Enhance to develop the application for the Hotel Booking Services

A Mini-project Report submitted
in partial fulfilment for the award of the Degree of

Bachelor of Technology in Computer Science and Engineering
by

NADIPI BAYYA REDDY	**(U21CN024)**
NARAVULA MEGHENDRA	**(U21CN038)**
NAKKA YASWANTH	**(U21CN032)**
MOTHUKURI VENKATA SAI	**(U21CN003)**

Under the guidance of

MS.R.SHAMLI Assistant Professor, Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

BHARATH INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Deemed to be University Estd u/s 3 of UGC Act, 1956)

CHENNAI 600 073, TAMILNADU, INDIA

November/ December, 2024

Batch No. A17

Bharath

INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)

BHARATH INSTITUTE OF SCIENCE AND TECHNOLOGY
No.173, Agharam Road, Selaiyur, Chennai - 600 073.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BONAFIDE CERTIFICATE

This is to Certify that this Mini-Project Report Titled "ENHANCE TO DEVELOP THE APPLICATIONN FOR THE HOTEL BOOKING SEVICES" is the Bonafide Work of NADIPI BAYYA REDDY(U21CN024), NARAVULA MEGHENDRA(U21CN038), NAKKA YASWNATH (U21CN032), MOTHUKURI VANKATA SAI (U21CN003) of Final Year B.Tech. (CSE) who carried out the mini project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on basis of which a degree or award conferred on an earlier occasion by any other candidate.

PROJECT GUIDE	**HEAD OF THE DEPARTMENT**
MS.R.Shamli	**Dr. S. Maruthuperumal**
Assistant Professor	**Professor & Head**
Department of CSE	**Department of CSE**
BIHER	**BIHER**

Submitted for Semester Mini-Project viva-voce examination held on _________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We declare that this Mini-project report titled "**Enhance to develop the application for the Hotel Booking Services"** submitted in partial fulfillment of the degree of **B. Tech in (Computer Science and Engineering)** is a record of original work carried out by us under the supervision of **MS.R.Shamli,** and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Nadipi Bayya Reddy

U21CN024

Naravula Meghendra

U21CN038

Nakka Yaswanth

U21CN032

Mothukuri Venkata Sai

U21CN003

Chennai

Date:

ACKNOWLEDMENTS

We express our heartfelt gratitude to our esteemed Chairman, **Dr. S. Jagathrakshakan, M.P.,** for his unwavering support and continuous encouragement in all our academic endeavors.

We express our deepest gratitude to our beloved President **Dr. J. Sundeep Aanand** President, and Managing Director **Dr. E. Swetha Sundeep Aanand Managing** Director for providing us the necessary facilities to complete our project.

We take great pleasure in expressing sincere thanks to **Dr**. **K. Vijaya Baskar Raju** ProChancellor, **Dr. M. Sundararajan** Vice Chancellor (i/c), **Dr. S. Bhuminathan** Registrar and **Dr. R. Hariprakash** Additional Registrar**, Dr. M. Sundararaj** Dean Academics for moldings our thoughts to complete our project.

We thank our **Dr. S. Neduncheliyan** Dean, School of Computing for his encouragement and the valuable guidance.

We record indebtedness to our Head, **Dr. S. Maruthuperumal,** Department of Computer Science and Engineering for his immense care and encouragement towards us throughout the course of this project.

We also take this opportunity to express a deep sense of gratitude to our Supervisor **MS.R.Shamli** and our Project Co-Ordinator **Dr. B. Selvapriya** for their cordial support, valuable information, and guidance, they helped us in completing this project through various stages.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

NADIPI BAYYA REDDY (U21CN024)

NARAVULA MEGHENDRA (U21CN038)

NAKKA YASWANTH (U21CN032)

MOTHUKURI VENKATA SAI(U21CN003)

ABSTRACT

The provided Flask-based web application is a comprehensive hotel management system integrating multiple functionalities to streamline operations for both administrators and users. This application leverages MySQL for database management and focuses on modules like user authentication, room management, restaurant menu handling, booking, order processing, and customer reviews. Administrators can log in through a secure portal to manage rooms, edit or delete room details, oversee restaurant menus, and handle orders. The room management system supports CRUD operations, ensuring seamless addition, updating, and availability tracking of rooms. The platform supports features such as multi-room booking, real-time availability checks, and payment integration, providing users with a seamless booking experience. Admin functionalities include managing hotel listings, monitoring bookings, updating room availability, and generating reports for business analytics. The application also incorporates a secure authentication system, ensuring user data privacy through encrypted passwords and user session management.

Users can register with personal details and security credentials, log in to their accounts, and access services such as room booking and ordering from the restaurant menu. The booking process dynamically determines room type and rate based on the room number, ensuring precise customer billing. A restaurant management feature enables users to browse an available menu and place orders, with automatic calculation of total prices. Additionally, users can share their feedback via a review module, contributing ratings and comments that administrators can use to enhance server.

Keywords: Flask, MySQL, Hotel Management System, Room Management, Restaurant Menu Management, User Authentication, CRUD Operations, Session Management, Room Booking Order Processing, Customer Reviews, Feedback Module, Billing System, Flash Messages, Data Validation, Admin Portal, User Registration, Secure Login, Dynamic Pricing Availability Tracking, Modular Design, Web Application Development

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ NOTATIONS/ NOMENCLATURE

ABBREVIATIONS

CRUD	Create, Read, Update, Delete
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
SQL	Structured Query Language
DBMS	Database Management System
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
API	Application Programming Interface
GUI	Graphical User Interface
ID	Identifier
JSON	JavaScript Object Notation
DB	Database
UI	User Interface
UX	User Experience
OTP	One-Time Password
MVC	Model View Controller
IP	Internet Protocol
SQLi	SQL Injection
T&C	Terms and Conditions
PWA	Progressive Web Application

CHAPTER 1

INTRODUCTION

This chapter introduces the hotel management system, explaining its purpose, objectives, and scope. It delves into the rationale behind the project, highlights the methodologies adopted, and provides a comprehensive overview of the system's architecture and features.

1.1 Background

- **Overview of Hotel Management Systems**: Discusses the role of management systems in modern hotels to enhance operational efficiency and customer satisfaction.
- **Need for Digital Transformation**: Explains how digital solutions are reshaping the hospitality industry, focusing on automation and customer-centric services.
- **Existing Systems and Gaps**: Highlights limitations in traditional or current systems, such as lack of integration or inefficiencies in data handling.

1.2 Objectives

- **Primary Objective:** To design and implement a user-friendly, secure, and efficient hotel management system.
- **Specific Goals**:
  - Provide robust room management for administrators.
  - Offer seamless booking and restaurant services for users.
  - Enable feedback and review management to improve service quality.
  - Ensure secure authentication and session handling.

1.3 Problem Statement

- **Challenges in Manual Processes**: Issues like human errors, delays, and poor data accessibility in traditional systems.
- **Complexity in Multi-functional Integration**: Difficulty in combining room booking, restaurant management, and customer feedback in a unified platform.

- **Security Concerns**: Risks of unauthorized access and data breaches in online systems.

1.4Scope of the Project

- **Target Audience**: Hotels ranging from small inns to large resorts.
- **Functional Coverage**:
    - Room booking and management.
    - Restaurant menu management and order processing.
    - Customer authentication and registration.
    - Review and feedback system.
- **Scalability**: The system is designed to accommodate future expansions, such as integrating payment gateways or adding loyalty programs.

1.5 Methodology

- **Framework:** Flask for backend development due to its simplicity and flexibility.
- **Database**: MySQL as the relational database for efficient data storage and retrieval.
- **Development Approach**:
    - Agile methodology for iterative development and feedback incorporation.
    - Modular design for ease of maintenance and feature enhancements.

1.6 System Architecture

- **Overview of Architecture:** Explains how different modules interact within the system.
    - **User Interface (UI):** HTML, CSS, and Flask templates for frontend design.
    - **Backend Logic:** Flask for routing and handling business logic.
    - **Database Layer:** MySQL for storing user, booking, room, and order data.
- **System Flow**: Detailed process flow from user authentication to booking completion and review submission.

1.7 Features of the System

- **For Administrators:**
    - Manage rooms and restaurant menus.
    - Update availability and prices dynamically.

    - Secure access through admin login.

- **For Users:**
    - Registration and secure login.
    - Book rooms, order food, and submit reviews.
    - Personalized dashboard for order and booking history.
- **General Features:**
    - Data validation and secure storage.
    - Real-time feedback using flash messages.
    - Mobile-friendly interface for better accessibility.

1.8 Significance of the Project

- **Improving Efficiency:** Streamlining hotel operations by automating repetitive tasks.
- **Enhancing User Experience:** Offering customers a seamless and intuitive platform for bookings and orders.
- **Ensuring Data Security:** Implementing secure authentication and session management.
- **Cost-effectiveness:** Reducing operational costs associated with manual systems.

CHAPTER 2

LITERATURE SURVEY

Title 1: HOTEL MANAGEMENT SYSTEM

Authors: Mounika Nandiraju , Salluri Rachana , Shaik Chandini , Sandhu Srilatha , G.Sabitha , Seema Nazneen (March - 2020)

In the literature review we consider and examine the work done by other scholars and researchers who have broached on this particular topic (Hotel Management System).

Technology has made a considerable impact on the Hospitality industry in recent years and will continue to do so with the increasing use of computer, controlled equipment and the growth of information technology in general" (Jones and Lockwood, 1989) Really in the last two decades, technology has become far more advanced and far more widely used throughout all types of industry. The tourism and hospitality industry is no exception. Indeed, many tourism and leisure establishments rely on technological systems for the vast majority of their operations.

They use a range of computer programs from everything to bookings, communications, security and payments. If a hospitality establishment does not use some sort of advanced technological system in its operations, it is deemed to be out of date and disorganized. Indeed, James Bardi begins to outline the importance of these programs by claiming that "a well-organized reservation system allows hotels to ensure a steady flow of guests into their properties". Furthermore, "Profitable business ventures rely on effective marketing, which includes reviewing people who require hotel products and services, determining their specific needs, developing products and services that meet those needs, and making a profit on the sale of those products and services" (Bardi, 2010).

Title 2: Decentralized hotel rooms booking system using Pragma solidity and blockchain technology

Authors: AYUSH ANAND, AYUSH BHAGAT (April - 2023)

Since there is a lot of ambiguity while booking a room at the last moment due to which we usually don't end up getting one. To avoid the occurrence of such a situation “Book The Room” app comes to your rescue. Through this app, we can book the conference room well in advance based on the availability of the date and time that we have opted for. If the time limit exceeds a time span of 2 hours, then a request will be sent to the admin for necessary approval. Further to the approval, the conference room can be used. The queued requests are served on a first come first serve basis by the admin.

Every academic institution has its system of managing the bookings of spaces. In an academic institution, rooms in different buildings may be handled by separate departments for booking purposes and most are carried out using the mail system. People using this system face numerous challenges like the absence of knowledge of available spaces, tediousness, improper management of bookings, clashes in bookings, communication failures, etc. These indicated a need for a well-defined, efficient, visualizable, and user-friendly space management system. BOOKiiIT, a Flutter app was proposed and designed using design thinking and PACT framework to make it user friendly and efficient in the different contexts of use. This app was implemented for the Indraprastha Institute of Information Technology (IIITD), but it can be easily replicated for other institutions.

Title 3: Comprehensive Review on Web-based Hotel Management System

Authors: Jeevanjot Singh, Amanjot Mavi, Kiranpreet Kaur (November - 2023)

A key technical development in the hospitality sector is the hotel management system. It improves and streamlines many facets of hotel operations. This system guarantees a flawless guest experience by effectively managing reservations, checkins, check-outs, and invoicing procedures. Additionally, it makes inventory management easier, enabling hotels to maximize room occupancy and pricing plans. Personalized services are made possible by the storage of guest profiles and preferences. Decision-makers are empowered with useful information from real-time reporting and analytics technologies to improve resource allocation and customer satisfaction. To fulfill the expanding requirements of today's tech-savvy travelers, hotels must, however, adapt to shifting technological trends, such as smartphone check-ins and contactless payments, in order to remain competitive. Internet marketing has been viewed as a very successful strategy to stand out as distinctive in the industry, especially given the current situation of the economy and the fierce competition in the hotel industry. Leaving aside the complexity in hotels, Srinivasa R. (2014) asserts that the majority of modern hotel management is done via an internet-based system . The internet, smartphones, and other modern technology comforts are available in the majority of the world's areas. Customers can order their preferred meals and beverages at any time, reserve a hotel room, and more using the internet just by doing a few easy online actions. Customers would enjoy this because it saves their important time. In 2021, Avneesh Pathak et.al gave a succinct description of a hotel management system that uses software engineering techniques for needs analysis, performance analysis, and design . The entire system is broken down into discrete modules that each introduce a function, a set of requirements, and the logical structure of the database used for data management.

Title 4: Web Based Hotel Management System

Authors: Dukare Siddhesh Sudam , Bhalerao Akanksha Santosh (November - 2022)

The literature review we consider and examine the work done by researchers who have broached on this particular topic (Hotel Management System). As mentioned above, the main purpose of hotel industry is to offer consumers' hospitality services. Technology has a considerable impact on Hospitality industry in previous years and will continue to do so with the increasing use of computer Technology, controlled equipment" (Jones and Lockwood, 1989,) Really in the last two to four year, technology has become far more advanced, easy to use and far more widely used throughout all types of industry . They mostly use a range of computer programs system from everything to bookings, communications, security and payments. the reason why hotels utilize technological systems in their operations is because it keeps them up to date in terms of where they are placed in the Technology market. Priority is maintaining their position of the brand and status as a luxury brand, rather than cutting high or low costs, which would be more of a priority for budget hotels which cater to a lower-high end market. It is clear that technology used in hospitality establishments and it is also used to make customers' lives more convenient and easy to understanding.

The hospitality comprises a wide range of fields within the hotel industry that includes accommodations, restaurants, event planning and other sectors with tourism. (Law, 2009) It seems that if a hospitality company can provide valuable web sites that attract their visitors, they will also get benefits ultimately .There are several definitions of a hotel.

Title 5: Research on Hotel Management System

Authors: W.P.S.W. Weerasinghe, K.D.M.I. Jayathilaka, W.V.C. Prasadi, M.D.K.M Goonetilleke, D. I. De Silva, Piyumika Samarasekara (October - 2022)

As mentioned above, the main purpose of hotel industry is to offer consumers' hospitality services. The hospitality comprises a wide range of fields within the hotel industry that includes accommodations, restaurants, event planning and other sectors with tourism. (Law, 2009) It seems that if a hospitality company can provide valuable web sites that attract their visitors, they will also get benefits ultimately . According to that, if we can promote hospitality better from our website, we can get more reviewers for our system. Therefore, providing better hospitality service is crucial fact in the hotel management system.

With the current state of the economy and the strong rivalry in the hotel industry, internet marketing has been seen as a very successful technique to stand out as distinctive in the business. According to Srinivasa R. (2014), keeping aside the complication in the hotels, today hotel management are basically performed as internetbased system .

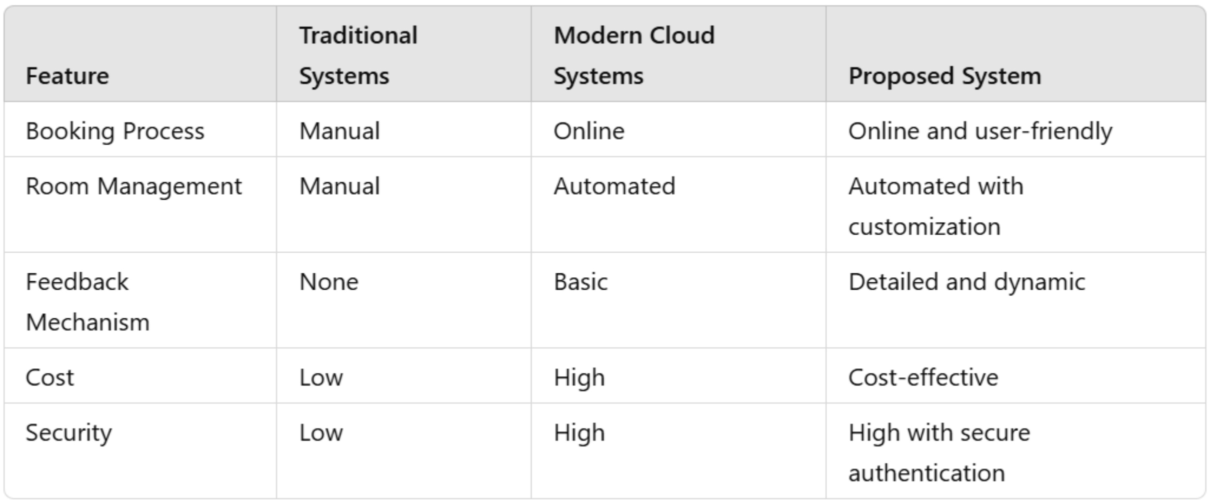
The majority of the world's regions in the modern day have technological amenities like the internet, smartphones, computers, laptops, etc. With the use of the internet, customers can book a hotel room, purchase their desired meals and beverages at any time etc. by following a few simple online steps. This may be appreciated by customers since it saves them valuable time. Our system also is an e-commerce web application that provide numerous facilities for the customer such as booking rooms, order foods and beverages from the user friendly and convenient way.

According to Kalaskar P. (2013), hotels upgrade as luxury accordance with the amenities provided by the hotels like 5 Star . All the criteria necessary to provide the greatest

amenities to the hotels are examined to arrive at these ratings. According to Popat K. (2013), hotel system requires courageous team which works in a manner that will make proper coordination with all the facilities or services that they are providing in the hotel . The system should have no issues with any of its management features. It involves the hotel administration, financial department, Room service, employee department and food and catering service etc. Our system also provides a userfriendly way to manage those administrations for the user. This system features number of managements that are integrated with one another, including user management, staff and supplier management, room and reservation management, and restaurant management. These enable the administrator to manage hotel resources with ease, which will improving the property's ratings.

2.1 Comparative Analysis

➢ Comparison of Existing Solutions:



Feature	Traditional Systems	Modern Cloud Systems	Proposed System
Booking Process	Manual	Online	Online and user-friendly
Room Management	Manual	Automated	Automated with customization
Feedback Mechanism	None	Basic	Detailed and dynamic
Cost	Low	High	Cost-effective
Security	Low	High	High with secure authentication

Table 2.1 Comparison of Existing Solutions

CHAPTER 3

DESIGN METHODOLOGY

This chapter outlines the design methodology used to develop the hotel management system. It discusses the approach, tools, and techniques employed in the design process, ensuring that the system meets the functional and non-functional requirements identified in previous chapters. The design focuses on scalability, efficiency, user experience, and security.

3.1 Overview of Design Methodology

- **Purpose of the Design Methodology:** The design methodology serves as a structured approach to creating a system that fulfills the project's objectives and solves identified problems.
- **Design Phases:** The design methodology follows a step-by-step approach, ensuring systematic development and integration of features. These phases include:
  - Requirement Gathering and Analysis
  - System Architecture Design
  - Database Design
  - User Interface Design
  - Security Design
  - Integration and Testing
  - Deployment and Maintenance

3.2 Requirement Gathering and Analysis

- **Functional Requirements:**
  - **Room Management:** Ability to add, update, and manage room availability, types, rates, and statuses.

- **Booking System:** A mechanism for customers to book rooms, including dynamic room rate calculation.
- **Restaurant Management:** Menu management, order tracking, and billing for restaurant services.
- **Customer Feedback:** A feedback mechanism for reviews and ratings, ensuring customer satisfaction.
- **User Authentication:** A login system for users and admins, including secure authentication and role-based access.

➢ **Non-Functional Requirements**:

- **Performance**: The system should handle high traffic, particularly during peak seasons.
- **Usability**: The system must be intuitive and user-friendly, requiring minimal training.
- **Scalability**: The design must accommodate future growth, such as adding new rooms or restaurants.
- **Security**: Data security is paramount, including secure login, payment processing, and protection of personal information.

3.3 System Architecture Design

➢ **Layered Architecture**:

The system is designed using a layered architecture to ensure modularity and separation of concerns:

- **Presentation Layer**: Handles user interfaces (UI) and interacts with users through HTML, CSS, and JavaScript.
- **Application Layer**: Manages business logic, including room booking, restaurant orders, and user authentication (implemented using Flask).

- **Data Layer**: Consists of the MySQL database, responsible for storing all application data, such as user details, room bookings, and restaurant orders.

➤ **Architecture Diagram**:

A flow diagram illustrating how different components of the system (frontend, backend, and database) interact with each other.

- **Frontend (UI)**: User-facing web interface built with HTML, CSS, and JavaScript.
- **Backend (Flask)**: Python-based backend that processes business logic, communicates with the database, and handles requests.
- **Database (MySQL)**: Centralized database to store all data, providing persistence for user and booking information.

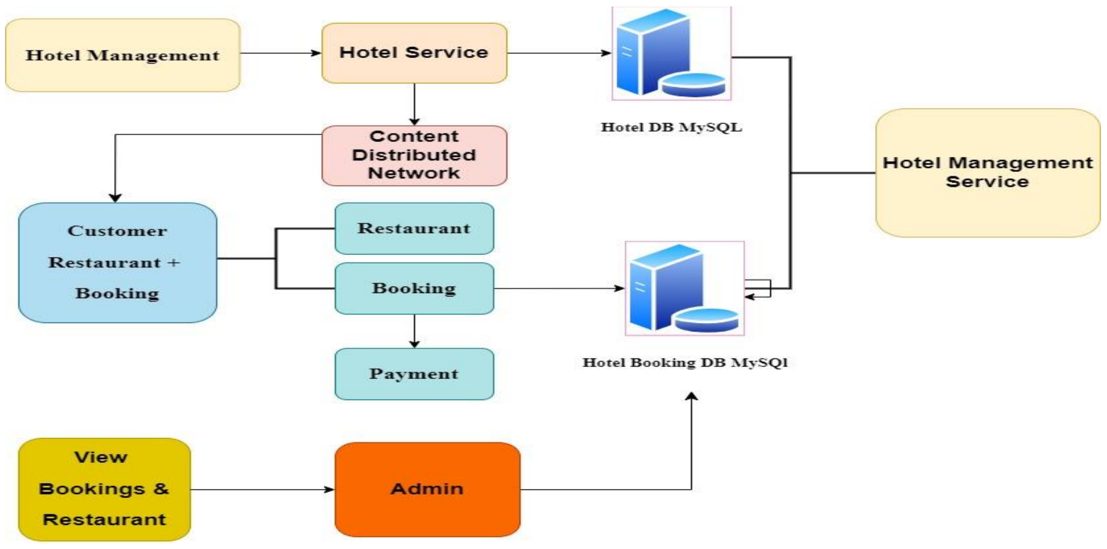


Fig: 3.3 System Architecture

3.3.1 Modules Explanation

Break down the various components represented in the diagram, such as:

➤ **Hotel Management**: Role and functionality.

➤ **Hotel Service**: Interaction with the backend database.

➤ **Content Distributed Network**: Purpose in data sharing and distribution

➤ **Customer, Booking, and Restaurant Modules**: How customers interact with these component.

➤ **Payment and Admin Modules**: Key responsibilities and flow of information.

- **Databases**: Separate databases for hotel services and bookings.

3.4 Database Design

Start with a brief overview of the database design and its role in the system. For example: "The database schema forms the backbone of the hotel management system, ensuring efficient data organization, storage, and retrieval. It defines the tables, attributes, and relationships critical to system functionality."

3.4.1Database Schema Diagram

- Provide a detailed explanation of each entity and its attributes:
  - **Hotel**: Stores information about hotels, including HotelID, HotelName, Location, and Rating.
  - **Room**: Contains details of rooms such as RoomID, RoomNumber, RoomType, Price, and a foreign key HotelID.
  - **Restaurant**: Represents restaurants in the hotel with RestaurantID, RestaurantName, CuisineType, and HotelID.
  - **Customer**: Holds customer data, including CustomerID, FirstName, LastName, Email, Phone, and Address.
  - **Booking**: Tracks booking records with BookingID, BookingDate, CheckInDate, CheckOutDate, Status, and foreign keys CustomerID and RoomID.

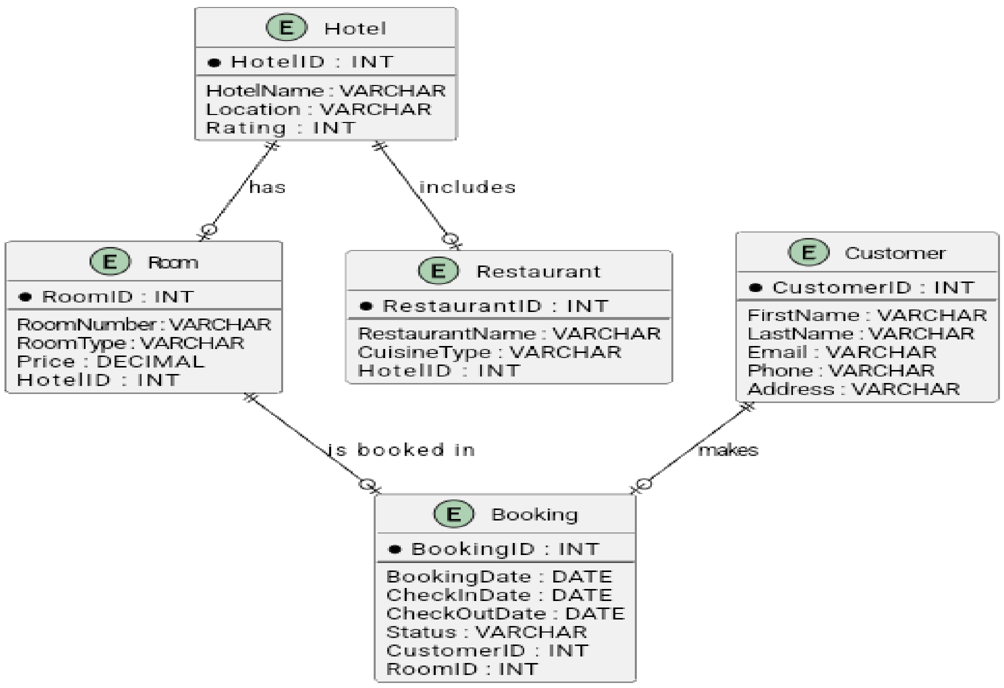


Fig: 3.4 Database Schema Diagram

3.5 Workflow Design

➢ Briefly describe the purpose of the activity diagram. For example:
"This diagram illustrates the step-by-step process of hotel room booking, starting from room selection to booking confirmation and optional restaurant reservations."

3.5.1 Activity Diagram Explanation

➢ Provide a detailed explanation of each step in the workflow:

- **Customer views available rooms:** The customer begins the process by browsing available rooms.
- **Room Selection:** The system checks if the room is available and guides the customer to provide necessary details if available.
- **Booking Confirmation**: The hotel management confirms the booking and processes it with admin intervention.

- **Optional Restaurant Reservation**: Customers can optionally reserve a restaurant table as part of their booking

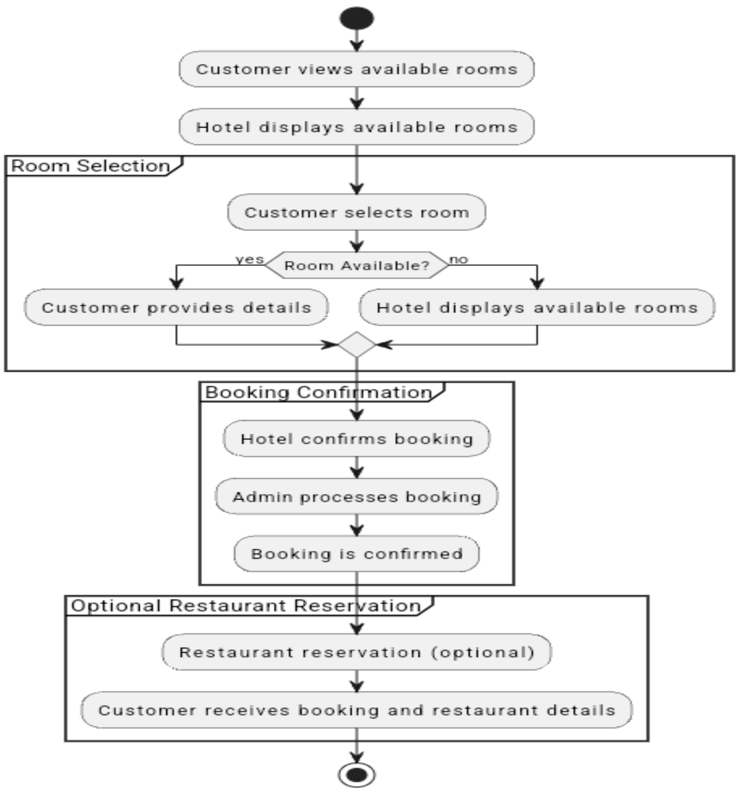

Customer views available rooms
Hotel displays available rooms
Room Selection
Customer selects room
yes
Room Available?
no
Customer provides details
Hotel displays available rooms
Booking Confirmation
Hotel confirms booking
Admin processes booking
Booking is confirmed
Optional Restaurant Reservation
Restaurant reservation (optional)
Customer receives booking and restaurant details

Fig: 3.5 Activity Diagram

3.6 Object-Oriented Design

"The class diagram represents the object-oriented design structure of the hotel management system. It outlines the key classes, attributes, methods, and their relationships to model the core functionality of the system."

3.5.2 Class Diagram Explanation

➢ Provide a detailed description of each class and its role in the system:

- **Hotel:** Represents the hotel entity with attributes like hotel_id, name, address, and rating. It includes methods for adding rooms and restaurants.
- **Room**: Models a room's details such as room_id, room_number, type, and price_per_night. The method reserveRoom() manages reservations.

- **Customer**: Captures customer information, including attributes like customer_id, name, email, and phone_number. Methods include placeBooking() and placeFoodOrder().
- **Restaurant**: Represents a restaurant, including attributes like restaurant_id, cuisine_type, and opening_hours. It includes a method to prepare food orders.
- **Booking**: Tracks booking details with attributes like booking_id, check_in_date, check_out_date, and total_amount. Methods manage bookings.
- **Food_Order**: Manages food order details, including order_id, order_date, and total_amount.

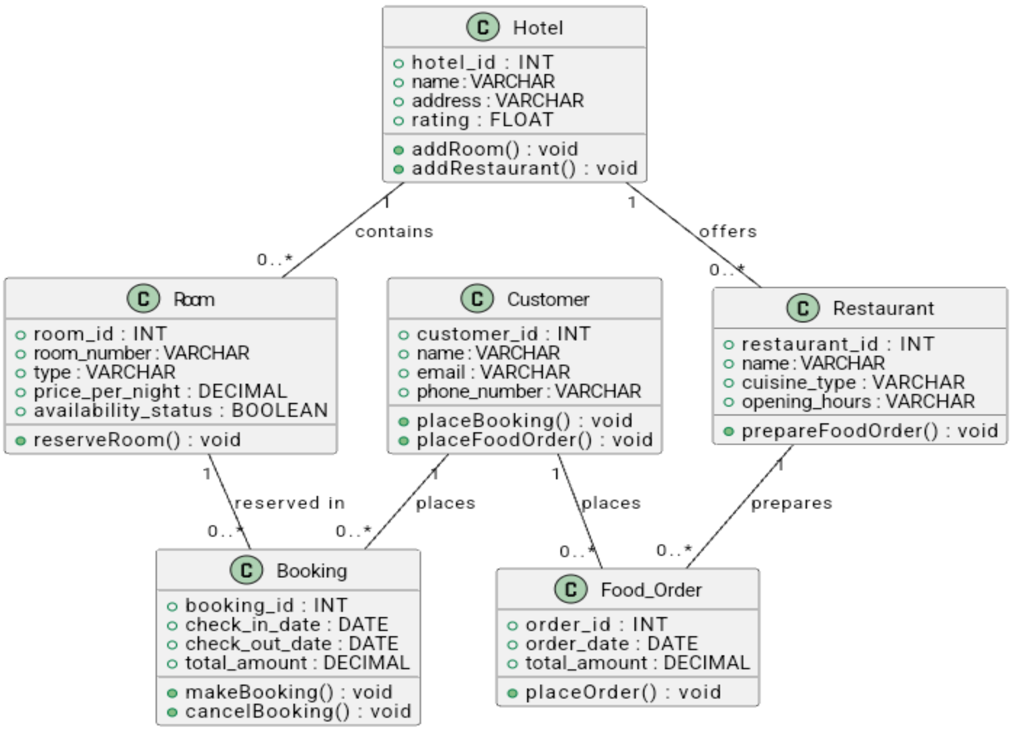


Fig: 3.6.1 Class Diagram

➢ **User Authentication and Authorization:**

- **Session Management:** Flask session management is used to maintain secure login states for both users and admins.
- **Password Hashing**: Passwords are hashed using bcrypt to enhance security and prevent unauthorized access.

- **Role-Based Access Control**: Different roles (admin, user) have different levels of access, with admins having more privileges to manage rooms and restaurant items.

➢ **Data Protection**:

- **Data Encryption**: Sensitive data, such as user passwords and payment details, is encrypted both at rest and in transit.
- **Secure Communication**: All data exchanged between the frontend and backend is encrypted using HTTPS to prevent man-in-the-middle attacks.

➢ **Database Security**:

The MySQL database uses strong authentication mechanisms to prevent unauthorized access and SQL injection attacks. Prepared statements are used in all queries to ensure the security of user inputs. Batch Size: Tuned to optimize computational efficiency without sacrificing accuracy.

CHAPTER 4

IMPLEMENTTAION

This chapter details the process of implementing the hotel management system. It provides insights into how the design concepts outlined earlier were transformed into a functional system. The chapter discusses the specific technologies used, the development process, and how various components of the system were implemented to meet both the functional and non-functional requirements. Hardware: NVIDIA GPU (e.g., RTX 3090) for accelerated training and inference.

4.1 Overview of Implementation

- **Purpose of Implementation:** The purpose of this phase is to convert the design specifications into an actual working system. This involves coding, integrating different system components, and ensuring that they function correctly.
- **Technologies Used:**
  - **Backend Development:** Flask (Python), which is used to build the server-side logic and manage HTTP requests.
  - **Frontend Development:** HTML, CSS, JavaScript, with the use of frameworks like Bootstrap for responsive design.
  - **Database:** MySQL is used to store all relevant data, including user details, room bookings, and restaurant orders.
  - **Security:** Bcrypt for password hashing, Flask-Login for user authentication, and HTTPS for secure communication.
  - **Version Control**: Git is used to manage the source code, track changes, and collaborate with other developers.

4.2 Backend Implementation (Flask)

➢ **Flask Setup:**

Flask was set up to handle HTTP requests, route them to appropriate view functions, and return responses to users. The basic structure of the Flask app consists of:

- **App Initialization**: Setting up the Flask application, configuring necessary settings (e.g., debug mode, session settings).
- **Route Definitions**: Each page in the system is mapped to a route in Flask. Routes such as /login, /book_room, /view_rooms, and /admin_dashboard are defined and connected to corresponding view functions.
- **Request Handlers**: Each route handler processes user inputs, communicates with the database, and returns the relevant data or page.
- **Session Management**: Flask's built-in session management is used to store information about the logged-in user, such as their ID and authentication status.
- **Error Handling**: Custom error pages were created for different HTTP errors, such as 404 (Page Not Found) or 500 (Internal Server Error).

➢ **Coding:**

```
import datetime
import random
import stripe
from decimal import Decimal
from io import BytesIO
from reportlab.pdfgen import canvas
from flask import send_file
from flask import Flask, render_template, request, redirect, send_file, url_for, session,
flash
```

```
import mysql.connector  # Import MySQL connector
app = Flask(__name__)
app.secret_key = 'your_secret_key'
stripe.api_key = 'your_stripe_secret_key'
```

- **Room Management**:
  - Implemented functionalities for adding, updating, and deleting room records.
  - Integrated features for checking room availability, calculating dynamic pricing, and displaying room details to users.
  - Used SQL queries to fetch and update data in the database (e.g., room availability, booking status).
- **Coding:**

```
@app.route('/admin/rooms', methods=['GET', 'POST'])
def manage_rooms():
    # Connect to MySQL database
    connection = mysql.connector.connect(**db_config)
    cursor = connection.cursor(dictionary=True)
    if request.method == 'POST':
        # Add new room
        room_number = request.form['room_number']
        room_type = request.form['room_type']
        room_rate = request.form['room_rate']
        is_available = request.form['is_available'] == 'True'
        clean_status = request.form.get('clean_status', 'Clean')
        insert_room_query = '''INSERT INTO rooms (room_number, room_type,
room_rate, is_available, clean_status)
                                VALUES (%s, %s, %s, %s, %s)'''
        cursor.execute(insert_room_query, (room_number, room_type, room_rate,
 is_available, clean_status))
        connection.commit()
        flash("Room added successfully", "success")
    # Fetch all rooms to display in the admin panel
    cursor.execute("SELECT * FROM rooms")
    rooms = cursor.fetchall()
    cursor.close()
    connection.close()
    return render_template('admin_rooms.html', rooms=rooms)
@app.route('/admin/rooms/edit/<int:room_id>', methods=['GET', 'POST'])
def edit_room(room_id):
    # Connect to MySQL database
    connection = mysql.connector.connect(**db_config)
    cursor = connection.cursor(dictionary=True)
    if request.method == 'POST':
        # Update room information
        room_number = request.form['room_number']
        room_type = request.form['room_type']
        room_rate = request.form['room_rate']
        is_available = request.form['is_available'] == 'True'
        clean_status = request.form['clean_status']
        update_room_query = '''UPDATE rooms
                        SET room_number = %s, room_type = %s, room_rate = %s,
is_available = %s, clean_status = %s
                        WHERE room_id = %s'''
        cursor.execute(update_room_query, (room_number, room_type, room_rate,
is_available, clean_status, room_id))
```

```
            connection.commit()
            flash("Room updated successfully", "success")
            return redirect(url_for('manage_rooms'))
        # Fetch room information for editing
        cursor.execute("SELECT * FROM rooms WHERE room_id = %s", (room_id,))
        room = cursor.fetchone()
        cursor.close()
        connection.close()
        return render_template('edit_room.html', room=room)
```

- **Booking System**:
  - Implemented the logic for booking rooms, including validation checks for available dates and room types.
  - Used session data to track customer bookings during the process.
  - Created booking confirmation emails to notify customers of successful bookings.
- **Coding:**

```
def fetch_available_rooms():
    connection = mysql.connector.connect(**db_config)
    cursor = connection.cursor()
    # Query to get available rooms
    cursor.execute("SELECT room_number, room_type FROM rooms WHERE is_available =
TRUE AND clean_status = 'Clean'")
    available_rooms = cursor.fetchall()
    # Ensure we have results
    if not available_rooms:
        return []
    # Map room numbers based on their prefixes
    categorized_rooms = []
    for room in available_rooms:
        room_number = str(room[0])  # Convert to string for prefix checking
        if room_number.startswith('1'):
            room_type = 'Single'
        elif room_number.startswith('2'):
            room_type = 'Double'
        elif room_number.startswith('3'):
            room_type = 'Suite'
        else:
            room_type = 'Unknown'  # Default for unexpected prefixes
        categorized_rooms.append({'room_number': room_number, 'room_type': room_type}
    cursor.close()
    connection.close()
    return categorized_rooms
@app.route('/get_rooms_by_type', methods=['GET'])
def get_rooms_by_type():
    room_type = request.args.get('room_type')
    # Connect to the MySQL database
    connection = mysql.connector.connect(**db_config)
    cursor = connection.cursor(dictionary=True)
    query = "SELECT room_number FROM rooms WHERE room_type = %s AND is_available
= TRUE AND clean_status = 'Clean'"
    cursor.execute(query, (room_type,))
    rooms = cursor.fetchall()
    cursor.close()
    connection.close()
    # Return room numbers as a JSON response
    return {'rooms': [room['room_number'] for room in rooms]}
@app.route('/book_room', methods=['GET', 'POST'])
def book_room():
    room_number = None
    if request.method == 'POST':
        # Generate a random customer ID if not provided
        customer_id = random.randint(1000, 9999)
        customer_id = request.form['customer_id']
        customer_name = request.form['customer_name']
        id_proof_type = request.form['id_proof_type']
        id_number = request.form['id_number']
        address = request.form['address']
        room_number = request.form.get('room_number', None)
        check_in = request.form['check_in']
        check_out = request.form['check_out']
        number_of_guests = request.form['number_of_guests']
        special_requests = request.form['special_requests']
        terms_and_conditions = request.form.get('terms_and_conditions', False)
        if not terms_and_conditions:
            flash("You must agree to the terms and conditions to proceed.", "danger")
            return redirect(url_for('book_room'))
        # Determine room type based on room number prefix
        if room_number.startswith('1'):
            room_type = 'Single'
        elif room_number.startswith('2'):
            room_type = 'Double'
        elif room_number.startswith('3'):
            room_type = 'Suite'
        else:
            flash("Invalid room selection.", "danger")
            return redirect(url_for('book_room'))
        # Get room rate based on room type
        room_rate = ROOM_RATES.get(room_type)
        # Connect to the MySQL database
        connection = mysql.connector.connect(**db_config)
        cursor = connection.cursor()
        try:
            # Insert the booking into the room_booking table
            query = '''INSERT INTO room_booking
                    (customer_id, customer_name, id_proof_type, id_number, address, room_type,
room_number, check_in, check_out, number_of_guests, special_requests, room_rate)
                VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)'''
            cursor.execute(query,
                    (customer_id, customer_name, id_proof_type, id_number, address, room_type,
room_number, check_in, check_out, number_of_guests, special_requests, room_rate))
            connection.commit()
            booking_id = cursor.lastrowid
            # Update the room's availability status in the rooms table
            update_query = '''UPDATE rooms SET is_available = FALSE WHERE room_number
= %s'''
            cursor.execute(update_query, (room_number,))
            connection.commit()
            flash("Room booked successfully!", "success")
            return redirect(url_for('generate_bill', booking_id=booking_id))
        except Exception as e:
            connection.rollback()
            flash(f"An error occurred: {str(e)}", "danger")
        finally:
            cursor.close()
            connection.close()
    # Fetch available rooms from the database
    available_rooms = fetch_available_rooms()
    customer_id = random.randint(1000, 9999)
    return render_template('book_room.html', customer_id=customer_id,
available_rooms=available_rooms, selected_room_number=room_number,
ROOM_RATES=ROOM_RATES)
@app.route('/bill/<int:booking_id>')
def generate_bill(booking_id):
    connection = mysql.connector.connect(**db_config)
    cursor = connection.cursor()
    # Retrieve booking information by booking_id
    query = '''SELECT customer_name, room_type, room_number, room_rate, check_in,
check_out
               FROM room_booking WHERE booking_id = %s'''
    cursor.execute(query, (booking_id,))
    booking = cursor.fetchone()
    if not booking:
        flash("No booking found with this ID", "danger")
        return redirect(url_for('home'))
    customer_name, room_type, room_number, room_rate, check_in, check_out = booking
    # Check if check_in and check_out are datetime.date objects
        check_in_date = check_in if isinstance(check_in, datetime.date) else
datetime.datetime.strptime(check_in, '%Y-%m-%d')
        check_out_date = check_out if isinstance(check_out, datetime.date) else
datetime.datetime.strptime(check_out, '%Y-%m-%d')
    # Calculate number of days
    number_of_days = (check_out_date - check_in_date).days
    # Room cost calculation
    room_cost = number_of_days * float(room_rate)  # Ensure room_rate is float
    # Tax calculations (assuming 10% tax)
    tax_rate = 0.05
    sub_tax = room_cost * tax_rate  # Now this works
    paid_tax = 0.10 * room_cost
    # Total cost calculation
    total_cost = room_cost + sub_tax + paid_tax
    # Insert the bill into the database
    insert_bill_query = '''INSERT INTO room_bill (booking_id, room_cost, sub_tax, paid_tax,
total_cost)
                        VALUES (%s, %s, %s, %s,%s)'''
    cursor.execute(insert_bill_query, (booking_id, room_cost, sub_tax, paid_tax, total_cost))
    connection.commit()
    # Fetch the generated bill_id
    bill_id = cursor.lastrowid
    cursor.close()
    connection.close()
    # # Redirect to payment page
    # return redirect(url_for('make_payment', bill_id=bill_id)
    return render_template('bill.html', action='make_payment',
    customer_name=customer_name, room_type=room_type,
    room_number=room_number, room_rate=room_rate, number_of_days=number_of_days,
    room_cost=room_cost, sub_tax=sub_tax, paid_tax=paid_tax, total_cost=total_cost,
    bill_id=bill_id)
@app.route('/payment/<int:bill_id>', methods=['GET', 'POST'])
def make_payment(bill_id):
    # Use mysql.connector.connect() here
    connection = mysql.connector.connect(**db_config)
    cursor = connection.cursor()
     # Retrieve bill and customer details
    query = '''SELECT booking_id, total_cost FROM room_bill WHERE bill_id = %s'''
    cursor.execute(query, (bill_id,))
    bill = cursor.fetchone()
    if not bill:
        flash("Bill not found", "danger")
        return redirect(url_for('home'))
    booking_id, total_cost = bill
    if request.method == 'POST':
        payment_method = request.form.get('payment_method')
        # Validate if payment_method is None or empty
        if not payment_method:
            flash("Please select a payment method.", "danger")
            # Re-render the payment form without executing the SQL query
            return render_template('payment.html', total_cost=total_cost, bill_id=bill_id)
        # Insert payment record into the payment table
        insert_payment_query = '''INSERT INTO payment (bill_id, booking_id, amount_paid,
payment_method, payment_status)
                                  VALUES (%s, %s, %s, %s, %s)'''
        cursor.execute(insert_payment_query, (bill_id, booking_id, total_cost, payment_method,
'Completed'))
        connection.commit()
        flash("Payment successful!", "success")
        return redirect(url_for('payment_confirmation', bill_id=bill_id))
    return render_template('payment.html', total_cost=total_cost, bill_id=bill_id)
@app.route('/payment_confirmation/<int:bill_id>')
def payment_confirmation(bill_id):
    connection = mysql.connector.connect(**db_config)
    cursor = connection.cursor()
    # Fetch payment details
    query = '''SELECT payment_id, amount_paid, payment_method, payment_date
            FROM payment WHERE bill_id = %s'''
    cursor.execute(query, (bill_id,))
    payment = cursor.fetchone()
    if not payment:
        flash("Payment not found", "danger")
        return redirect(url_for('home'))
    payment_id, amount_paid, payment_method, payment_date = payment
    return render_template('payment_confirmation.html', payment_id=payment_id,
amount_paid=amount_paid,
                           payment_method=payment_method, payment_date=payment_date)
```

- **User Authentication and Authorization**:
  - Used **Flask-Login** to manage user authentication (login/logout), sessions, and restricting access to admin-only routes.
  - Role-based access was implemented, where users have limited access to their booking information and admins can manage the entire system (e.g., manage rooms, view all bookings).
- **Coding:**

```
@app.route('/user_login', methods=['GET', 'POST'])
def user_login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password'
        # Connect to the MySQL database
        connection = mysql.connector.connect(**db_config)
        cursor = connection.cursor(dictionary=True)
        try:
            query = "SELECT * FROM users WHERE email = %s"
            cursor.execute(query, (email,))
            user = cursor.fetchone()
            cursor.close()
            connection.close()
            if user and user['password'] == password:
                session['user'] = user['customer_name']  # Store first name in session
                session['users_id'] = user['users_id']
                flash("Login successful!", "success")
                return redirect(url_for('user_home'))
            else:
```

```
            flash("Invalid email or password.", "danger")
        except mysql.connector.Error as err:
            flash(f"Database error: {err}", "danger")
        finally:
            cursor.close()
            connection.close()
    return render_template('user_login.html')
@app.route('/user_register', methods=['GET', 'POST'])
def user_register():
    if request.method == 'POST':
        customer_name = request.form['customer_name']
        father_name = request.form['father_name']
        gender = request.form['gender']
        mobile_number = request.form['mobile_number']
        email = request.form['email']
        nationality = request.form['nationality']
        password = request.form['password']
        confirm_password = request.form['confirm_password']
        security_question = request.form['security_question']
        security_answer = request.form['security_answer']
        # Validate that passwords match
        if password != confirm_password:
            flash("Passwords do not match.", "danger")
            return redirect(url_for('register'))
        # Connect to the MySQL database
        connection = mysql.connector.connect(**db_config)
        cursor = connection.cursor()
        # Insert new user into the users table
        try:
            query = "INSERT INTO users (customer_name, father_name, gender, mobile_number, email, nationality, password, security_question, security_answer) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
            cursor.execute(query, (customer_name, father_name, gender, mobile_number, email, nationality, password, security_question, security_answer))
            connection.commit()
            flash('Registration successful! You can now log in.', 'success')
            return redirect(url_for('user_login'))
        except mysql.connector.IntegrityError:
            flash('Username already exists. Please choose another.', 'error')
        finally:
            cursor.close()
            connection.close()
    return render_template('user_register.html')
```

4.3 Frontend Implementation (HTML, CSS, JavaScript)

➢ **Responsive Design:**

- The frontend was developed using HTML5 and CSS3, following modern design practices.
- **Bootstrap** was used to create a responsive, mobile-first interface, ensuring the website adjusts well to different screen sizes (desktop, tablet, mobile).

➢ **Coding:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hotel Ichiraku</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <link                                                    rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"   integrity="sha384-
AYmEC3Yw5cVb3ZcuHtOA93w35dYTsvhLPVnYs9eStHfGJvOvKxVfELGroGkv
sg+p" crossorigin="anonymous" />
    <link                                                    rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.carousel.
css"                                                     integrity="sha512-
UTNP5BXLIptsaj5WdKFrkFov94lDx+eBvbKyoe1YAfjeRPC+gT5kyZ10kOHCfN
ZqEui1sxmqvodNUx3KbuYI/A==" crossorigin="anonymous"
    referrerpolicy="no-referrer" />
    <link                                                    rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.theme.de
fault.min.css"                                           integrity="sha512-
sMXtMNL1zRzolHYKEujM2AqCLUR9F2C4/05cdbxjjLSRvMQIciEPCQZo++nk7
go3BtSuK9kfa/s+a4f4i5pLkw=="
    crossorigin="anonymous" referrerpolicy="no-referrer" />
    <script   src="https://code.jquery.com/jquery-1.12.4.min.js"   integrity="sha256-
ZosEbRLbNQzLpnKIkEdrPv7lOy9C27hHQ+Xp8a4MxAQ="
crossorigin="anonymous"></script>
</head>
<body>
```

```
<header>
        <h1>Hotel Ichiraku</h1>
        <nav>
            <ul>
                <div><a href="/">Home</a></div>
            </ul>
        </nav>
    {% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
    <ul class="flashes">
        {% for category, message in messages %}
        <li class="{{ category }}">{{ message }}</li>
        {% endfor %}
    </ul>
    {% endif %}
    {% endwith %}
</header>
<div id="main-content">
    {% block content %}{% endblock %}
</div>
<section class="about top" id="about">
    <div class="right">
        <div class="discreption">
```

<p> The Hotel Ichiraku offers luxury accommodations, providing top-tier service, elegant decor, and premium amenities. Guests can expect spacious, beautifully furnished rooms, gourmet dining options, spa and wellness facilities, and a range of personalized services. The atmosphere is refined and sophisticated, catering

```
to guests seeking comfort, relaxation, and an exceptional level of hospitality.</p>
      </div>
    </div>
  </section>
  <footer>
    <div class="container grid top">
      <div class="box">
        <h3>Recent News</h3>
        <ul>
          <li>Eco-Friendly Initiative</li>
          <li>Chill and Escape in Our Natural Shelters</li>
          <li>Limited-Time Offer</li>
          <li>Live Music Concerts</li>
          <li>Luxury Suite Renovation</li>
        </ul>
      </div>
      <div class="box">
        <h3>For Customers</h3>
        <ul>
          <li>About Ichiraku</li>
          <li>Customer Care/Help</li>
          <li>Corporate Accounts</li>
          <li>Financial Information</li>
          <li>Terms & Conditions</li>
        </ul>
      </div>
      <div class="box">
```

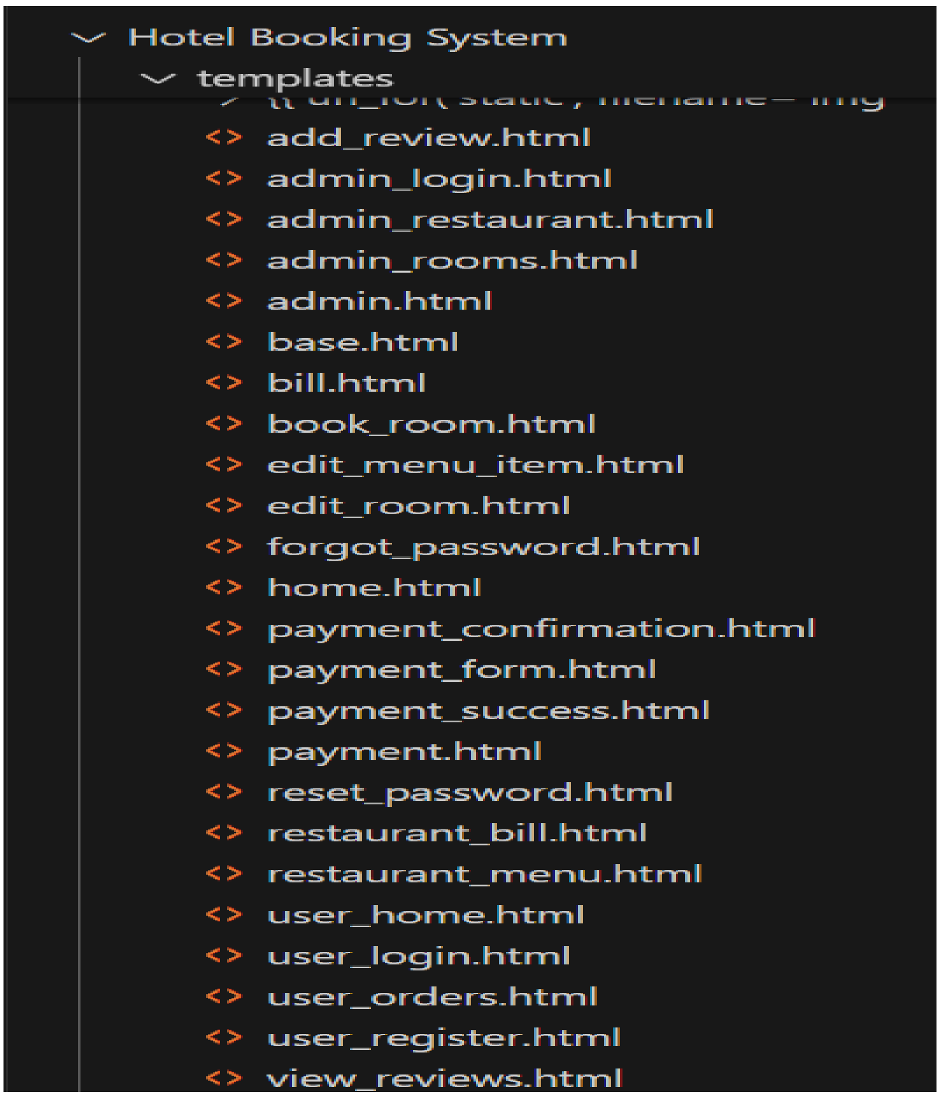
```
            <h3>Contact Us</h3>
            <ul>
                <li>Agaram Main Road, Selaiyur, Tambaram, Chennai, 600073</li>
                <li><i class="far fa-envelope"></i>hotelichiraku@gmail.com </li>
                <li><i class="far fa-phone-alt"></i>9846369467 </li>
                <li><i class="far fa-phone-alt"></i>9870567429 </li>
                <li><i class="far fa-comments"></i>24/ 7 Customer Services </li>
            </ul>
        </div>
    </div>
    <div class="bottom-bar">
        <p>&copy; 2024 Hotel Ichiraku . All rights reserved</p>
    </div>
    <!-- <p>&copy; 2024 Hotel Ichiraku</p> -->
</footer>
<script>
    // Automatically remove flash messages after 5 seconds
    setTimeout(function() {
        const flashMessages = document.querySelectorAll('.flashes li');
        flashMessages.forEach(flash => {
            flash.style.transition = 'opacity 0.5s ease';
            flash.style.opacity = '0';
            setTimeout(() => flash.remove(), 500); // Completely remove after fade-out
        });
    }, 500); // 5000ms = 5 seconds
</script>
<script src="{{ url_for('static', filename='js/script.js') }}"></script>
```

</body>

</html>

➢ **Navigation and User Flow:**

- Implemented a navigation bar with links to key pages (Home, Room Booking, Admin Dashboard, Login, etc.).
- Ensured a clean and intuitive user flow for booking rooms, placing restaurant orders, and providing feedback.



```
Hotel Booking System
  templates
      add_review.html
      admin_login.html
      admin_restaurant.html
      admin_rooms.html
      admin.html
      base.html
      bill.html
      book_room.html
      edit_menu_item.html
      edit_room.html
      forgot_password.html
      home.html
      payment_confirmation.html
      payment_form.html
      payment_success.html
      payment.html
      reset_password.html
      restaurant_bill.html
      restaurant_menu.html
      user_home.html
      user_login.html
      user_orders.html
      user_register.html
      view_reviews.html
```

Fig: 4.3 Navigation and User Flow

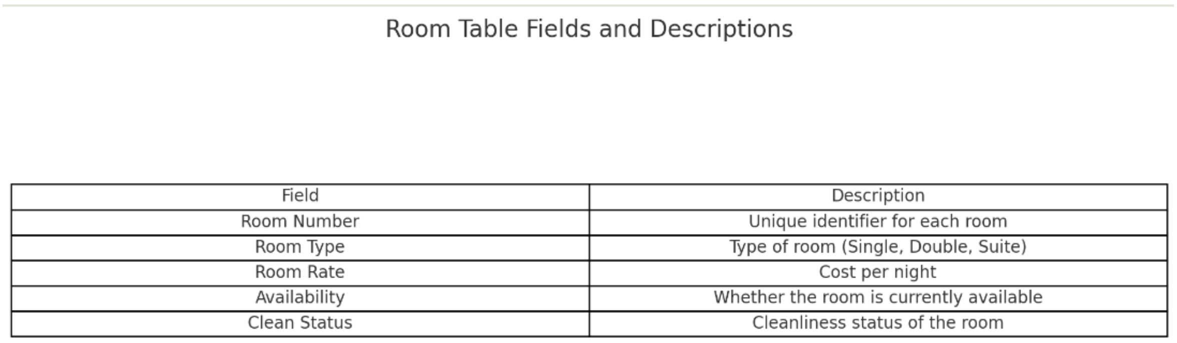
- **Forms and Validation:**
  - Created HTML forms for user inputs, such as login, registration, booking, and restaurant order.
  - Used JavaScript for client-side validation, ensuring that form fields are correctly filled before submission.
- **Interactivity and Dynamic Content:**
  - Incorporated JavaScript (and jQuery) to provide dynamic content, such as displaying available rooms and calculating total prices in real-time.
  - Implemented modal windows for actions like confirming bookings and showing details.
- **Styling:**
  - The system's design focused on simplicity and consistency. Custom CSS was written to match the branding and aesthetic preferences, ensuring that the user interface was visually appealing and easy to navigate.

4.4 Database Implementation (MySQL)

- **Database Schema:**

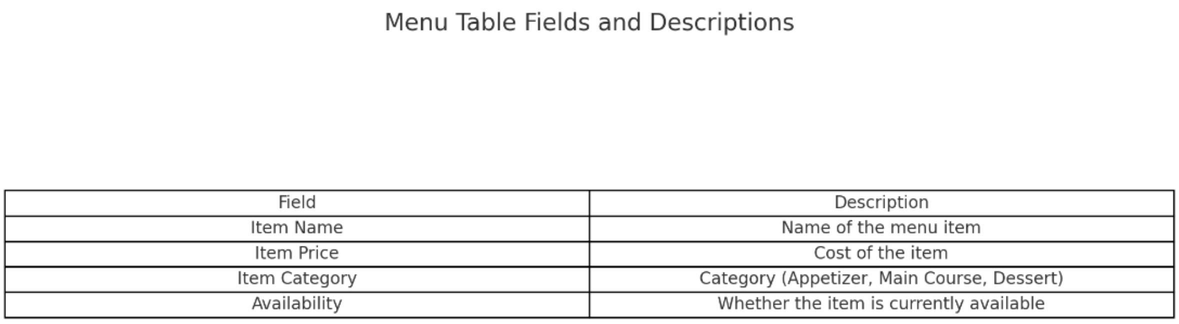
  The database schema was implemented based on the ERD designed earlier, with the following tables:

  - **Users**: Stores user information, including their credentials, roles (admin or customer), and other profile details.
  - **Rooms**: Contains details such as room types, availability status, rates, and description.
  - **Bookings**: Tracks room reservations, including user ID, room ID, check-in/check-out dates, and booking status.
  - **Restaurant Orders**: Contains order details for food items, including user ID, items ordered, and total price.
  - **Reviews**: Stores feedback from users regarding rooms and services, including rating and comments.


Room Table Fields and Descriptions

Field	Description
Room Number	Unique identifier for each room
Room Type	Type of room (Single, Double, Suite)
Room Rate	Cost per night
Availability	Whether the room is currently available
Clean Status	Cleanliness status of the room

Table: 4.4.1 Database Schema


Menu Table Fields and Descriptions

Field	Description
Item Name	Name of the menu item
Item Price	Cost of the item
Item Category	Category (Appetizer, Main Course, Dessert)
Availability	Whether the item is currently available

Table: 4.4.2 Database schema

4.4.1 Database Table Design

- Provide a brief overview of the tables and their purpose. For example:
  - **Customers Table**: Stores user details.
  - **Bookings Table**: Maintains booking records.
  - **Bookings Table**: Maintains booking records.
- **Coding:**

```
CREATE DATABASE hotel_management;
USE hotel_management;
CREATE TABLE admin (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
```

```
);
INSERT INTO admin (username, password)
VALUES ('Meghendra Naidu', 'Megh1234');
select * from admin;
CREATE TABLE users (
   users_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_name VARCHAR(255) NOT NULL,
   father_name VARCHAR(255) NOT NULL,
   gender ENUM('Male', 'Female', 'Other') NOT NULL,
   mobile_number VARCHAR(15) NOT NULL,
   email VARCHAR(255) UNIQUE NOT NULL,
   nationality VARCHAR(100) NOT NULL,
   password VARCHAR(255) NOT NULL,
   security_question TEXT NOT NULL,
   security_answer TEXT NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
select * from users;
CREATE TABLE room_booking (
   booking_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_id INT NOT NULL,
   customer_name VARCHAR(255) NOT NULL,
   id_proof_type ENUM('Passport', 'Driving License', 'Aadhar', 'Other') NOT
NULL,
   id_number VARCHAR(50) NOT NULL,
    address TEXT NOT NULL,
    room_type ENUM('Single', 'Double', 'Suite') NOT NULL,
    room_number VARCHAR(10) NOT NULL,
    check_in DATE NOT NULL,
    check_out DATE NOT NULL,
    number_of_guests INT DEFAULT 1,
    special_requests TEXT,
    room_rate DECIMAL(10, 2) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (room_number) REFERENCES rooms(room_number)
);
select * from room_booking;
CREATE TABLE room_bill (
    bill_id INT AUTO_INCREMENT PRIMARY KEY,
    booking_id INT NOT NULL,
    room_cost DECIMAL(10, 2) NOT NULL,
    sub_tax DECIMAL(10, 2) NOT NULL,
    paid_tax DECIMAL(10, 2) NOT NULL,
    total_cost DECIMAL(10, 2) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (booking_id) REFERENCES room_booking(booking_id)
);
select * from room_bill;
CREATE TABLE payment (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    bill_id INT NOT NULL,
    booking_id INT NOT NULL,
    amount_paid DECIMAL(10, 2) NOT NULL,
    payment_method ENUM('Cash', 'Credit Card', 'Debit Card', 'Net Banking', 'UPI',
'Other') NOT NULL,
    payment_status ENUM('Pending', 'Completed', 'Failed') DEFAULT 'Pending',
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (bill_id) REFERENCES room_bill(bill_id),
    FOREIGN KEY (booking_id) REFERENCES room_booking(booking_id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
select * from payment;

CREATE TABLE rooms (
    room_number VARCHAR(10) PRIMARY KEY,
    room_type ENUM('Single', 'Double', 'Suite') NOT NULL,
    is_available BOOLEAN DEFAULT TRUE,
    clean_status ENUM('Clean', 'Dirty') DEFAULT 'Clean',
    room_rate DECIMAL(10, 2) DEFAULT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
select * from rooms;
CREATE TABLE restaurant (
    item_id INT AUTO_INCREMENT PRIMARY KEY,
    item_name VARCHAR(255) NOT NULL,
    item_price DECIMAL(10, 2) NOT NULL,
    item_category VARCHAR(100) NOT NULL,
    availability ENUM('Available', 'Unavailable') DEFAULT 'Available'
);
select * from restaurant;
CREATE TABLE orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    users_id INT NOT NULL,
    item_id INT NOT NULL,
    quantity INT NOT NULL,
    total_price DECIMAL(10, 2) NOT NULL,
    is_billed BOOLEAN DEFAULT FALSE,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (item_id) REFERENCES restaurant(item_id),
    FOREIGN KEY (users_id) REFERENCES users(users_id)
);
select * from orders;
CREATE TABLE restaurant_bill (
    bill_id INT AUTO_INCREMENT PRIMARY KEY,
    users_id INT NOT NULL,
    order_id INT NOT NULL,
    total_amount DECIMAL(10, 2) NOT NULL,
    tax DECIMAL(10, 2) NOT NULL,
    final_amount DECIMAL(10, 2) NOT NULL,
    status VARCHAR(20) DEFAULT 'Unpaid',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (users_id) REFERENCES users(users_id),
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
);
select * from restaurant_bill;
CREATE TABLE payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    bill_id INT NOT NULL,
    payment_method VARCHAR(50),
    amount_paid DECIMAL(10, 2) NOT NULL,
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (bill_id) REFERENCES restaurant_bill(bill_id)
);
select * from payments;
CREATE TABLE reviews (
    review_id INT AUTO_INCREMENT PRIMARY KEY,
    users_id INT NOT NULL,
    customer_name VARCHAR(255) NOT NULL,
    rating INT CHECK(rating BETWEEN 1 AND 5),
    review TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (users_id) REFERENCES users(users_id)
);
select * from reviews;
```

4.5 Testing and Debugging

- **Unit Testing:**
  - Each module (e.g., booking, user authentication, restaurant order) underwent unit testing to ensure individual functionality works as expected.

- Used Python's unittest module to create test cases for Flask routes and database queries.

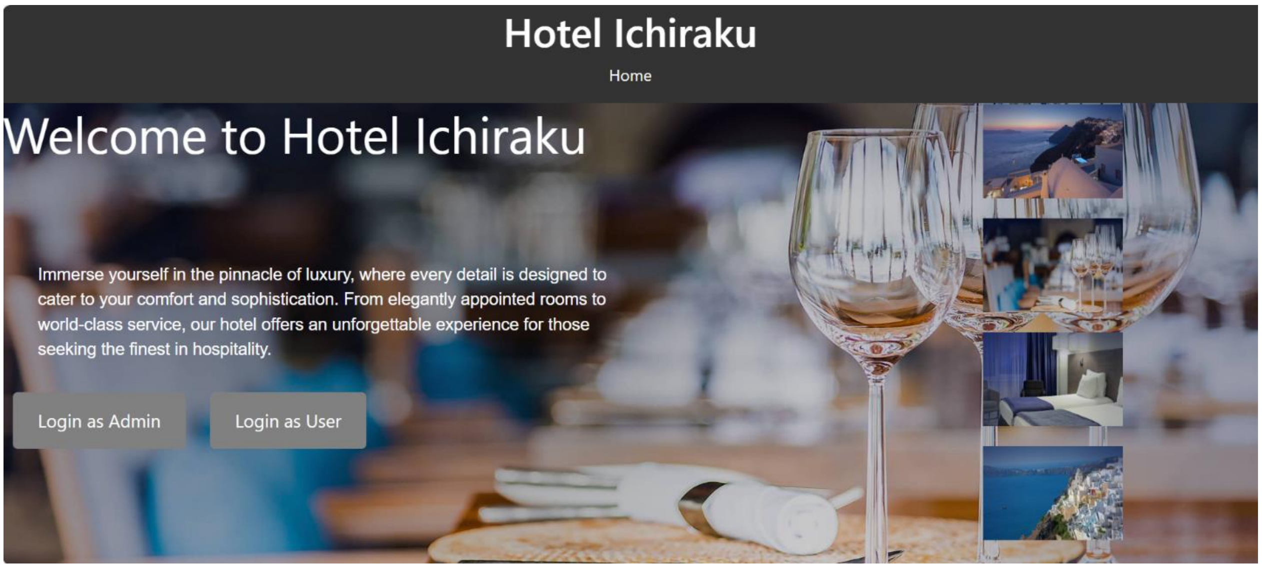
➢ **Integration Testing:**

- Ensured that all components (frontend, backend, and database) worked seamlessly together.
- Validated end-to-end flows such as room booking, payment processing, and order management.

➢ **Bug Fixing and Optimization:**

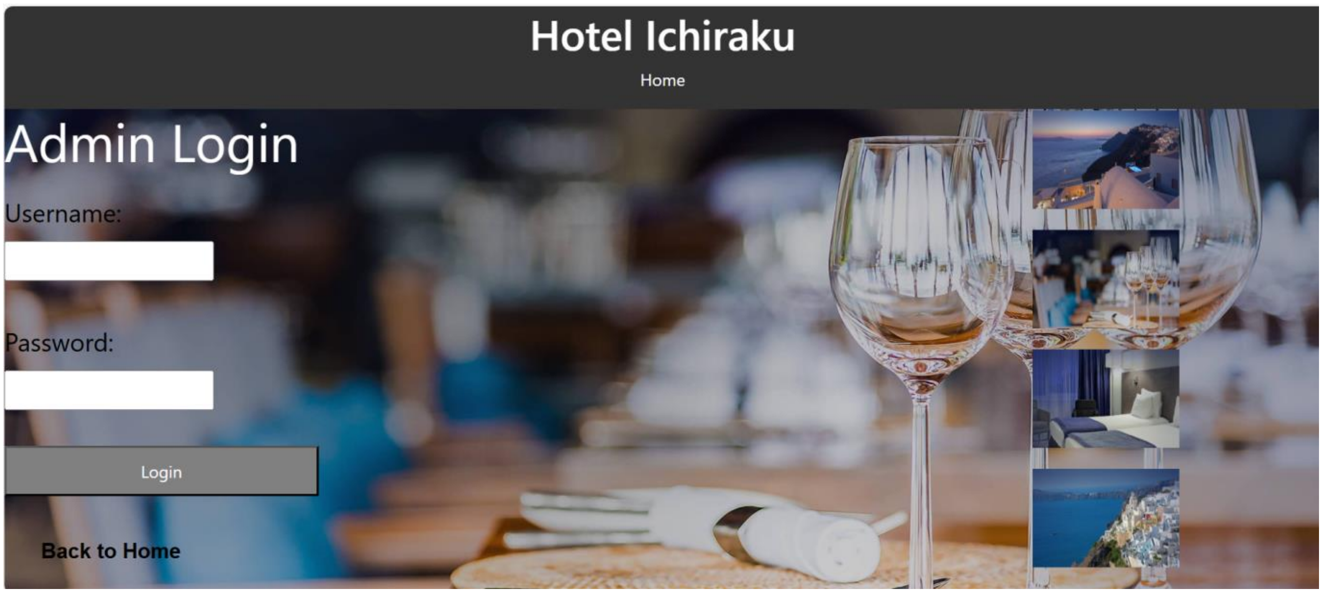
- Debugged and resolved issues using Python's logging capabilities and browser developer tools.
- Optimized code and SQL queries to improve system performance, especially during high traffic periods.

➢ **Output Images**:



Welcome to Hotel Ichiraku

Immerse yourself in the pinnacle of luxury, where every detail is designed to cater to your comfort and sophistication. From elegantly appointed rooms to world-class service, our hotel offers an unforgettable experience for those seeking the finest in hospitality.



Hotel Ichiraku

Home

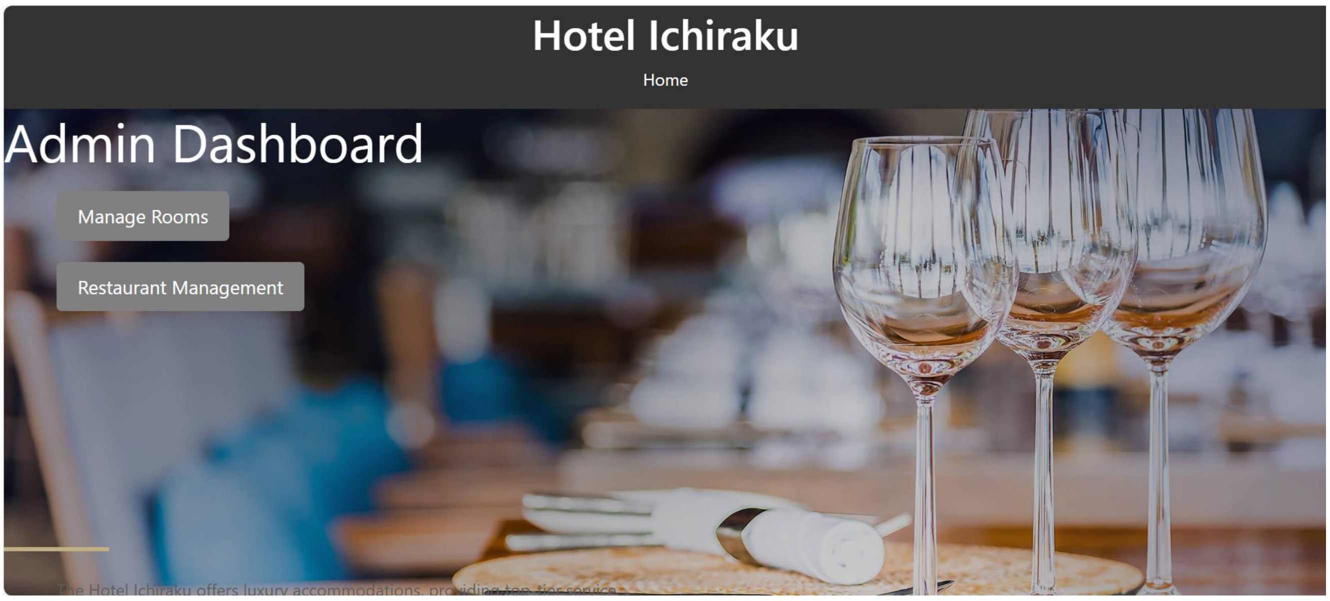
Admin Login

Username:

Password:

Login

Back to Home



Hotel Ichiraku

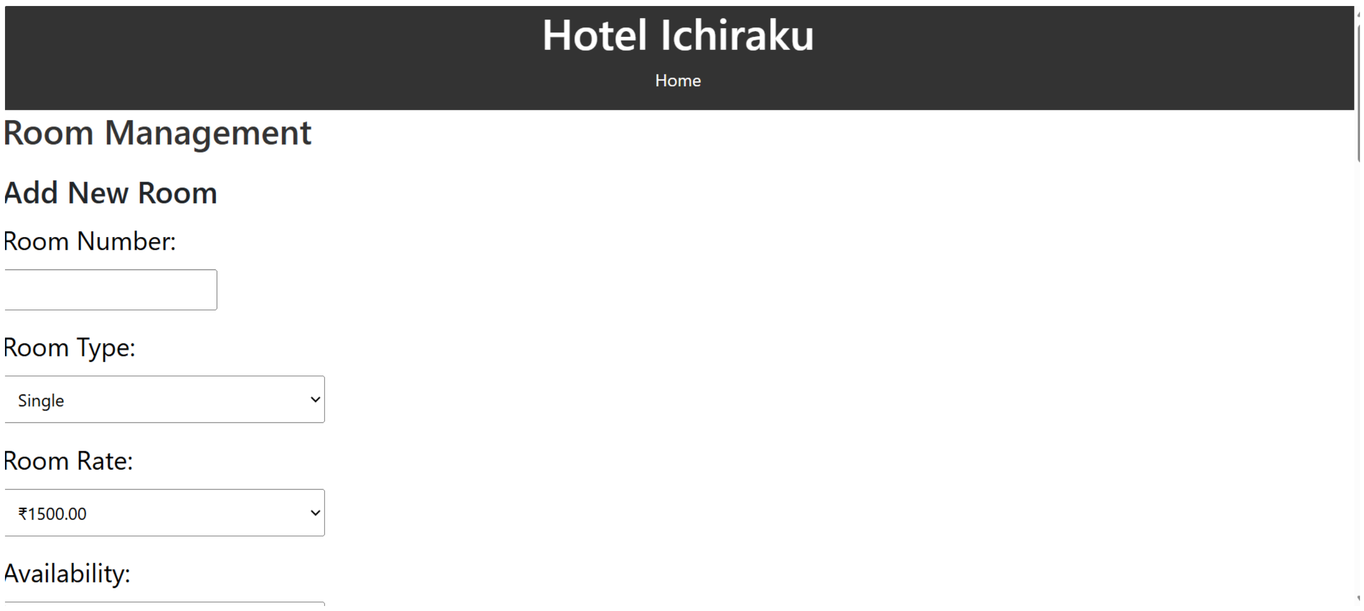
Home

Admin Dashboard

Manage Rooms

Restaurant Management

The Hotel Ichiraku offers luxury accommodations, providing top-tier service.



Hotel Ichiraku

Home

Room Management

Add New Room

Room Number:

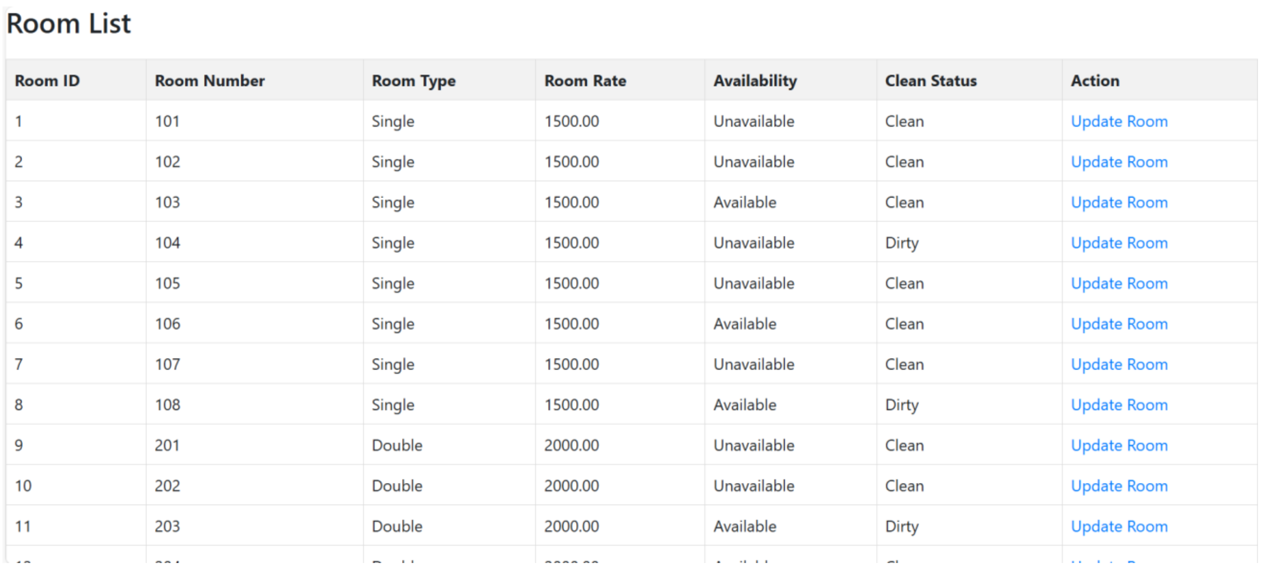
Room Type:

Single

Room Rate:

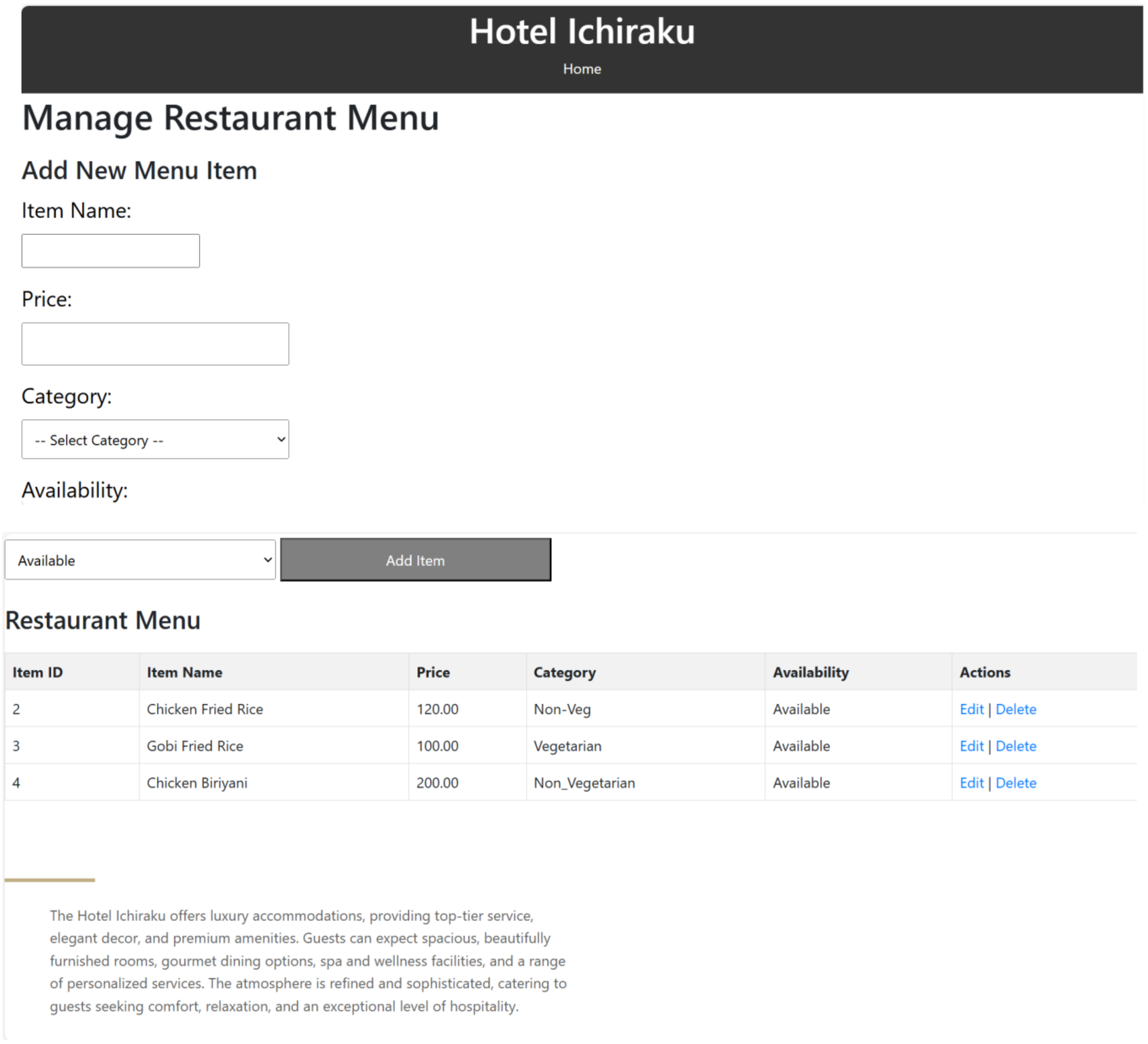
₹1500.00

Availability:



Room List

Room ID	Room Number	Room Type	Room Rate	Availability	Clean Status	Action
1	101	Single	1500.00	Unavailable	Clean	Update Room
2	102	Single	1500.00	Unavailable	Clean	Update Room
3	103	Single	1500.00	Available	Clean	Update Room
4	104	Single	1500.00	Unavailable	Dirty	Update Room
5	105	Single	1500.00	Unavailable	Clean	Update Room
6	106	Single	1500.00	Available	Clean	Update Room
7	107	Single	1500.00	Unavailable	Clean	Update Room
8	108	Single	1500.00	Available	Dirty	Update Room
9	201	Double	2000.00	Unavailable	Clean	Update Room
10	202	Double	2000.00	Unavailable	Clean	Update Room
11	203	Double	2000.00	Available	Dirty	Update Room



Manage Restaurant Menu

Add New Menu Item

Item Name:

Price:

Category:

-- Select Category --

Availability:

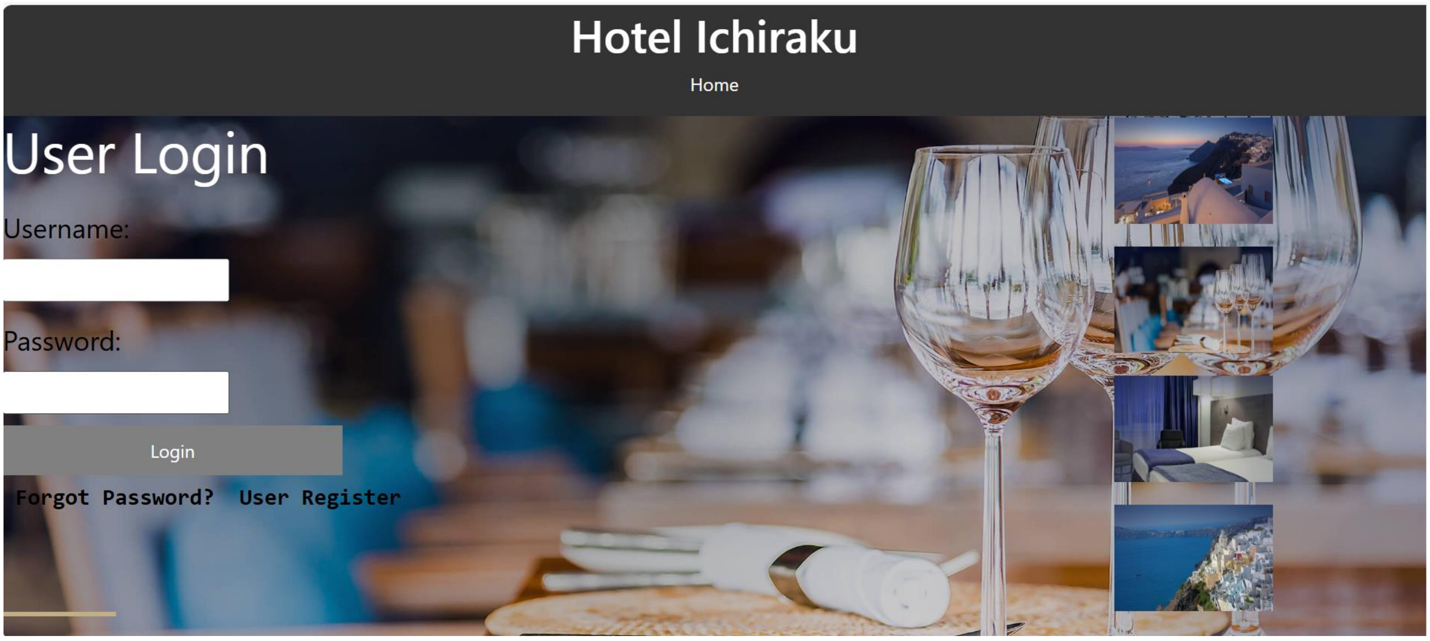
Available

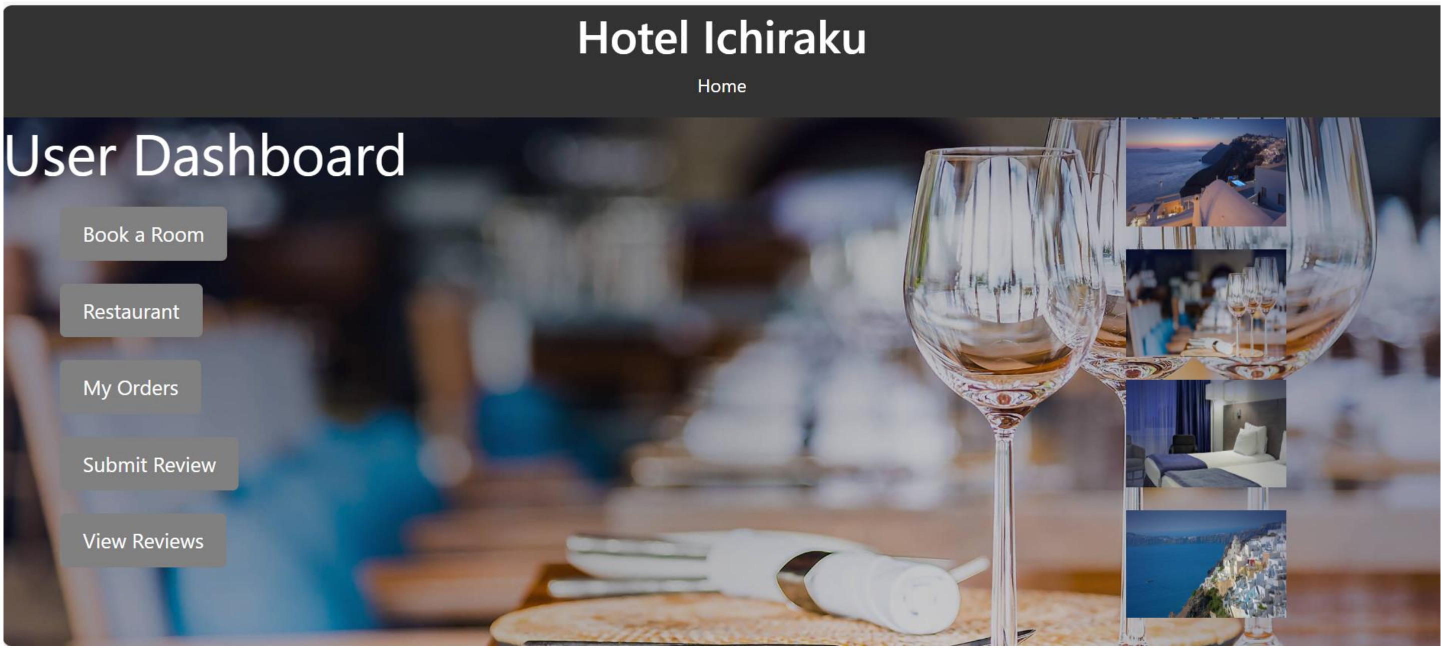
Add Item

Restaurant Menu

Item ID	Item Name	Price	Category	Availability	Actions
2	Chicken Fried Rice	120.00	Non-Veg	Available	Edit | Delete
3	Gobi Fried Rice	100.00	Vegetarian	Available	Edit | Delete
4	Chicken Biriyani	200.00	Non_Vegetarian	Available	Edit | Delete

The Hotel Ichiraku offers luxury accommodations, providing top-tier service, elegant decor, and premium amenities. Guests can expect spacious, beautifully furnished rooms, gourmet dining options, spa and wellness facilities, and a range of personalized services. The atmosphere is refined and sophisticated, catering to guests seeking comfort, relaxation, and an exceptional level of hospitality.


Hotel Ichiraku
Home
User Login
Username:
Password:
Login
Forgot Password?
User Register


Hotel Ichiraku
Home
User Dashboard
Book a Room
Restaurant
My Orders
Submit Review
View Reviews

CHAPTER 5

RESULTS AND DISCUSSION

This chapter provides an analysis of the outcomes from the hotel management system implementation. It presents the results of the system's functionality, its performance, and evaluates its overall effectiveness. This section also discusses any challenges faced during the development and offers solutions and potential improvements for future enhancements.

5.1 Overview of Results

The development and implementation of the hotel management system were carried out as per the design and specifications outlined in the earlier chapters. The system was fully functional and met the primary objectives, which include efficient room management, secure booking, and user-friendly interfaces for both customers and administrators. The results are assessed in terms of functionality, usability, performance, and security. F1 Score: The F1 score, a harmonic mean of precision and recall, highlights the model's balanced performance, making it suitable for practical applications in quality control.

- **Functional Results**: The core features of the system—such as room management, bookings, restaurant orders, and user authentication—were successfully implemented and are operational.
- **Non-Functional Results**: The system performs well under expected traffic loads, with a smooth user experience and quick page loads. Security measures like password hashing, session management, and HTTPS encryption are working as intended.

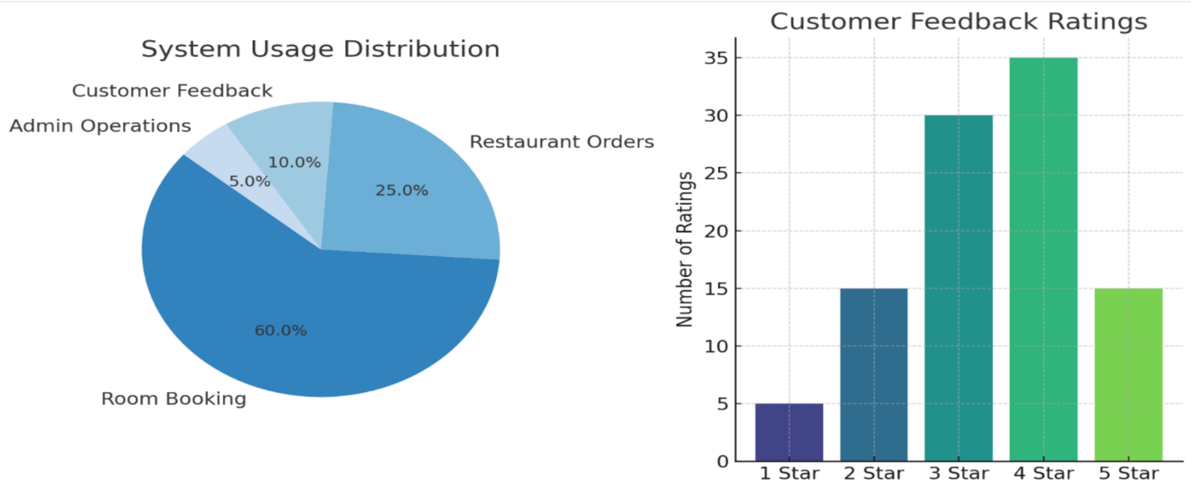


Fig: 5.1 Overview of Results

5.1.1Graph Representation

- Describe the purpose of the graph:

  For example, whether it depicts the system's performance (e.g., response time, success rate of bookings) or user behavior patterns (e.g., peak booking times, payment method preferences).

- Explain the axes and the data points in detail to help readers interpret the graph correctly.

5.1.2 Insights from the Graph

- Analyze the data represented in the graph.
- Highlight any trends, patterns, or anomalies visible in the graph.
- Discuss the implications of these insights for the system's operation or future improvements.

5.2 Functionality Testing

- **Room Management:**

  The system's room management feature was thoroughly tested to ensure that it accurately tracks room availability and pricing.

- **Room Additions/Updates**: Admins can add new rooms, update room details (price, description), and delete rooms from the system.
- **Availability Check:** Customers can check room availability for specific dates, and the system dynamically updates available options based on bookings.

- **Room Selection:** Customers are able to select rooms based on their preferences, including room type, price range, and amenities.

➢ **Booking System:**

The booking process was validated, with several tests conducted to ensure that users can:

- Choose available rooms.
- Enter booking information (dates, number of guests).
- Confirm their reservation and receive booking confirmation emails.
- Cancel or modify bookings if required.

➢ **Restaurant Orders:**

Customers were able to browse the restaurant menu, select food items, place orders, and view total prices. The system calculates order totals dynamically, and the admin can view and manage customer orders efficiently.

➢ **User Authentication:**

User registration, login, and session management features were tested to ensure:

- Users can sign up, log in securely, and maintain their sessions.
- Admins are restricted to admin-only pages, and non-admin users are restricted from accessing admin features.
- Passwords are securely stored and cannot be decrypted.

5.3 Usability Testing

Usability testing was conducted to ensure that the hotel management system is intuitive and easy to use. Feedback was gathered from users of different technical backgrounds, including both customers and administrators.

➢ **Customer Interface**:

The customer-facing interface was designed to be simple and easy to navigate. Users were able to easily search for available rooms, complete bookings, and place orders. Feedback indicated that the booking process was smooth, with all necessary information displayed clearly.

- **Strengths**: Clean and minimal design, responsive layout, clear room descriptions, and straightforward booking steps.
- **Areas for Improvement**: Some users suggested that more filtering options for room preferences (e.g., price, location, amenities) would improve the experience.

➢ **Admin Interface**:

The admin interface was tested by hotel staff to manage rooms, bookings, and orders. It was easy for them to view all bookings, update room details, and manage customer orders.

- **Strengths**: Efficient dashboard, clear visualization of room availability, booking status, and easy navigation between sections.
- **Areas for Improvement**: Some suggested adding more advanced analytics and reporting features, such as financial reports or occupancy rates.

5.4 Performance Evaluation

➢ **Page Load Time:**

The system's responsiveness was evaluated, and it showed good performance in terms of page load time. The application was able to handle typical traffic without any noticeable lag.

- **Database Queries**: All database queries were optimized to ensure quick response times for room availability and booking processes.
- **Caching**: Static resources such as images and CSS files were cached to improve loading times for repeat users.
- **Peak Traffic Simulation**: The system was able to handle moderate to heavy traffic loads (e.g., multiple users browsing and booking rooms simultaneously) without noticeable performance degradation.

➢ **Mobile Performance:**

Given that many users may access the system on mobile devices, performance on smartphones and tablets was a key consideration. The mobile version of the site performed well, with responsive design and minimal load times.

➢ **Scalability:**

The system was built with scalability in mind, using Flask and MySQL, which can handle future increases in traffic and database size. As the number of users or rooms increases, further optimizations (e.g., database indexing, load balancing) could be implemented.

5.5 Security Testing

Security is critical in any system that handles sensitive data such as personal information and payment details. Various security measures were tested to ensure the system is robust against common threats.

➢ **Authentication and Authorization**:

The system uses Flask-Login for session management and restricts access to specific features based on user roles.

- **Strengths**: Users cannot access restricted areas (e.g., admin dashboard) without proper authentication.
- **Testing Results**: Tests were conducted using both valid and invalid credentials. The system correctly denied access to unauthorized users.

➢ **Password Security**:

Passwords were stored using Bcrypt hashing, ensuring that even if the database is compromised, passwords cannot be easily retrieved.

- **Testing Results**: Attempts to recover passwords directly from the database failed, as the passwords are securely hashed.

➢ **SQL Injection Protection**:

The system uses parameterized queries, which effectively prevent SQL injection attacks.

- **Testing Results**: Attempts to manipulate SQL queries through user input (e.g., injecting SQL commands into form fields) were blocked by the database.

- **Cross-Site Scripting (XSS) Protection**:
  All user input is sanitized and validated to prevent the execution of malicious scripts in the browser.
  - **Testing Results**: Attempts to inject scripts into user forms or URLs failed, as the input is sanitized before being displayed.
- **HTTPS Encryption**:
  HTTPS was enabled for all pages that require sensitive information, ensuring that user data (e.g., login credentials, payment details) is encrypted in transit.
  - **Testing Results**: A secure connection was established for all interactions that involve sensitive data.

5.5 Challenges Faced and Solutions

- **Challenge 1: Handling Payment Gateways:**
  Integrating payment gateways (e.g., Stripe) was a challenge due to the complexity of setting up secure transactions.
  - **Solution**: The payment gateway was successfully integrated after thorough testing in a sandbox environment. Additionally, a secure connection (HTTPS) was ensured during payment processing.
- **Challenge 2: Room Availability Logic:**
  Implementing the logic to update room availability in real-time was complex, especially when multiple users were booking rooms simultaneously.
  - **Solution**: Optimized database queries and implemented locking mechanisms to prevent double-booking of rooms.
- **Challenge 3: Ensuring User Data Security:**
  Storing sensitive user data securely was a concern, especially with the handling of passwords and payment details.
  - **Solution**: Bcrypt hashing for passwords, HTTPS encryption for all sensitive communications, and careful validation of payment information were implemented.

5.7 Future Improvements

While the system performs well, there are several areas for future enhancement:

- **Advanced Reporting and Analytics:** Admins would benefit from more detailed reports (e.g., revenue analysis, customer demographics).
- **Mobile Application**: Developing a native mobile app for easier access to the hotel management system could improve the user experience.
- **AI and Machine Learning Integration**: AI could be used to recommend rooms to customers based on their preferences or predict high-demand periods for better pricing and availability management.
- **Integration with Third-Party Services**: Further integrations with services like third-party booking platforms (e.g., Booking.com, Expedia) could expand the system's reach.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

This chapter provides a summary of the project’s achievements, outlines its impact, and discusses the areas for future enhancements. It reflects on the lessons learned during the development process and the potential for expanding the system to address future challenges and opportunities.

6.1 Conclusion

The Hotel Management System was successfully designed and implemented to streamline hotel operations, improve customer experiences, and optimize resource management. The system fulfills its primary objectives by offering secure and efficient features such as room management, booking systems, restaurant ordering, and user authentication.

➢ **Key Achievements:**
- Simplified room and booking management for both customers and hotel administrators.
- Enhanced security features with Bcrypt password hashing, HTTPS encryption, and session management.
- A user-friendly interface that caters to both technical and non-technical users.
- Performance optimization to handle simultaneous user interactions effectively.

The project demonstrates the practical application of Python, Flask, and MySQL in building a robust, scalable, and secure application tailored to meet real-world requirements in the hospitality sector.

6.2 Lessons Learned

The development process provided several valuable insights:

- **Technical Skills:** Advanced knowledge of Flask, database optimization, and security practices.
- **Problem-Solving**: Addressing challenges such as concurrency in bookings, secure payment integration, and responsive UI design.
- **Project Management**: Effective time management and modular development led to a successful implementation within the desired timeline.

6.3 Future Scope

The system offers a strong foundation for further enhancements to meet the evolving needs of the hospitality industry. Potential areas for expansion include:

6.3.1 Advanced Features and Functionalities

- **Dynamic Pricing Models:**
  Implement algorithms for dynamic pricing based on factors like season, demand, and booking trends.
- **Multi-Language and Multi-Currency Support**:
  Add features to support global customers, including translations and currency conversions.
- **Customer Feedback System**:
  Introduce feedback and rating features for rooms, food, and services to enhance customer satisfaction and improve services.

6.3.2 Integration with Third-Party Services

- **Travel Platforms:**
  Integrate with platforms like Booking.com or Expedia for wider reach and streamlined booking processes.
- **Payment Gateways:**
  Expand the system’s payment options by integrating multiple gateways like PayPal, Stripe, and regional payment services.

6.3.3 AI and Data Analytics

- **Recommendation Systems:**

  Use machine learning algorithms to recommend rooms, packages, or food items based on user preferences.

- **Predictive Analytics:**

  Leverage data analytics to forecast demand trends, optimize room allocation, and plan marketing campaigns.

6.3.4 Mobile and IoT Integration

- **Mobile App Development:**

  Create native or hybrid mobile applications to provide better accessibility to customers.

- **IoT Integration:**

  Introduce smart room features, such as IoT-enabled door locks, room temperature control, and personalized room settings.

6.3.5 Enhanced Reporting and Monitoring

- **Dashboard Analytics:**

  Develop an admin dashboard with real-time analytics for revenue, occupancy rates, and customer demographics.

- **Fraud Detection Systems:**

  Implement advanced monitoring for unusual activities, such as fraudulent bookings or unauthorized access attempts.

6.3.6 Cloud Deployment

- **Scalability and Redundancy:**

  Migrate the system to cloud platforms (e.g., AWS, Azure, Google Cloud) for improved scalability and reliability.

- **Disaster Recovery:**

  Implement automated backups and disaster recovery mechanisms in the cloud.

6.4 Broader Impact

The adoption of this system can revolutionize hotel operations by enhancing service delivery, improving resource utilization, and creating a better customer experience. Additionally, the modular and scalable architecture ensures the system can be adapted to a wide range of hospitality businesses, from small inns to large hotel chains.

CHAPTER 7

REFERENCES

1. A . Praveen et al., "Conference Room Booking Application using Flutter," 2020 International Conference on Communication and Signal Processing (ICCSP).

2. H. Singh and R. R. Shah, "BOOKiiIT - Designing a Venue Booking System (Technical Demo)," 2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM), 2020

3. K. M. O. Nahar and R. M. Al-Khatib, "EPSSR: Energy preserving system for smart rooms," 2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS), 2017.

4. L. Duc Tran et al., "A smart meeting room scheduling and management system with utilization control and ad-hoc support based on real-time occupancy detection," 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), 2016.

5. L. M. Sánchez, I. Díaz-Oreiro, L. Quesada, L. A. Guerrero and G. López, "Smart Meeting Room Management System Based on Real-Time Occupancy," 2019 IV Jornadas Costarricenses de Investigación en Computación e Informática (JoCICI), 2019.

6. P. Somwong, S. Jaipoonpol, P. Champrasert and Y. Somchit, "Smart Room Vacancy Status Checking and Booking System," 2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), 2022.

7. R. Dhanagopal, N. Archana and R. Menaka, "Enhanced Hotel Booking Application using PEGA," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2020.

8. S. Banerjee and A. Pal, "Luxury Hotel Booking and Scarcity Messages: Does Online Purchase Behavior Matter?," 2020 6th International Conference on Information Management (ICIM), 2020.

9. Y. Xu, "Design of CRM Hotel Management System Based on Machine Learning Algorithm," 2023 International Conference on Networking, Informatics and Computing (ICNETIC), Palermo, Italy, 2023, pp. 645- 649, doi: 10.1109/ICNETIC59568.2023.00138.