object (in terms of object-oriented programming) complete with its attributes instead of a row of fields from a database table. This allows the client (or user) modules to be created with a higher level of abstraction. This kind of model could be implemented by creating a class of data access methods that directly reference a corresponding set of database stored procedures. Another implementation could potentially retrieve or write records to or from a file

system. The DAL hides this complexity of the underlying data store from the external world.

Control layer

The control layer is responsible for communication between business and presentation layer. It connects the logic and data with each other and gives a better connectivity and separation between layers.

4.2 Pseudo Code for Major Functionalities

Front Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <html>
 <head>
 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
 <title>Four Seasons</title>
 <meta name="keywords" content="" />
 <meta name="description" content="" />
       <link href="{{url_for('static',filename='style/main.css')}}" rel="stylesheet" type="text/css" media="screen" />
 <link href="{{url_for('static',filename='style/animate.css')}}" rel="stylesheet" >
 </head>
 <body background="{{url_for('static',filename='hotel12.jpg')}}">

    <div class ="supreme">
    <h1>NOT FOUND </h1>
    <p> {{message}} </p>
  </div>
  </body>
</html>
```

Add employees

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <html lang=en>
 <head>
 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
 <title>Add Employee</title>
 <meta name="keywords" content="" />
 <meta name="description" content="" />
       <link href="{{url_for('static',filename='style/main.css')}}" rel="stylesheet" type="text/css" media="screen" />
 <link href="{{url_for('static',filename='style/animate.css')}}" rel="stylesheet" >
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-rc.2/css/materialize.min.css">

  <!-- Compiled and minified JavaScript -->
                <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-rc.2/js/materialize.min.js"></script>
 </head>
 <body background="{{url_for('static',filename='hotel12.jpg')}}">


   {% if error is defined %}
     <div>
       <h1>{{ error }}</h1>
     </div>
   {% else %}
     <div>
      <div>
        <h1>Enter the employee details</h1>
       <form action ="/employees/addemp/" method="post">
          <div class ="col-3">
```

```
            Employees ID <input type = "text" name ="empid" /><br>
            Employee Name <input type = "text" name ="empname"/><br>
            Age<input type = "text" name ="empage" /><br>
            Designation<input type = "text" name ="empdes" /><br>
            Phone<input type = "text" name ="empphone" /><br>
            Address<input type = "text" name ="empaddress" />
          </div>
          <div class ="col-4">
            <input type ="submit" value ="Enter"/>
          </div>
        </form>
        </div>

    <div>{{ msg }}</div>
  {% endif %}
  </body>
</html>
```

Add Room

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html lang=en>

<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Add Rooms</title>
  <meta name="keywords" content="" />
  <meta name="description" content="" />
        <link  href="{{url_for('static',filename='style/main.css')}}"  rel="stylesheet"
type="text/css" media="screen" />
  <link href="{{url_for('static',filename='style/animate.css')}}" rel="stylesheet">
```

```
                                                    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-
rc.2/css/materialize.min.css">

    <!-- Compiled and minified JavaScript -->
                <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-
rc.2/js/materialize.min.js"></script>
</head>

<body background="{{url_for('static',filename='hotel12.jpg')}}">


   {% if error is defined %}
   <div>
      <h1>{{ error }}</h1>
   </div>
   {% else %}
   <div>
      <div>
         <h1>Enter the room details</h1>
         <form action="/rooms/addrooms/" method="post">
            <div class="col-3">
               Room no
               <input type="text" name="roomno" />
               <br>

            </div>
            <label>
               enter the room type
               <br>
               <select id="serviceQuality" name="type">
                  <option disabled selected value="0">-- Choose an option --</option>
                  <option value="3">type 5</option>
                  <option value="2">type 4</option>
```

```
                <option value="1.5">type 3</option>
                <option value="1">type 2</option>
                <option value="0.5">type 1</option>
              </select>
            </label>
            <div class="col-4">
              <input type="submit" value="Enter" />
            </div>
          </form>
        </div>

        <div>{{ msg }}</div>
        {% endif %}
</body>

</html>
```

Rooms

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <html>
 <head>
 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
 <title>Rooms</title>
 <meta name="keywords" content="" />
 <meta name="description" content="" />
       <link    href="{{url_for('static',filename='style/main.css')}}"    rel="stylesheet"
type="text/css" media="screen" />
 <link href="{{url_for('static',filename='style/animate.css')}}" rel="stylesheet" >
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-
rc.2/css/materialize.min.css">
```

```
  <!-- Compiled and minified JavaScript -->
                <script       src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-
rc.2/js/materialize.min.js"></script>
  </head>
  <body background="{{url_for('static',filename='hotel12.jpg')}}">

     <div class ="supreme">
     <h1>This shows list of rooms</h1>
     <a href="/rooms/addrooms/"><button>Add</button></a>
     <a href="/rooms/lstroom/"><button>Manage</button></a>
    </body>
          </html>
```

CHAPTER 5

RESULTS, SNAPSHOTS & DISCUSSIONS

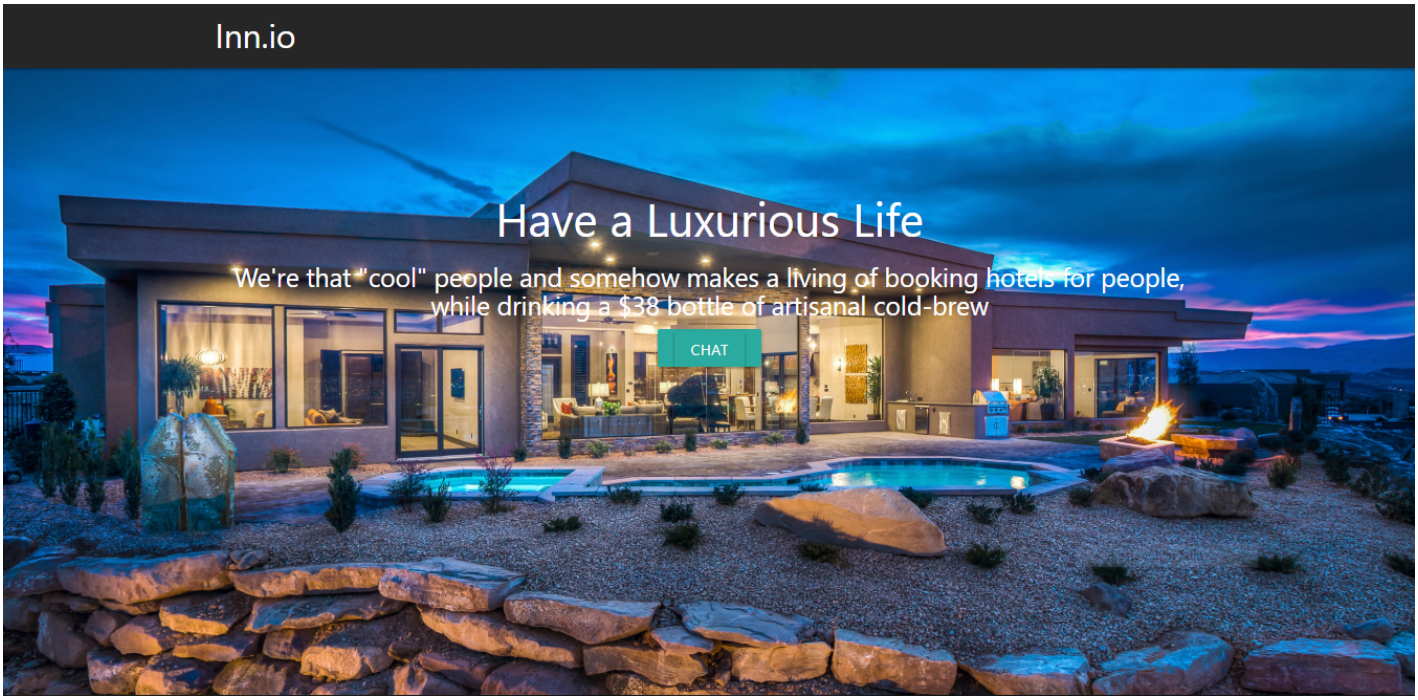

Figure 5.1

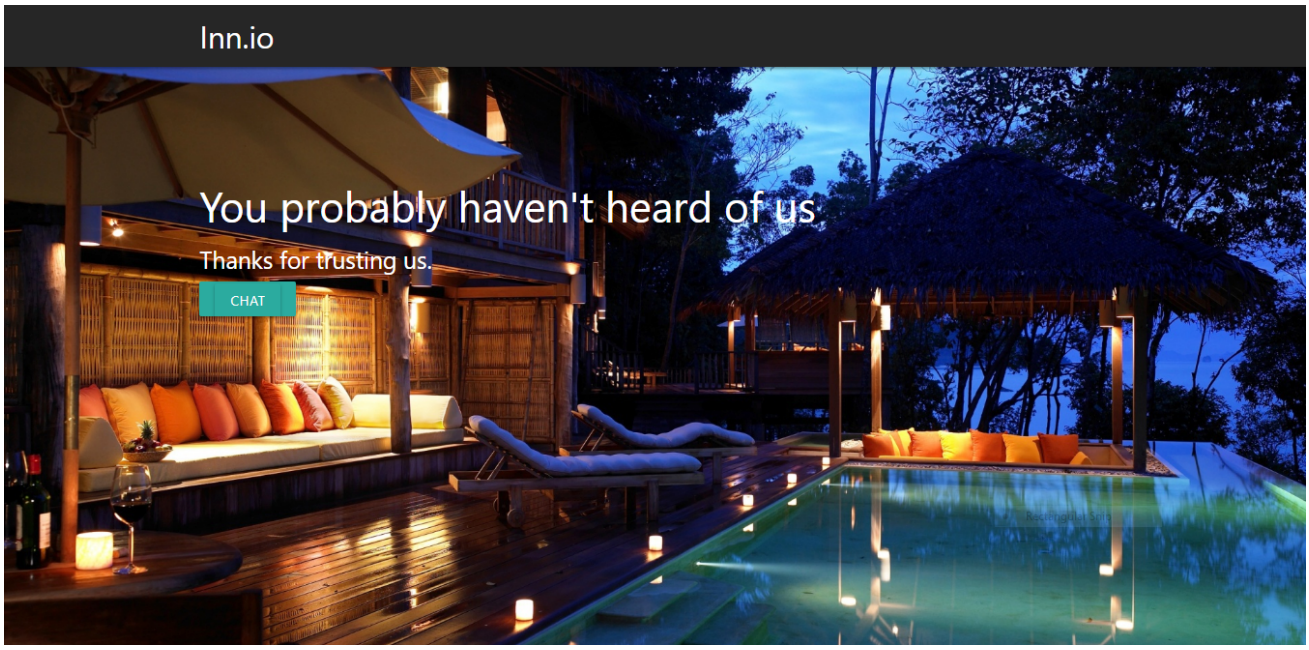

Figure 5.2

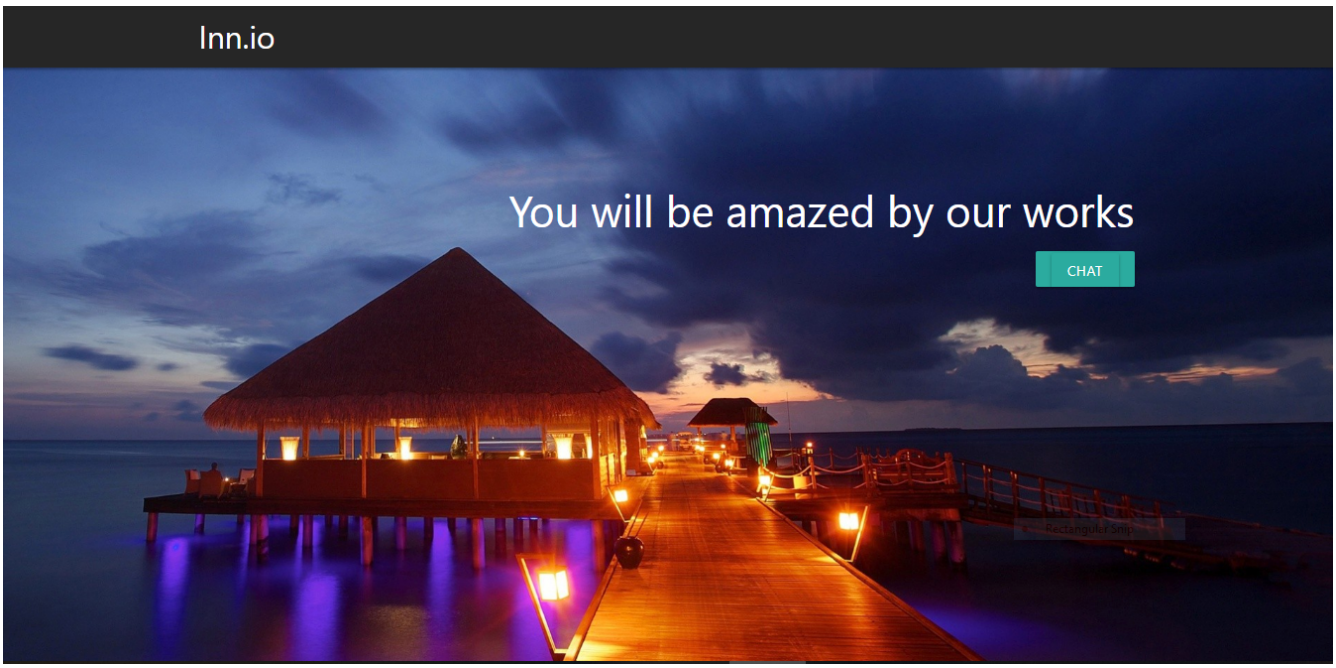

Figure 5.3

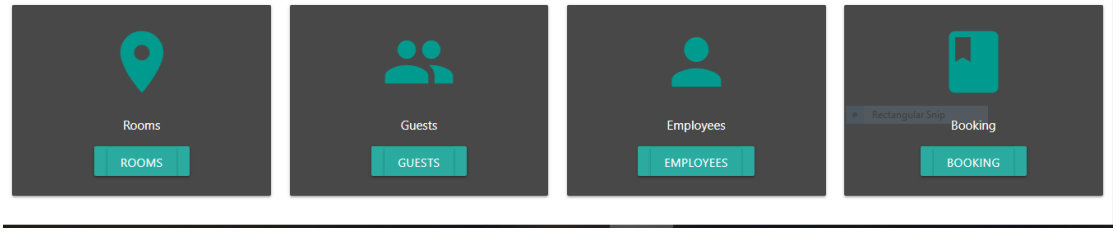

Figure 5.4

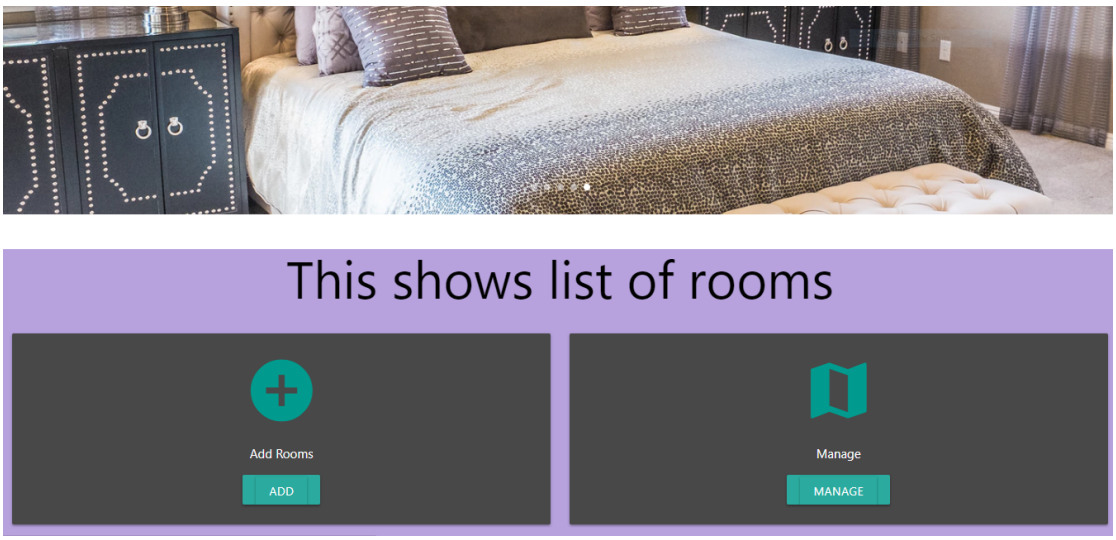

Figure 5.5

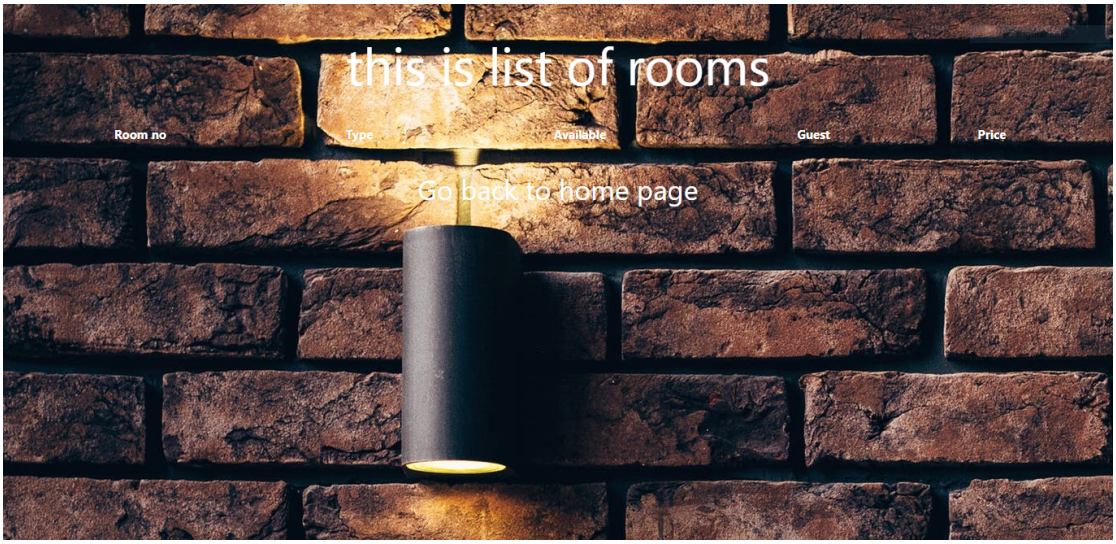

Figure 5.6

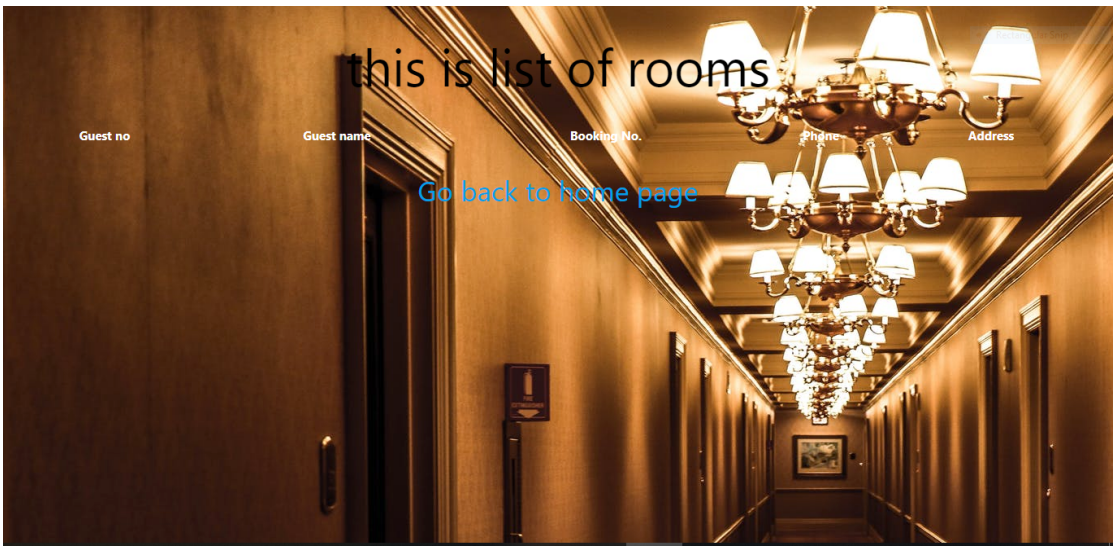

Figure 5.7

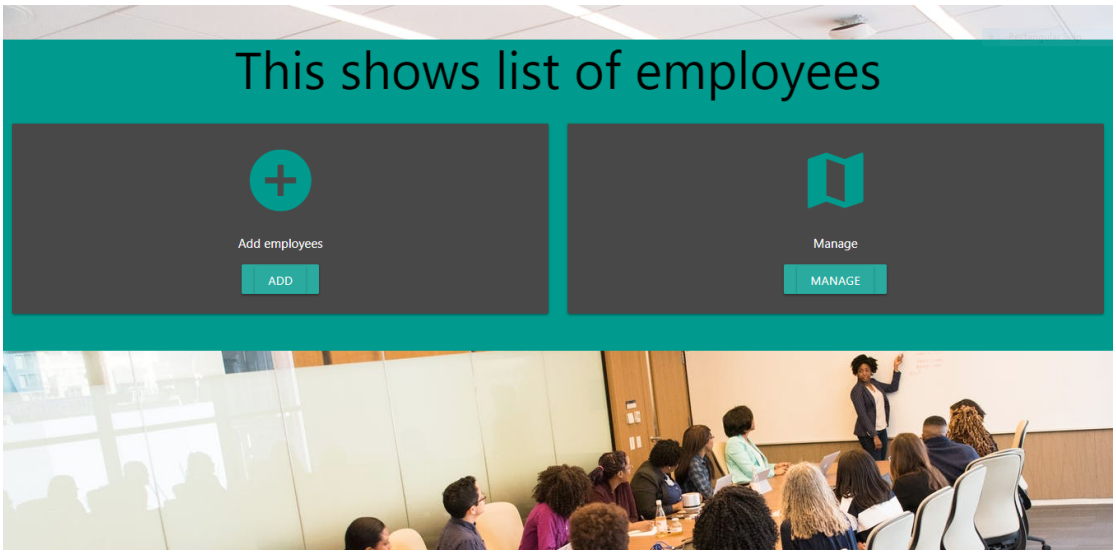

Figure 5.8

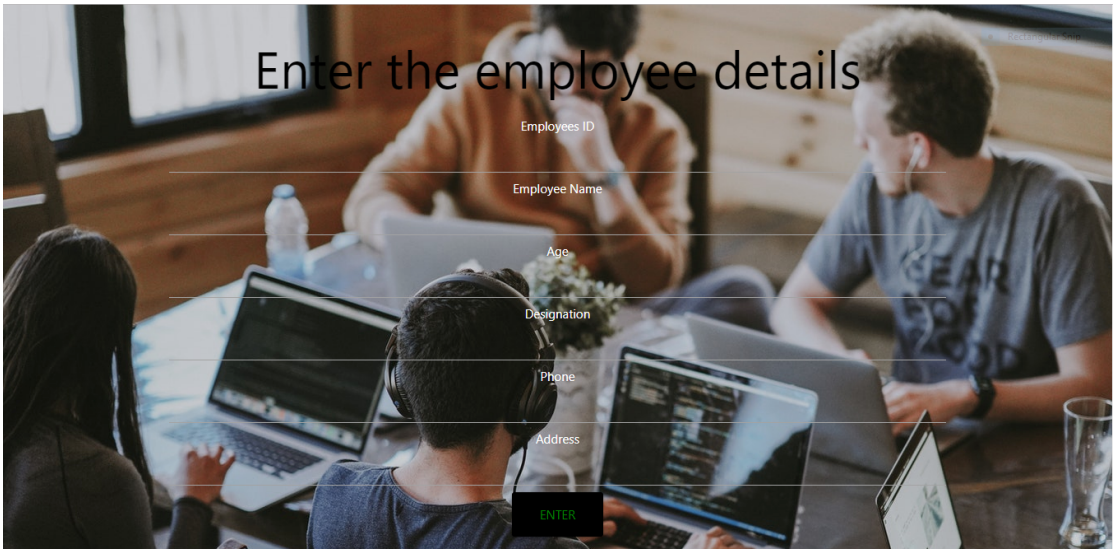

Figure 5.9

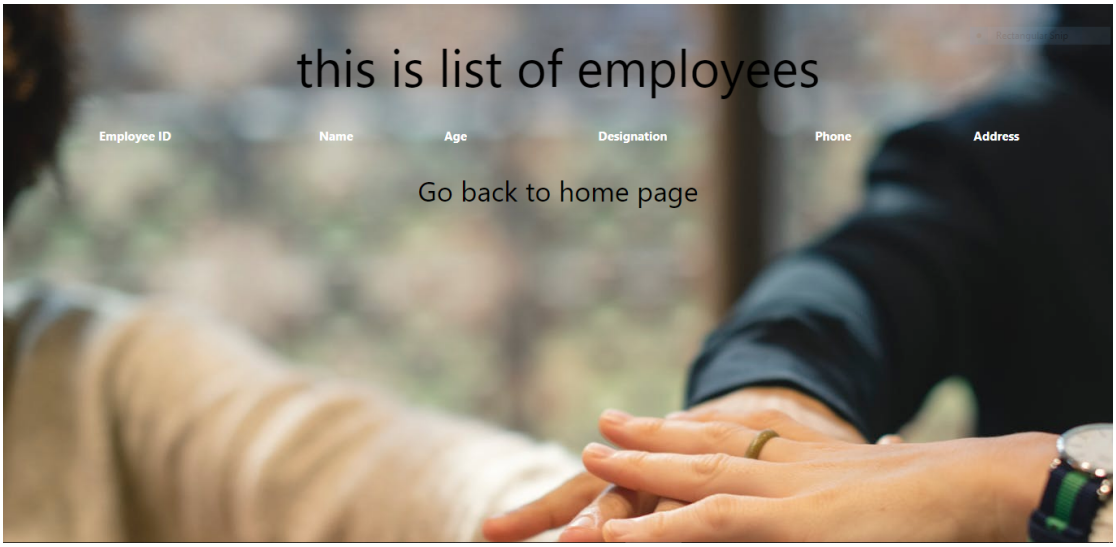

Figure 5.10

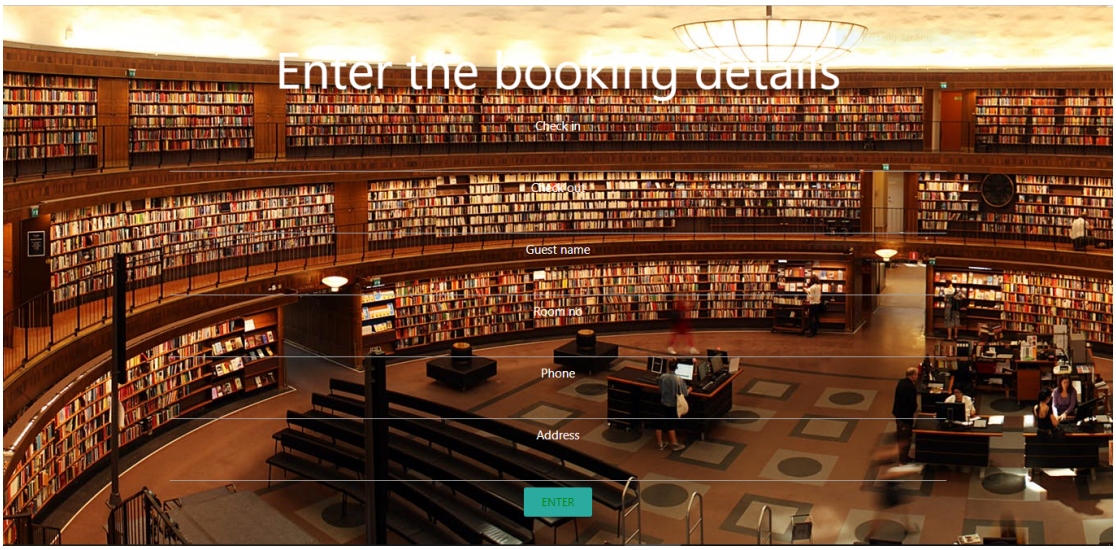

Figure 5.11

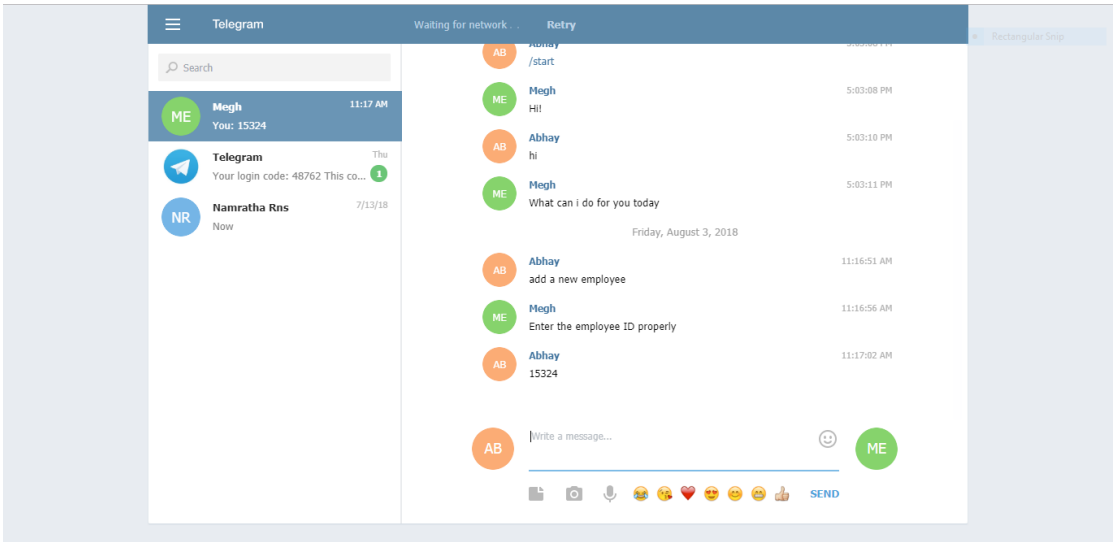

Figure 5.12

Discussion

The above snapshots are self-explanatory. They contain the different screenshots of our website. They include the Website pages and how the manager can add or remove employees , add or remove rooms, manage the rooms allocated. It shows the various details of the employee which are considered before adding them.

CHAPTER 6

CONCLUSION & FUTURE ENHANCEMENTS

6.1 Conclusion

Computer has got clear advantage over the manual system. The computerized system is more reliable, efficient and fast at the end of the project, I can say that computer play a very crucial role in the development of firm. All the daily reports generated by the system are to be checked by the concerned official so as to ensure that all the transactions have been put through in appropriate accounts and this is tallied with the new vouchers.

We have built a web application on hotel management system where the manager can manage the employees and rooms. It has a machine learning component which is implemented using IBM Watson and it is also deployed on the cloud.
We were able to gain knowledge on Web development, IBM Watson, Telegram and back end using SQLite.

6.2 Future Enhancements

We are currently managing employees. We hope to provide security to the system so that it can be actually implemented and is secure.
We can also try to add a feature which allows the customer to order food. Manage food orders from the hotel and the bills to be generated.
We can add a feature for number of days of stay and cumulative bills and also house keeping for maintenance and feedback from the customer can be taken.

CHAPTER 7

BIBLIOGRAPHY

[1] Wikipedia

https://en.wikipedia.org/wiki/flask

https://en.wikipedia.org/wiki/SQLite

[2] W3Schools

https://www.w3schools.com/html/

https://www.w3schools.com/css/

https://www.w3schools.com/bootstrap/

https://www.w3schools.com/sql/

[3] Internshala Web Course

https://trainings.internshala.com/web-development