Chongqing University of Science and Technology

Graduation Project (Dissertation)

Title Hotel: Management System Application

College: School Of Intelligent Technology & Engineering

Major: Computer Science & Technology

Student Name: Meem Afsana Akter

Student Number: 2017490104

Supervisor: Ying Wu

Reviewer: ____________________

20th May, 2021

Declaration of originality of students' graduation project (dissertation)

I declare with my reputation: the submitted graduation project (dissertation) is the design (research) work and results obtained under the guidance of the supervisor. The project (dissertation) quotes literature, data, drawings, and materials from other people have been clearly marked, the conclusions and results in the dissertation are completed independently by myself, and do not include the achievements of others and the use of their materials for obtaining degrees or certificates from Chongqing University of Science and Technology or other educational institutions. The participant who works with me has clearly stated in the paper and expressed gratitude for any contributions to this design (research).

Author: Meem Afsana Akter
19/05/2021

ABSTRACT

As the name specifies “HOTEL MANAGEMENT SYSTEM” is software developed for managing various activities in the hotel. For the past few years, tourism has become very popular in every country worldwide. Thereby the number of hotels are also increasing for the accommodation of the tourists. And hence there is a lot of strain on the person who is running the hotel and software’s are not usually used in this context. This particular project deals with managing a hotel and avoids the problems that occur when carried manually. The Django Hotel Management System is developed using Python Django, HTML, CSS, and JavaScript, This Hotel Booking App In Django is a fully responsive website(for both mobile and larger screens) based on google material design. Also, this project has a Custom dashboard for both the Customer and the Room Manager. A Django Hotel Reservation System contains different features like Login and Sign-up functionality for both manager and customer users, Customers can book rooms based on availability and See the booking history, and edit orders from the dashboard. Identification of the drawbacks of the existing system leads to designing a computerized system that will be compatible with the existing system, which is more user-friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the drawbacks of the existing system.

1. Less human error
2. The strength and strain of manual labor can be reduced
3. High security,
4. Data redundancy can be avoided to some extent,
5. Data consistency,
6. Easy to handle,
7. Easy data updating,
8. Easy record keeping,

Keywords: hotel management system, django project, python django, hotel booking, django, search hotels, hotels near me, hotel room.

摘要

Abstract in Chinese…
顾名思义，“酒店管理系统”为 开发用于管理酒店各种活动 件。在过去的几年中，旅游在每个国家都变得非常流行。
 世界各地的国家/地区。 因此，酒店的数量也 增加游客的住宿。 因此那里给经营酒店的人很大的压力，在这种情况下通常不使用该软件。 这个特别的该项目涉及管理酒店，并避免了手动携带时发生。 Django酒店管理系统是使用Python Django，HTML，CSS和JavaScript，这是Django中的酒店预订应用程序，功能全面响应型网站（适用于移动设备和大屏幕），基于
 谷歌材料设计。 此外，该项目还有一个“自定义”信息中心
 为客户和房间经理。 Django酒店 预订系统包含不同的功能，例如登录和 经理和客户用户的注册功能，客户可以根据空房情况预订房间，并查看 预订历史记录，并从信息中心编辑订单。
 识别现有系统的弊端导致 设计一个与计算机兼容的计算机化系统 现有系统，它更加人性化和更多GUI面向。 我们可以提高系统效率，从而 克服了现有系统的弊端。
 1.减少人为错误
 2.可以减轻体力劳动的强度和压力
 3.高安全性
 4，可以在一定程度上避免数据冗余
 5，数据一致性
 6.易于处理
 7，方便的数据更新
 8.轻松保存记录，
 题为“酒店管理系统的设计与开发（使用Django）”的项目已部分提交给以下受邀的审查委员会成员 到2021年5月满足计算机科学与技术理学学士学位要求 学生，并被认为是令人满意的。

关键字：酒店管理系统，django项目，python django，酒店预订，django，搜索酒店，我附近的酒店，酒店客房。

TABLE OF CONTENTS

CHAPTER 1

CHAPTER 1

INTRODUCTION

1.1 Problem definition

HOTEL MANAGEMENT SYSTEM was developed following system development stages for smooth running and management of Daxuecheng hotel. The seven months provided by the School of Intelligent Technology and Engineeringenabled the system analyst to recognize and define the problem in the current manual system at the hotel. After an information-gathering process from several hotels managed by manual and computerized systems, the system analyst saw that the hotel indeed needed a computerized management system. After a close analysis of samples collected during the problem definition stage, the analyst found that all the hardware and software requirements needed for implementation and maintenance of the system are readily available in the market and cheaply affordable by the hotel. The system was carefully designed to ensure maximum efficiency of the system at the hotel. The system was skillfully and carefully coded to seal any possible loopholes in the system. The system was developed using visual basic for applications (Microsoft access) language. This system will indeed help the hotel management and the esteemed staff members to manage and steer the hotel's functionality and transactions to realize its maximum potential in addition to its competence in the hotel business field.
Identification of the drawbacks of the existing system leads to the designing of a computerized system that will be compatible with the existing system the system which is more user-friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the drawbacks of the existing system.

1.2 Historical Background

1.2.1 Earlier Research

1. Django user authentication.
2. Django REST API implementation.
3. Payment integration system develop.
4. Database management.
5. Request handling, backend & frontend connection.
6. Django packages handling.

1.2.2 Recent Research

- Django chat app implementation.
- Django blog app implementation
- Django Saleor.

1.2.3 Features

1. Fully responsive website(for both mobile and larger screens) based on google material design.
2. Log in/Sign up/Logout feature for both Customer and Room Manager.
3.
4. Custom dashboard for both Customer and Room Manager
5.
6. Facility to add, delete, update rooms by Room Manager.
7. The room Manager can see the details of the user that has booked one of his rooms.
8. The customer can cancel the room booking.
9. Contact form support for every visitor of the website.
10. Superusers have access to all the functionality listed above.

1.3. Goal and objectives of the project

The main purpose of the Hotel Management System is to provide an easy way to do all the marketing work. The main reason for designing an HMS website is managing the right purchase. Some of the goals are:

- Building a Hotel Management System
- Social Media Promotion
- Promotion with google ads
- Marketing by email
- E-commerce marketing
- Collecting customer data
- Targeting the target audience Digital Marketing Increases profitability and sales price.

CHAPTER 2

CHAPTER 2

SYSTEM ANALYSIS

2.1 Overview

System analysis is the process of collecting and interpreting facts, diagnosing problems, and using data to recommend system development. System analysis is a problem-solving task that requires in-depth communication between system users and system developers.
System analysis or study is an important part of any system development process. The system is considered complete, inputs are identified and the system is considered to close the survey to identify problem areas. Solutions are offered as a suggestion. The suggestion is reviewed at the user's request and appropriate changes are made. This opens as soon as the user is satisfied with the suggestion.

2.2 Existing System

The current system for Hotel Management System is to visit the shop manually and from the available product choose the item customer want and buying the item by payment of the price of the item.
1.It is less user-friendly.
2.User should go shopping and choose products.
3. It is difficult to find the required product.
4. Product description is limited.
5.It is a time-consuming process
6. Not available for remote users.

2.3 Proposed System

In the proposed system customers do not have to go to the store prohibiting hotel rooms. He can order the hotel rooms he wants to book using his Smartphone. The hotel owner will be the program admin. The owner can appoint hotel managers who will assist the owner in customer management and room order.

2.4 System Requirement Specification

2.4.1 General description

Product Description:
The program consists of two parts. A web application that can provide an online shopping service and an Android customer application by which a user can access the web service from his Smartphone. The Web application should be able to assist the customer in selecting its product and assist the owner in managing customer orders. Customers have a proactive experience and save time by shopping online. Competing with those products online, if stores offer an online portal where their customers can shop online and find products at their door will increase the number of customers.

2.4.2 System Requirement

Non-Functional Requirement

- Efficiency Requirement
  when the online shopping cart began to use the customized shopping product effectively.
- Reliability Requirement
  The system should provide a reliable environment for both the customer and the owner. All orders must reach the administrator without error.
- Usability Requirement
  The Android app is designed for natural and easy use.
- Implementation Requirement
  System startup CSS and Html will be used for data communication. And the database part was developed by myself. Responsive web design is used to make a website compatible with any type of screen.
- Delivery Requirement
  The entire program is expected to be submitted within four months of time through a weekly evaluation by the project guide

2.5 Functional Requirements

1. Reservation/Booking

- record reservations
- record the customer's user name
- record the customer's password
- record the customer's email
- record the customer's phone number
- record the customer's address
- record the customer's state and pin code
- take inputs for room no from the manager
- take inputs for room type from the manager
- take inputs for room price from the manager
- take inputs for room available start date from the manager
- take inputs for room image from the manager
- display the default room rate
- display whether or not the room is guaranteed
- generate a unique confirmation number for each
- record the expected check-in date and time
- The system shall record the expected checkout date and time
- The system shall check in customers
- The system shall allow reservations to be modified without having to reenter all the customer information
- The system shall checkout customers
- The system shall charge the customer for an extra night if they checkout after 11:00 m. The system shall mark guaranteed rooms as "must pay" after 6:00 pm on the check-in date.
- The system shall record customer feedback

2. Management

  - record the manager’s username
  - record the manager’s password
  - record the manager’s email
  - record the manager’s profile pic
  - record the manager’s phone number
  - record the manager’s gender
  - display the hotel occupancy for a specified period of time (days; including the past, present, and future dates).
  - display projected occupancy for a period of time (days).
  - display an exception report, showing where default room and food prices have been overridden
  - allow for the addition of information, regarding rooms, rates, menu items, prices, and user profiles
  - allow for the deletion of information, regarding rooms, rates, menu items, prices, and user profiles
  - allow for the modification of information, regarding rooms, rates, menu items, prices, and user profiles
  - allow managers to assign user passwords

- Associate every online booking with an account
- Limit every account to a single user
- Enable users to search and find the most relevant booking options
- Accept date and time to check available rooms for that particular time
- Booking confirmation should be sent to the specified contact details
- Calculate and display accommodation charges and other utilities
- Cancel bookings
- Cancel bookings
- Display and change records of guests
- Change rooms

2.5.1 USER

User login
Description of feature
This is the feature that the user uses to sign in to the system. The user must log in with their username and password in the system after registration. If this method is invalid, the user is not allowed to enter the system.
Functional Requirement
- Username and password will be provided after user registration is confirmed.
- The password should be hidden from others while you type it in the field

Register New User
Description of feature
The new user will have to sign up for the system by providing important information to view system products.
Purchasing an Item

Description of feature
The user can book the desired product to his or her account by clicking order to the favorite hotel. He can view his ordered hotel room with the click of a booked button. All products rooms to the booking list can be viewed in the room list. The user can cancel an order from the list. After confirming the rooms in the cart the user can send the cart by providing the delivery address. Successfully exporting a cart will be empty.
Functional Requirement
The system must ensure that only a registered customer can purchase goods.

2.5.2 ADMIN

a. Manage User

Description of feature
The administrator can add user, remove user, view user

b. Manage Products

Description of feature
The admin can add the product, remove the product and view the product.
c. Manage Orders

Description of feature
The manager can view orders and cancel orders.
Functional Requirements

-The system must indicate the administrator login.

Admin account must be protected so that only the store owner can access that account

2.5.2 Manager

a. Manage Rooms

Description of feature
The hotel manager has a personal account and can add/remove and update the room status of the hotel.

b. Manage Dates

Description of feature
The room manager has access to change or add the available dates in which a hotel room is available for booking.

c. Manage Prices

Description of feature
The manager can view the room's price and can update the room's price.

2.6 Non-functional Requirements

- Use encryption to avoid bots from booking
- Search results should populate within acceptable time limits
- Users should be helped appropriately to fill in the mandatory fields, in case of invalid input
- The system should accept payments via various payment methods
- Easy to use, efficient, and accessible
- Keep track of documentation, activities, and responses

2.7 Use Case Diagram

CUSTOMER

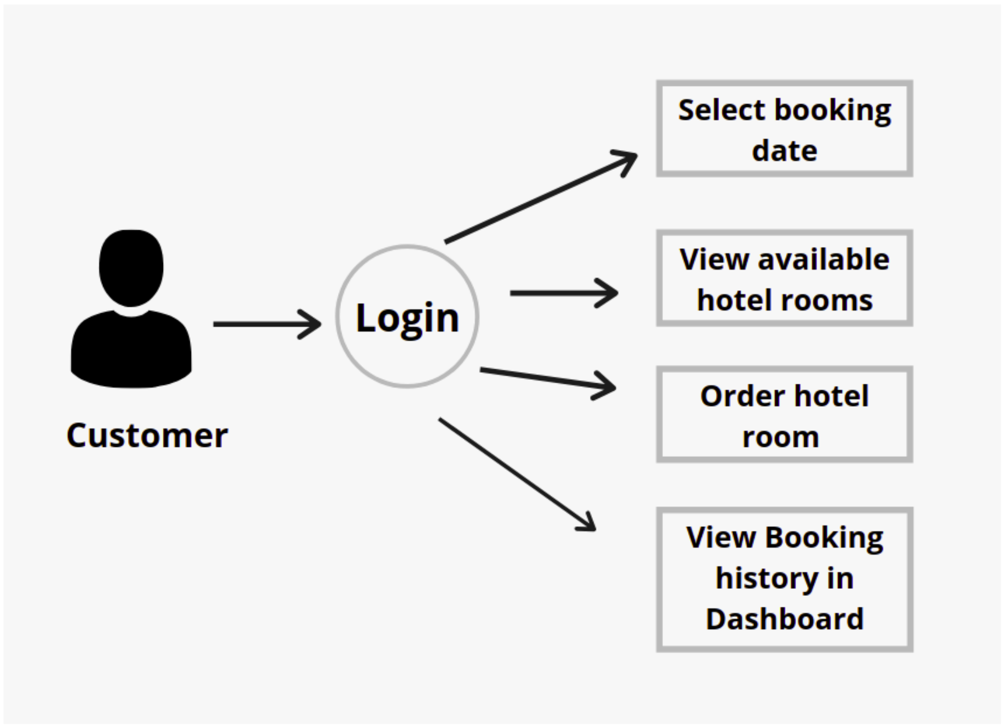


Figure 1: Use Case for Customer

MANAGER

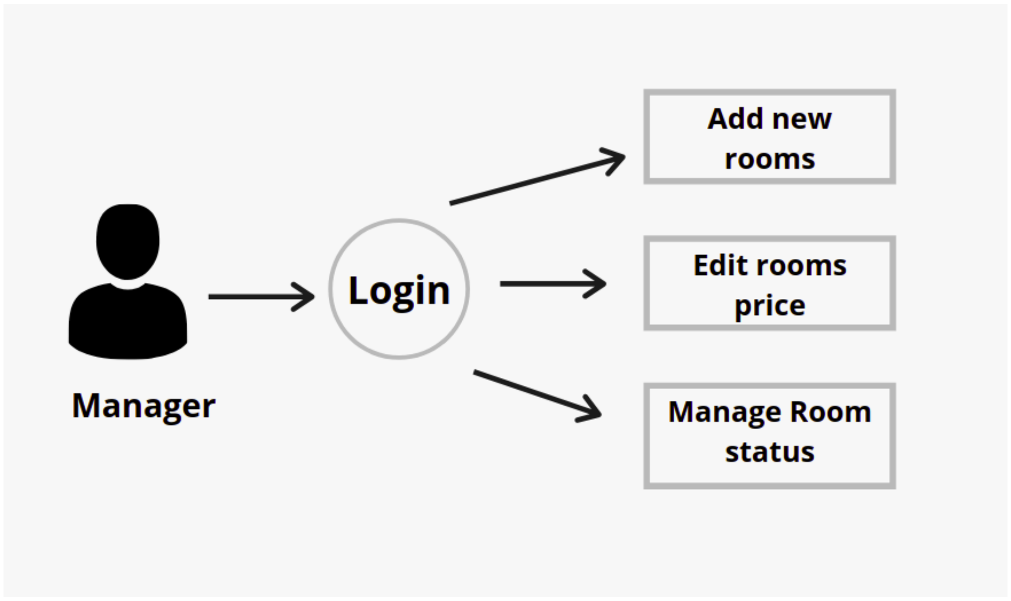


Figure 2: Use Case for Manager

ADMIN

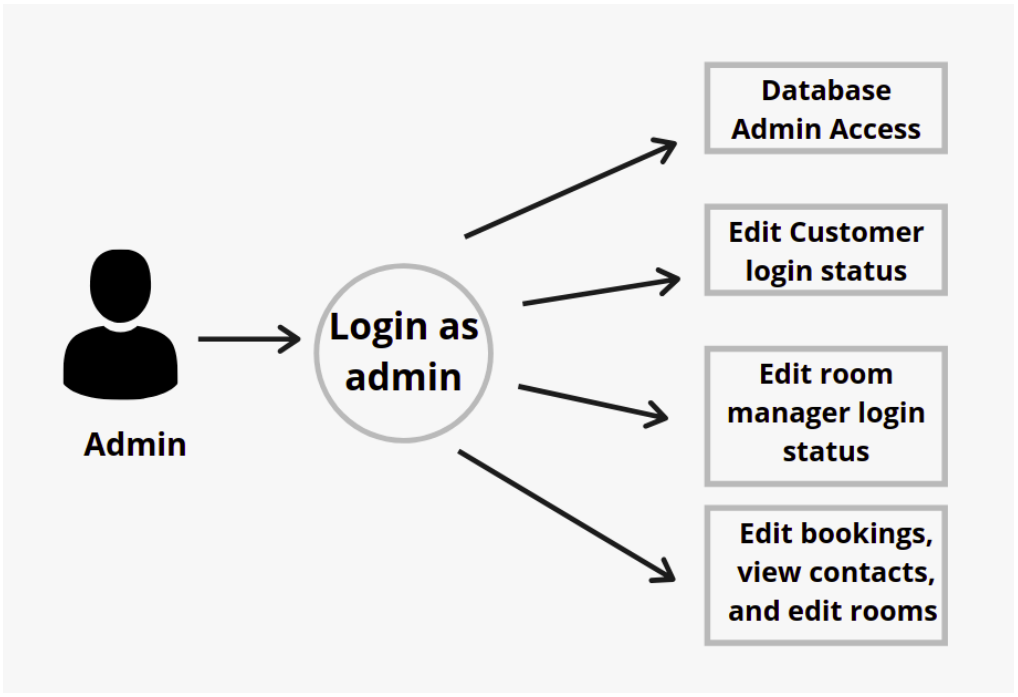

Admin
Login as admin
Database Admin Access
Edit Customer login status
Edit room manager login status
Edit bookings, view contacts, and edit rooms

Figure 3: Use Case for Admin

CHAPTER 3

CHAPTER 3

SYSTEM DESIGN

3.1 Overview

System design is the solution for building a new system. This section focuses on the detailed implementation of the program. It focuses on the translation of the application specification details. The design of the system has two stages of development

a. Logical Design
b. Physical Design

Body structure logical design phase the analyst describes inputs (sources), output s (locations), data details (data wounds), and processes (data flow) all in a format that meets the needs of the user. The analyst also clarifies user needs at a level that determines the flow of information within and outside the system and data resources. Logical design is done with data flow diagrams and information formats. Visual design is followed by physical design or coding. Body design produces a functional plan by defining the design details, specifying exactly what the candidate's plan should do. The editors write the required programs that receive input from the user, perform the required processing on the received data and produce the required report on hard copy or display it on screen.

3.2 Input and Output Design

3.2.1 Input Design

Input design is a link that binds the information system in the world of its users. Input design includes input data validation, minimizing data input, and providing space for multiple users. Inaccurate inputs are the most common cause of data usage errors. Errors inserted in data entry operators can be controlled by input build.

3.2.2 Output Design

Computer output is a very important and direct source of information for the user. The design of the output is a very important phase because the output needs to be efficient. An effective and intuitive output design enhances system-user relationships and assists decision-making. Allowing the user to view a screen sample is important because the user is the best output quality, judge. The output module of this program is the notifications selected.

3.3 Architecture Diagram

An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components.

3.3.1 Activity UML diagram of Hotel Management System

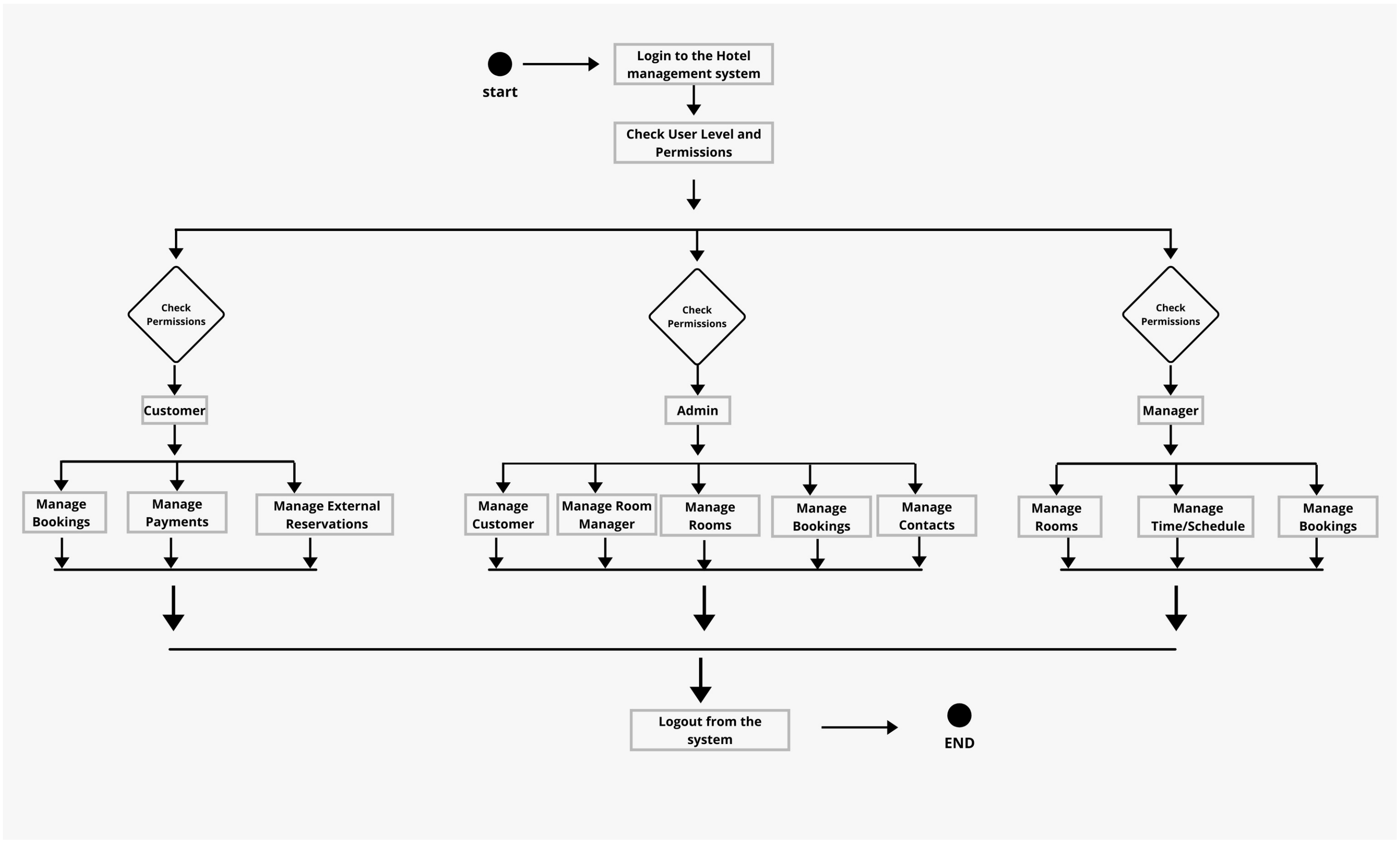


Figure 4: The activity UML diagram of hotel management system

3.3.2 Class Based Architecture Diagram

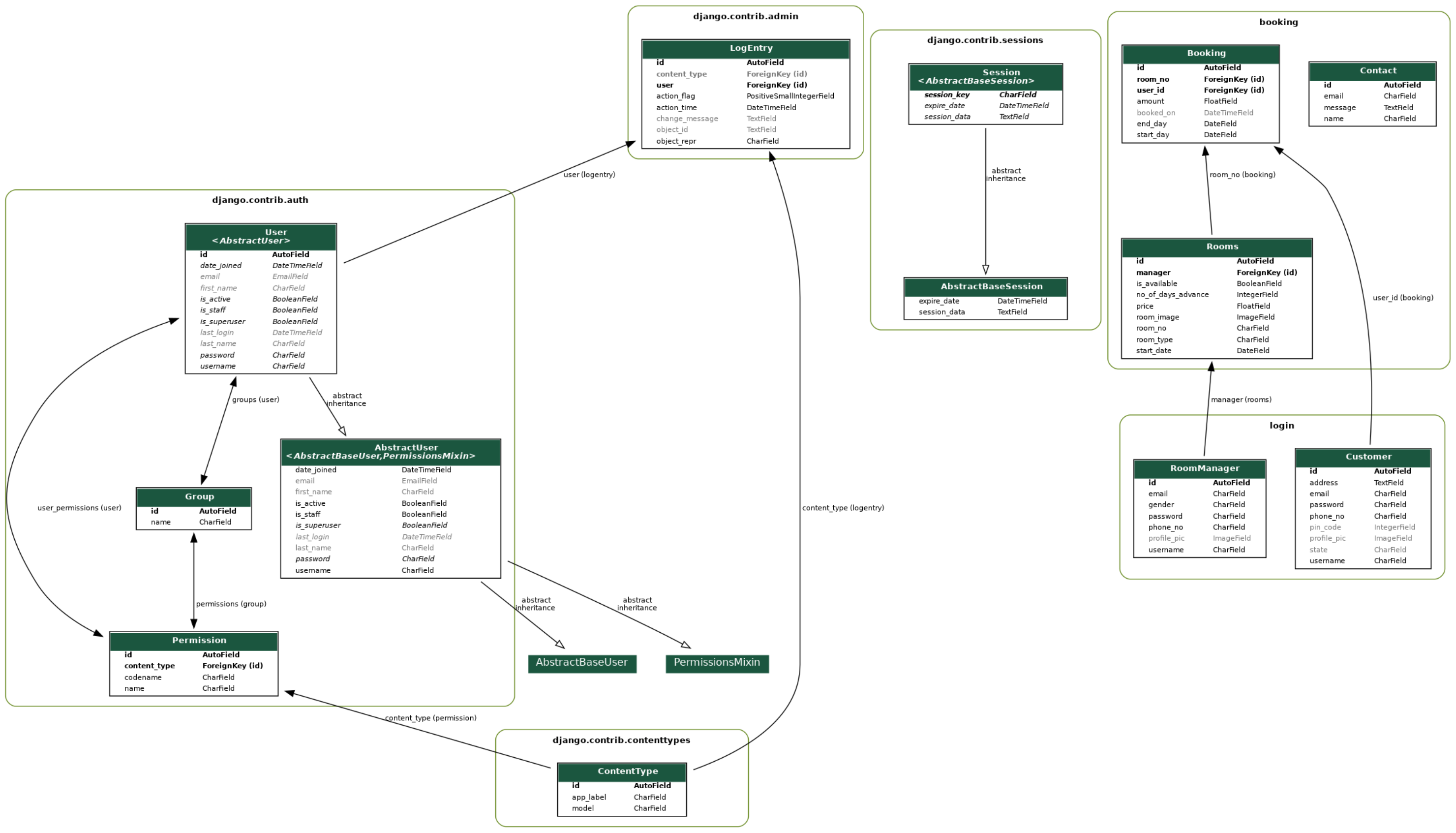


Figure 5: The class based architectural diagram of hotel management system.

3.4 Database

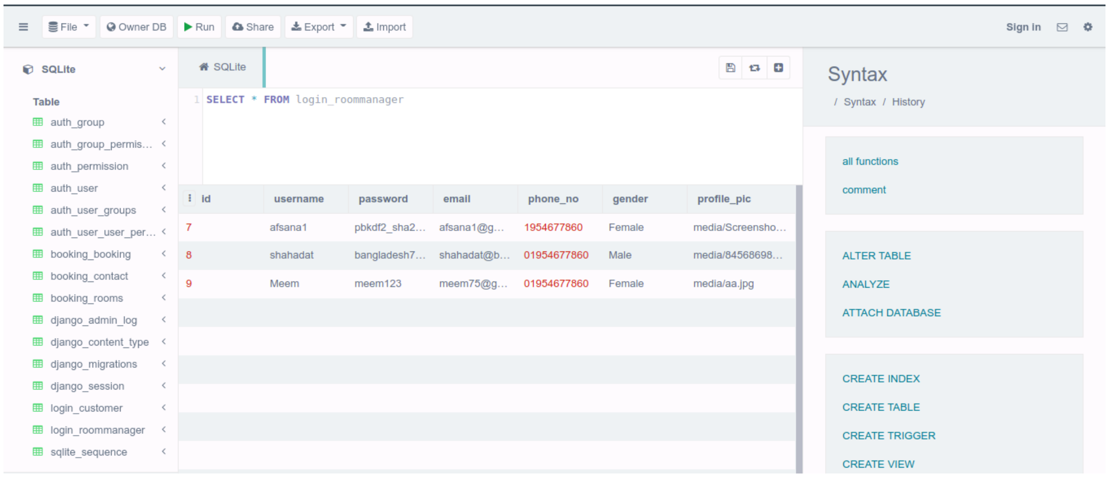
Database Design
Databases are data repositories used in software applications. The data is stored in tables within the database. Several tables are for the use of system information. Two important database settings are

- Primary Key: A separate field for all record operations.
- Foreign Key: The field used to set the relationship between the tables.

Normalization is a technique to avoid redundancy in the tables.

Database Query:

SELECT * FROM login_roommanager



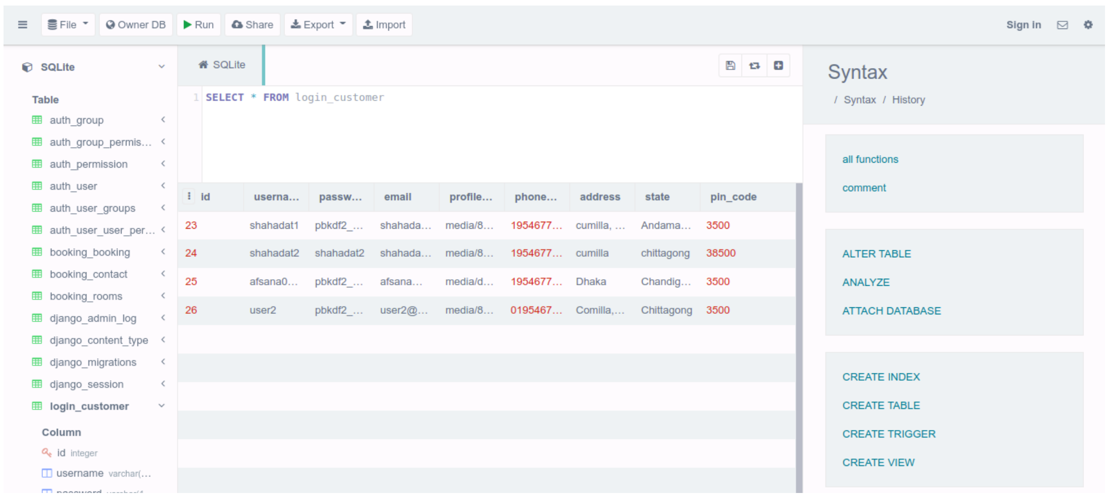
```
SELECT * FROM login_roommanager
```

id	username	password	email	phone_no	gender	profile_pic
7	afsana1	pbkdf2_sha2…	afsana1@g…	1954677860	Female	media/Screensho…
8	shahadat	bangladesh7…	shahadat@b…	01954677860	Male	media/84568698…
9	Meem	meem123	meem75@g…	01954677860	Female	media/aa.jpg

Figure 6: Database Table 1, login_roommanager

Database Query:

SELECT * FROM login_customer



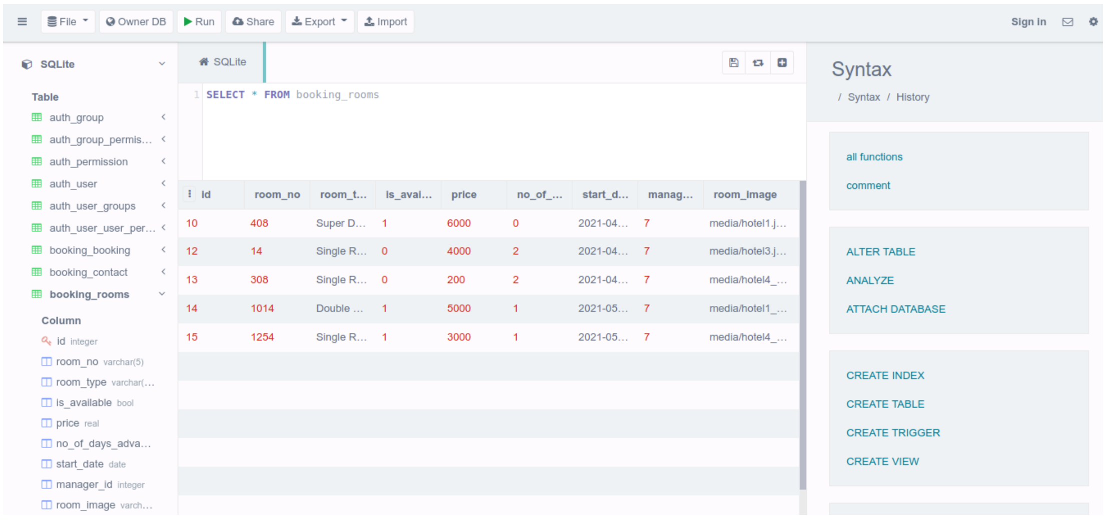
```
SELECT * FROM login_customer
```

id	userna…	passw…	email	profile…	phone…	address	state	pin_code
23	shahadat1	pbkdf2_…	shahada…	media/8…	1954677…	cumilla, …	Andama…	3500
24	shahadat2	shahadat2	shahada…	media/8…	1954677…	cumilla	chittagong	38500
25	afsana0…	pbkdf2_…	afsana…	media/d…	1954677…	Dhaka	Chandig…	3500
26	user2	pbkdf2_…	user2@…	media/8…	0195467…	Comilla,…	Chittagong	3500

Figure 7: Database Table 2, login_customer

Database Query:

SELECT * FROM booking_rooms



```
SELECT * FROM booking_rooms
```

id	room_no	room_t...	is_avai...	price	no_of_...	start_d...	manag...	room_image
10	408	Super D...	1	6000	0	2021-04...	7	media/hotel1.j...
12	14	Single R...	0	4000	2	2021-04...	7	media/hotel3.j...
13	308	Single R...	0	200	2	2021-04...	7	media/hotel4_...
14	1014	Double ...	1	5000	1	2021-05...	7	media/hotel1_...
15	1254	Single R...	1	3000	1	2021-05...	7	media/hotel4_...

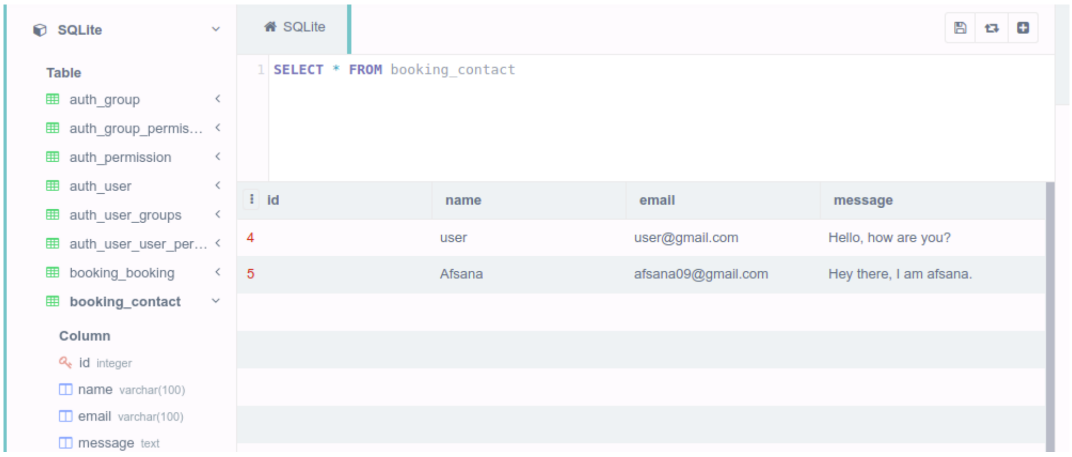
Table: auth_group, auth_group_permis..., auth_permission, auth_user, auth_user_groups, auth_user_user_per..., booking_booking, booking_contact, booking_rooms

Column: id integer, room_no varchar(5), room_type varchar(..., is_available bool, price real, no_of_days_adva..., start_date date, manager_id integer, room_image varch...

Figure 8: Database Table 3, booking_rooms

Database Query:

SELECT * FROM booking_contact



```
SELECT * FROM booking_contact
```

id	name	email	message
4	user	user@gmail.com	Hello, how are you?
5	Afsana	afsana09@gmail.com	Hey there, I am afsana.

Table: auth_group, auth_group_permis..., auth_permission, auth_user, auth_user_groups, auth_user_user_per..., booking_booking, booking_contact

Column: id integer, name varchar(100), email varchar(100), message text

Figure 9: Database Table 4, booking_contact

3.5 System Tools

The various planning tools used to create both the front and back end of the project are discussed in this chapter.

3.5.1 Front End

HTML, CSS, JAVASCRIPT are utilized to implement the frontend.

HTML:

Hypertext Markup Language (HTML) is the standard language for marking texts designed to be displayed in a web browser.

CSS:

Cascading Style Sheets (CSS) is a style language used to describe the presentation of a document written in HTML or XML.

JAVASCRIPT:

JavaScript (JS) is a structured language that is lightweight, translated, or up-to-date with first-class tasks. Although it is widely known as the writing language of Web pages, many non-browser sites also use it, such as Node.js, Apache Couch DB, and Adobe Acrobat.

Bootstrap:

Bootstrap is a large collection of usable, usable codes written in HTML, CSS, and JavaScript. There are also prior developments framework that enables developers & designers to quickly build fully responsive websites.

3.5.2 Back End

Python:
Python is a programming language that is translated to the highest level of the world. The Python architecture philosophy emphasizes the readability of the code with its remarkable application of critical direction. Its language is structured and its object-oriented approach aims to assist editors in writing clear, logical code for small and large projects.

Django:
Django is a state-of-the-art Python web framework that enables the rapid development of secure and secure websites. Built by experienced engineers, Django takes great care of the web development hassles, so you can focus on writing your app without having to refresh the wheel.

3.5.3 Database

SQLite:
SQLite is a software library that provides a database management system. Lite in SQLite means survival in terms of setup, data management, and required resources. SQLite has the following visual features: self-contained content, sub-server, zero-configuration, functionality.

CHAPTER 4

CHAPTER 4
METHODOLOGY

4.1: Methodology

We have used incremental method on the proposed system.

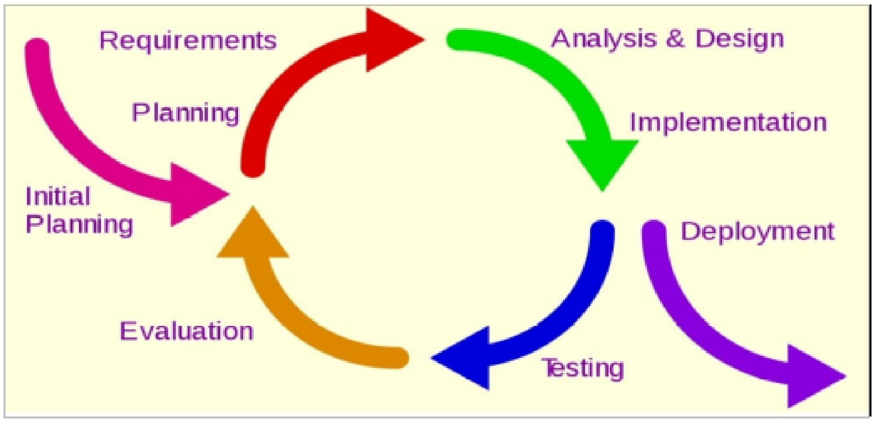


Figure 10: Methodology Plan

4.2 Project Structure:

Before we start building, let’s take a look at the basic structure of this project. The Data Flow Diagram (DFD) represents the flow of data and the transformations in the Hotel management system. These transformations occur as data enters and exits a system. In the DFD, input, processing, and output are used to represent and define the overall system.

The following are the flows that the Hotel Management System can generate:

- Managing Customers
- Assigning Rooms
- Managing Employees
- Facilitates Reservation
- Manage Transaction

Here’s the Data flow Diagram levels 0 1 2 for Hotel Management System and a step-by-step way on how to create a Hotel management system DFD.

4.3 DFD for Hotel Management System Level 0

To start with, let us familiarize what is Hotel management system level 0.

The hotel management system level 0 is also known as the context diagram. It’s supposed to be an abstract view, with the mechanism represented as a single process with external parties.

This DFD for the system depicts the overall structure as a single bubble. It comes with incoming/outgoing indicators showing input and output data.

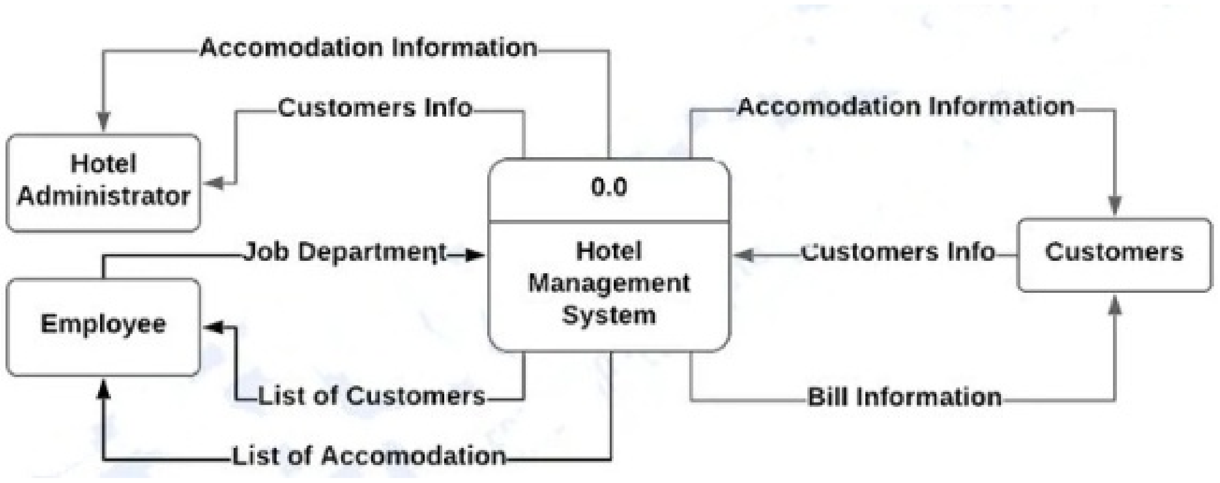


Figure 11: DFD Level 0

4.4 DFD for Hotel Management System Level 1

Next to the context diagram is the level 1 data flow diagram.

The content of the Hotel management system DFD level 1 must be a single process node from the context diagram and is broken down into sub-processes

At this level, the system must display or reveal further processing information. And the actors that are going to use this system were the customers, hotel administrators,s and hotel employees.

The following are essential data to accommodate:

- Customer Information
- Employees Information
- Rooms
- Reservation

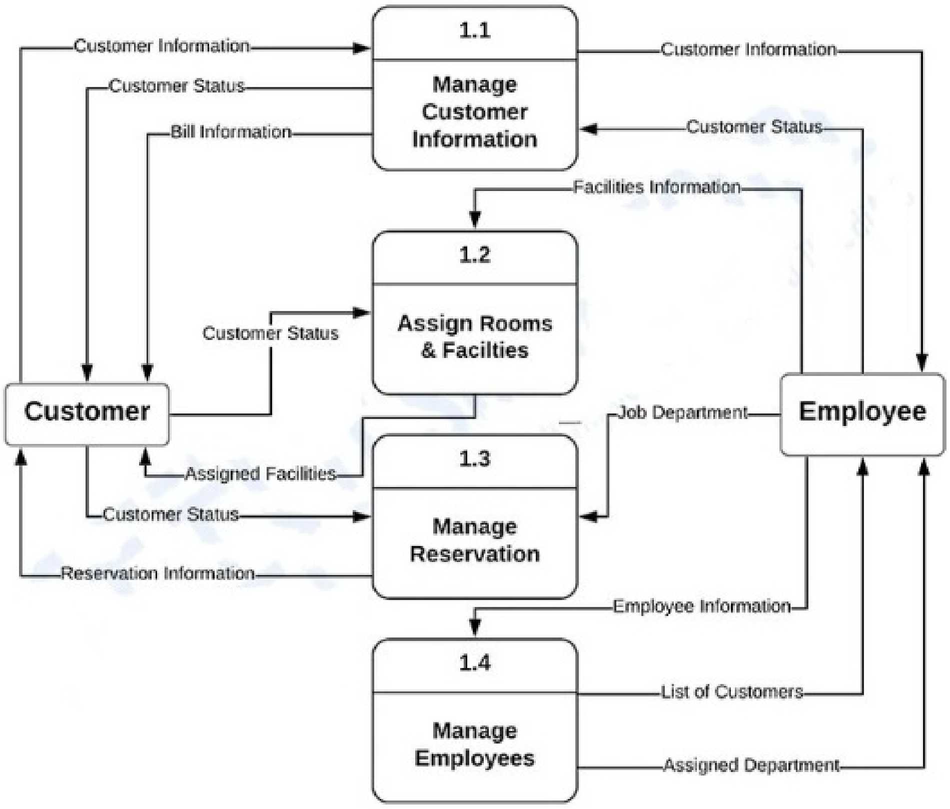


Figure 12: DFD Level 1

These procedures require information such as a list of customers, rooms, employees, and facilities from which served as the bases for admin to manage hotel transactions. This type of data is represented by a data store.

With being knowledgeable about the DFD level 1 of the Hotel Management System, you will know then its broaden context terms.

In addition to that, this may also serve as your reference on how the inputs or data fed on the system. Then you will be also informed about the outputs that the system gives.

These processes shown in the DFD were all based on the concept of Hotel Management System.

4.5 DFD for Hotel Management System Level 2

After presenting the Hotel management system DFD levels 0 and 1, next to that is level 2.

Here's what you need to consider in creating data flow diagram level 2 for the Hotel management system.
• The Level 2 DFD for the system should represent the basic modules as well as data flow between them.
• Since the DFD level 2 is the highest abstraction level, its Hotel management system processes must be detailed that is based on the DFD level 1.

Finally, after figuring the processes given in the system, the user will now have their request being processed.

The Processes that the system should prioritize are as follows:
• Managing Clients
• Assigning Rooms
• Managing Employees
• Facilitates Reservation
• Manage Transactions

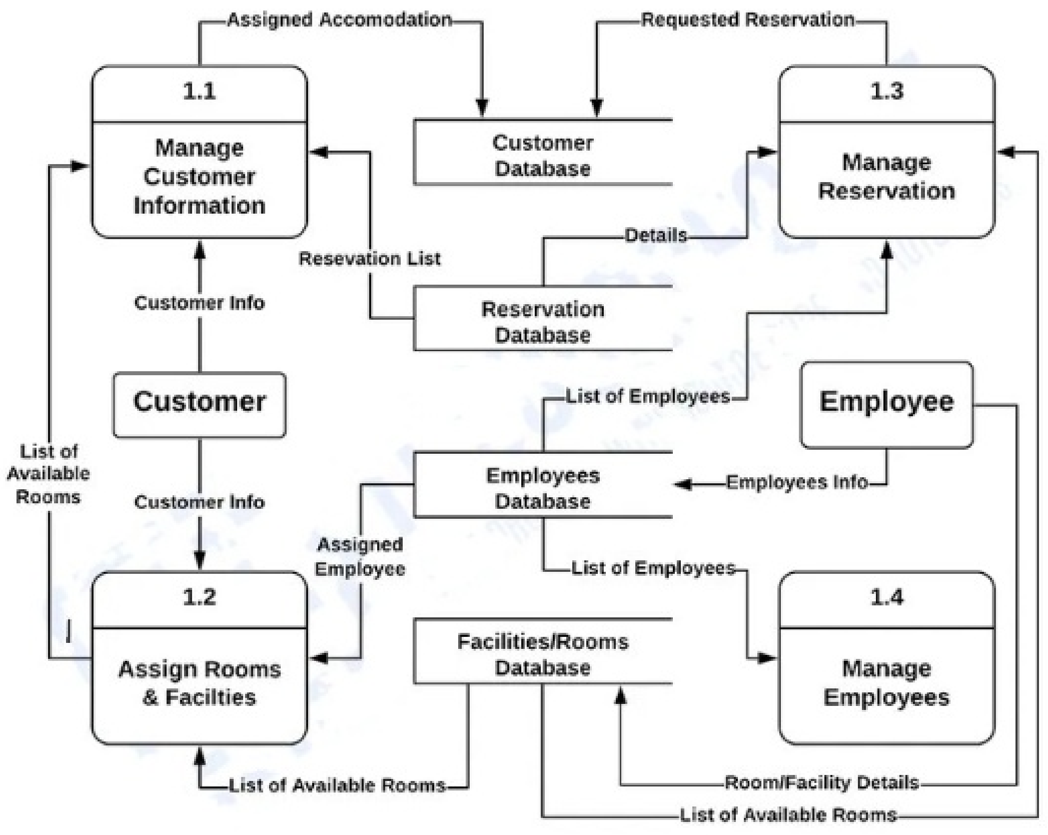


Figure 13: DFD Level 2

DFD level 2 lets us know the ideas on where does the data inputs goes and inputs comes within the Hotel management system. Considering the the data flow levels mentioned above, you can determine well the importance of breaking the processes into more specific manner.

The presented level not only shows you the detailed processes of system, but also gives you precise destination of the data that flows in the system.

This DFD will also be your references as you make your own management system DFD levels 0, 1 and 2.

4.6 Modules

This project will consist of 5 models, so let's summarize each one:

1.USER- Built on Django user model, an example created for each customer who
subscribes.

2.CUSTOMER- Along with the user model, each customer will contain a Customer model that governs the person-to-person relationship for each user. (One to one field).

3. ROOM_MANAGER - Along with the user model, each room manager will contain a model that governs the person-to-person relationship for each user. (One to one field).
4. BOOKING- The booking model represents the room booking the customers have booked.

5. ROOMS – The rooms model represents the existing rooms in the hotel added by the room manager.

6. CONTACT- The contact model represents the inbox message send by website visitors or customers.

CHAPTER 5

CHAPTER 5
IMPORTANT CODE

5.1 URLS

A URL is a web address. You can see a URL every time you visit a website – it is visible in your browser's address bar. Every page on the Internet needs its own URL. This way your application knows what it should show to a user who opens that URL. In Django, we use something called URLconf (URL configuration). URLconf is a set of patterns that Django will try to match the requested URL to find the correct view. It's where you define the mapping between URLs and views. A mapping is a tuple in URL patterns like – from django. conf. urls import patterns include url from django.

Source Code: login/urls.py

```
from django.urls import path, include
from . import views

urlpatterns = [
    path('login', views.user_login, name='user_login'),
    path('login1', views.manager_login, name='manager_login'),
    path('signup', views.user_signup, name='user_signup'),
    path('signup1', views.manager_signup, name='manager_signup'),
    path('dashboard/', include('customer.urls')),
    path('dashboard1/', include('room_manager.urls')),
    path('add-room/', include('room_manager.urls'))
]
```

5.2 SERIALIZERS

Serializers allow complex data such as querysets and model instances to be converted to native Python datatypes that can then be easily rendered into JSON, XML, or other content types. Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data.

Source Code: serializers.py

```
from rest_framework import serializers
from login.models import Customer
from booking.models import Booking, Rooms

class UserSerializer(serializers.ModelSerializer):
class Meta:
 model = Customer
 fields = ['id', 'username', 'password', 'email', 'state', 'pin_code', 'address',
'profile_pic']

class RoomsSerializer(serializers.ModelSerializer):
class Meta:
 model = Rooms
 fields = ['id', 'manager', 'room_no', 'room_type', 'is_available', 'price',
'no_of_days_advance', 'start_date',
'room_image']
```

```
class BookingSerializer(serializers.ModelSerializer):
class Meta:
 model = Booking
 fields = ['id', 'room_no', 'user_id', 'start_day', 'end_day', 'amount', 'booked_on']
```

5.3 User Sign up

I made a signup view for handling the user signup functionality, This manages the POST requests from the front end and works as defined in the backend.

Source Code: login/views.py

```
def user_signup(request):
if request.session.get('username', None) and request.session.get('type',
None) == 'customer':
return redirect('user_dashboard')
if request.session.get('username', None) and request.session.get('type',
None) == 'manager':
return redirect('manager_dashboard')
if request.method == "POST":
 username = request.POST['username']
 email = request.POST['email']
if Customer.objects.filter(username=username) or
Customer.objects.filter(email=email):
 messages.warning(request, "Account already exist, please Login to
continue")
else:
 password = request.POST['password']
 address = request.POST['address']
 pin_code = request.POST['pin_code']
 profile_pic = request.FILES.get('profile_pic', None)
 phone_no = request.POST['phone_no']
 state = request.POST['state']
 error = []
 if (len(username) < 3):
  error.append(1)
  messages.warning(request, "Username Field must be greater than 3
 character.")
 if (len(password) < 5):
  error.append(1)
  messages.warning(request, "Password Field must be greater than 5
 character.")

if (len(email) == 0):
 error.append(1)
 messages.warning(request, "Email field can't be empty")
if (len(phone_no) != 11):
 error.append(1)
 messages.warning(request, "Valid Phone number is a 10 digit-integer.")
if (len(error) == 0):
 password_hash = make_password(password)
 customer = Customer(username=username, password=password_hash,
email=email, phone_no=phone_no,
address=address, state=state, pin_code=pin_code, profile_pic=profile_pic)
 customer.save()
 messages.info(request, "Account Created Successfully, please Login to
continue")
 redirect('user_signup')
else:
 redirect('user_signup')


else:
 redirect('user_signup')
return render(request, 'login/user_login.html', {})
```

5.4 User LOGIN

I made a signup view for handling the user login functionality, This manages the POST and GET requests from the front end and works as defined in the backend. This function is used for the login into a user account.

Source Code: login/views.py

```
def user_login(request):
if request.session.get('username', None) and request.session.get('type',
None) == 'customer':
return redirect('user_dashboard')
if request.session.get('username', None) and request.session.get('type',
None) == 'manager':
return redirect('manager_dashboard')
if request.method == "POST":
 username = request.POST['username']
 password = request.POST['password']
if not len(username):
 messages.warning(request, "Username field is empty")
 redirect('user_login')
elif not len(password):
 messages.warning(request, "Password field is empty")
 redirect('user_login')
else:
pass
if Customer.objects.filter(username=username):
 user = Customer.objects.filter(username=username)[0]
 password_hash = user.password
 res = check_password(password, password_hash)
```

```
if res == 1:
 request.session['username'] = username
 request.session['type'] = 'customer'
return render(request, 'booking/index.html', {})
else:
 messages.warning(request, "Username or password is incorrect")
 redirect('user_login')
else:
 messages.warning(request, "No, Account exist for the given Username")
 redirect('user_login')
else:
 redirect('user_login')
return render(request, 'login/user_login.html', {})
```

5.5 Logout

This function is used to logout from the users account.

Source Code: login/views.py

```
def logout(request):
    if request.session.get('username', None):
        del request.session['username']
        del request.session['type']
        return render(request, "login/logout.html", {})
    else:
        return render(request, "login/user_login.html", {})
```

5.6 Manager Dashboard View

Every Manager has an individual account, and each account has a dashboard for showing specific detailed information. This function handles the dashboard functionality.

Source Code: room_manager/views.py

```
def dashboard(request):
  if not request.session.get('username',None):
     return redirect('manager_login')
  if request.session.get('username',None) and
request.session.get('type',None)=='customer':
       return redirect('user_dashboard')
  if request.session.get('username',None) and
request.session.get('type',None)=='manager':
     username=request.session['username']
     data=RoomManager.objects.get(username=username)
     room_data=data.rooms_set.all()
     booked=room_data.filter(is_available=False).count()
     print(booked)
     return render(request,"manager_dash/index.html",
{"room_data":room_data,"manager":data,"booked":booked})
  else:
     return redirect("manager_login")
```

5.7 Customer Dashboard View

Every customer has an individual account, and each account has a dashboard for showing specific detailed information. This function handles the dashboard functionality.

Source Code: customer/views.py

```
def dashboard(request):
    if request.session.get('username', None) and
    request.session.get('type', None) == 'manager':
    return redirect('manager_dashboard')
    if request.session.get('username', None) and
    request.session.get('type', None) == 'customer':
     username = request.session['username']
     data = Customer.objects.get(username=username)
     booking_data = Booking.objects.filter(user_id=data).order_by('-id')
     counts =
    booking_data.filter(end_day__lt=datetime.datetime.now()).count()
     available = len(booking_data) - counts
    return render(request, "user_dash/index.html", {"data":
    booking_data, "count": counts, "available": available})
    else:
    return redirect("user_login")
```

5.8 Customer Detail

This detail function handles the customer detail functionalities.

Source Code: customer/views.py

```
def details(request, id, booking_id):
        if not request.session.get('username', None):
            return redirect('manager_login')
        if request.session.get('username', None) and
    request.session.get('type', None) == 'customer':
            return redirect('user_dashboard')
        try:
            booking_data = Booking.objects.get(id=booking_id)
            user = Customer.objects.get(id=id)
            return render(request, "user_dash/details.html", {"user": user,
    "booking_data": booking_data})
        except:
            return redirect("/manager/dashboard1/")
```

5.9 Room Book

This book function handles the requests from the customer side and manages the room booking functionalities.

Source Code: booking/views.py

```
def book(request):
    if request.method == "POST":
     start_date = request.POST['start_date']
     end_date = request.POST['end_date']
     request.session['start_date'] = start_date
     request.session['end_date'] = end_date
```

```
start_date = datetime.datetime.strptime(start_date, "%d/%b/%Y").date()
 end_date = datetime.datetime.strptime(end_date, "%d/%b/%Y").date()
 no_of_days = (end_date - start_date).days
 data = Rooms.objects.filter(is_available=True,
no_of_days_advance__gte=no_of_days, start_date__lte=start_date)
 request.session['no_of_days'] = no_of_days
return render(request, 'booking/book.html', {'data': data})
else:
return redirect('index')
```

5.10 Cancel Room

If the customer wants to cancel any ordered room from his order list this cancel_room function handles this function.

Source Code: booking/cancel_room.py

```
def cancel_room(request, id):
    data = Booking.objects.get(id=id)
    room = data.room_no
    room.is_available = True
    room.save()
    data.delete()
    return HttpResponse("Booking Cancelled Successfully")
```

5.11 Relationships Between Data Tables

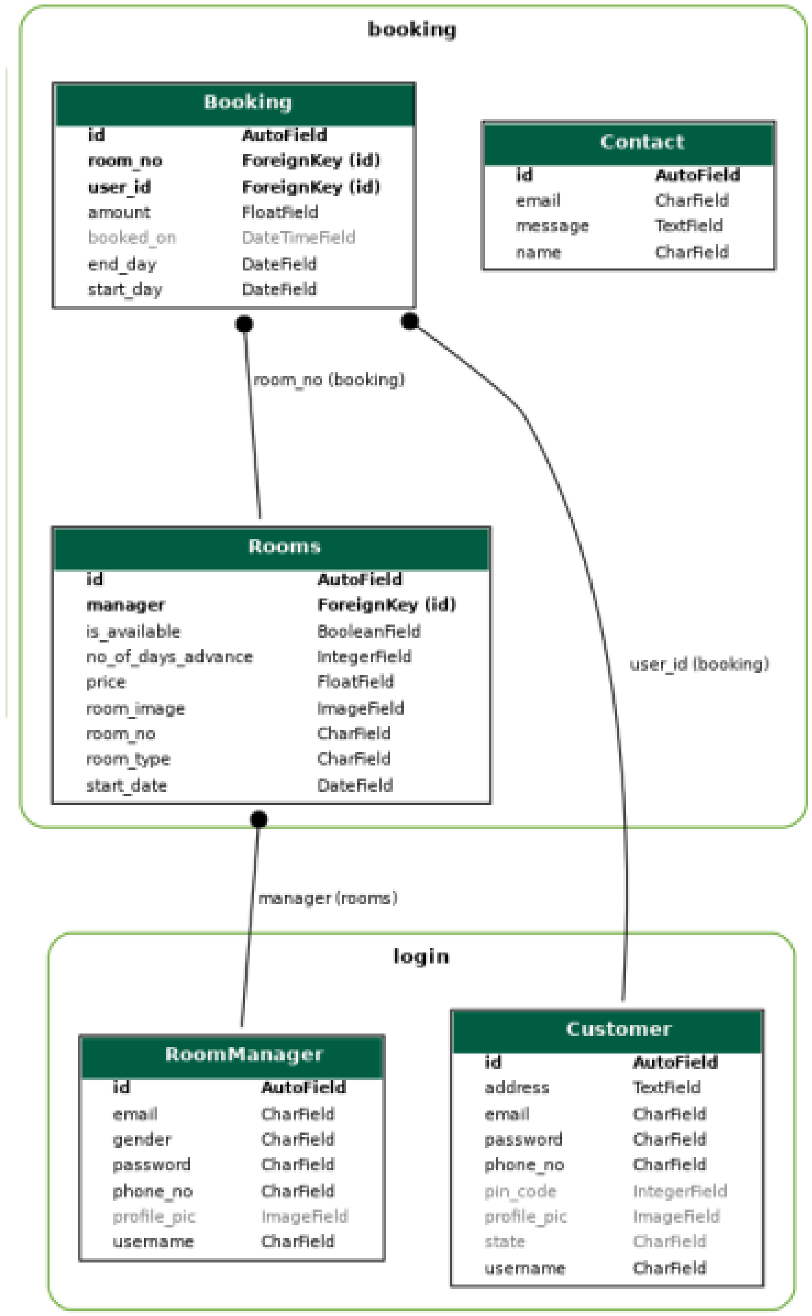


Figure 14: Visualization of the relationships between database tables.

CHAPTER 6

CHAPTER 6
SCREENSHOT WITH PROJECT DETAIL

6.1 Main Interface

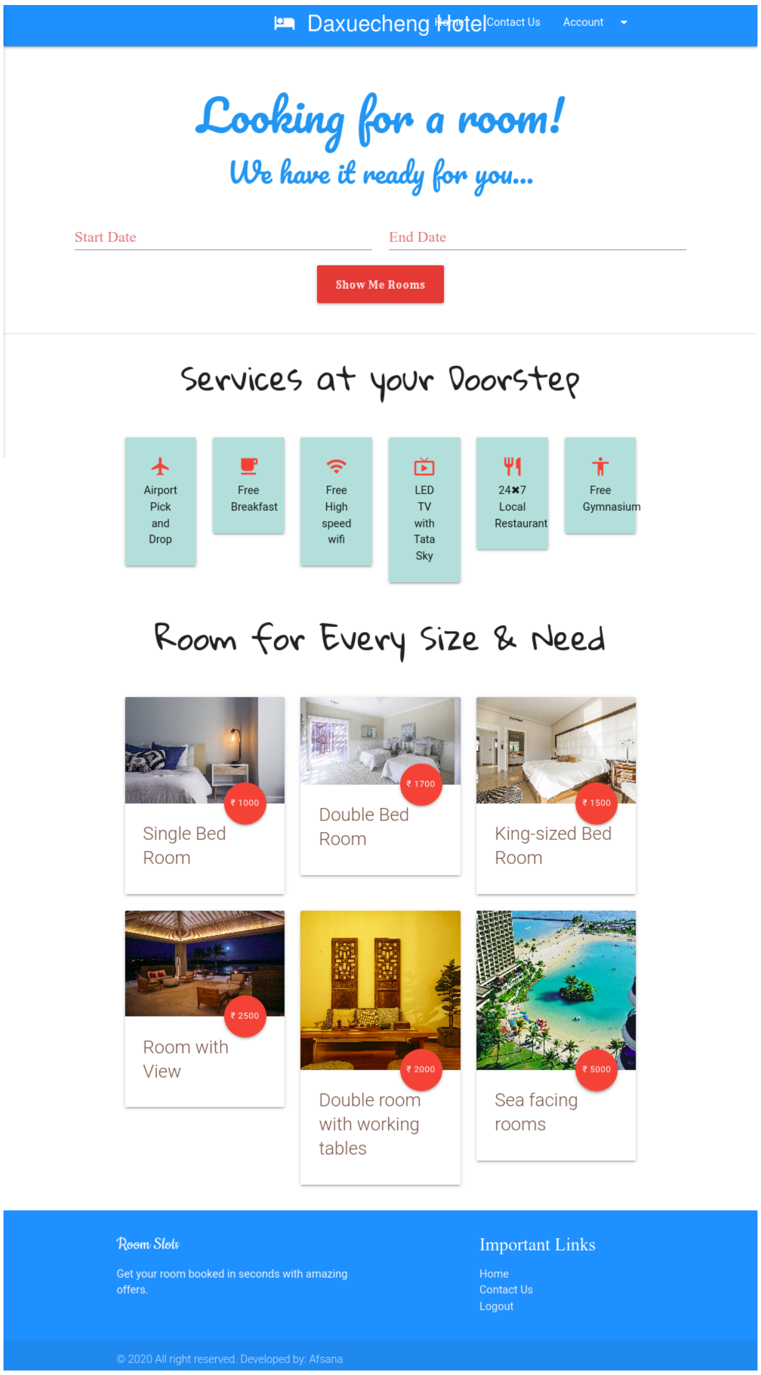
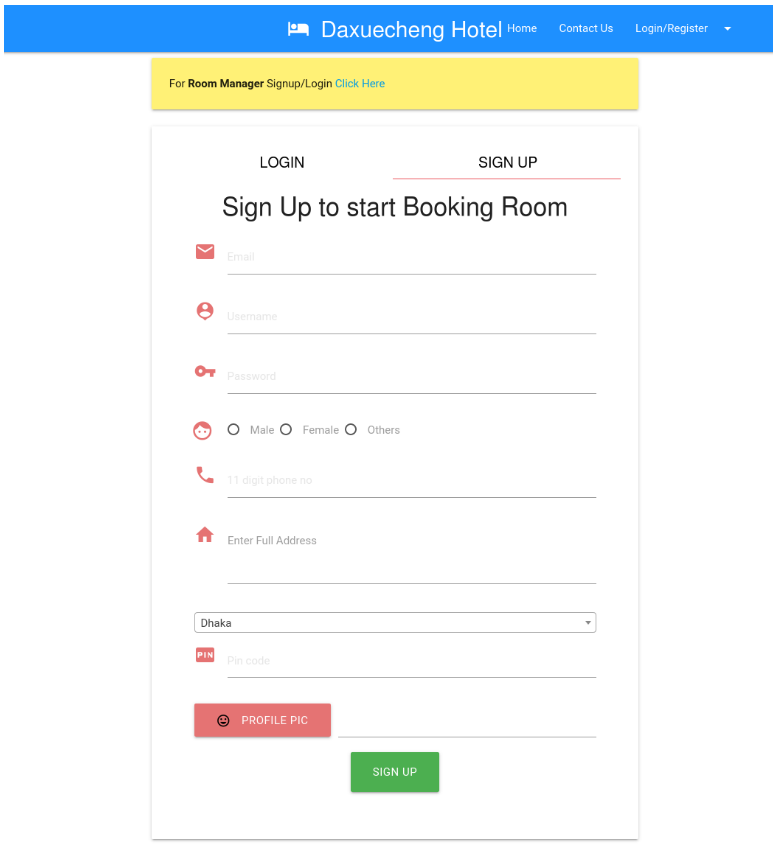


Figure 15: HMS home view

6.2 Customer Signup Interface

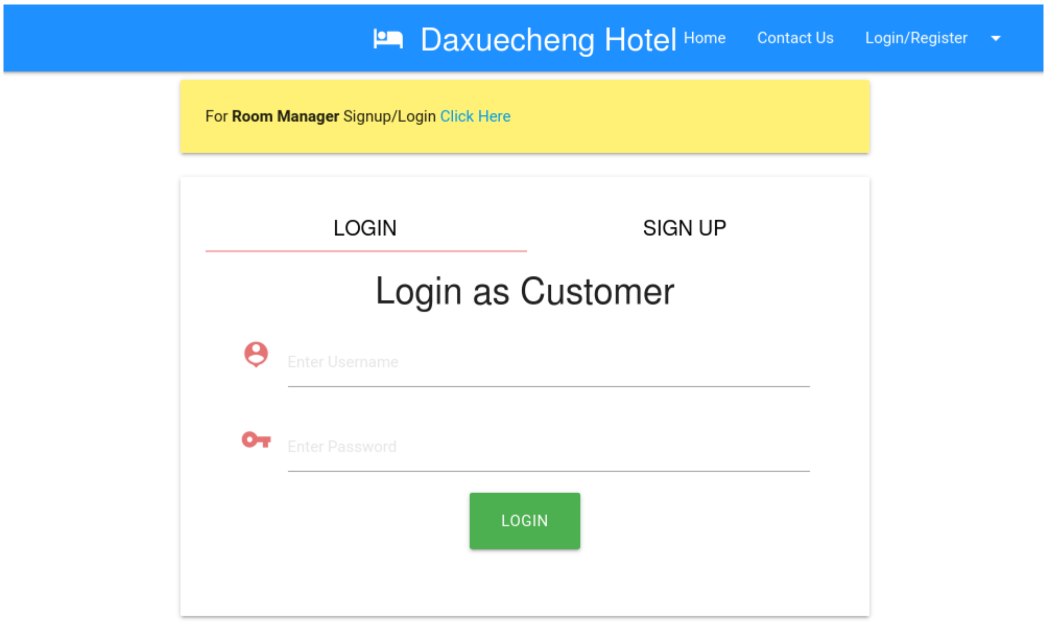


For **Room Manager** Signup/Login Click Here

Sign Up to start Booking Room

Figure 16: Customer signup view

6.3 Customer Login Interface

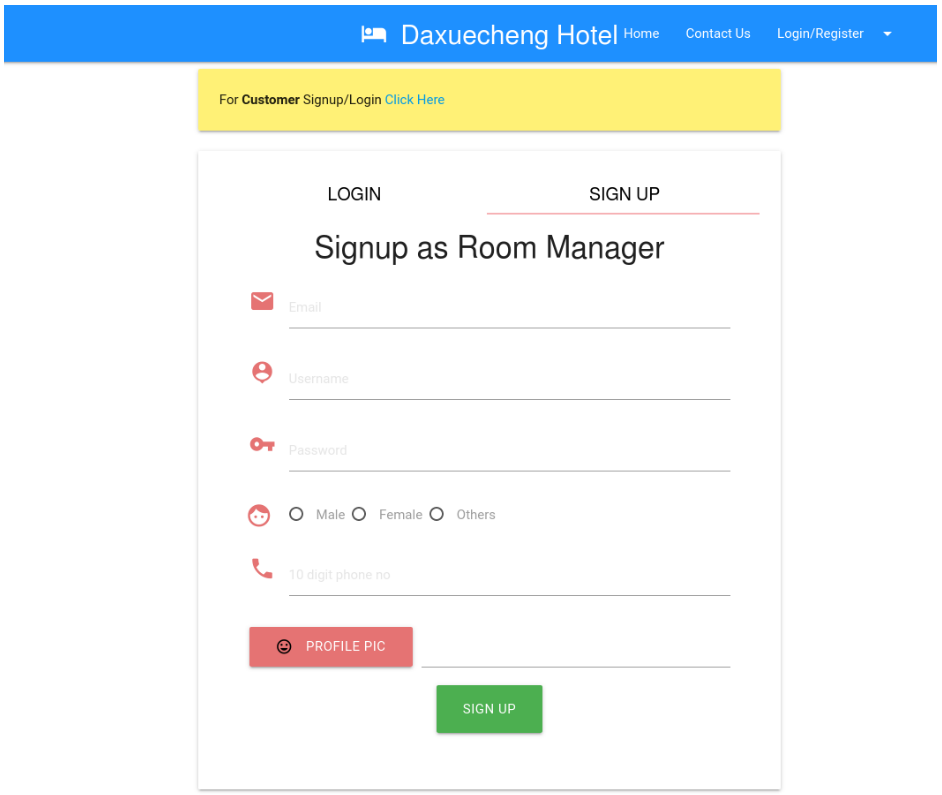


For **Room Manager** Signup/Login Click Here

Login as Customer

Figure 17: Customer login view

6.4 Manager Signup Interface

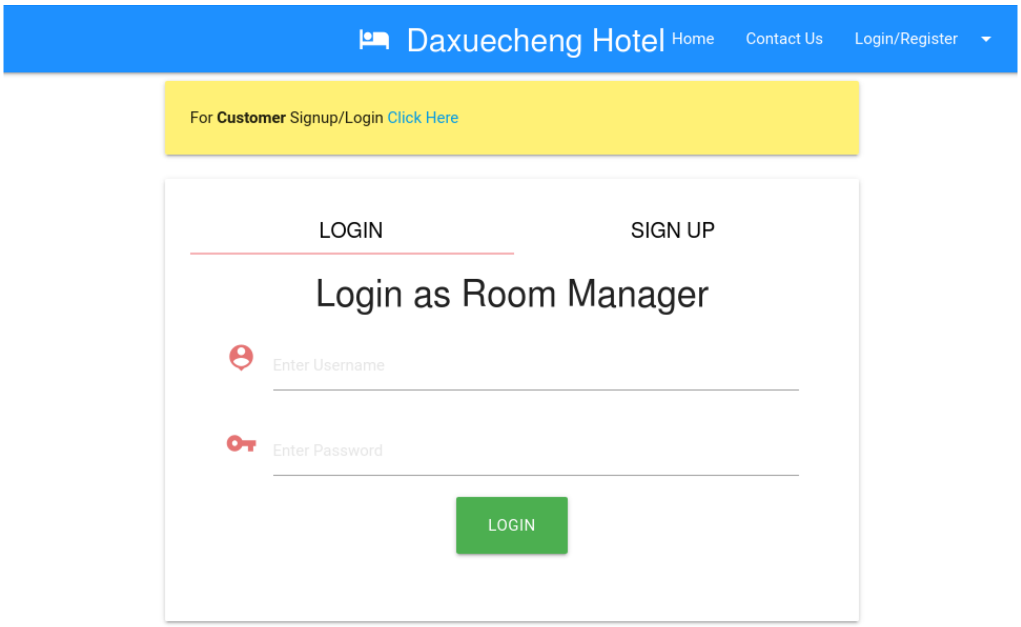


For **Customer** Signup/Login Click Here

Signup as Room Manager

Figure 18: Manager signup view

6.5 Manager Login Interface

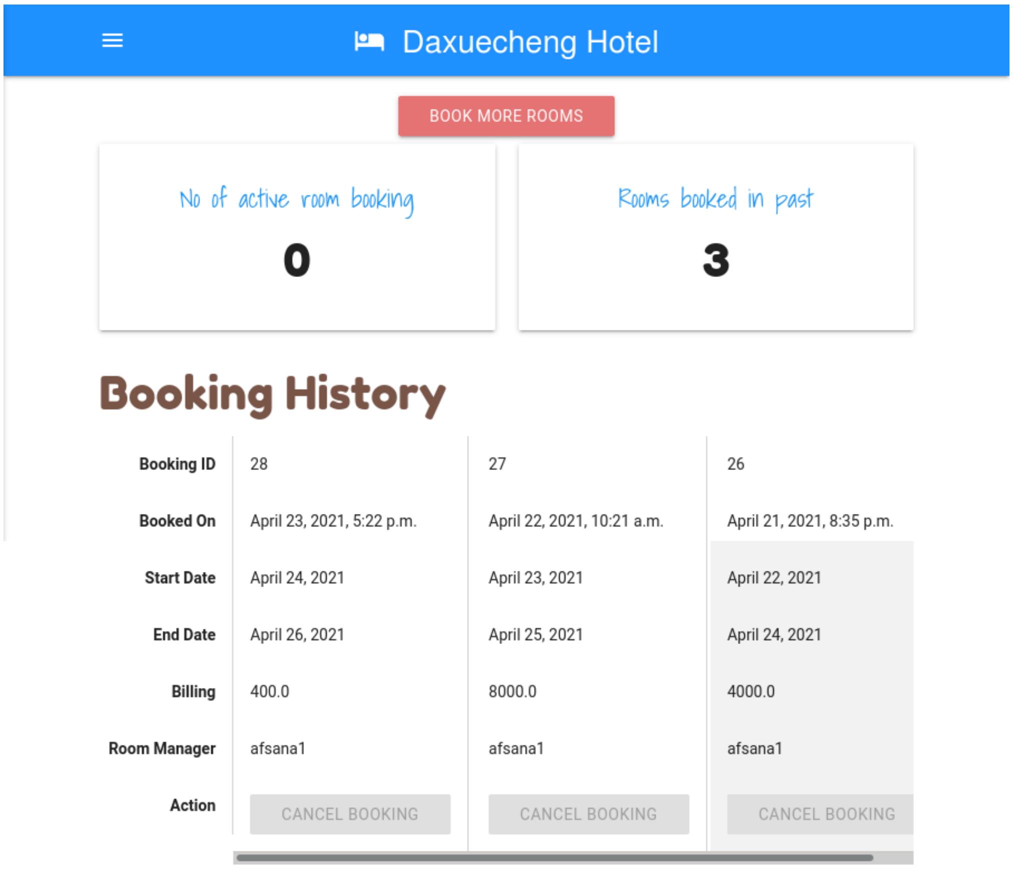


For **Customer** Signup/Login Click Here

Login as Room Manager

Figure 19: Manager login view

6.6 Customer Dashboard



No of active room booking
0

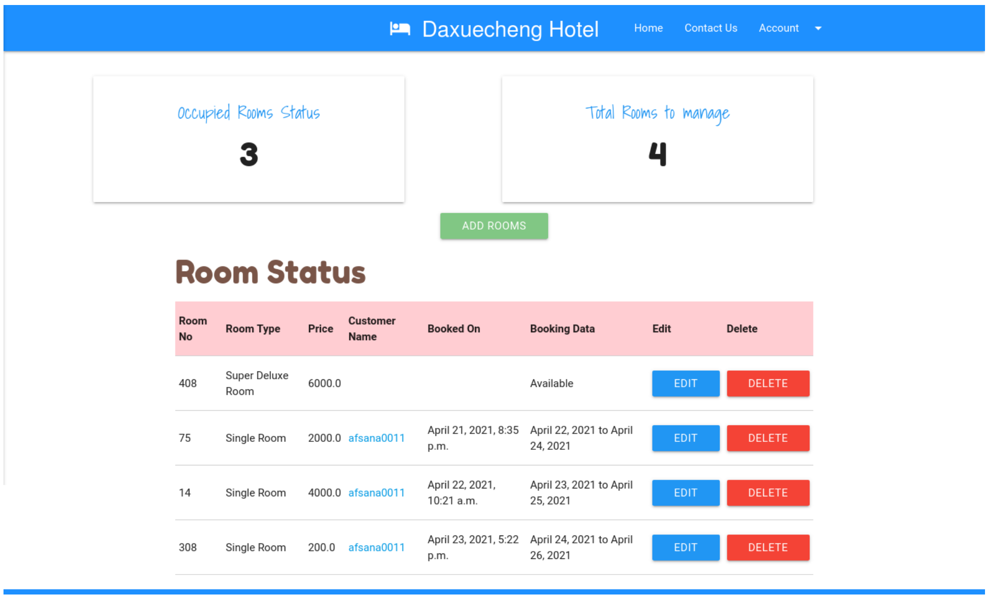
Rooms booked in past
3

Booking History

Booking ID	28	27	26
Booked On	April 23, 2021, 5:22 p.m.	April 22, 2021, 10:21 a.m.	April 21, 2021, 8:35 p.m.
Start Date	April 24, 2021	April 23, 2021	April 22, 2021
End Date	April 26, 2021	April 25, 2021	April 24, 2021
Billing	400.0	8000.0	4000.0
Room Manager	afsana1	afsana1	afsana1
Action	CANCEL BOOKING	CANCEL BOOKING	CANCEL BOOKING

Figure 20: Customer dashboard view

6.7 Manager Dashboard



Occupied Rooms Status
3

Total Rooms to manage
4

Room Status

Room No	Room Type	Price	Customer Name	Booked On	Booking Data	Edit	Delete
408	Super Deluxe Room	6000.0			Available	EDIT	DELETE
75	Single Room	2000.0	afsana0011	April 21, 2021, 8:35 p.m.	April 22, 2021 to April 24, 2021	EDIT	DELETE
14	Single Room	4000.0	afsana0011	April 22, 2021, 10:21 a.m.	April 23, 2021 to April 25, 2021	EDIT	DELETE
308	Single Room	200.0	afsana0011	April 23, 2021, 5:22 p.m.	April 24, 2021 to April 26, 2021	EDIT	DELETE

Figure 21: Manager dashboard view

6.8 Available Room View

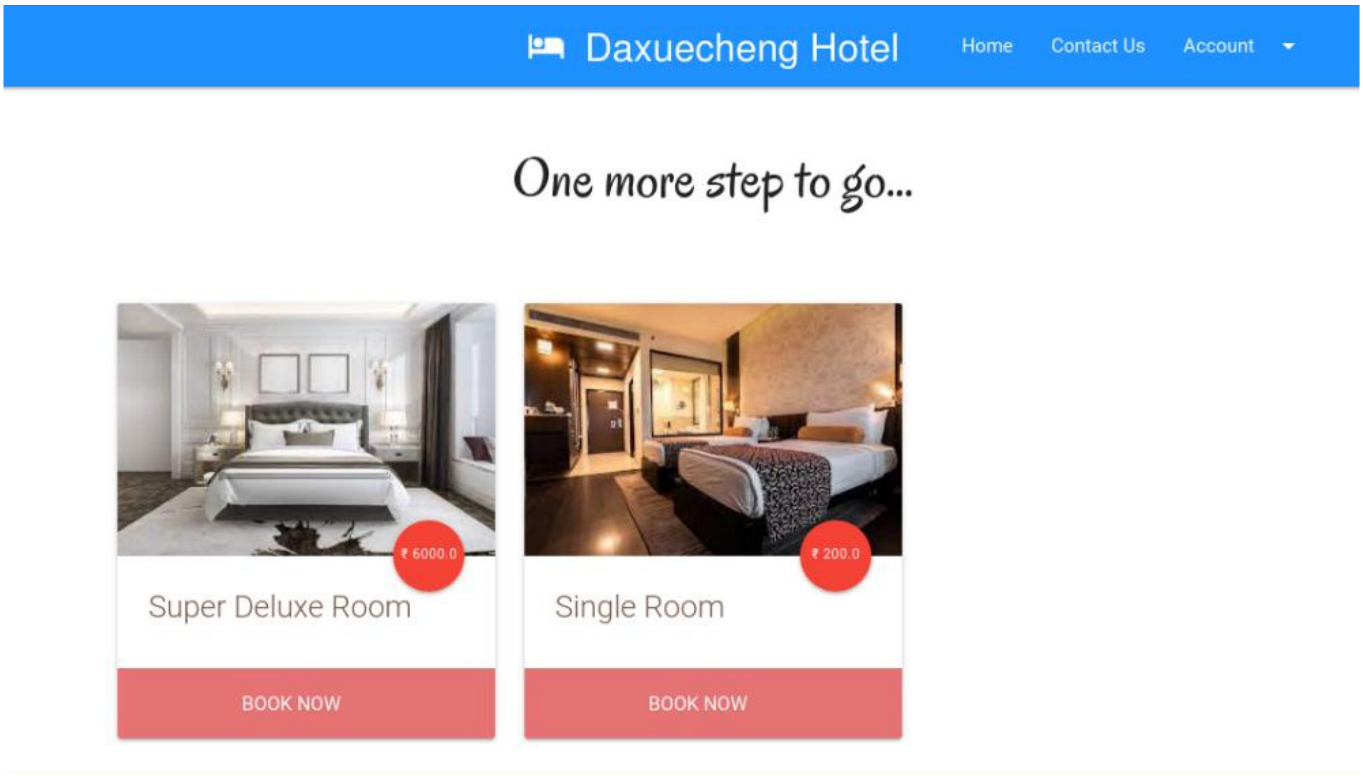
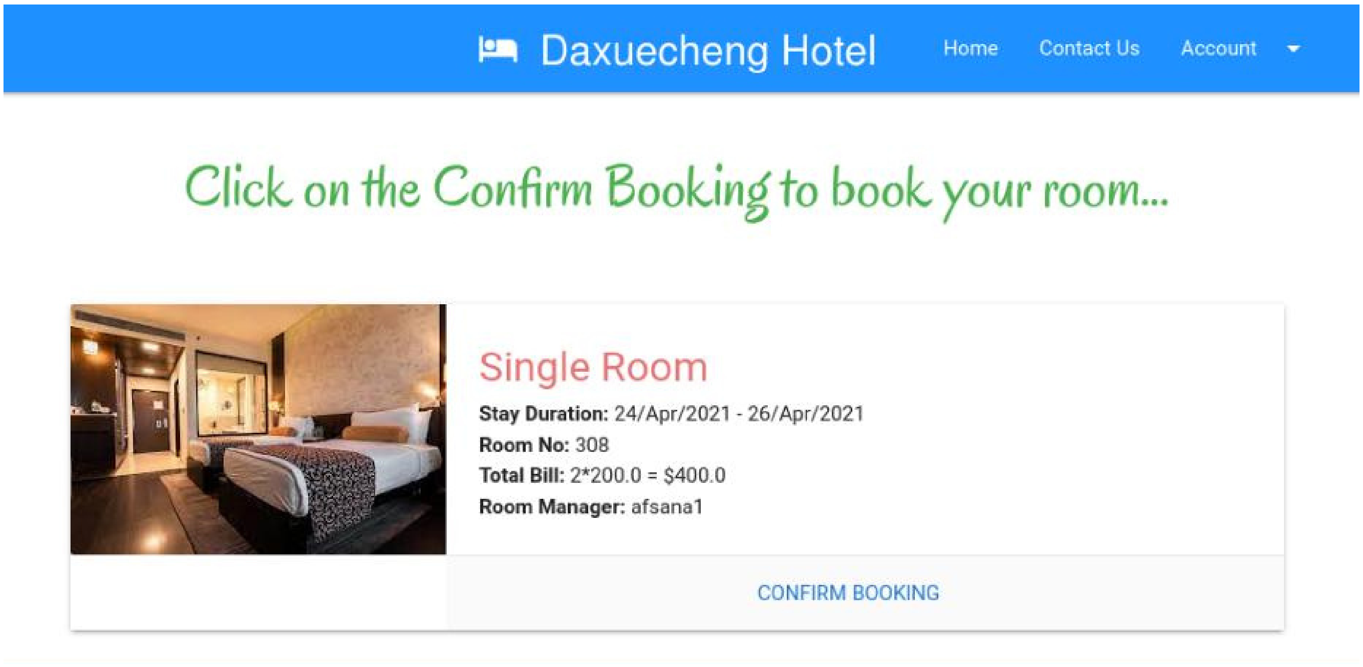


Figure 22: Available hotel room view

6.9 Booking Page View



Single Room

Stay Duration: 24/Apr/2021 - 26/Apr/2021

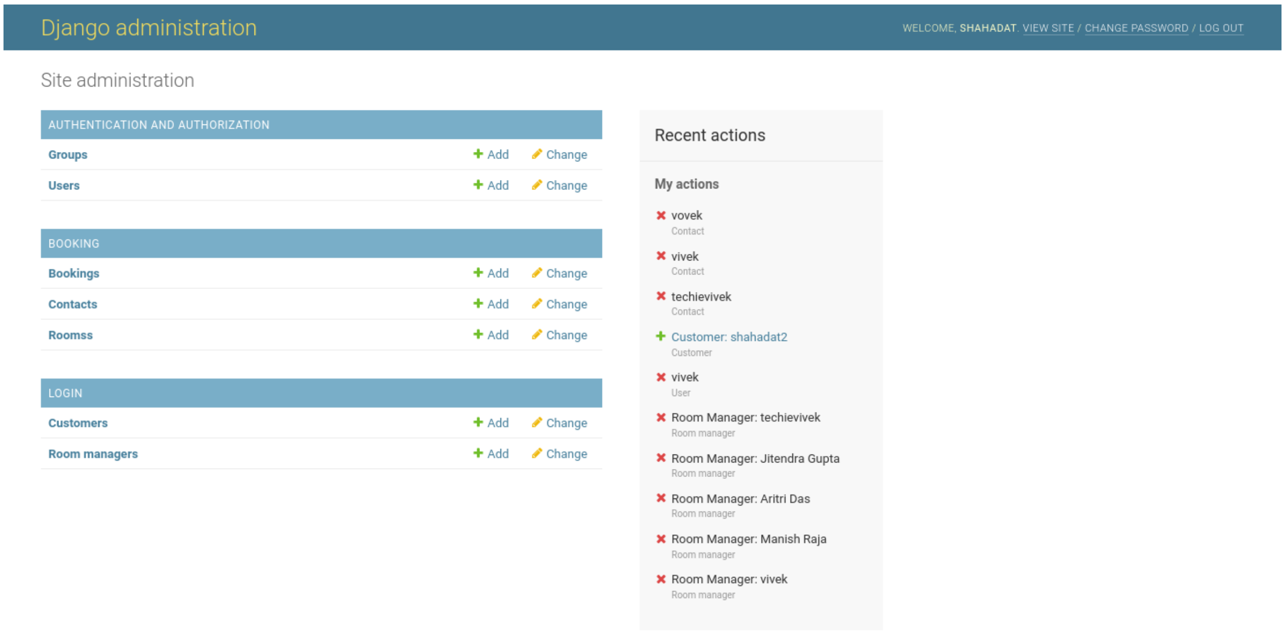
Room No: 308

Total Bill: 2*200.0 = $400.0

Room Manager: afsana1

Figure 23: Booking page view

6.10 Admin Panel



Django administration

WELCOME, SHAHADAT. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

BOOKING		
Bookings	+ Add	Change
Contacts	+ Add	Change
Roomss	+ Add	Change

LOGIN		
Customers	+ Add	Change
Room managers	+ Add	Change

Recent actions

My actions

- ✗ vovek
  Contact
- ✗ vivek
  Contact
- ✗ techievivek
  Contact
- + Customer: shahadat2
  Customer
- ✗ vivek
  User
- ✗ Room Manager: techievivek
  Room manager
- ✗ Room Manager: Jitendra Gupta
  Room manager
- ✗ Room Manager: Aritri Das
  Room manager
- ✗ Room Manager: Manish Raja
  Room manager
- ✗ Room Manager: vivek
  Room manager

Figure 24: Admin view

CHAPTER 7

CHAPTER 7
SYSTEM TESTING

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is quality assurance.

Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies a test plan is carried out on each module. The various tests performed in “Network Backup System” are unit testing, integration testing, and user acceptance testing.

7.1 Unit Testing

The software units in a system are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches the type and size supported by java. The various controls are tested to ensure that each performs its action as required.

7.2 Integration Testing

Data can be lost across any interface, one module can have an adverse effect on another, sub-functions when combined, may not produce the desired major functions. Integration testing is systematic testing to discover errors associated with the interface. The objective is to take unit-tested modules and build a program structure. All the modules are combined and tested as a whole. Here the Server module and Client module options are integrated and tested. This testing provides the assurance that the application is a well-integrated functional unit with a smooth transition of data.

7.3 User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the system users at the time of developing and making changes whenever required.

7.4 Writing tests

Django’s unit tests use a Python standard library module: unittest. This module defines tests using a class-based approach.
Here is some test example which subclasses from django.test.TestCase, which is a subclass of unittest.TestCase that runs each test inside a transaction to provide isolation:

7.4.1 CustomerTestCases

We put some test data for Customer username, email, pin_code and phone_number. There are three functions, setUp(), test_customer_username() and test_customer_email(), The setUP() creates two customer objects initially. Next the other two functions works for the testing of the customer username and email.

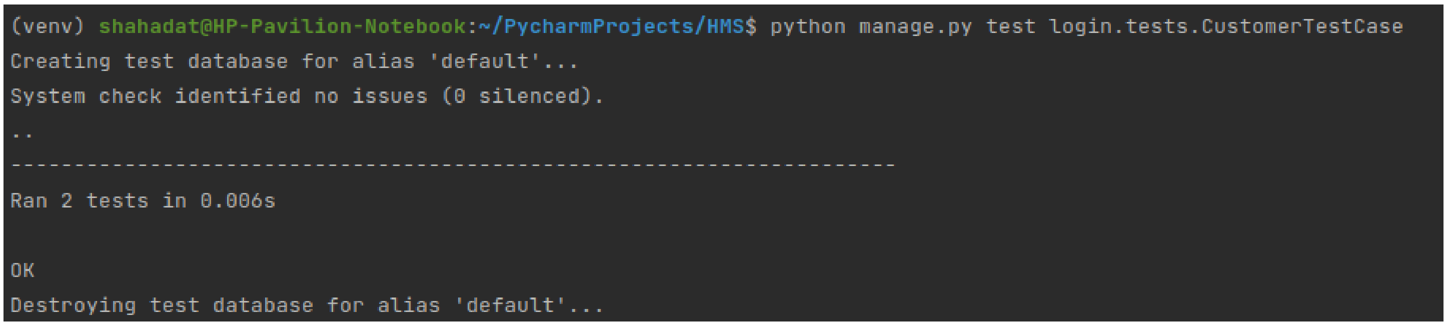
Source Code: TestCase

```
from django.test import TestCase
from .models import Customer
from .models import RoomManager

class CustomerTestCase(TestCase):
    def setUp(self):
        Customer.objects.create(username='vivek',
email='vray6640@gmail.com', pin_code=799046,
phone_no='8794173921')
        Customer.objects.create(username='vikas', email='vikas@gmail.com',
pin_code=799046, phone_no='9090909090')

    def test_customer_username(self):
        # I am just trying to take different case to prove the point
        customer1 = Customer.objects.get(phone_no='8794173921')
        customer2 = Customer.objects.get(username='vikas')
        self.assertEqual(customer1.username, 'vivek')
        self.assertEqual(customer2.username, 'vikas')
```

```
def test_customer_email(self):
        customer1 = Customer.objects.get(email='vray6640@gmail.com')
        customer2 = Customer.objects.get(username='vikas')
        self.assertEqual(customer1.email, 'vray6640@gmail.com')
        self.assertEqual(customer2.email, 'vikas@gmail.com')
```

Running tests

Once we've written tests, it's time to run them using the test command of our project's manage.py utility:



```
(venv) shahadat@HP-Pavilion-Notebook:~/PycharmProjects/HMS$ python manage.py test login.tests.CustomerTestCase
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------------------------------------
Ran 2 tests in 0.006s

OK
Destroying test database for alias 'default'...
```

Figure 25: customer test case output

7.4.2 Room ManagerTestCases

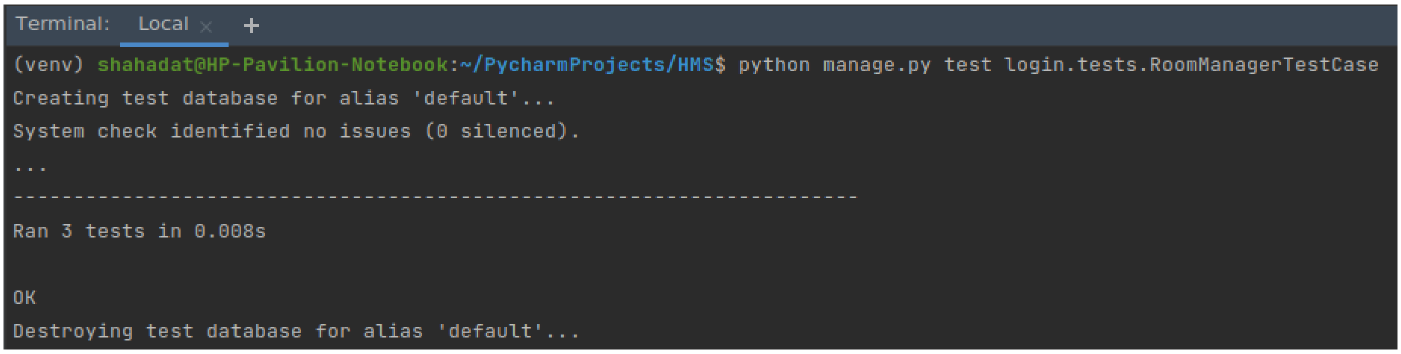
We put some test data for Customer username, email, and phone_number. There are four functions, setUp(), test_roomManager_username(), test_roomManager_email(), and test_roomManager_phone().
 The setUP() creates two RoomManager objects initially. Next, the other three functions work for the testing of the room manager username, email and phone number

Source Code: TestCase

```
from django.test import TestCase
from .models import Customer
from .models import RoomManager

class RoomManagerTestCase(TestCase):
   def setUp(self):
       RoomManager.objects.create(username='vivek',
email='vray6640@gmail.com', phone_no='8794173921')
       RoomManager.objects.create(username='vikas',
email='vikas@gmail.com', phone_no='9090909090')

   def test_roomManager_username(self):
       manager1 = RoomManager.objects.get(username='vivek')
       manager2 = RoomManager.objects.get(username='vikas')
       self.assertEqual(manager1.username, 'vivek')
       self.assertEqual(manager2.username, 'vikas')

   def test_roomManager_email(self):
       manager1 =
RoomManager.objects.get(email='vray6640@gmail.com')
       manager2 = RoomManager.objects.get(email='vikas@gmail.com')
       self.assertEqual(manager1.email, 'vray6640@gmail.com')
       self.assertEqual(manager2.email, 'vikas@gmail.com')

   def test_roomManager_phone(self):
       manager1 = RoomManager.objects.get(phone_no='8794173921')
       manager2 = RoomManager.objects.get(phone_no='9090909090')
       self.assertEqual(manager1.phone_no, '8794173921')
       self.assertEqual(manager2.phone_no, '9090909090')
```

Running tests

Once we’ve written tests, it's time to run them using the test command of our project’s manage.py utility:



```
(venv) shahadat@HP-Pavilion-Notebook:~/PycharmProjects/HMS$ python manage.py test login.tests.RoomManagerTestCase
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 0.008s

OK
Destroying test database for alias 'default'...
```

Figure 26: room manager test case output

7.4.3 Rooms TestCases

We put some test data for Rooms Manager object, room number, room type, price room available status, number of days advanced and start date.. There are four functions, setUp(), test_room_no(), test_room_type(), and test_price().

The setUP() creates two objects for Rooms initially. Next, the other three functions work for the testing cases.

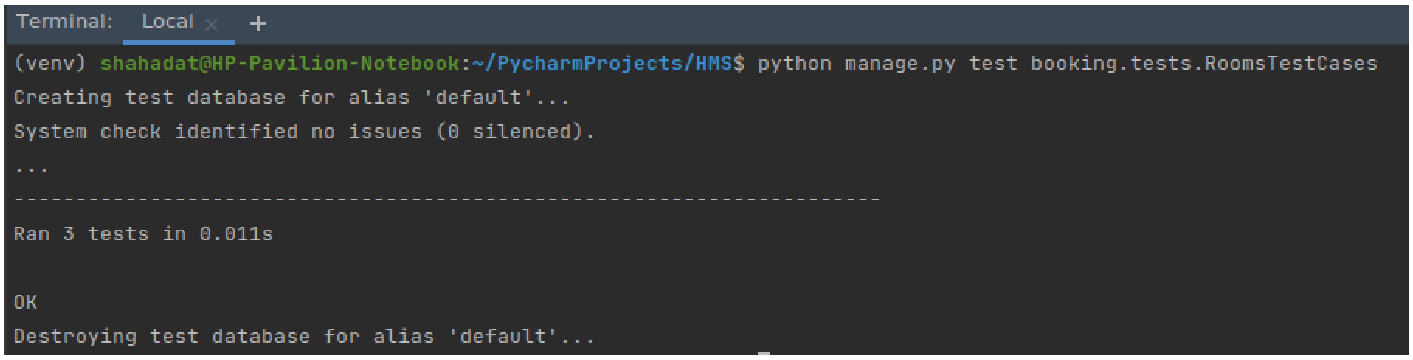
Source Code: TestCase

```
from django.test import TestCase, Client
from .models import Rooms
from .models import Booking
from login.models import Customer, RoomManager
from django.urls import reverse
```

```
class RoomsTestCases(TestCase):
    def setUp(self):
        RoomManager.objects.create(username='vivek')
        manager = RoomManager.objects.get(username='vivek')
        Rooms.objects.create(manager=manager, room_no=300,
room_type='Deluxe', price=3000, is_available=True,
                          no_of_days_advance=10, start_date='2020-03-20')
        Rooms.objects.create(manager=manager, room_no=301,
room_type='Super Deluxe', price=5000, is_available=True,
                          no_of_days_advance=15, start_date='2020-03-26')

    def test_room_no(self):
        room1 = Rooms.objects.get(room_no='300')
        room2 = Rooms.objects.get(room_no='301')
        self.assertEqual(room1.room_no, '300')
        self.assertEqual(room2.room_no, '301')

    def test_room_type(self):
        room1 = Rooms.objects.get(room_type='Deluxe')
        room2 = Rooms.objects.get(room_type='Super Deluxe')
        self.assertEqual(room1.room_type, 'Deluxe')
        self.assertEqual(room2.room_type, 'Super Deluxe')

    def test_price(self):
        room1 = Rooms.objects.get(price=3000)
        room2 = Rooms.objects.get(price=5000)
        self.assertEqual(room1.price, 3000)
        self.assertEqual(room2.price, 5000)
```

Running tests

Once we've written tests, it's time to run them using the test command of our project's manage.py utility:



```
Terminal:   Local ×  +
(venv) shahadat@HP-Pavilion-Notebook:~/PycharmProjects/HMS$ python manage.py test booking.tests.RoomsTestCases
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 0.011s

OK
Destroying test database for alias 'default'...
```

Figure 27: rooms test case output

7.4.4 Booking TestCases

We put some test data for Rooms booking. For setUP() we have created several objects for testing booking. The objects are RoomManager, Customer, Rooms, User, rooms, and Booking. There are four functions, setUp(), test_booking_username(),test_booking_amount(),andtest_booking_roomManager().The setUP() creates several objects for Room booking initially. Next, the other three functions work for the testing cases.

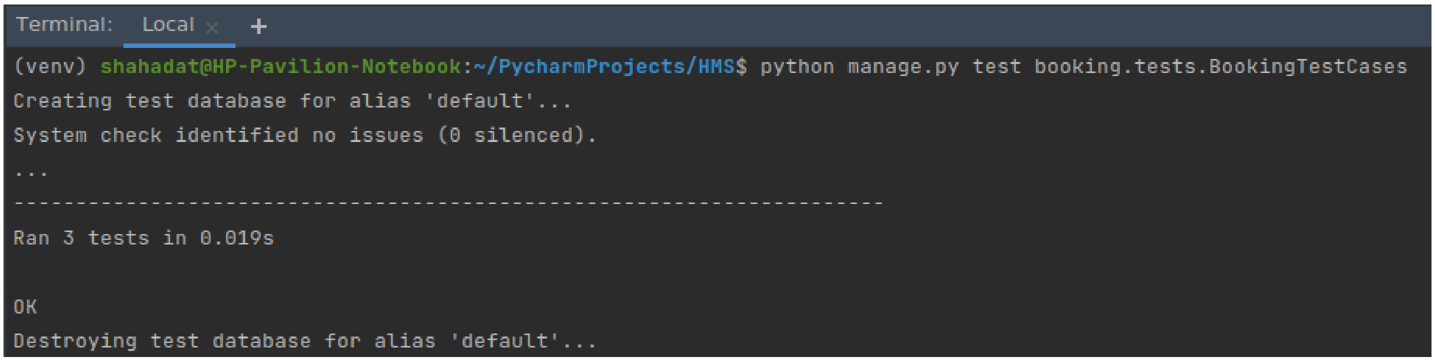
Source Code: TestCase

```
from django.test import TestCase, Client
from .models import Rooms
from .models import Booking
from login.models import Customer, RoomManager
from django.urls import reverse
```

```
class BookingTestCases(TestCase):
    def setUp(self):
        RoomManager.objects.create(username='rahul')
        manager = RoomManager.objects.get(username='rahul')
        Customer.objects.create(username='vivek', pin_code=799046)
        Customer.objects.create(username='vikas', pin_code=799046)
        Rooms.objects.create(room_no='300', no_of_days_advance=10,
start_date='2020-03-09', manager=manager)
        user = Customer.objects.get(username='vivek')
        user1 = Customer.objects.get(username='vikas')
        room = Rooms.objects.get(room_no='300')
        Booking.objects.create(user_id=user, room_no=room, amount=10000,
start_day='2020-03-10', end_day='2020-03-10')
        Booking.objects.create(user_id=user1, room_no=room, amount=5000,
start_day='2020-03-26', end_day='2020-03-28')

    def test_booking_username(self):
        user = Customer.objects.get(username='vivek')
        booking1 = Booking.objects.get(user_id=user)
        user1 = Customer.objects.get(username='vikas')
        booking2 = Booking.objects.get(user_id=user1)
        self.assertEqual(booking1.user_id.username, 'vivek')
        self.assertEqual(booking2.user_id.username, 'vikas')

    def test_booking_amount(self):
        booking1 = Booking.objects.get(amount=10000)
        booking2 = Booking.objects.get(amount=5000)
        self.assertEqual(booking1.amount, 10000)
        self.assertEqual(booking2.amount, 5000)

    def test_booking_roomManager(self):
        booking = Booking.objects.get(amount=5000)
        self.assertEqual(booking.room_no.manager.username, 'rahul')
```

Running tests

Once we’ve written tests, it's time to run them using the test command of our project’s manage.py utility:



```
(venv) shahadat@HP-Pavilion-Notebook:~/PycharmProjects/HMS$ python manage.py test booking.tests.BookingTestCases
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 0.019s

OK
Destroying test database for alias 'default'...
```

Figure 28: booking test case output

CHAPTER 8

CHAPTER 8
ANALYSIS

8.1 The Convenience Aspect

HMS:

As technology continues its rapid advance across all industries around the world, hospitality is really beginning to see the benefits. Hotel management software has come a long way in helping hoteliers improve the way their business operates, and there is little doubt as to its transformative impact. A modern property management system helps us to streamline administrative systems and processes,
as well as boost the company's overall operations. It's easy to see why the vast majority of hotel owners believe an excellent management system is essential for their business.

8.2 The Merchandise Aspect

HMS:

We can usually find a comprehensive view of every product on offer, and stores have limited shelf space. It can also be easy to find clarity on each item we like. And we can read online reviews from others. There are countless advantages to a reliable hotel management software system. Whether it’s time-saving on manual tasks or increasing direct bookings, every element of a hospitality system should be working towards the end goals of improving efficiency and enhancing the guest experience.

A more streamlined check-in and check-out experience will boost your guest's happiness. And that’s only the tip of the iceberg – anything from improved communication and additional services will also heighten guest loyalty. Choosing the best property management software will likely mean an increased level of retention in both guests and staff.

In-Store:
We physically see that we find it, take it, and show it to you. Much depends on what we buy. Buying an iPod online is easy because we know we get it. But choosing fabrics o shoes that way is a problem because we can't try them on first.

8.3 The Cost Aspect

HMS:
Because they do not have to hire sales staff or pay rent, some online deals can be found much cheaper at the moment in the mall. And the product range is also wide

In-Store:
Unless there is a sale, we can end up paying more if we buy it at the store. But if there is a dispute, we can at least argue with the manager about something we can do on our computer, unless we buy something at Hotel According to this, the demand for online shopping is increasing. Compared to in-store purchases due to the proliferation of mobile and laptop.

8.4 Economic Feasibility

This study was conducted to look at the economic impact it would have on the organization's future. The amount of funds that a company can contribute to research and program development is limited. Expenses must be allowed. Therefore, the system has been improved and within budget and this has been achieved because most of the technology used is freely available. Only personalized products should be purchased.

8.5 Technical Feasibility

This study was conducted to look at the feasibility of the technology, which is the technical requirements of the system. Any plan made should not have a high demand for available resources. This will result in greater demands placed on the consumer. The upgraded system should have a modest requirement, such as minor or non-existent changes to use the system.

8.6 Operational Feasibility

The study feature is to assess the level of acceptance of the system by the user. This includes the process of training the user to use the system properly. The user should not feel threatened by the system but should accept it as a necessity. The level of user acceptance depends only on the methods used to educate the user about the program and get him or her acquainted with it. His level of self-confidence should be enhanced so that he can take constructive criticism, which is well received, as he is the one who makes full use of the system.

CHAPTER 9

CHAPTER 9
FUTURE PLAN AND CONCLUSION

9.1 Future Plan

- Want to expand the facilities of this project.
- Want to add a print option.
- Want to improve the graphical design.
- Want to publish this website online.
- Want to remove all restrictions.

9.2 Conclusion

After learning about the online hotel booking system, we can see a big change in people's behavior with many habits like their attitude and buying pattern. In the past, people used to buy with their hands but now as time has changed people are busy and because of this technology has brought a new revolution in online shopping.
When we first did the survey, we noticed that young person like the 15-30 use of online shopping options because of time and energy saving. But the middle class is not very selective because it has the wrong idea that by seeing the product one can get good quality goods. And some people don't choose to use plastic money, credit cards.
However, online shopping has a bright future but to be successful, it is necessary to spread awareness about your profits.

The conclusion of this project is A Hotel management system is a computerized management system. This system keeps the records of hardware assets besides the software of this organization. The proposed system will keep a track of Workers, Residents, Accounts, and the generation of reports regarding the present status. This project has GUI-based software that will help in storing, updating, and retrieving the information through various user-friendly menu-driven modules. The project “Hotel Management System” is aimed to develop to maintain the day-to-day state of admission/Vacation of Residents, List of Workers, payment details, etc. The main objective of this project is to provide a solution for hotels to manage most there work using a computerized process. This software application will help the admin to handle customers' information, room allocation details, payment details, billing information. etc. Detailed explanations about modules and design are provided in the project documentation. The existing system is a manually maintained system. All the Hotel records are to be maintained for the details of each customer, Fee details, Room Allocation, Attendance, etc. All these details are entered and retrieved manually, because of this there are many disadvantages like Time Consuming, updating process, inaccuracy of data. For avoiding this we introduced or proposed a new system in the proposed system the computerized version of the existing system. provides easy and quick access over the data.

CHAPTER 10

CHAPTER 10

REFERENCE:

[1] djangoproject, [Online]. Available: Django documentation | Django documentation | Django (djangoproject.com)

[2] Stackoverflow, [Online]. Available: Stack Overflow - Where Developers Learn, Share, & Build Careers

[3] https://www.udemy.com/topic/django/

[4] https://www.w3schools.com/python/

[5] https://www.coursera.org/specializations/django

[6] https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

[7] Jackson W. Android Content Providers: Access to Datastores. Learn Android App Development. Apress, 2013: 451-486

[8] Venezia, Paul. How to get started with MySQL[J]. ProQues.2013,(5):1-2

[9] AN Li-li ,XUE Qiu, Low-carbon Economy and Eco-design of Commodity Packing[J]. Computer Aided Drafting,Design and Manufacturing.2011(01):87-89

[10] Osman,Mohd Azam,Zakaria, Nasriah,Bo,Tan.Adoption of e-commerce online shopping in Malaysia. IEEE international conference on E-BusinessEngineering2010.2010（02）:47-485.

[11] Roya Gholami, Augustine Ogun, Elizabeth Koh, John Lim.Factors Affecting e-Payment Adoption in Nigeria[J].Journal of Electronic Commerce in Organizations (JECO) .2010

ACKNOWLEDGMENT

I take this occasion to thank God, almighty for blessing me with his grace and taking my endeavor to a successful culmination. I extend our sincere and heartfelt thanks to my esteemed guide, Ying Wu, to provide me with the right guidance and advice at the crucial junctures and for showing me the right way. I extend my sincere thanks to our respected head of the division Xiang Yi, for allowing me to use the facilities available. I would like to thank the other faculty members also, on this occasion. Last but not the least, I would like to thank friends for the support and encouragement they have given us during the course of my work.

Meem Afsana Akter
Student Number: 2017490104