## ONLINE ROOM RENTAL SYSTEM

**A Project Report Submitted In Partial fulfillment of the Requirements**

## for the degree of

Master in Computer Application

**By**

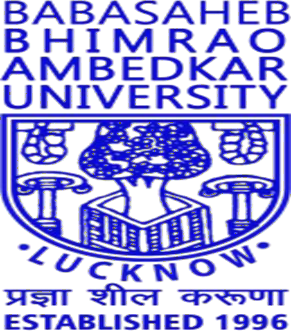
**Name singh**

### (Enrolment No. 605/18) (Roll No. 180902 )

Under the Supervision of

**Mrs. Name singh Mrs. Name Singh**

**Department of Computer Science**



## Babasaheb Bhimrao Ambedkar University

(A Central University)

## Lucknow, UP

**June, 2021**

## CANDIDATE’S DECLARATION

I hereby declare that the work presented in this report entitled “Online Room Rental system”,

was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution.

I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

## Name: Name singh Enroll No: 605/18 Roll No: 00000

**CERTIFICATE**

This is to certify that Name Singh(Enroll No: 605/18 ) has carried out the project work presented in this report entitled “Online Room Rental System” for the award of Master of Computer Application from Babasaheb Bhimrao Ambedkar Uiversity, Lucknow under my/our supervision. The project embodies result of original work, and studies are carried out by the student himself and the contents of the project do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/ Institution.

**Signature**

**Mrs. Name**

**Assistant Professor Mrs. Versha Verma**

**Signature**

**(Prof. Sanjay K. Dwivedi, DCS) Head of the Department**

## INTERNSHIP CERTIFICATE



**ACKNOWLEDGEMENT**

An assignment puts to litmus test of an individual’s knowledge, experience and thus sole effect of an individual are not sufficient to accomplish the successful completion of project involving interests and effort of many people . so this becomes my obligator thanks to all of them.

I would also like to thank you Mrs.Deepa Raj and Mrs. Versha Verma for his constant support and guidance all through the making the project. I acknowledge unforgettable guidance, invaluable suggestion and constant moral support to my classmates during this period. I am also grateful to technical staff for providing me the right kind of environment and resources whenever needed.

My sincere thanks to all staff members for their contribution in coordinating and Co-operating with me for successful completion of my project.

## Ashakti Agnihotri

**Date: 30/06/2020 Place: Lucknow**

## TABLE OF CONTENTS

INTRODUCTION PROBLEM DEFINITION OBJECTIVE

REQUIREMENT SPECIFICATION MODULE DISCRIPTION PROBLEM FORMULATION FEASIBILITY STUDY

PROBLEM SPECIFICATION SYSTEM ARCHETECTURE DESIGN PROBLEM SPECIFICATION

MANAGEMENT SYSTEM ARCHETECTURE DESIGN TESTING

ER DIAGRAMS

DATA FLOW DIAGRAM

PROJECT PLANING AND SCHEDULING DATABASE TABLES

PROJECT SCREENSHOT COADING OF PROJECT LIMITATIONS CONCLUSION

FUTURE SCOPE OF THE PROJECT REFERENCE

# INTRODUCTION

Room booking is a part of e-commerce. E-commerce or business through net means search room, book room and servicing of products or services over electronic systems such as the Internet and other computer networks. Thus if we own a business we need to upload menu online to attract potential customers.

The online room booking system gives booking ability to increase

booking and expand their business by giving customers the facility to book room online.

With an online book menu ordering system, customers can book room online 24

\*7. Thus it is a simple, fast and convenient room booking system giving an edge over the competition at an affordable price.

An Online Room Rental System will provide the Information about Rooms/Flats/Houses which is available for Rent.

Room Rental system will make easy to find the location of Rooms/Flats/Houses, select no. of rooms and other facts by the renter.

It will make easy to upload the location, contact No., expected rent, No. of rooms, Facilities and information by landlord/room-master.

With [Online Room Rental System], we can set up we room online and the customers can easily book room with a simple mouse click. Also with a book room online we can easily track the facility, maintain customer's database and

improve the room facility service. We can receive order through e-mails/ fax or directly view on internet.

Internet has seen a tremendous growth in terms of coverage and awareness.

So giving the business an online presence has become very crucial and important.

With [Online Room Booking System], we can set up we room online and the customers can easily book room with a simple mouse click. Also with a book room online we can easily track the facility, maintain customer's database and improve the room facility service. We can receive order through directly view on internet.

# PROBLEM DEFINITION

At present there are many online room booking systems that maintain their day to day transactions manually. These have a number of consumers, staff as a result they need to track of all these clients requirements. At this point ROOM BOOKING SYSTEM will play an important role in helping the admin to perform all these operation on a single click. This project will handle all the necessary data as well as every minute details of the campus and properly.

#### Purpose

The purpose of this document is to specify requirements and to give guidelines for the development of above said project. In particular it gives guidelines on how to prepare the above said project. The old system was suffering from a series of the drawbacks. Since whole system was to be maintained with hands the process of keeping, maintaining and retrieving the information was very tedious and lengthy. The records were never used to be in a systematic order as a result there used to be lot of difficulties in associating any particular transaction with a particular context. If any information was to be found it was required to go through different registers, documents there would never exist anything like report generation. There would be unnecessary consumption of time while entering records and retrieving the records. One more problem was that it was very difficult to find errors while entering records as a result if one record was entered then it was difficult to update these records.

#### Scope

As this is generic software it can be used by a wide variety of room to automate the process of manually maintaining the records related to the subject of maintaining the entries and updating these records that too in a single click.

# OBJECTIVE

Today’s world is computer world because most of work is doing with the help of computer. Dependency on computer is behind the few reasons. We cannot easily manage to store large number of data or information single handle. If we will be need some information or data in urgency then we cannot manage in manually these works are very difficult if we cannot use computer.

As the generic software it can be used by a wide verity of rooms to automate the process of manually maintaining records related to the subject of maintain the records of each personnel in room.

This software is basically updating the manual work of systems. So that organization can manage their record in efficient and organize them.

The main objective of the project is to develop software that facilitates the data storage, data maintenance and its retrieval for the restaurant in an igneous way.

To store the records of the customers. To save manpower.

It will speed the processing of data and transaction.

It will provide best security features such as provisions of passwords To develop a user friendly system that requires minimal user training.

#### System Objective

Today’s world is computer world because most of work is doing with the help of

computer. Dependency on computer is behind the few reasons. We cannot easily

manage to store large number of data or information single handle. If we will be need some information or data in urgency then we cannot manage in manually these works are very difficult if we cannot use computer.

#### System Context

This section clearly depicts the environment and boundaries of ROOM BOOKING SYSTEM and the entities with which it interacts. It helps us see how the system fits into the existing scheme of things. What the system will do by itself.

#### Functional Requirement

This Software must request Username and Password for access to data, only after authentication will allow access to the system. The Software must allow input of products data from administrator and secured access.

#### Non-Functional Requirement

In this Software Input error will be returned in red with appropriate message box. System should automatically update after every transaction.

# REQUIREMENTS SEPECIFICATIONS

***HARDWARE AND SOFTWARE SPECIFICATION***

* 1. **Hardware Requirements**
     1. Pentium IV Processor
     2. 512 MB RAM
     3. 40 GB HDD
     4. Color Monitor
     5. Keyboard, Mouse

### Software Requirements

* + 1. Python 3 (Any version)
    2. Any text editor (PyCharm, Notepad++ etc.)

# 5. MODULE DISCRIPTION

### Registration (User)

The User fills the registration form by giving the personal information and successfully registers with the website.

1. **Login**

The ROOM BOOKING SYSTEM first activates the login page. Here the user enters USER NAME AND PASSWORD and the system starts authentication process in which the USER NAME AND PASSWORD is matched with the existing USER NAME AND PASSWORD in the database. If the password matches then it is allowed to the main page else it warns the user for invalid USER NAME AND PASSWORD.

After the successful authentication the system activated menus. The activity log also prepared for failure and security There are two types of users using this software i.e., admin, student, faculties and staff.

Admin (Description/Images/Status) :After entering into admin home page will add new room by filling add room form, by providing sufficient details about room such as room number, available dates, rent etc and upload the image of that room or house .It consists of following sub modules:1.

1. **Update Room**

Admin can update the existing room details such as rent, available dates, etc.

***Update Product:*** Admin can update the existing room details such as rent, available dates, etc.

***Delete Product:*** admin can also delete the room.

1. **Data base maintenance**

The data provided by the admin such as room details, personal details, etc. and data provided by the customer such as feedback and booking details will be maintained in a data base.

1. **Searching and Booking the room**

The Customer after accessing the site searches for room, if he/she finds the required room then he/she need to fill the signup form and submit to the database.

1. **Authentication**

Authentication is nothing but providing security to the system . Here everyone must enter into the system through login page.

# PROBLEM FORMULATION

#### Introduction

Problem introduction or problem starting is the starting point of the software development activity. The objective of this statement is to answer: Exactly *what must the system do*? The software project is initiated by the client’s need. In the beginning, these needs are on the minds of various people in the client’s organization. The analyst has to identify the requirements by talking to the people and understanding to their needs. It goes without saying that an accurate and through understanding of software requirement are essentials to the success of software development effort. All further development like system analysis.

System design and coding will depends on how accurate and well understood the requirements are poorly analyzed and specified software will disappoint the user and will bring brief to the developer. No matter how well designed and well appearances are often deceiving. Chances of misinterpretation are very high, ambiguity is probable and communication gap between customer and developer is bound to bring confusions. Requirements understanding begin with a clear and concise heading stating in sentence the task to be performed. Then requirements are describe in a technical manner precise statement.

# Feasibility study

All projects are feasible given unlimited resources and infinite time! Unfortunately, the development of computer based system is more likely to be plagued by a scarcity of resources. It is both necessary and prudent to evaluate the feasibility of the project at the earliest possible time. Months or years of effort, Money loss and untold professional embarrassment can be averted I few better understand the project at its study time.

This type of study determines if an application can and should be developed. Once it has been determining that, application is feasible. After that analyst can go ahead and prepares the project specification, which finalizes project requirements. Feasibility studies are undertaken within tight time constraints.

1. Technical Feasibility
2. Operational Feasibility
3. Economic Feasibility
4. Legal Feasibility

#### Technical Feasibility

As we know the technical feasibility is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may vary considerably, but might include:

The facility to produce outputs of advertisements, booking and mailing in a given time for ease of use.

Response time under certain condition is minimal.

Ability to process a certain volume of transaction at a particular speed. Facility to communicate data to distinct location.

In examining the technical feasibility, configuration of the system is given more importance than the actual make of hardware. The configuration should give the complete picture about the system’s requirements- how many workstations are required, how these units are interconnected so that they could operate and communicate smoothly.

#### Operational Feasibility

Proposed projects are beneficial only if they can be turned into information system that will meet the financial management requirements of the business/organization. This test of feasibility asks if the system will work when it developed and installed. Are there major barriers to implementation?

Some of the important questions that are useful to test the operational feasibility of a project are given below:

Is there sufficient support for the project from the implementation? From user? If the present system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.

Are current business methods acceptable to the user? If they are not, user may welcome a change that will bring about a more operational and useful system.

Have the user been involved in the planning and development of the Project? If they are involved at the earliest stage of project development, the chances of resistance can be possibly reduced.

Will the proposed system cause harm? Will it produce poorer result in any case or area?

Will the performance of staff member fall down after implementation? Issue that

Appears to be quite minor at the early stage can grow into major problem after Implementation.

#### Economical Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of the proposed system. More commonly known as cost/benefits analysis, the procedure is to determine the benefits and savings that are expected from the purposed system and compared with costs.

If benefits outweigh cost, a decision is taken to design and implement the system. Otherwise, further justification or alternative of the proposed system will have to be made if it has a chance of being approved. This is an ongoing effort that improves in accuracy at each phase of the system life cycle. The analysis part also clears the doubt of economic problems which could be possible in developing the system. As already mentioned that the company has to just pay the developed software cost and not other investment is needed at the time of implementation of the new system as the preliminary requirements already exist in the company.

#### Legal Feasibility

In the legal feasibility is necessary to check that the software we are going to develop is legally correct which means that the ideas which we have taken for the proposed system will be legally implemented or not so, it is also an important step in feasibility study.

# PROBLEM SPECIFICATION

The definition of our problem lies in manual system and a fully automated system.

#### Manual System

The system is very time consuming and lazy. This system is more prone to error and sometimes the approach to various problems is unstructured.

#### Technical System

With the advent of latest technology if we do not update our system then our business result in losses gradually with time. The technical system contains the tools of latest trend i.e. computers, printers, FAX, Internet etc the system with the technology are very fast, accurate, user friendly and reliable.

#### Need of ROOM BOOKING System

ROOM BOOKING SYSTEM website is very needy for various rooms. This website helps them maintain day to day entries of the customers who are their client or those who desire to be, daily bases booking of the rooms in computer.

#### The Proposed System

The proposed system helps them in many ways. It helps them do billing very easily. Account maintenance also becomes easier. They can keep track of their sales, stocks of the supplements and account details of their customers regarding their packages and many more. The software is provided with all the master entries to enter any new product, or customer or trainer to add or modify and delete.

#### Existing System

At present various Room booking systems maintain their day to day transactions manually. These have thousands of products they need to track of all these products to check the stock, order date etc. To find the product is also another risky job and this also applies for in finding the details of the customer. So proper system is required. They need full pledged software to maintain their day to day transactions.

# SYSTEM ARCHETECTURE DESIGN

### Fundamental design Concepts

A set of fundamental design concepts are evolved over the past three decades. Although the degree of interest in each concept has varied over the years, each has stood the test of time. Each provides the software designer with a foundation from which more sophisticated design methods can be applied. Fundamental design concepts provide the necessary framework for “getting it right”.

Abstraction

Abstraction permits one to concentrate on a problem at some level of generalization without regard to irrelevant low level details, use of abstraction also permits one to work with concepts and terms that are familiar in the problem environment without having to transform them to an unfamiliar structure. Two types of abstraction are there, one is procedural abstraction and data abstraction. A procedural abstraction is a named sequence of instructions that has a specific and limited function. A data abstraction is a named collection of data that describes a data object.

### Modularity

Modularity is the single attribute software that allows a program to be intellectually manageable. Software architecture embodies modularity, that is, software is divided into named and addressable components, called modules that are integrated to satisfy problem requirements.

### Software Architecture

Software Architecture alludes to “the overall structure of the software and the ways in which that structure provides conceptual integrity for a system”. Control hierarchy also called program structure”, represents the organization of control. The tree structure used to represent the control hierarchy.

### Structural Partitioning

The program structure should be partitioned both horizontally and vertically. Horizontal partitioning defines separate branches of the modular hierarchy for each major program function, Vertical partitioning called factoring, suggest that control and work should be distributes top-down in the program architecture. Top level modules should perform control functions and do little actual processing work. Modules reside low in the architecture should be the workers, performing all input, computational, an output tasks.

### Data Structure

Data Structure is a representation of logical relationship among individual elements of data. Because the structure of information will invariably affects the final procedural design, data structure is very important as the program structure to the representation of the software architecture. Data structure dictates the organization, methods of access, degree of associatively, and processing alternatives for information. The organization and complexity of a data structure

are limited only by the ingenuity of the designer. Scalar item array and linked list are some of the representations of the data structure.

### Software Procedure

Program structure defines control hierarchy without regard to the sequence of processing and decisions. Software procedure focuses on the processing details of each module individually. Procedure must provide a precise specification of processing, including sequence of events, exact, decision points, repetitive operations and even data organization / structure. Information hiding suggests

that modules be “characterized by design decisions that hide from all others.” In other words, modules should be specified and designed so that information contained within module is inaccessible to other module.

Design is defining a model of the new system and continues by converting this model to a new system. The method is used to convert the model of the proposed system into computer specification. Data models are converted to a database and processes and flows to user procedures and computer programs. Design proposes the new system that meets these requirements. This new system may be built by a fresh or by changing the existing system. The detailed design starts with three activities, database design, user design and program design. Database design uses conceptual data model to produce a database design. User procedure design uses those parts of the DFD outside the automation boundary to design user procedures.

Our system design includes mainly the design of the UML diagrams and constructing the code. The code construction and its description are also given as separate topic. System design may also include the data flow diagrams (DFD) which models a system by using external entities from which data flows to a process, which transforms the data and creates output data flows which go to other processes or external entities or data stores. Stored data may also flow to

processes as inputs. The main merit of DFD is that it can provide an overview of what data a system would process, what transformation of data are done, what data are stored and which stored data are used, and where the results flow. The graphical representation of the system makes it a good communication tool between user and an analyst on the one hand and the analyst and the system designer on the other hand.

DFD is mainly used for the representation of the dataflow in the system, which uses database for the storing and retrieving the data. But as our system doesn’t use any database, we haven’t provided any data flow diagram here.

### Component Diagram

The component diagram contains components and dependencies. Components represent the physical packaging of a module of code. The dependencies between the components show how changes made to one component may affect the other components in the system.Dependencies in a component diagram are represented by a dashed line between two or more components. Component diagrams can also show the interfaces used by the components to communicate to each other.

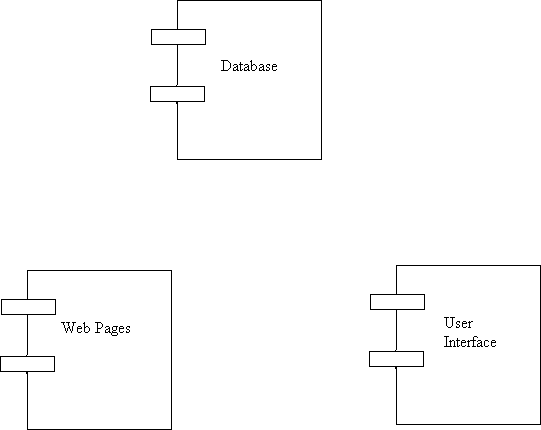


Figure of Component Diagram

### Deployment Diagram

Deployment and component diagram combines the features of both diagrams into one diagram.

The deployment diagram contains nodes and connections. A node usually represents a piece of hardware in the system. A connection depicts the communication path used by the hardware to communicate and usually indicates a method such as TCP/IP.

DB

UI

Web Server

Web pages

Figure of Deployment Diagram

### PROBLEM SPECIFICATION

The definition of our problem lies in manual management system and a fully automated management system.

#### Manual Management system

The management system is very time consuming and lazy. This management system is more prone to error and sometimes the approach to various problems is unstructured.

#### Technical Management system

With the advent of latest technology if we do not update our management system then our business result in losses gradually with time. The technical management system contains the tools of latest trend i.e. computers, printers, FAX, Internet etc the management system with the technology are very fast, accurate, user friendly and reliable.

#### The Proposed Management system

The proposed management system helps them in many ways. It helps them do billing very easily. Account maintenance also becomes easier. They can keep track of their sales, stocks of the supplements and account details of their students regarding their packages and many more. The software is provided with all the master entries to enter any new product, or student or trainer to add or modify and delete.

### MANAGEMENT SYSTEM ARCHETECTURE DESIGN

**1 DEFINING A SYSTEM**

Collection of component, which are interconnected, and work together to realize some objective, from a system. There are three components in every system, namely input, processing and output

Input Output

Processing

### SYSTEM DEVELOPMENT LIFE CYCLE:-

The **System development life cycle (SDLC)**, or **Software development process** in [systems](http://en.wikipedia.org/wiki/Systems_engineering) [engineering,](http://en.wikipedia.org/wiki/Systems_engineering) [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is a process of creating or altering information systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems. In software engineering, the SDLC concept underpins many kinds of [software](http://en.wikipedia.org/wiki/Software_development_methodologies) [development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process.](http://en.wikipedia.org/wiki/Software_development_process)

Broadly, following are the different activities to be considered while defining the system development life cycle for the said project:

Problem Definition System Analysis

Study of existing system Drawback of the existing system Proposed system

System Requirement study Data flow analysis Feasibility study

System design

Input Design (Database & Forms) Updating

Query /Report design Administration Testing Implementation Maintenance

### SYSTEM ANALYSIS

**Systems analysis** is the study of sets of [interacting](http://en.wikipedia.org/wiki/Interaction) [entities](http://en.wikipedia.org/wiki/Entity), including computer systems analysis. This field is closely related to [requirements analysis](http://en.wikipedia.org/wiki/Requirement_analysis) or [operations research](http://en.wikipedia.org/wiki/Operations_research). It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made.

System development can generally be thought of having two major components: systems analysis and systems design. In System Analysis more emphasis is given to understanding the details of an existing system or a proposed one and then deciding whether the proposed system is desirable or not and whether the existing system needs improvements. Thus, system analysis is the process of investigating a system, identifying problems, and using the information to recommend improvement to the system.

### SYSTEM DESIGN

Systems design is the process of defining the architecture, components, modules, interfaces, and [data](http://en.wikipedia.org/wiki/Data) for a [system](http://en.wikipedia.org/wiki/System) to satisfy specified [requirements](http://en.wikipedia.org/wiki/Requirement). One could see it as the application of [systems](http://en.wikipedia.org/wiki/Systems_theory) [theory](http://en.wikipedia.org/wiki/Systems_theory) to [product development](http://en.wikipedia.org/wiki/Product_development). There is some overlap with the disciplines of [systems analysis](http://en.wikipedia.org/wiki/Systems_analysis), [systems architecture](http://en.wikipedia.org/wiki/Systems_architecture) and [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering). If the broader topic of [product development](http://en.wikipedia.org/wiki/Product_development) "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing [systems](http://en.wikipedia.org/wiki/System) to satisfy specified [requirements](http://en.wikipedia.org/wiki/Requirement) of the user. Until the 1990s systems design had a crucial and respected role in the [data processing](http://en.wikipedia.org/wiki/Data_processing) industry. In the 1990s [standardization](http://en.wikipedia.org/wiki/Standardization) of hardware and software resulted in the ability to build [modular](http://en.wikipedia.org/wiki/Modularity_(programming)) systems. The increasing importance of software running on generic platforms has enhanced the discipline of [software](http://en.wikipedia.org/wiki/Software_engineering) [engineering.](http://en.wikipedia.org/wiki/Software_engineering)

[Object-oriented analysis and design](http://en.wikipedia.org/wiki/Object-oriented_analysis_and_design) methods are becoming the most widely used methods for computer systems design. The [UML](http://en.wikipedia.org/wiki/Unified_Modeling_Language) has become the standard language in object-oriented analysis and design. It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations

# TESTING

#### Testing Plan

* + - ***Testing Strategy***

#### Testing Methods

* + - ***Test cases and result***

**Testing**

**Generally, it has been specified thought for testing that:**

“Testing is the critical element of any software quality assurance and represents the ultimate review of specification, design and code generation.”

Software testing has a dual function; it is used to establish the presence of defects in program and it is used to help judge whether or not the program is usable in practice. Thus software testing is used for validation and verification, which ensure that software conforms to its specification and meets the need of the software customer.

Developer resorted Alpha testing, which usually comes in after the basic design of the program has been completed. The project scientist will look over the program and give suggestions and ideas to improve or to correct the design. They also report and give after a program has been created.

**Testability**

Software Testability is simply how easily a computer program can be tasted. The check list that follows provides a set of characteristics that lead to testable software.

* Operability
* Observables
* Controllability
* Decomposability
* Simplicity
* Stability
* Understandability

Following are the attributes of the Good Test

* A good test has a high probability of finding an error.
* A good test is not redundant.
* A good test should be “Best of Breed”.
* A good test would be neither too simple nor too complex.

**Compartmentalization**

In this step we divide the project into number of manageable activities and task like

* Selection Module
* System Admin data module
* Dept. Admin data module
* Store and assign rendered module
* Employee data module
* Task creation data module
* Task allocation and reply data module
* Insert category and password Authority module
* Testing Module
* Documentation Module

**Interdependency**

Interdependencies of each compartmentalized activity were then found out. Some tasks must occur in sequence while there are many tasks, which can occur in parallel.

**Software Inspection**

Analyze and check system representations such as the requirements document, design, diagrams and the program source code. They may be applied at all the stages of process.

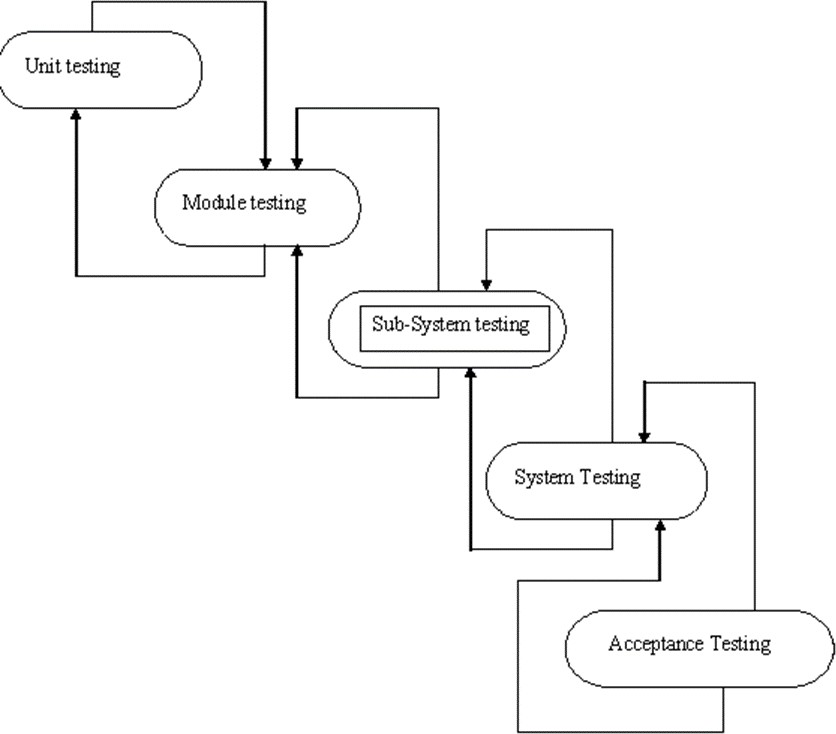


Figure of Testing Process

**Test Plan:-**

**THE TESTING PROCESS**

Developer tests the software process activity such as design, implementation and the requirement engineering. Because, design errors are very costly to repair when the system has been started to operate. Therefore, it is quite obvious to repair them at early stage of the system. So, analysis is the most important process of any project.

**REQUIREMENTS TRACTABILITY**

As most interested portion is whether the system is meeting its requirements or not, for that testing should be planned so that all requirements are individually tested. Developer checked the output of certain combinations of input, which gives desirable results, or not. Strictly stick to our requirements specifications, give you the path to get desirable results from the system.

**TESTED ITEMS**

Our tested items are like:

* Data fetching from the database
* Data insertion, updating and deleting in the database
* Form access to particular login

**TESTING SCHEDULE**

Developer has tested each procedure back to back so that errors and omissions can be found as earliest as possible. Once the system has been developed by fully developer tested it on other machines, which differs in configuration.

**TESTING METHODS**

Software testing involves executing an implementation of the software which tests data and examining the outputs of the software and its operational behavior to check that it is performing as required.

**STATISTICAL TESTING**

Statistical testing is used to test the program’s performance and reliability and to check how it works under

operational conditions. Tests are designed to reflect the actual user inputs and their frequency. The stages involved in the static analysis for this system are follows:

* + Control Flow Analysis

1. Unreachable code
2. Unconditional branches into loops
   * Data Use analysis
3. Variable used before initialization
4. Variables declared but never used
5. Variables assigned twice but never used between assignments
6. Possible array bound violations
7. Declared variables
   * Interface Analysis
8. Parameter type mismatches
9. Parameter number mismatches
10. Non-usage of the results of function
11. Uncalled functions and procedures
    * Storage Management Faults
12. Data not stored in proper tables
13. Data cannot be fetched from proper table

**DEFECT TESTING**

Defect testing is intended to find inconsistencies between a program and its specifications. These inconsistencies are usually due to the program faults or defects.

**UNIT TESTING**

The Developer carries out unit testing in order to check if the particular module or unit of code is working fine. The unit testing comes at the very basic level as it is carried out as and when the unit of the code is developed or a particular functionality is built.

In this application we test one most important module as task allocation which is as follows:

**LOOP TESTING**

Tester has tested the some condition in a code of application. So they test the looping in source code of application for finding miss route or any error or wrong direction of flow in code.

**BLACK-BOX TESTING**

In black box testing or functional testing, the developer is concerned about the output of the module and software, i.e. whether the software gives proper output as per the requirements or not. In another words, these testing aims to test a program behavior against specification without making any reference to the internal structure of the program or the algorithms used. Therefore, the source code is not needed, and so even purchased modules can be tested. The program just gets a certain input and its functionality is examined by observing the output.

This can be done in the following way:

* Input Interface
* Processing
* Output Interface

The tested program gets certain inputs. Then the program does its job and generates a certain output, which is collected by a second interface. This result is then compared to the excepted output, which has been determined before the test.

**WHITEBOX TESTING**

It is also called ‘GLASS BOX’ or ‘STRUCTURAL’ testing. Tester has access to the system design.

* Simple Loops
* Nested Loops
* Concatenated Loops
* Unstructured Loops
* Continuous Loops



*They can:*

* Examine the design document
* View source code
* Individual path examine
* Logical path examine one time
* Logical decision on their true and false

The intention in white-box testing is to ensure that all possible feasible flow of control path through a sub- program is traversed while the software is under tested. This is not the same as saying that all statements in the sub-program will be executed as it is possible for all statements to be executed but for not all of the possible paths to be traversed. However, the conversed is true; whether all the possible paths through a sub- program are traversed then all statements in sub-programs will necessarily be executed.

When considering the number of possible paths through a sub-program two other factors need to be remembered. The first is that some of the possible paths through a sub-program turnout upon investigation to be non- feasible paths. The second consideration is that the number of possible paths to a sub-program indicated by a flow-graph analysis will indicate the minimum number of paths to ensure complete coverage. This may be less than the total number of paths which are possible when combinations of paths are allowed. White-box testing is used as an important primary testing approach. Here, code is inspected to see what it does. Tests are designed to exercise the code. Code is tested using code scripts driver etc. which are employe to directly interfaced with and drive the code.

The tester can analyze the code and used the knowledge about the structure of a component to derive the test data.

**Advantages of White box testing**

* As the knowledge of internal coding structure is prerequisite, it becomes very easy to find out which type of input / data can help in testing the application effectively.
* The other advantage of white box testing is that it helps in optimizing the code.
* It helps in removing the extra lines of code, which can bring in hidden defects.



**STRUCTURE TESTING**

Developer has done his path testing to exercise every independent execution path through a component or program. If every independent path is executed then all statements in the components must have been executed at least once. The structure of our program is also cheeked.

**INTEGRATION TESTING**

After our individual modules Developer tested out Developer go to the integrated to create a complete system. This integration process involves building the system for problems that arise from component interactions.

Developer has applied top-down strategy to validate high-level components of a system before design and implementations have been completed. Because, our development process started with high-level components and Developer worked down the component hierarchy.

**PERFORMANCE TESTING**

Performance testing is designed to test the runtime performance of the system within the context of the system. These tests Developer performed as module level as developer as system level. Individual modules developers tested for required performance.

**CONDITION TESTING**

Condition testing is a test case design method that exercises the logical conditions contained in a program module. If the condition is incorrect, then as least one part of the condition is incorrect. It may include:

* Boolean variable error
* String index out of Bound error
* Null pointer Assignment
* Input Output Connection Exceptions
* Arithmetic expression error
* Parsing ( conversion) errors
* Image unloaded errors



**INTERFACE TESTING**

Interface testing is integral part of integration testing. Therefore, developer checked for the following.

* Interface misuse
* Interface misunderstanding

Developer examined the code to be tested and explicitly list each call to an external component. In the system, standards tests for various modules have been performed, which are follows.

All the menu functions and sub menu functions have been checked. Validations for all inputs are done.

All required fields are not left blank.

**OBJECT TESTING**

Object testing is to test object as individual components, which are often larger than single function. Here following activities have taken place.

* Testing the individual operations associated with object.
* Testing individual object classes.
* Testing cluster of objects.
* Testing object oriented systems.

Now we will discuss how testing and debugging is done of this application. Our application can be divided into parts like;

* To make queries to database and retrieve information from it.
* Reduce some memory requirements for the database.
* Maintain database so that unauthorized access can not affect.

# ENTITY RELATION DIAGRAMS

The Entity Relation Model or Entity Relation Diagram (ERD) is a data model or diagram for high-level description of conceptual data model, and it provides a graphical notation for representing such data models in the form of entity relationship diagrams. Such models are typically used in the first stage of Management information management system design; they are used for example, to describe information needs and/ or the type of information that is to be stored in the Database during the requirement analysis. The data modeling technique, however, can be used to describe any ontology ( i.e an overview and classification of used term and their relationships) for a certain universe of discourse (i.e area of interest ).

In the case of design a Management Information Management system that is based on a database, the conceptual data model is, a later stage( usually called logical design), mapped to a logical data model such as, relational data model; this is turn in mapped to a physical model during physical design. Note that sometimes, both of the phases are referred a “physical design”. There are number of convention for entity- relation diagrams (ERDs). The classical notation is describe in the remainder of this article, and mainly related to the conceptual modeling. There is a range of notation more typically employed in physical and logical database design.

### ER DIAGRAM

contact

name

View user

email

Admin

User

signup

gender

profil addres

Change pass.

login

Admin

Delete User

manage

name

contact

email

status

type

bookin

Booking

Booking

status

price

type

Room no.

image

# DATA FLOW DIAGRAM

The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It’s also known as BUBBLE CHART. The purpose of DFD is major transformation that will become in system design symbols used in DFD.

In the DFD, four symbols are used and they are as follows.

1. A square defines a source (originator) or destination of system data.
2. An arrow identifies data flow-data in motion. It is 2a pipeline through which information flows.
3. An open rectangle is a data store-data at rest, or a temporary Repository of data.
4. A circle or a “bubble “(Some people use an oval bubble) represents a process that transfers informing data flows into outgoing data flows.

### Context Level Data Flow Diagram

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. Online book store is shown as one process in the context diagram; which is also known as zero level DFD, shown below. The context diagram plays important role in understanding the system and determining the boundaries. The main process can be broken into sub-processes and system can be studied with more detail; this is where 1st level

DFD comes into play.

User/customer

Admin

Room Rental Services

Cancel Booking

Book room

Signup

Update profile

See Room

Feed Back

Zero Level Data Flow Diagram

See contact Details

Manage Rooms

Add Rooms

Manage User

Manage Booking

Manage Feedback

### First Level DFD:-

See

See

Update

profile

Access

Logi

Check status

confirm/canc

Actio

This level (level 1) shows all processes at the first level of numbering, data stores, external entities and the data flows between them. The purpose of this level is to show the major high- level processes of the system and their interrelation. A process model will have one, and only one, level-1 diagram. A level-1 diagram must be balanced with its parent context level diagram,

i.e. there must be the same external entities and the same data flows, these can be broken down to more detail in the level 1.

Check Action

Register

Submit

Signup

Get ID/Password

Feedback

Check

Login Table

See and

Reques

Booking Request

confirm

Book

Room

Room

Status

See Status

Send Status Available or

non

Change Room

Status

Submit

Check

Admin

Admin

Contact Table

Any

Customer

Admin

See

Customer

Customer

Check

Add/Delete

Delet

See Details

Check Room

Table Action

One level data flow diagram

* 1. **Project Planning & Project Scheduling**

### PERT CHART

A project plan needs to be created to ensure the timely completion of the project. As part of project analysis, we break the project down to a number of stages and use a Gantt chart and PERT chart to describe specific tasks and status.

The Work Breakdown Structure of our proposed system “E-Commerce” is shown below:

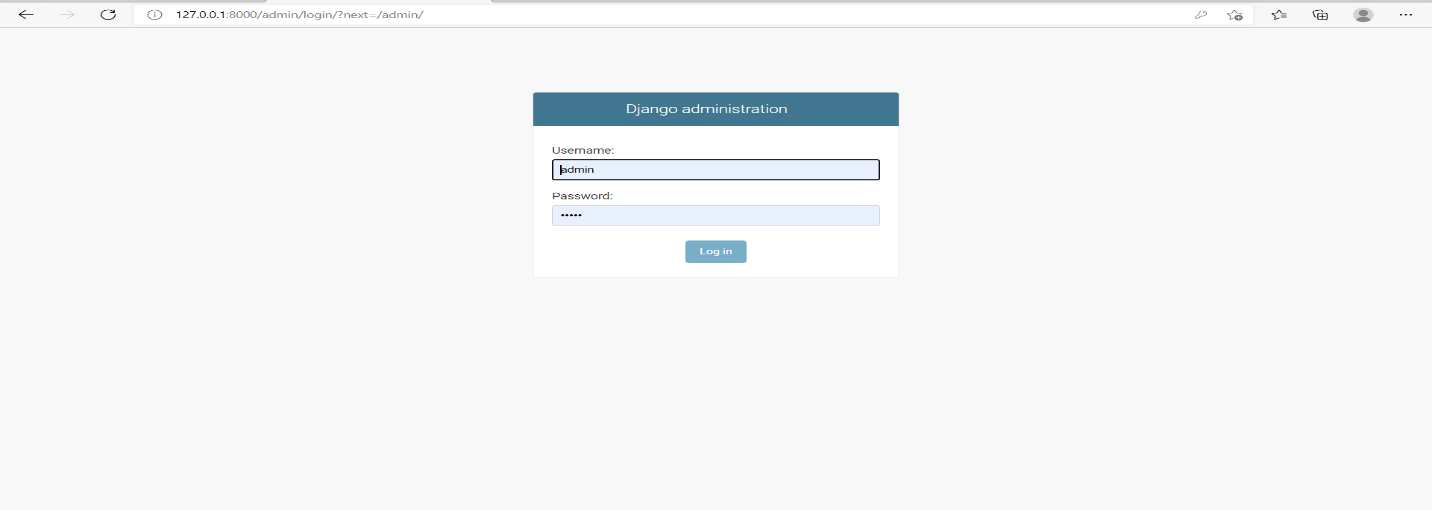
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Task Name | Duration | Start | Finish | Predecessors | Resource  Names |
| 1 | **Project Initiation** | 2 days | Sat 6/02/2021 | Mon  08/02/2021 |  |  |
| 2 | Draft Project Plan | 2 days | Tue 09/02/2021 | Thu  11/02/2021 |  |  |
| 3 | **Analysis Phase** | 7 days | Fri  12/02/2021 | Fri  19/02/2021 |  |  |
| 4 | Plan User  Interviews | 2 days | Sat  20/02/2021 | Mon  22/02/2021 |  |  |
| 5 | Schedule users  Interviews | 3 days | Tue  23/02/2021 | Fir  26/06/2021 |  |  |
| 6 | Conducting users  Interviews | 2 days | Mon  1/03/2021 | Wed  04/03/2021 |  |  |
| 7 | **System Design** | 20 days | Fri  05/03/2021 | Sat  27/03/2021 | 6 |  |
| 8 | Modules Design | 10 days | Wed  31/8/18 | Sat  10/04/2021 |  |  |
| 9 | Data Structure  Design | 3 days | Mon  12/04/2021 | Thu  15/04/2021 | 8 |  |
| 10 | User Interface  Design | 3 days | Fri  16/04/2021 | Mon  19/04/2021 |  |  |
| 11 | Coding Phase | 36 days | Tue | Thu |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 20/04/2021 | 25/05/2021 |  |  |
| 12 | **Testing Phase** | 9 days | Wed 26/05/2021 | Fri  04/06/2021 |  |  |
| 13 | Integration  Testing | 5 days | Sat  05/06/2021 | Thu  10/06/2021 |  |  |
| 14 | System Level  Testing | 4 days | Fri  11/06/2021 | Thu  15/06/2021 |  |  |
| 15 | Implementation | 4 days | Fri  16/06/2021 | Sun  20/06/2021 |  |  |
| 16 | Post-  Implementation Review | 2 days | Mon 21/06/2021 | Wed 23/06/2021 |  |  |

# DATABASE TABLES

### Admin and User Login Table

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO.** | **Fields** | **Data type** | **Size** |
| **1** | **username** | CharField | 30 |
| **2** | **passwodr** | CharField | 30 |
| **3** | **email** | EmailField | Auto size by python |



1. **User Signup Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO.** | **Fields** | **Data type** | **Size** |
| **1** | **username** | CharField | 30 |
| **2** | **email** | EmailField | 40 |
| **3** | **first\_name** | CharField | 30 |
| **4** | **last\_name** | CharField | 30 |
| **5** | **Mobile** | IntegerField | 40 |
| **6** | **gender** | CharField | 10 |
| **7** | **dob** | DateField | Auto size by python |
| **8** | **address** | CharField | 100 |
| **9** | **image** | FileField | Auth size by Python |

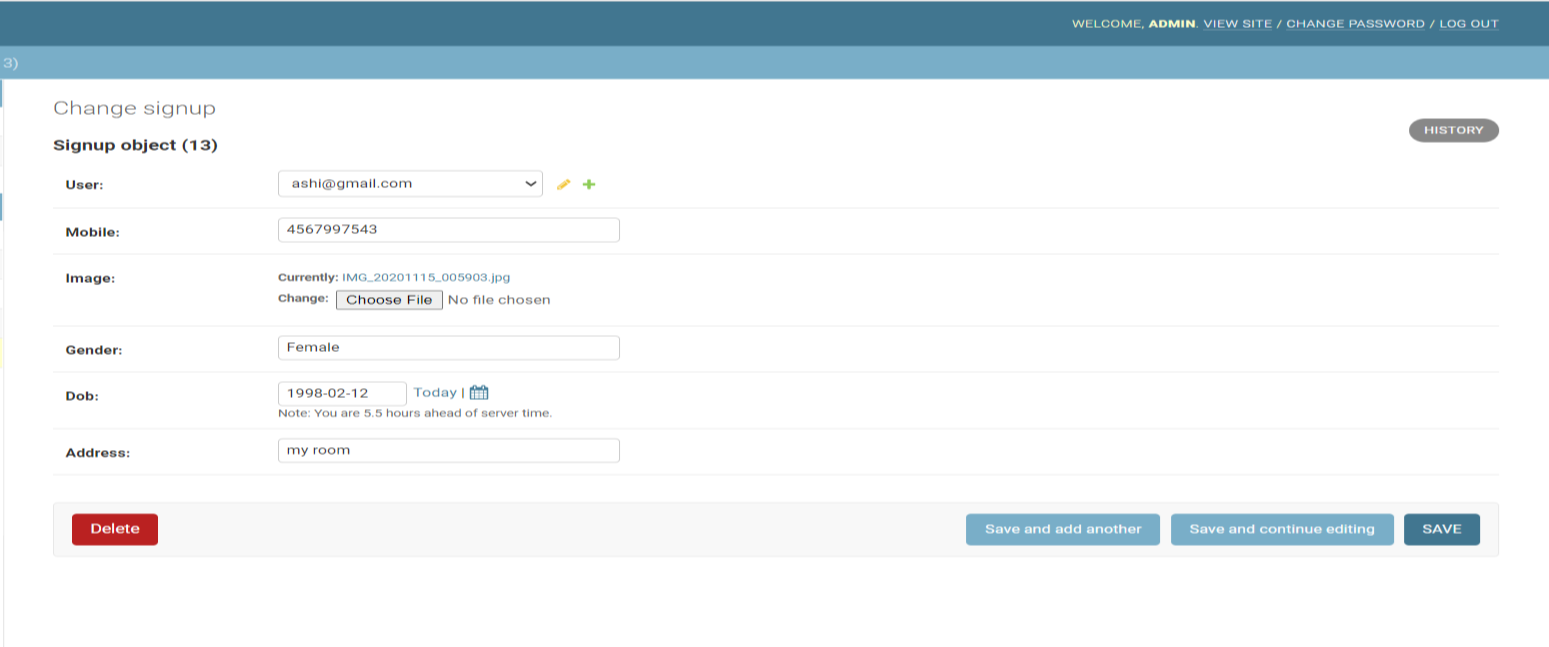
class Signup(models.Model): user=models.ForeignKey(User,on\_delete=models.CASCADE)

mobile = models.CharField(max\_length=20,null=True) image = models.FileField(null=True)

gender = models.CharField(max\_length=10,null=True) dob = models.DateField()

address = models.CharField(max\_length=50,null=True) def \_str\_(self):

return self.user.username



### 3.- Feedback Table

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO.** | **Fields** | **Data type** | **Size** |
| **1** | **feedback\_name** | CharField | 30 |
| **2** | **feedback\_contact** | CharField | 40 |
| **3** | **feedback\_email** | EmailField | 30 |
| **4** | **feedback\_comment** | CharField | 30 |

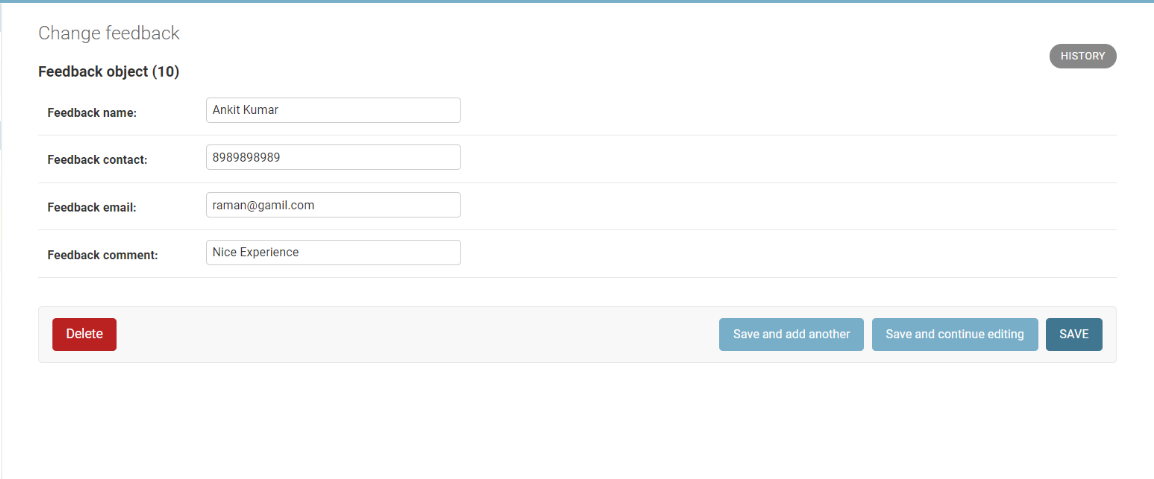
class Feedback(models.Model): feedback\_name =

models.CharField(max\_length=20,null=True) feedback\_contact = models.CharField(max\_length=30,

null=True)

feedback\_email = models.CharField(max\_length=10,null=True)

feedback\_comment = models.CharField(max\_length=15,null=True)

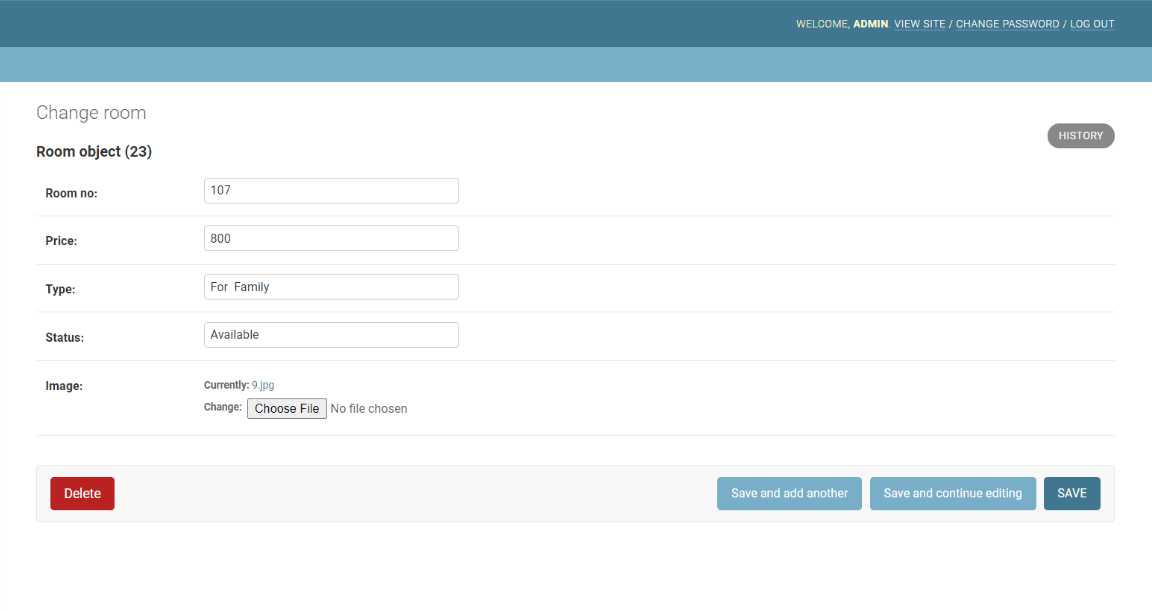


**4.- Room Deatil Table**

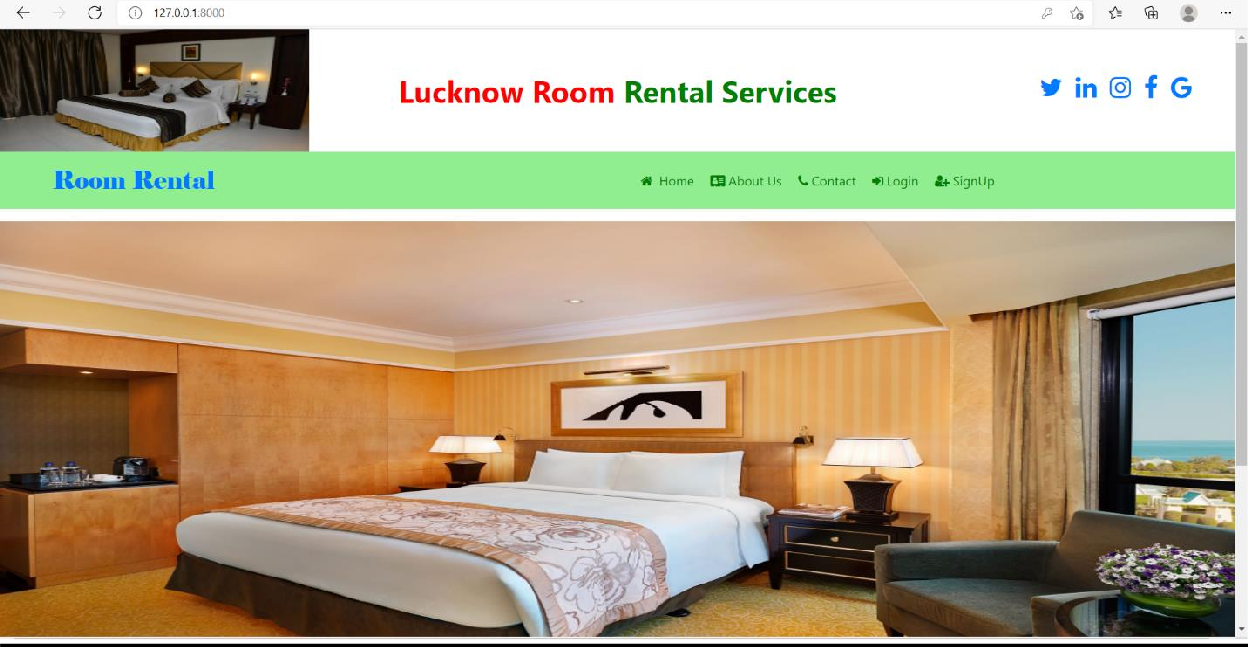
|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO.** | **Fields** | **Data type** | **Size** |
| **1** | **room\_no** | IntegerField | 10 |
| **2** | **price** | CharField | 20 |
| **3** | **type** | CharField | 10 |
| **4** | **status** | CharField | 10 |
| **5** | **image** | FileField | Auto Size by python |

class Room(models.Model):

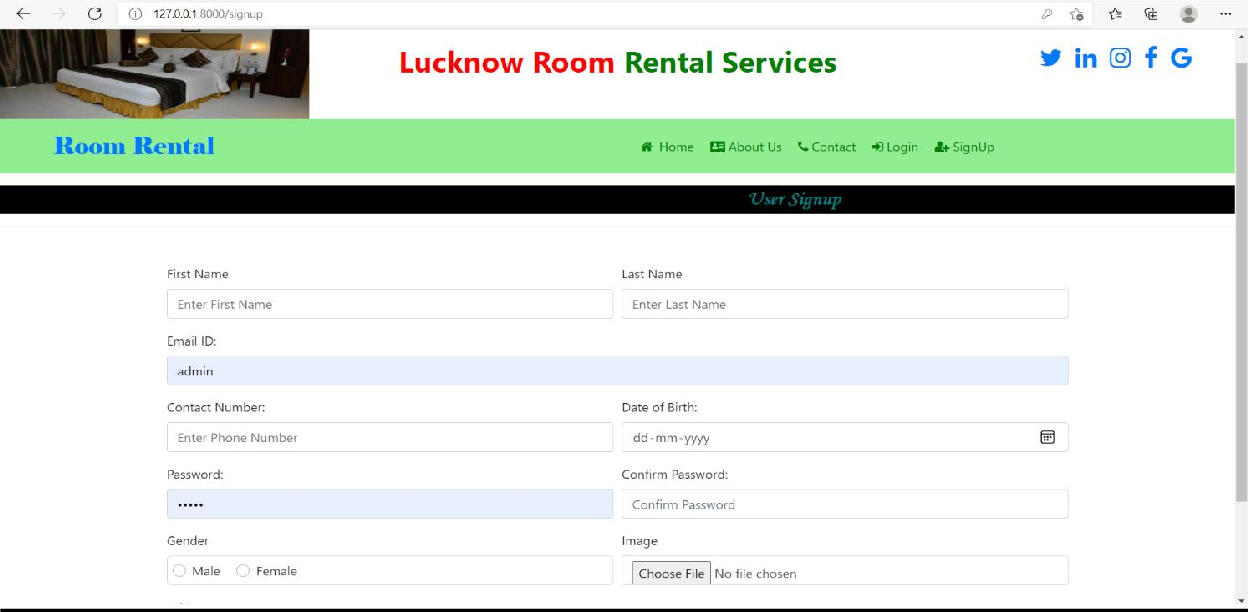
room\_no = models.CharField(max\_length=20,null=True) price = models.CharField(max\_length=30, null=True) type = models.CharField(max\_length=10,null=True) status = models.CharField(max\_length=15,null=True) image = models.FileField(null=True)



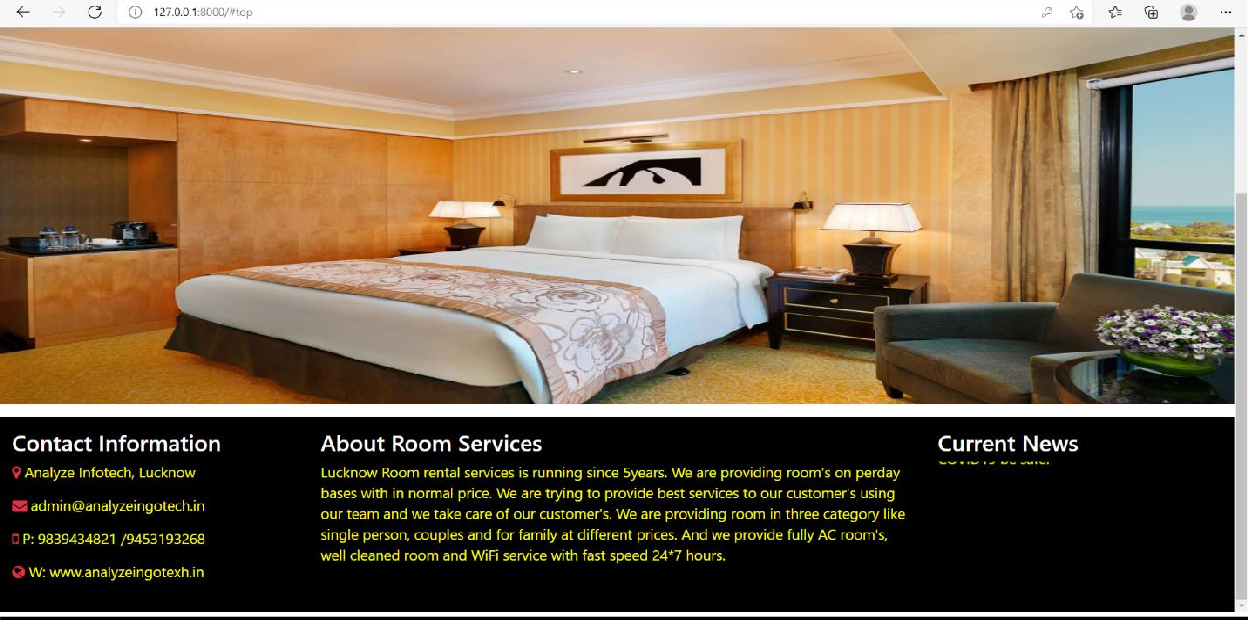
### PROJECT SCREENSHOT

**HOME PAGE**

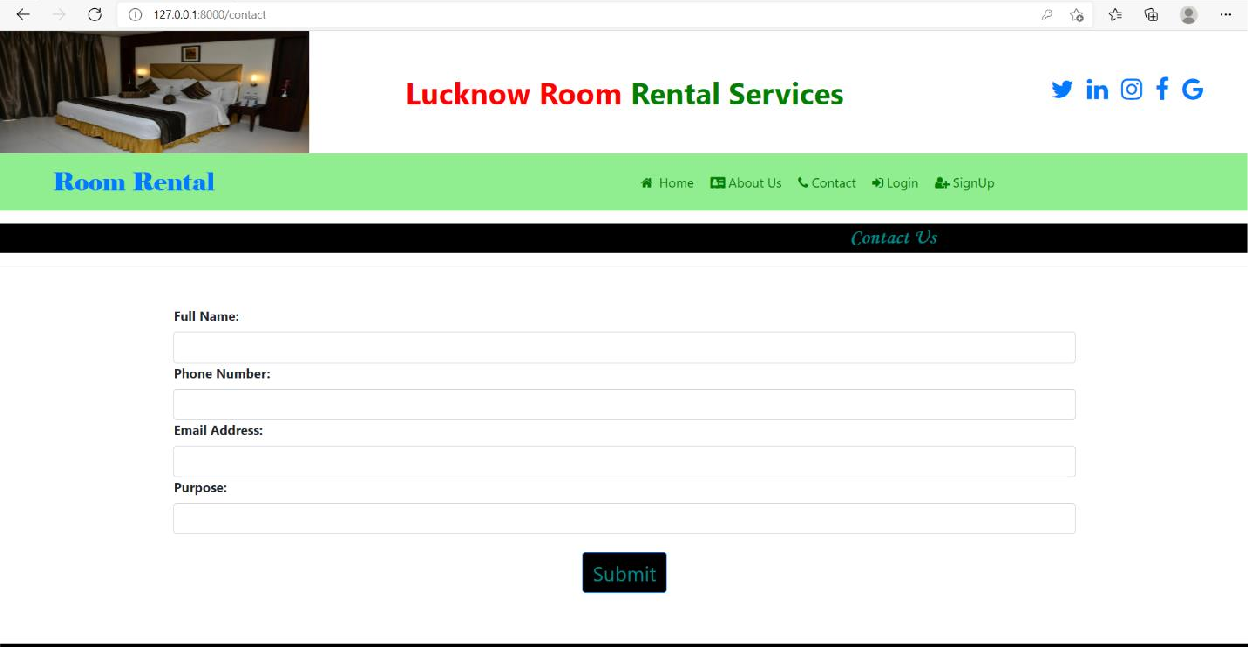
### User Signup Page



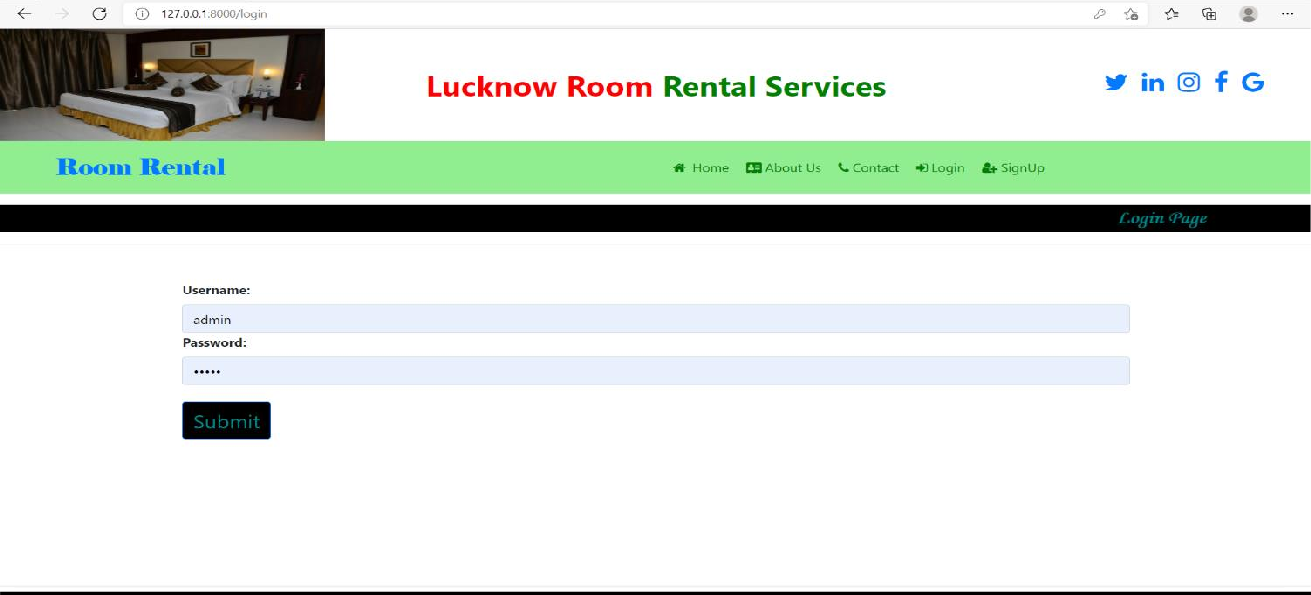
**About Us**



### Contact Us



**User & Admin Login Page**



### View User



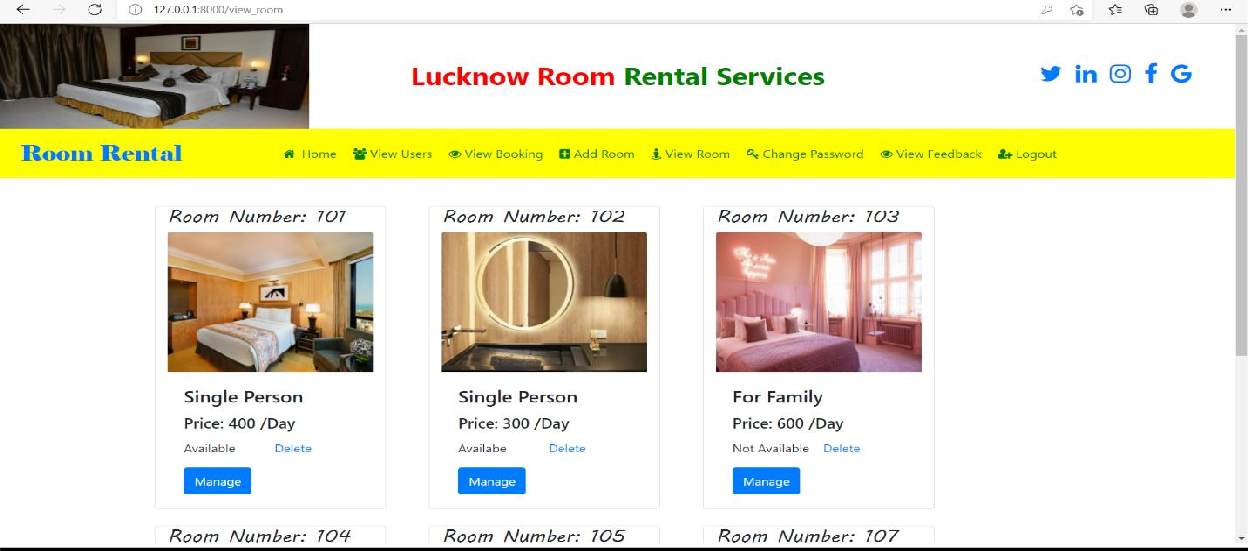
**View Booking**



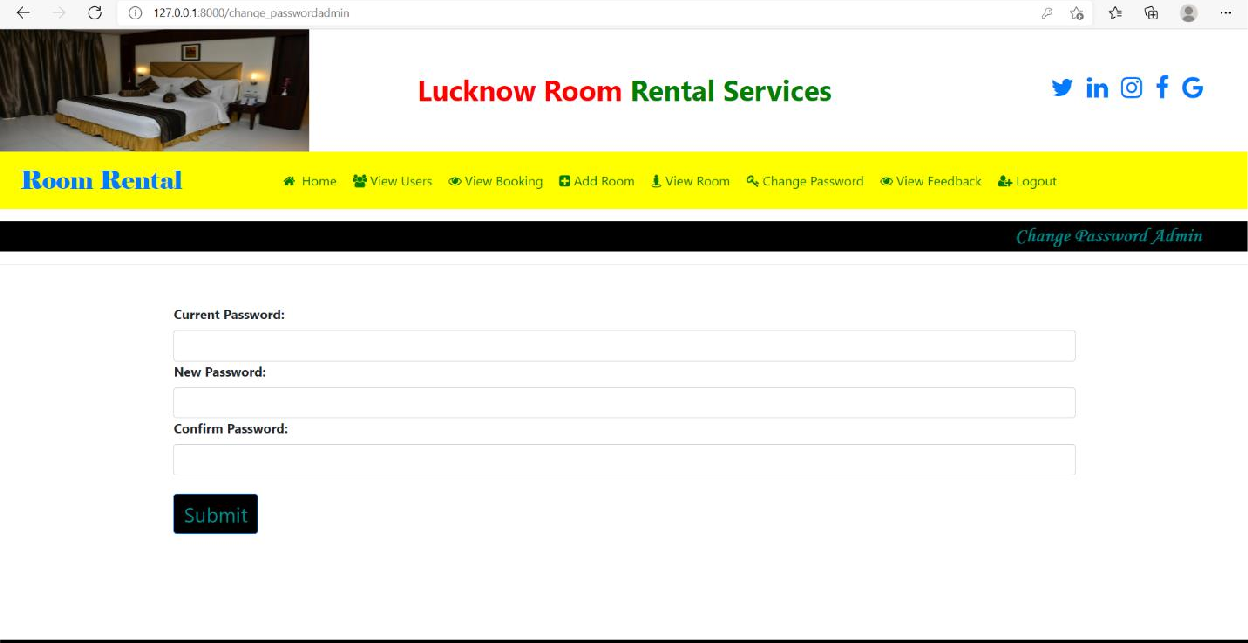
### Add Room



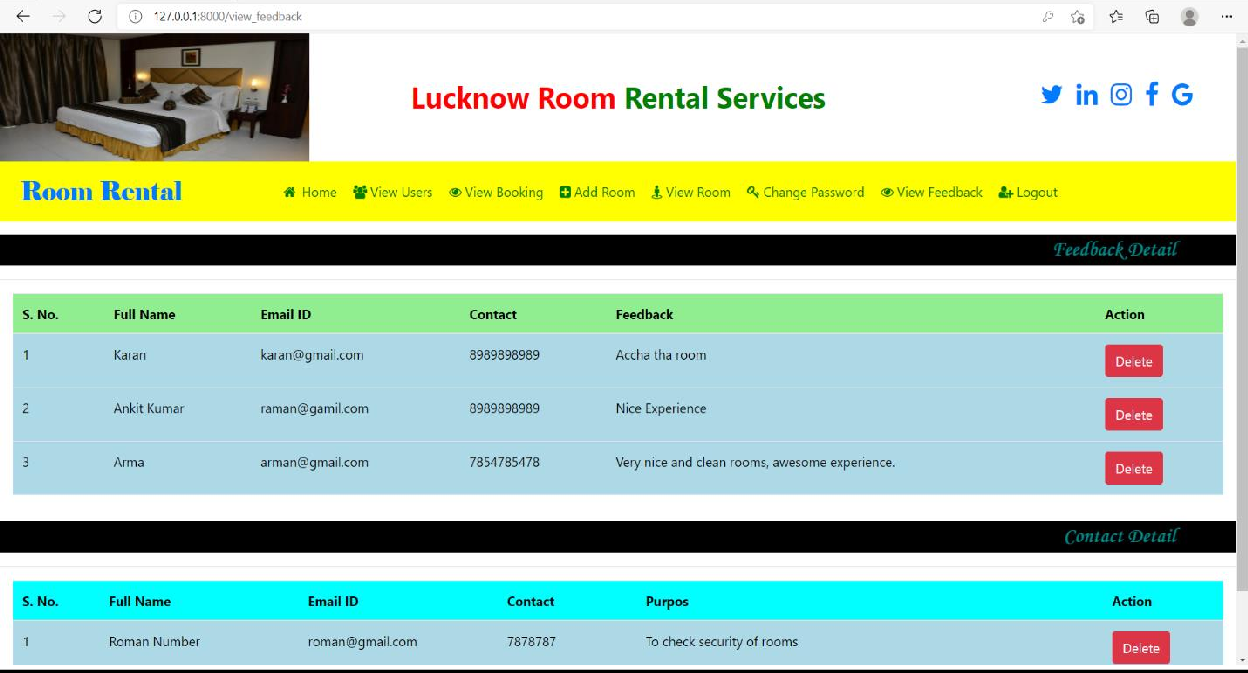
**View Room**



### Change Password



**View Feedback**



### COADING OF PROJECT

**Views.py file code**

from django.shortcuts import render,redirect from django.contrib.auth.models import User,auth from django.contrib.auth import authenticate,login,logout

from .models import \*

*# Create your views here.*

def index(request):

return render(request,**'index.html'**)

def login(request): error=**""**

if request.method == **"POST"**: ur = request.POST[**'uname'**] pd = request.POST[**'pwd'**] user =

auth.authenticate(username=ur,password=pd) try:

if user.is\_staff: auth.login(request,user) error = **"no"**

elif user is not None: auth.login(request,user) return redirect(**'user\_home'**) error = **"not"**

else:

error = **"yes"**

except:

error = **"yes"**

d = {**'error'**:error}

return render(request,**'admin\_login.html'**,d)

def admin\_home(request):

if not request.user.is\_authenticated: return redirect(**'login'**)

return render(request,**'admin\_home.html'**)

def user\_home(request):

if not request.user.is\_authenticated: return redirect(**'login'**)

user = request.user

user2 = Signup.objects.get(user=user) error = **""**

if request.method == **'POST'**: f = request.POST[**'fname'**] l = request.POST[**'lname'**]

con = request.POST[**'contact'**] gen = request.POST[**'gender'**] add = request.POST[**'address'**] user2.user.first\_name = f user2.user.last\_name = l user2.mobile = con

user2.gender = gen user2.address = add

try:

user2.save() user2.user.save() error = **"no"**

except:

error = **"yes"**

try:

i = request.FILES[**'image'**] user2.image = i user2.save()

error = **"no"**

except:

pass

d = {**'user2'**: user2, **'error'**: error} return render(request,**'user\_home.html'**,d)

def change\_passworduser(request):

if not request.user.is\_authenticated: return redirect(**'user\_login'**)

error=**""**

if request.method==**"POST"**:

c = request.POST[**'currentpassword'**] n = request.POST[**'newpassword'**] try:

u = User.objects.get(id=request.user.id) if u.check\_password(c):

u.set\_password(n) u.save() error=**"no"**

else:

error=**"not"**

except:

error=**"yes"** d = {**'error'**:error}

return render(request,**'change\_passworduser.html'**,d) def change\_passwordadmin(request):

if not request.user.is\_authenticated: return redirect(**'user\_login'**)

error=**""**

if request.method==**"POST"**:

c = request.POST[**'currentpassword'**] n = request.POST[**'newpassword'**] try:

u = User.objects.get(id=request.user.id) if u.check\_password(c):

u.set\_password(n) u.save() error=**"no"**

else:

error=**"not"**

except:

error=**"yes"** d = {**'error'**:error} return

render(request,**'change\_passwordadmin.html'**,d)

def signup(request): error = **""**

if request.method==**'POST'**: f=request.POST[**'fname'**] l=request.POST[**'lname'**] e = request.POST[**'email'**]

con = request.POST[**'contact'**] p = request.POST[**'pwd'**]

gen = request.POST[**'gender'**] i=request.FILES[**'image'**] addr=request.POST[**'address'**] d=request.POST[**'dob'**]

try:

user=User.objects.create\_user(first\_name=f,last\_name=l, username=e,password=p)

Signup.objects.create(user=user,mobile=con,image=i,gend er=gen,address=addr,dob=d)

error=**"no"**

except:

error=**"yes"** d={**'error'**:error}

return render(request,**'signup.html'**,d)

def view\_users(request):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

data = Signup.objects.all() d = {**'data'**:data}

return render(request,**'view\_users.html'**,d)

def delete\_user(request,pid):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

student = User.objects.get(id=pid) student.delete()

return redirect(**'view\_users'**)

def delete\_feedback(request,id):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

feedback = Feedback.objects.get(id=id) feedback.delete()

return redirect(**'view\_feedback'**)

def delete\_contact(request,id):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

contact = Contact.objects.get(id=id) contact.delete()

return redirect(**'view\_feedback'**)

def add\_room(request):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

error=**""**

if request.method==**'POST'**:

n = request.POST[**'roomno'**] p = request.POST[**'price'**]

rt = request.POST[**'rtype'**] s = request.POST[**'status'**] i = request.FILES[**'image'**] try:

Room.objects.create(room\_no=n,price=p,type=rt,status=s, image=i)

error=**"no"** except:

error=**"yes"** d={**'error'**:error}

return render(request,**'add\_room.html'**,d)

def view\_room(request):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

data = Room.objects.all() d = {**'data'**:data}

return render(request,**'view\_room.html'**,d)

def view\_roomuser(request):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

data = Room.objects.all() d = {**'data'**:data}

return render(request,**'view\_roomuser.html'**,d)

def book\_roomuser(request,id):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

user = request.user

data = Room.objects.get(id=id) user = request.user

data2 = Signup.objects.get(user=user) error = **""**

if request.method == **'POST'**: f = request.POST[**'fname'**] l = request.POST[**'lname'**] e = request.POST[**'email'**]

con = request.POST[**'contact'**]

con2 = request.POST[**'contact2'**] d = request.POST[**'booking\_date'**] dy = request.POST[**'select\_days'**] g = request.POST[**'gender'**]

p = request.POST[**'price'**] addr = request.POST[**'address'**] pr = int(p)\*int(dy)

try:

Booked.objects.create(first\_name=f,last\_name=l,email=e, mobile=con,mobile2=con2,booking\_date=d,gender=g,days=dy

,price=pr, address=addr,status=**"pending"**) error = **"no"**

except:

error = **"yes"**

d = {**'data'**: data,**'user'**:user,**'data2'**:data2,**'error'**: error}

return render(request,**'book\_roomuser.html'**,d)

def view\_booking(request):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

data = Booked.objects.all() d = {**'data'**: data}

return render(request,**'view\_booking.html'**,d)

def edit\_room(request,pid):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

data = Room.objects.get(id=pid) if request.method==**'POST'**:

n = request.POST[**'roomno'**] p = request.POST[**'price'**] rt = request.POST[**'rtype'**] s = request.POST[**'status'**] data.room\_no = n data.price = p

data.type = rt data.status = s

try:

data.save() error=**"no"**

except:

error=**"yes"** d = {**'data'**:data}

return render(request,**'edit\_room.html'**,d)

def delete\_booking(request,id): booking=Booked.objects.get(id=id) booking.delete()

return redirect(**'view\_booking'**)

def change\_roomstatus(request,id):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

booking=Booked.objects.get(id=id) if request.method == **"POST"**:

rs = request.POST[**'rstatus'**] booking.status = rs

try:

booking.save() error = **"no"**

except:

error = **"yes"**

d = {**'booking'**: booking}

return render(request,**'change\_roomstatus.html'**,d)

def cancel\_booking(request,id): booking=Booked.objects.get(id=id) booking.delete()

return redirect(**'user\_booking'**)

def delete\_room(request,id): emp=Room.objects.get(id=id) emp.delete()

return redirect(**'view\_room'**)

def feedback(request):

if not request.user.is\_authenticated:

return redirect(**'feedback'**) error=**""**

if request.method==**'POST'**:

n = request.POST[**'fname'**] p = request.POST[**'fphone'**] e = request.POST[**'femail'**]

c = request.POST[**'fcomment'**] try:

Feedback.objects.create(feedback\_name=n,feedback\_contac t=p,feedback\_email=e,feedback\_comment=c)

error = **"no"**

except:

error=**"yes"** d = {**'error'**:error}

return render(request,**'feedback.html'**,d)

def view\_feedback(request):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

data = Feedback.objects.all() data2 = Contact.objects.all()

d = {**'data'**:data,**'data2'**:data2}

return render(request,**'view\_feedback.html'**,d)

def user\_booking(request):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

user = request.user

data = Booked.objects.all() rms = Room.objects.all()

d = {**'data'**:data,**'rms'**:rms}

return render(request,**'user\_booking.html'**,d)

def contact(request): error=**""**

if request.method==**'POST'**:

n = request.POST[**'cname'**] pn = request.POST[**'cphone'**]

e = request.POST[**'cemail'**] p = request.POST[**'cpurpose'**] try:

Contact.objects.create(con\_name=n,con\_mobile=pn,con\_ema il=e,con\_purpose=p)

error = **"no"**

except:

error=**"yes"** d = {**'error'**:error}

return render(request,**'contact.html'**,d)

def change\_roomimage(request,id):

if not request.user.is\_authenticated: return redirect(**'admin\_login'**)

error=**""**

data = Room.objects.get(id=id) if request.method == **'POST'**:

l = request.FILES[**'image'**] data.image = l

try:

data.save() error=**"no"**

except:

error=**"yes"**

d = {**'error'**: error,**'data'**:data}

return render(request,**'change\_roomimage.html'**,d)

def Logout(request): logout(request)

return redirect(**'index'**)

### Urls.py file code

*"""room\_rantal URL Configuration*

*The `urlpatterns` list routes URLs to views. For more information please see:*

*https://docs.djangoproject.com/en/3.1/topics/http/urls/ Examples:*

*Function views*

1. *Add an import: from my\_app import views*
2. *Add a URL to urlpatterns: path('', views.home, name='home')*

*Class-based views*

1. *Add an import: from other\_app.views import Home*
2. *Add a URL to urlpatterns: path('', Home.as\_view(), name='home')*

*Including another URLconf*

1. *Import the include() function: from django.urls import include, path*
2. *Add a URL to urlpatterns: path('blog/', include('blog.urls'))*

*"""*

from django.contrib import admin from django.urls import path from room.views import \*

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path(**'admin/'**, admin.site.urls), path(**''**, index,name=**"index"**), path(**'login'**,login,name=**"login"**), path(**'Logout'**,Logout,name=**"Logout"**), path(**'signup'**,signup,name=**"signup"**),

path(**'admin\_home'**,admin\_home,name=**"admin\_home"**), path(**'user\_home'**,user\_home,name=**"user\_home"**),

path(**'change\_passworduser'**,change\_passworduser,name=**"ch ange\_passworduser"**),

path(**'change\_passwordadmin'**,change\_passwordadmin,name=**" change\_passwordadmin"**),

path(**'view\_users'**,view\_users,name=**"view\_users"**),

path(**'delete\_user\<int:pid>'**,delete\_user,name=**"delete\_u ser"**),

path(**'delete\_feedback/<int:id>'**,delete\_feedback,name=**"d**

**elete\_feedback"**),

path(**'delete\_contact/<int:id>'**,delete\_contact,name=**"del ete\_contact"**),

path(**'add\_room'**,add\_room,name=**"add\_room"**), path(**'feedback'**,feedback,name=**"feedback"**), path(**'contact'**,contact,name=**"contact"**),

path(**'view\_feedback'**,view\_feedback,name=**"view\_feedback"**

),

path(**'view\_room'**,view\_room,name=**"view\_room"**),

path(**'view\_roomuser'**,view\_roomuser,name=**"view\_roomuser"**

),

path(**'edit\_room<int:pid>'**,edit\_room,name=**"edit\_room"**),

path(**'delete\_room/<int:id>'**,delete\_room,name=**"delete\_ro om"**),

path(**'delete\_booking/<int:id>'**,delete\_booking,name=**"del ete\_booking"**),

path(**'cancel\_booking/<int:id>'**,cancel\_booking,name=**"can cel\_booking"**),

path(**'change\_roomimage/<int:id>'**,change\_roomimage,name=

**"change\_roomimage"**),

path(**'book\_roomuser/<int:id>'**,book\_roomuser,name=**"book\_ roomuser"**),

path(**'view\_booking'**,view\_booking,name=**"view\_booking"**), path(**'user\_booking'**,user\_booking,name=**"user\_booking"**),

path(**'change\_roomstatus/<int:id>'**,change\_roomstatus,nam e=**"change\_roomstatus"**),

]+static(settings.MEDIA\_URL,document\_root=settings.MEDI A\_ROOT)

### Models.py file code

from django.db import models

from django.contrib.auth.models import User class Signup(models.Model):

user=models.ForeignKey(User,on\_delete=models.CASCADE) mobile = models.CharField(max\_length=20,null=True) image = models.FileField(null=True)

gender = models.CharField(max\_length=10,null=True) dob = models.DateField()

address = models.CharField(max\_length=50,null=True) def \_str\_(self):

return self.user.username

class Room(models.Model):

room\_no = models.CharField(max\_length=20,null=True) price = models.CharField(max\_length=30, null=True) type = models.CharField(max\_length=10,null=True) status = models.CharField(max\_length=15,null=True) image = models.FileField(null=True)

class Feedback(models.Model): feedback\_name =

models.CharField(max\_length=20,null=True) feedback\_contact = models.CharField(max\_length=30,

null=True)

feedback\_email = models.CharField(max\_length=10,null=True)

feedback\_comment = models.CharField(max\_length=15,null=True)

class Contact(models.Model): con\_name =

models.CharField(max\_length=20,null=True) con\_mobile = models.CharField(max\_length=30,

null=True)

con\_email = models.CharField(max\_length=10,null=True)

con\_purpose = models.CharField(max\_length=15,null=True)

class Booked(models.Model): first\_name =

models.CharField(max\_length=20,null=True) last\_name = models.CharField(max\_length=30,

null=True)

email = models.CharField(max\_length=30,null=True) gender = models.CharField(max\_length=10,null=True) mobile = models.CharField(max\_length=30,null=True) mobile2 = models.CharField(max\_length=30,null=True) booking\_date = models.DateField()

days = models.CharField(max\_length=10,null=True) price = models.CharField(max\_length=20,null=True) address = models.CharField(max\_length=50,null=True) status=models.CharField(max\_length=20,null=True)

# Limitations

* It can run only on windows servers. It is not compatible with LINUX servers.
* It is not cent percent safe from professional hackers.
* Only registered user can buy and sell their items using this website.

# CONCLUSION

Effectively resolving the apartment issues is important to the buyer's long-term future, the Room Rental Project will be an important tool for creating rental Room stability by helping tenants speak with greater credibility through initiating and documenting communications and building productive relationships with User.

# FUTURE SCOPE OF THE PROJECT

In future our project is meant to satisfy the needs of room rental . Several user friendly interfaces have also been adopted. This package shall prove to be a powerful in satisfying all the requirements of the users. It is with utmost faith that I present this software to you hoping that it will solve your problems and encourage you to continue appreciating technology because it is meant to change and ease all our work that seems to be very difficult. I don't mean that my project is the best or that I have used the best technology available it just a simple and a humble venture that is easy to understand. When we can update the system, we can add GPS system in build and can give live chat online option to users. This project can also be extended to IOS Platform and several state Database can be included.

In future it can be modify, so that it can be done online. In order to meet these facility this is the major change which can be done in future regarding this project.

**Reference Books**

# References

**Head First PYTHON with django & SQLITE- By Lynn Beighley & Michael Morrison**

**The Complete Reference- PYTHON- with django (Guido van rossun)**

### Reference websites

* [**www.youtube.com**](http://www.scribed.com/)
* [**www.w3schools.com**](http://www.w3schools.com/)
* [**www.javatpoints.edu**](http://www.javatpoints.edu/)