Malnad College of Engineering

(An autonomous institution under Visvesvaraya Technological University, Belgaum)

Hassan – 573201, Karnataka, India

Department of Information Science and Engineering

Course Title: Full Stack Development

Project Title: Hotel Management System

Group Number: 11

Submitted By:

Name	USN
Praneeth M	4MC23IS073
Pranav Bharadwaj V	4MC23IS072
Dhanush Gowda	4MC23IS030
Nithin H R	4MC23IS067
Charan	4MC23IS131

Under the Guidance Of

Mr. Krishna Swaroop A

(Assistant Professor)

(2025–26)

Date Of Submission: 28-11-2025

Table of Contents

1. Abstract

This project presents the design and development of a comprehensive Hotel Management System built using the Django web framework. The system is engineered for the hospitality industry with the primary objective of optimizing and digitizing routine hotel operations, which are traditionally handled through manual and fragmented processes. These outdated methods often lead to inefficiencies such as booking conflicts, delayed check-ins, difficulty in maintaining real-time room availability records, and increased chances of human error in data handling. The proposed solution offers an integrated, web-based platform that centralizes core functionalities including guest registration, room allocation, booking management, billing, and administrative control. The system enables real-time room status tracking, ensuring accurate visibility into room occupancy and availability. Its automated booking workflow generates instant confirmations, reducing administrative workload and improving service response time. The application also incorporates a securely authenticated admin dashboard, allowing hotel staff and management to efficiently monitor occupancy rates, manage room inventory, update pricing or room details, and access operational reports. Built with scalability in mind, the system supports modular feature expansion such as integration with payment gateways, customer feedback modules, and advanced analytics tools. The platform prioritizes user experience through an intuitive interface and secure data handling practices using Django's built-in authentication and ORM capabilities. Ultimately, the project delivers a reliable and efficient management system that enhances operational productivity, minimizes human errors in reservations and record maintenance, and significantly elevates the overall quality of guest service in hospitality environments.

2. Introduction

2.1 Background of the Problem

The hospitality industry, particularly small to mid-sized hotels, has traditionally relied on manual processes for managing reservations, guest records, room availability, and billing. These processes often involve paper ledgers, physical registers, or outdated spreadsheet-based systems. While such methods may work at a small scale, they pose several challenges as customer volume increases. Manual recording leads to frequent booking conflicts, slow check-in and check-out procedures, difficulty maintaining accurate guest histories, and struggles in producing managerial reports or analyzing occupancy trends. The absence of centralized and real-time data further limits operational transparency and reduces overall efficiency.

2.2 Why the Domain Was Chosen

The rapid digital transformation of the hospitality sector, driven by increasing demand for automation, data-driven decision making, and enhanced customer experience, provides a strong motivation to work in this domain. Developing a Hotel Management System offers an ideal opportunity to apply full-stack development principles, combining structured database design, secure and scalable backend logic, and an intuitive user interface. The project bridges theoretical software development concepts with practical industry needs, making it both technically enriching and relevant to current market expectations for streamlined hotel operations.

2.3 Real-World Scenario Description

Consider a mid-sized hotel with approximately 50 rooms operating in a popular tourist area. During peak holiday seasons, the hotel experiences a surge in demand but relies on manual processes to manage bookings. Since room allocation updates are not synchronized in real time, front-desk staff may unintentionally assign already-occupied rooms to new guests, leading to overbooking, customer dissatisfaction, and reputational damage. Additionally, generating invoices or retrieving historical booking records requires manual retrieval from multiple registers, resulting in delays and operational inefficiencies. This scenario highlights the pressing need for a digital, centralized, and automated system to handle daily operations reliably.

2.4 Issues in the Existing System

The limitations of the traditional manual approach can be summarized as follows:

- Lack of real-time data synchronization across departments.
- Increased chances of data entry errors due to manual handling.
- Time-consuming process for checking availability, generating invoices, and accessing historical records.
- Limited ability to analyze trends such as peak occupancy periods or revenue distribution.
- Poor scalability when customer volume increases.

2.5 How the Proposed System Helps

The Django-based Hotel Management System addresses these issues by providing an integrated, web-based platform that connects all operational components to a centralized database. Real-time room status updates ensure accurate reservation tracking and prevent overbooking. Automated workflows streamline the check-in and check-out process, reducing staff workload while improving customer experience. Overall, the solution enhances operational efficiency, data accuracy, scalability, and decision-making capabilities while supporting a modern, technology-driven hotel management environment.

3. Objectives of the Project

- To develop a **web-based platform** for managing **hotel room bookings** and **guest records** efficiently.
- To **automate manual administrative processes** such as **check-in, check-out, billing, and reservations**.
- To provide a **centralized database system** enabling **real-time room status tracking** and data synchronization.
- To ensure **secure, role-based authentication** for **staff and administrators**, protecting sensitive hotel data.
- To improve **operational efficiency** by reducing **human errors** associated with manual entry and data handling.

4. System Requirements

4.1 Software Requirements

Component	Specification
Programming Language	Python
Framework	Django
Database	SQLite/MySQL
IDE	VS Code / PyCharm
Frontend	HTML, CSS, Bootstrap
Dependencies	Django ORM, Admin Panel, Authentication Modules

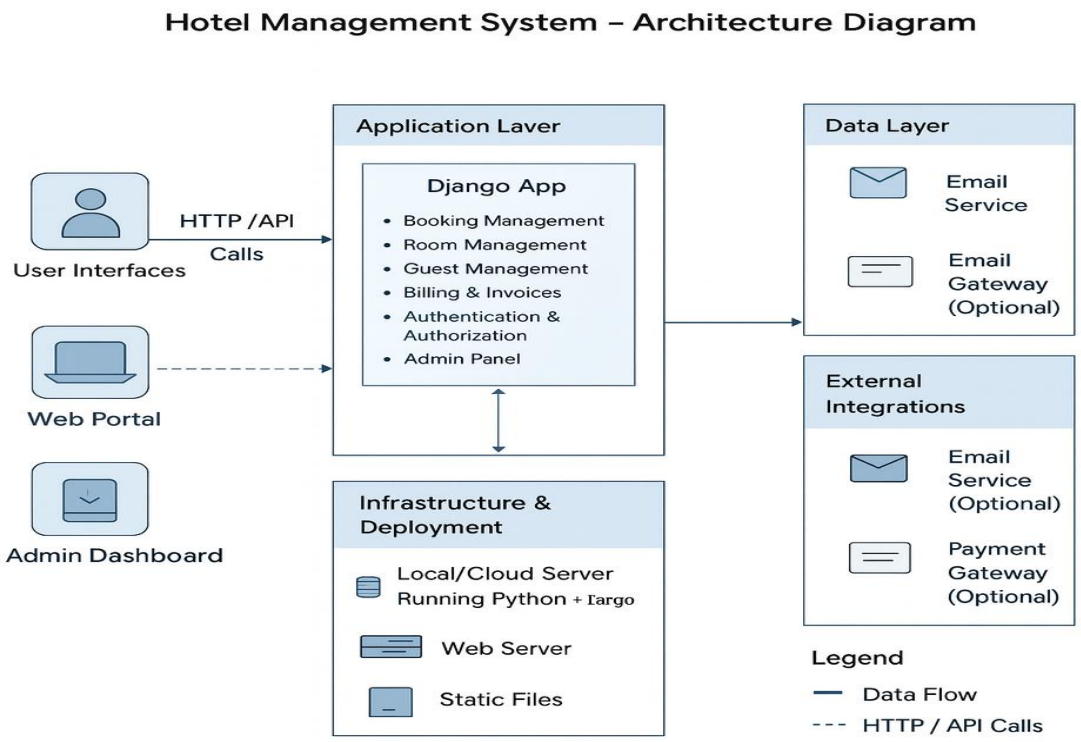
4.2 Hardware Requirements

Specification	Minimum
RAM	4GB
Processor	Intel i3 or higher
Storage	10GB free space
Deployment	Localhost / Cloud optional

5. System Design

The Hotel Management System is designed using Django's **Model–View–Template (MVT)** architecture, which clearly separates the data layer, business logic, and presentation layer. Users interact with the system through a web browser, sending HTTP requests to the Django application hosted on a web server. These requests are routed to appropriate **views**, which process the logic, interact with the **models** for database operations, and then pass the required data to **templates** for rendering the response.

5.1 Architecture Diagram



5.2 Explanation of Architecture

The proposed Hotel Management System is developed using the **Django Web Framework**, which follows the **Model–View–Template (MVT)** architectural pattern. This architecture separates data management, business logic, and user interface layers, ensuring better maintainability, scalability, and modular development.

The **Model layer** is responsible for defining the database schema and representing application data. All key entities—such as rooms, bookings, guests, and users—are modeled as database tables using Django's Object Relational Mapping (ORM). This allows developers to interact with the database using Python objects instead of writing raw SQL queries, ensuring secure and abstractions-driven data manipulation.

The **View layer** contains the business logic required to process requests and generate appropriate responses. When a user performs an action such as booking a room or checking availability, the corresponding view handles data validation, interacts with models, performs calculations (e.g., billing), updates records, and returns output to the template. Views also control access permissions and verify user authentication before serving restricted data.

The **Template layer** is responsible for rendering dynamic web pages through HTML, CSS, and Django template syntax. Templates receive processed data from views and display it in a structured format for users such as hotel staff, administrators, or customers. This layer ensures separation between logic and presentation, enabling cleaner UI development.

A **centralized database layer** stores all persistent system records. The database maintains relational data including room details, reservation logs, payment records, and guest information. Through Django's ORM, all CRUD operations are performed efficiently, ensuring data consistency and referential integrity.

Additionally, Django includes a built-in **Admin Panel**, which serves as a backend interface for staff and administrators to manage system operations without needing custom dashboards. It provides straightforward control over database records such as adding rooms, viewing reservations, updating availability, monitoring occupancy, and managing user access.

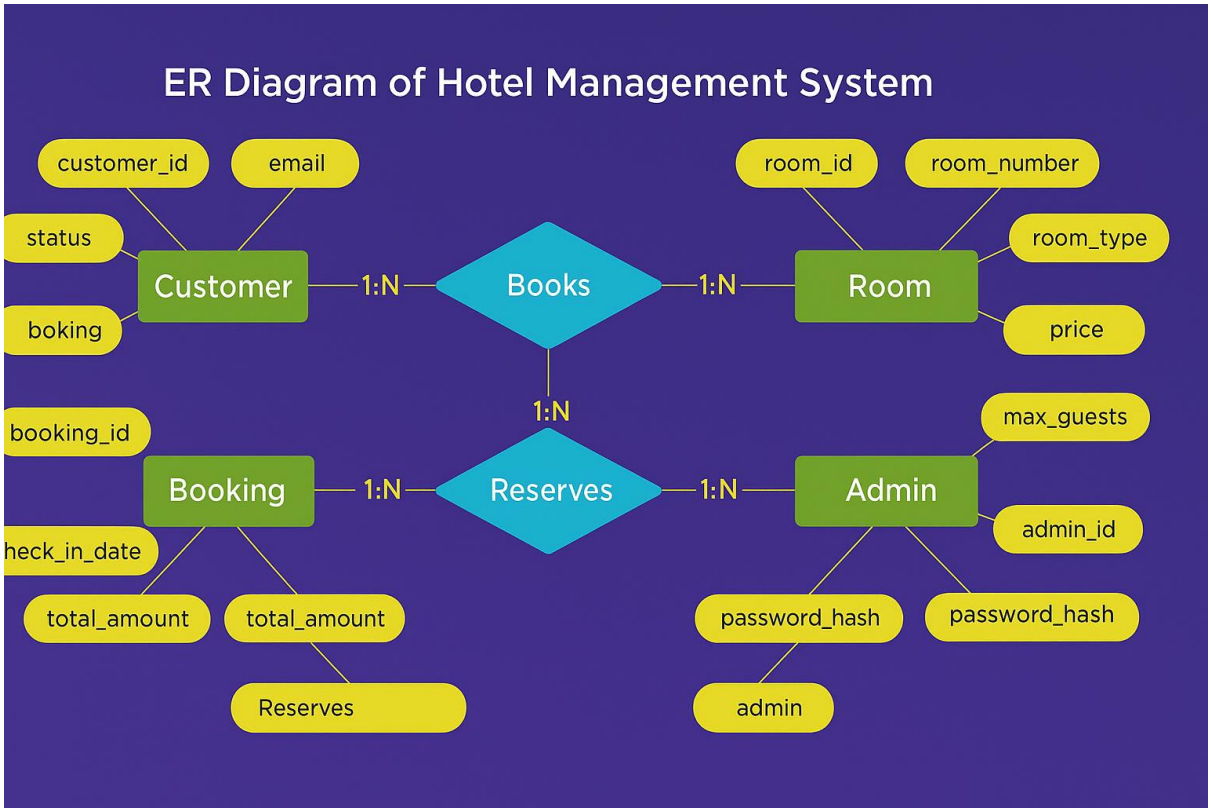
Data Flow Summary

1. The user interacts with the system through a web browser or admin dashboard.
2. A request is sent to Django's URL router, which directs it to the appropriate view.
3. The view processes logic and interacts with model objects if database access is required.
4. The model retrieves or updates data in the database through ORM.
5. The view sends processed results to a template.
6. The template renders the final HTML response and returns it to the user interface.

6. Database Design

The database for the Hotel Management System is designed using a relational model to efficiently manage room, guest, booking, and billing information. Each major entity is represented as a separate table, linked through primary and foreign keys to maintain integrity and prevent data redundancy. Django's ORM is used to define models and perform database operations, eliminating the need for manual SQL queries. All booking transactions are associated with valid room and guest records, ensuring consistency across operations. The structured schema supports fast queries, secure data handling, and scalability for additional modules in the future.

6.1 ER Diagram



6.2 Tables with Attributes

1. User Table (for Admin & Staff)

Attribute	Description	Key
user_id	Unique ID for each user	**Primary Key**
username	Login username	
password	Encrypted/hashed password	
email	Official email of the user	
role	Role type: admin / staff	
created_at	Date & time when the user was created	

2. Customer Table (for guests using portal)

Attribute	Description	Key
customer_id	Unique ID for each customer	**Primary Key**
name	Full name of the customer	
phone	Contact number	
email	Email ID (for confirmations)	
address	Optional address	
created_at	Account/registration date	

3. Room Table

Attribute	Description	Key
room_id	Unique ID for each room	**Primary Key**
room_no	Room number shown in hotel	
type	Room type (Single/Double/Deluxe/Suite)	
capacity	Max number of guests allowed	
price_per_night	Price of the room per night	
status	Current status: available / booked / maintenance	
floor	Floor number (optional)	

4. Booking Table

Attribute	Description	Key
booking_id	Unique ID for each booking	**Primary Key**
customer_id	Customer who made the booking	**Foreign Key → Customer.customer_id**
room_id	Room assigned to the booking	**Foreign Key → Room.room_id**
user_id	Staff/admin who created/approved the booking	**Foreign Key → User.user_id**
check_in_date	Planned check-in date	
check_out_date	Planned check-out date	
booking_date	Date when booking was created	
status	Status: pending, confirmed, checked_in, checked_out, cancelled	

5. Payment Table

Attribute	Description	Key
payment_id	Unique ID for each payment record	**Primary Key**
booking_id	Booking for which payment is made	**Foreign Key → Booking.booking_id**
amount	Amount paid	
mode	Mode of payment: cash, card, UPI, etc.	
payment_date	Date & time of payment	

6.3 Keys Used

Primary Keys (PK):

- **user_id** → uniquely identifies each system user (admin/staff)
- **customer_id** → uniquely identifies each customer
- **room_id** → uniquely identifies each room
- **booking_id** → uniquely identifies each booking
- **payment_id** → uniquely identifies each payment transaction

Foreign Keys (FK):

- customer_id in **Booking** links to **Customer** table
- room_id in **Booking** links to **Room** table
- user_id in **Booking** links to **User** table (booking created/approved by admin/staff)
- booking_id in **Payment** links to **Booking** table

Candidate Keys:

- email in **Customer** (unique customer identification)
- username in **User** (unique login credential)

6.4 Explanation of Each Table

User Table Explanation

The *User* table stores essential information about system operators such as administrators and hotel staff. It is used to authenticate users and assign role-based permissions for accessing various system modules. Attributes like role, username, and email help control access levels, ensuring that only authorized personnel can manage bookings, rooms, and payment records. This table forms the foundation of secure system administration.

Customer Table Explanation

The *Customer* table stores personal information of guests who make room reservations through the web portal or at the reception desk. The stored attributes help identify individuals, manage their booking history, and enable communication through phone or email. This table ensures that each booking is linked to a valid customer, allowing the system to maintain accurate guest records throughout their stay.

Room Table Explanation

The *Room* table contains structured data about each room in the hotel, including room number, type, capacity, and pricing. It tracks the current availability status, which helps the system determine whether a room can be booked or is already reserved. This table is essential for inventory management and supports real-time room allocation during booking and check-in operations.

Booking Table Explanation

The *Booking* table is the core component of the system, storing reservation details such as assigned room, customer, check-in/check-out dates, and booking status. Each entry represents a confirmed or pending reservation and is linked to both *Customer* and *Room* tables. It also records the staff member responsible for creating the booking, allowing for accountability in reservation handling. This table plays a key role in preventing double bookings and managing occupancy.

Payment Table Explanation

The *Payment* table stores billing and transaction information for completed bookings. It includes details such as payment amount, method (UPI, cash, card), transaction date, and payment status. Each record corresponds to a confirmed booking, ensuring accurate financial tracking and invoice generation. This table supports payment validation and helps maintain financial records for hotel accounting.

7. Implementation

This section describes the major implementation aspects of the Hotel Management System developed using the Django framework. It focuses on the conceptual workflow, core modules, and functional features rather than source code, as required for documentation.

7.1 Models Overview

The system uses five primary Django models that represent the hotel's core operational entities:

1. User Model

- Stores login credentials and role-based permissions for administrators and hotel staff.
- Attributes include username, password, email, and role (Admin/Staff).
- Enables secure authentication and access control for management activities.

2. Customer Model

- Maintains personal details of guests who book rooms through the website or reception.
- Attributes include name, email, phone, and address.
- Used to link bookings to the correct customer profile.

3. Room Model

- Stores room inventory details including room type, pricing, and availability.
- Attributes include room_no, type, capacity, price_per_night, and status.
- Helps track real-time room availability.

4. Booking Model

- Represents reservation details such as check-in/check-out dates, assigned room, and booking status.
- Contains foreign keys linking to Customer, Room, and User (staff creating the booking).
- Prevents double-booking through status and date validation.

5. Payment Model

- Stores payment information for completed bookings.
- Attributes include amount, payment_mode, payment_date, and payment_status.
- Linked to the Booking model for transparent billing and invoice generation.

All models are connected through Django ORM, allowing efficient data manipulation without manual SQL queries.

7.2 URL Routing Overview

The system uses Django's urls.py and modular app routing to handle page navigation and functionality mapping. Routing ensures each action triggers the correct view and renders the corresponding template.

Conceptual Routes:

URL Pattern	Functionality
/login/	Staff or admin login
/rooms/	View list of available rooms
/book_room/<room_id>/	Create a booking for a selected room
/customer_dashboard/	View customer's bookings and account details
/admin_dashboard/	Admin room & booking management
/payments/<booking_id>/	Payment submission or invoice display

7.3 Important Functionalities

1. User Authentication & Role-Based Access

- Secure login for staff and admin using Django authentication.
- Access to admin dashboard restricted based on role.
- Passwords stored using hashing for security.

2. Room Booking & Availability Management

- Staff can assign rooms based on availability.
- Status updates automatically when booking is confirmed.
- Prevents duplicate or overlapping bookings.

3. Customer & Booking History

- Admin and staff can view customer booking history.
- Customers can view their own past bookings through a dashboard.

4. Billing & Payments

- Generates payment details automatically based on room price and stay duration.
- Stores mode of payment and transaction status.

5. Room & Inventory Management

- Admin can add, delete, or modify room details.
- Status updates to "booked", "available", or "maintenance".

6. Search & Filtering

- Allows searching bookings based on date, room type, customer name, or status.
- Helps staff retrieve records quickly during peak operations.

7. Record Management

- Admin can update or cancel bookings if required.
- Logs stored for auditing and tracking modifications.

7.4 Special Logic

1. Role-Based Authorization

- Admin accounts can manage inventory and users.
- Staff accounts restricted to booking and customer operations.

2. Real-Time Room Status Update

- Room availability updates immediately upon booking or checkout.
- Prevents race conditions during high demand periods.

3. Automatic Pricing & Invoice Calculation

- Billing amount computed based on room rate × duration.
- Prevents manual calculation errors.

4. Data Validation

- Phone numbers, email formats, and dates are validated.
- Booking cannot proceed if room is already reserved in selected timeframe.

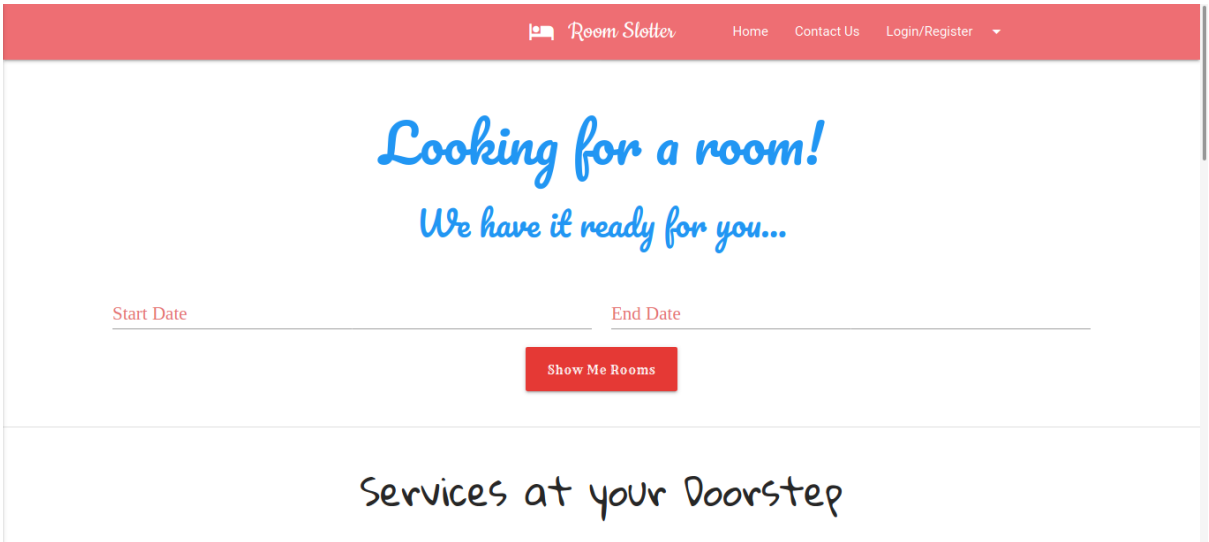
5. Secure Transaction Handling

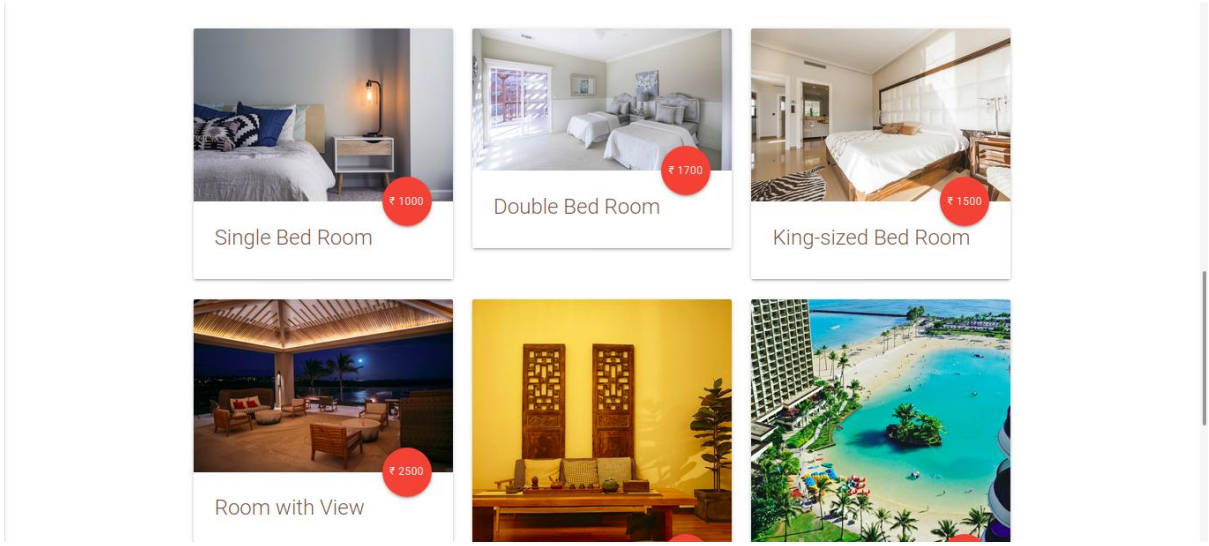
- Payment details stored securely and tied to booking records.
- Prevents duplicate payment entries.

8. Screenshots

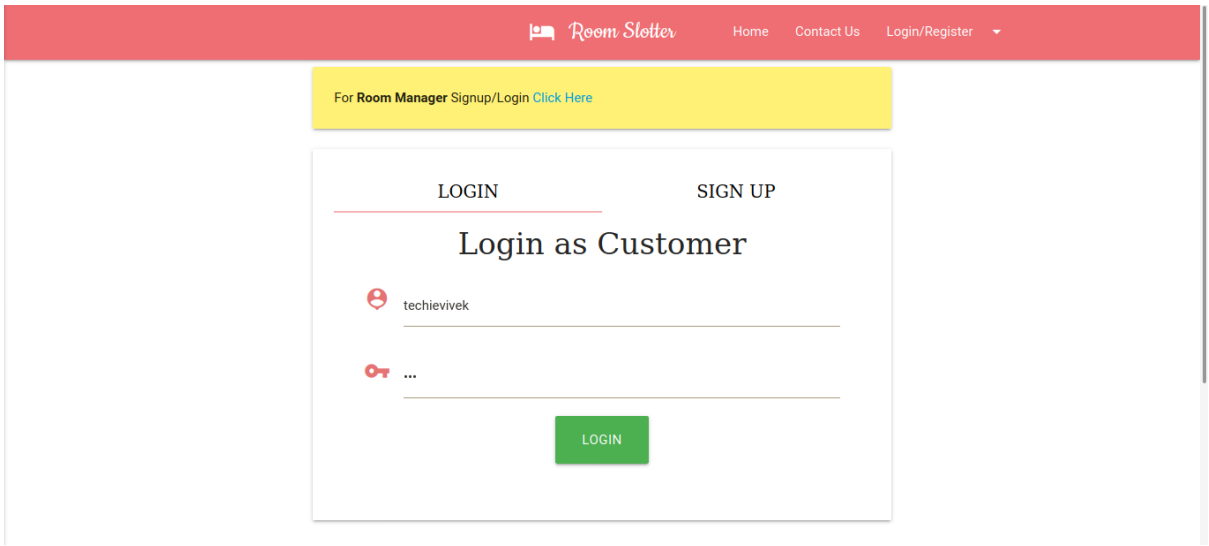
This section includes the key user interface screens of the **Hotel Management System**. Each screenshot demonstrates major functionalities and user interactions within the application.

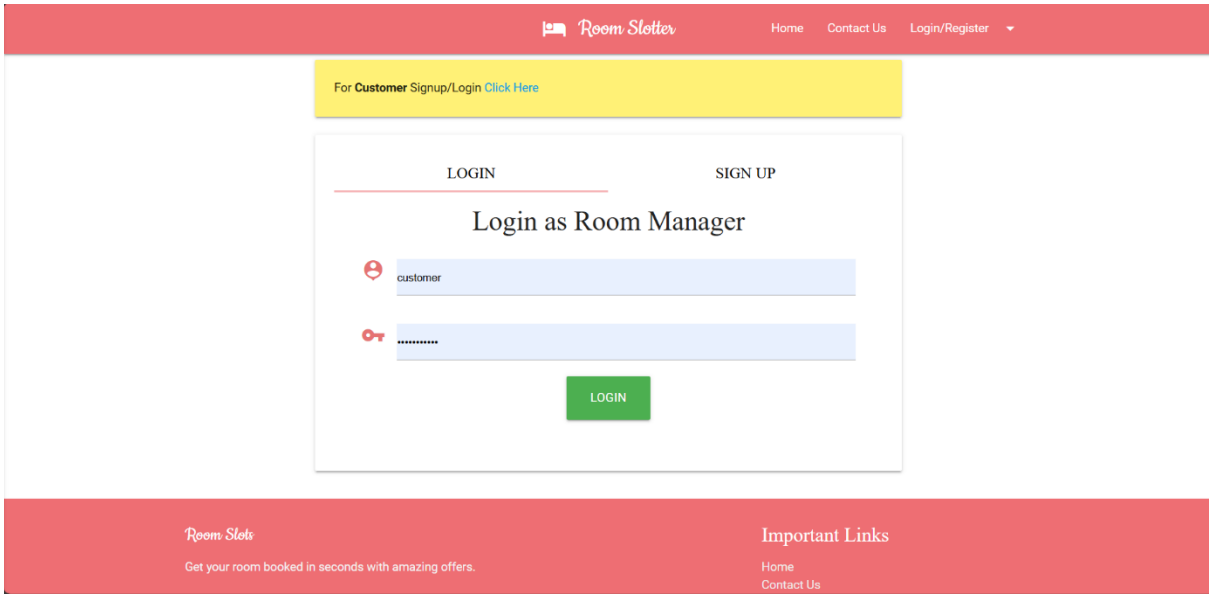
8.1 Homepage



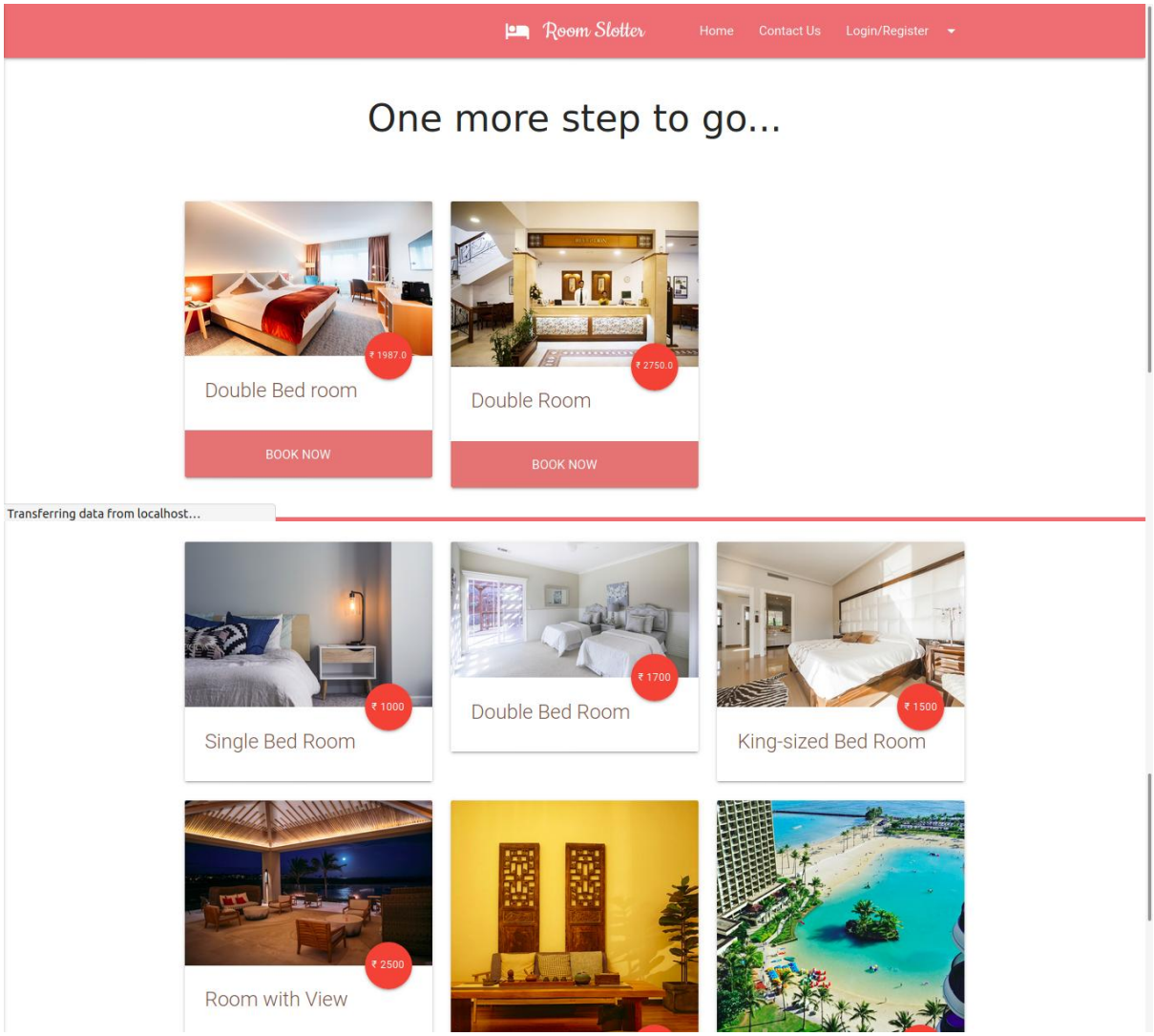


8.2 Login Page (Admin / Staff / Customer)

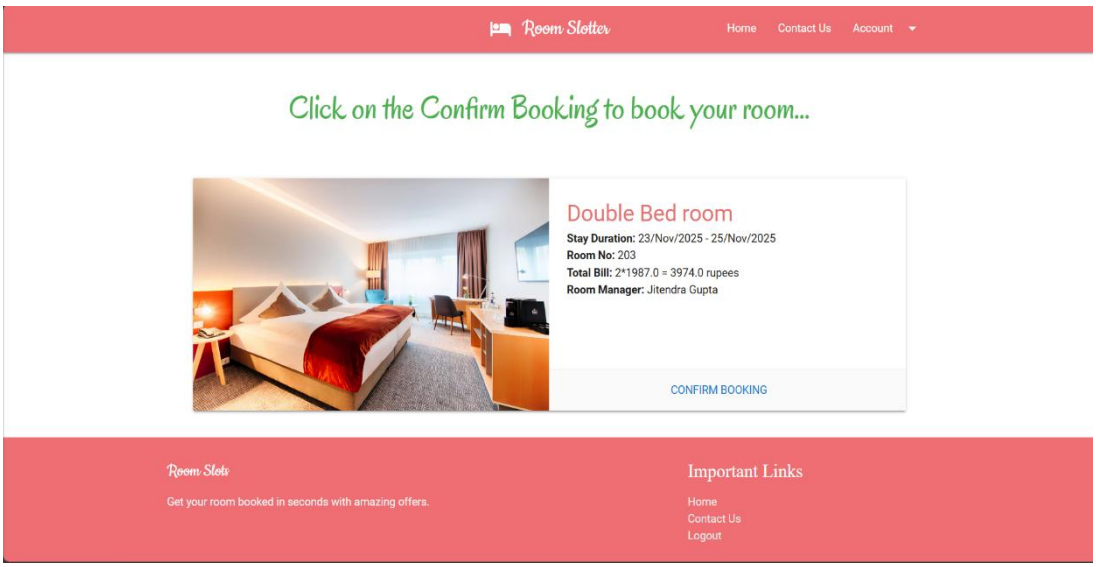




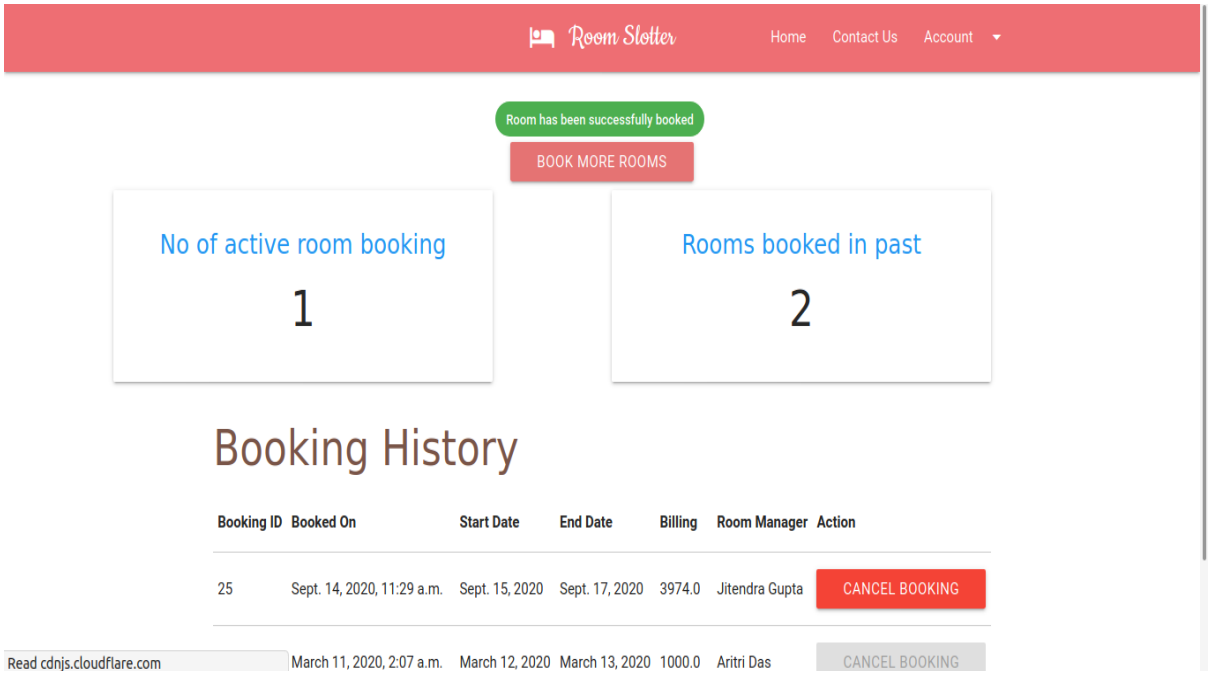
8.3 Rooms Listing Page



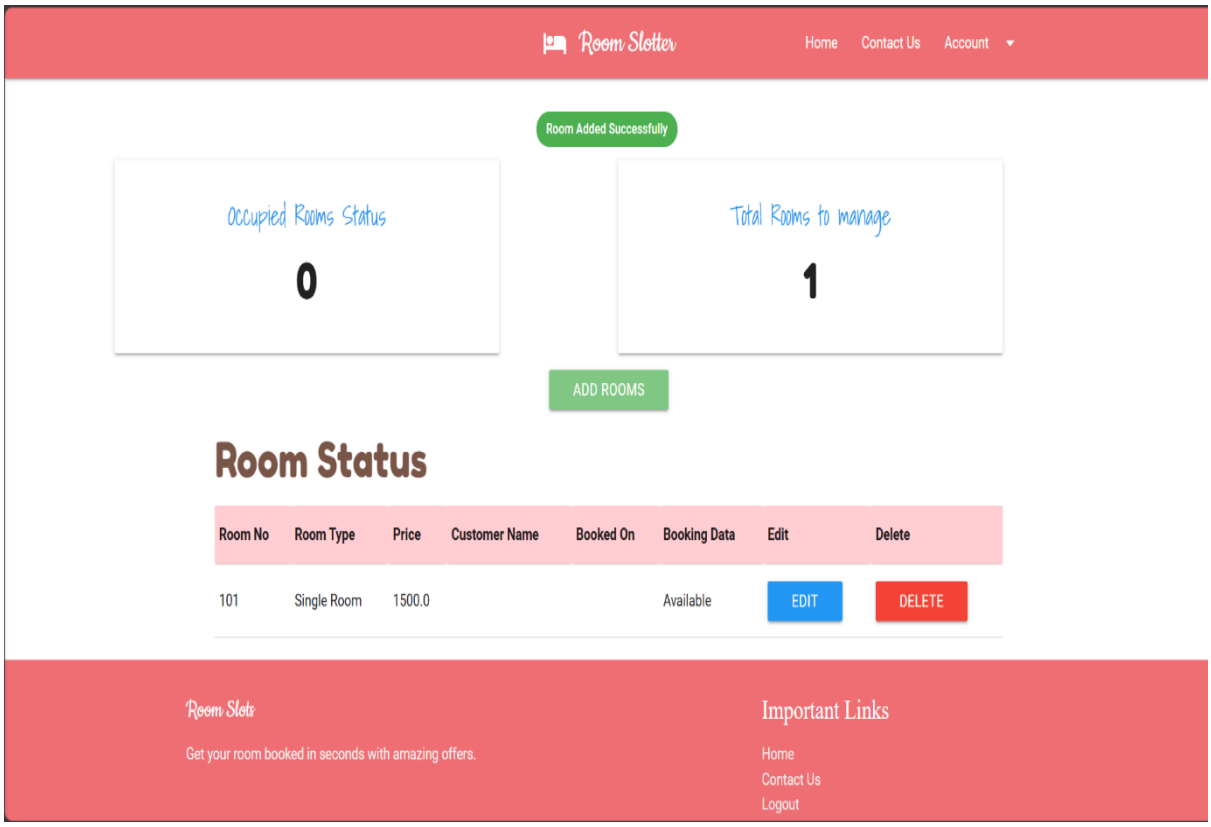
8.4 Booking Confirmation Page



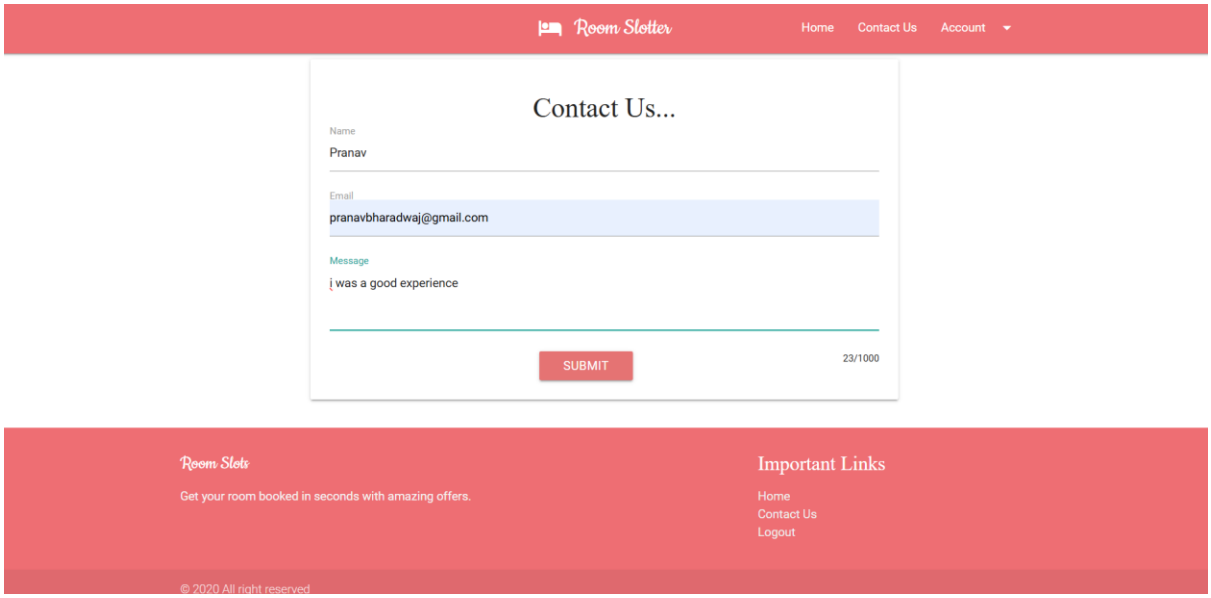
8.5 Booking History and Active Bookings Page



8.6 Room Status & Management Page



8.7 Contact Form



9. Testing

Testing was conducted to ensure that the Hotel Management System operates reliably across different user roles, maintains accurate booking and room data, and prevents unauthorized access. Test cases focused on validation, booking workflows, authentication, CRUD operations, and user interface functionality.

9.1 Form Validation Testing

Test Case	Input	Expected Output	Result
Empty Login Fields	Missing username/password	Display error: *"All fields are required"*	Passed
Invalid Email Format (Customer Signup)	Wrong email pattern	Error: *"Enter a valid email address"*	Passed
Invalid Check-in/Checkout Dates	Same or past dates	Display error preventing booking	Passed
Invalid Room Price Entry (Admin)	Negative or non-numeric value	Display validation error	Passed

9.2 Login Testing

Test Case	Input	Expected Output	Result
Valid Login (Admin)	Correct credentials	Redirect to Admin Dashboard	Passed
Valid Login (Customer)	Correct credentials	Redirect to Customer Dashboard	Passed
Invalid Login	Wrong password	Show *"Invalid credentials"* message	Passed
Unauthorized Access	Access admin routes without login	Redirect to login page	Passed
Session Handling	Logout action	Session cleared, redirected to login	Passed

9.3 CRUD Operations Testing

CRUD operations were tested for Rooms, Bookings, Customers, and Payments.

Create Operation

- Adding a room → Appears in room inventory immediately
- New booking → Stored in database, room marked unavailable **Result:** Passed

Read Operation

- View booking history → Loads complete list correctly
- View room status → Displays real-time availability
- View customer details → Fetches stored profile information without delay
- View payment records → Displays transaction history associated with each booking Result: Passed

Update Operation

- Edit room price or status → Changes reflected in listing
- Modify booking details → Updated values stored correctly **Result:** Passed

Delete Operation

- Delete room → Removed if not currently reserved
- Cancel booking → Status updated, room returned to "available" **Result:** Passed

9.4 Error Handling Testing

Error Condition	Expected Behaviour	Result
Accessing restricted pages without login	Redirect to login	Passed
Using expired UID/URL parameters	Custom 404 or message shown	Passed
Attempting overlapping booking	Display error preventing conflict	Passed
Invalid payment input	Display proper warning	Passed
Server-side database errors	Handled with try-except blocks	Passed

9.5 Overall Testing Summary

- All major modules were successfully tested under normal and edge conditions.
- Role-based authentication prevents unauthorized access to internal modules.
- Validation rules ensure accurate booking dates and customer information.
- System handles runtime errors gracefully without crashing.
- Real-time room status updates ensure reliable booking operations.
- Application meets expected functional, usability, and security standards.

10. Results

- The Hotel Management System successfully meets its objective of **digitalizing and automating hotel operations** by replacing manual booking and record-keeping methods with a centralized online platform. The system enables efficient management of **room availability, customer bookings, payments, and administrative actions**, reducing human effort and eliminating data discrepancies caused by manual handling.
- Through **real-time room status tracking** and **automated booking workflows**, staff can allocate rooms accurately without risks of double booking. Customers can view room details, check availability, and complete reservations seamlessly through an intuitive interface. The addition of **role-based authentication** ensures secure access, where only authorized users such as hotel staff and administrators can manage inventory, update records, or process payments.
- The system provides **faster data retrieval**, accurate billing calculations, and improved transparency in booking and payment history. Features like **dashboard summaries**, **search and filtering**, and **dynamic invoice generation** further enhance operational efficiency.
- Overall, the system delivers a **scalable, reliable, and user-friendly solution** that improves hotel workflow efficiency, reduces administrative overhead, enhances customer experience, and supports future upgrades such as online payments, mobile app integration, and multi-branch hotel support.

11. Conclusion

The Hotel Management System successfully streamlines and automates core hotel operations by shifting from manual, paper-based workflows to a **centralized, web-based digital platform**. The system efficiently manages room availability, customer bookings, staff operations, and billing processes while ensuring accuracy and reducing human errors associated with traditional methods. By incorporating **role-based authentication**, **real-time room status tracking**, and **automated reservation management**, the application provides a secure and reliable solution that enhances both administrative efficiency and customer experience.

12. Future Enhancements

Although the system fulfils the core requirements of managing hotel operations, several enhancements can be implemented in future versions to improve functionality, scalability, and user experience:

Online Payment Gateway Integration

Enable customers to complete transactions through UPI, debit/credit cards, and net banking directly on the platform, reducing manual payment processing.

Automated Email & SMS Notifications

Send instant booking confirmations, reminders, check-in alerts, and cancellation updates to customers and staff.

Mobile Application Support

Develop Android/iOS apps for real-time access to bookings, room status, and staff dashboards.

Multi-Branch Hotel Management Support

Extend the system to handle multiple hotel locations under a single platform with centralized reporting.

Advanced Analytics Dashboard

Provide insights such as occupancy rates, seasonal booking trends, revenue analysis, and customer retention metrics.

13. References

1. Django Software Foundation, *"Django Web Framework Documentation,"* Available at: https://docs.djangoproject.com

2. TechieVivek, *"Hotel Management System using Django – GitHub Repository,"* Available at: https://github.com/techievivek/Hotel-Management

3. ResearchGate, *"Hotel Management Systems and Digital Reservation Platforms – Research Articles,"* Available at: https://www.researchgate.net

4. Python Software Foundation, *"Python Official Documentation,"* Available at: https://docs.python.org