**[Hotel Booking System]**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology/Master of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**Neha Kujur**

**(AP23110010270)**

**Ayush Kumar**

**(AP23110010278)**

**Shivangi Dubey**

**(AP23110010306)**

**Jagrat Patel**

**(AP23110010338)**

**A picture containing text

Description automatically generated**

Under the Guidance of

**(Prof. Karnena Kavitha Rani)**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[November, 2024]**

# Certificate

Date: 16-Nov-22

This is to certify that the work present in this Project entitled “**Hotel Booking System**” has been carried out by **Neha Kujur,Ayush Kumar,Shivangi Dubey,Jagrat Patel** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

**Supervisor**

(Signature)

Prof. Karnena Kavitha Rani

Assistant Professor,

Department of Computer Science.

**Acknowledgements**

We express our heartfelt gratitude to **Prof. Karnena Kavitha Rani** for her invaluable guidance and support throughout the development of this project. Her expertise and encouragement have been instrumental in the successful completion of our hotel booking system.

We also extend our thanks to our team members, **Neha Kujur, Ayush Kumar, Shivangi Dubey, Jagrat Patel**, for their dedication, collaboration, and hard work in bringing this project to fruition. Finally, we are grateful to our university, SRM University AP, for providing us with the resources and platform to enhance our skills and showcase our learning.

Table of Contents

[Certificate 1](#_Toc182907175)

Acknowledgement…………………………………………………………………………..2

Table of Contents………..…………………………………………………………………..3

Abstract……………………………………………………………………………………….4

Statement of Contribution…………………………………………………………………..5

[1. Neha Kujur: 5](#_Toc182907176)

[2. Ayush Kumar: 5](#_Toc182907177)

[3. Shivangi Dubey: 5](#_Toc182907178)

[4. Jagrat Patel: 5](#_Toc182907179)

[List of Tables 6](#_Toc182907180)

[List of Figures 9](#_Toc182907181)

[Introduction 10](#_Toc182907182)

[Methodology 11](#_Toc182907196)

[1. User Input and Room Booking 11](#_Toc182907197)

[1.11Prompt the User: 11](#_Toc182907198)

[Ask for the type of room (roomType) and the number of guests (numGuests). 11](#_Toc182907199)

[1.12. Calculate Rooms Needed: 11](#_Toc182907200)

[1.13. Gather Additional Information: 11](#_Toc182907201)

[1.14. Capture Guest Names: 11](#_Toc182907202)

[1.15. Book the Room: 11](#_Toc182907203)

[2. Displaying Customer Details 11](#_Toc182907204)

[3. Checking Offers for a Specific Customer 11](#_Toc182907205)

[4. Exit Option 12](#_Toc182907206)

[5. Error Handling 12](#_Toc182907207)

[Discussion 13](#_Toc182907208)

[1. Functionality Overview 13](#_Toc182907209)

[2. Strengths 13](#_Toc182907210)

[2.1. Modular Approach: 13](#_Toc182907211)

[2.2. Scalability: 13](#_Toc182907212)

[2.3. Data Handling: 13](#_Toc182907213)

[3. Limitations and Suggestions for Improvement 13](#_Toc182907214)

[3.1. Validation of Inputs: 13](#_Toc182907215)

[3.2. Data Storage: 14](#_Toc182907216)

[3.3. Error Handling: 14](#_Toc182907217)

[3.4. User Experience: 14](#_Toc182907218)

[3.5. Security Concerns: 14](#_Toc182907219)

[4. Future Enhancements 14](#_Toc182907220)

[Concluding Remarks 15](#_Toc182907221)

[Strengths: 15](#_Toc182907222)

[1. Core Functionality: 15](#_Toc182907223)

[2. User-Friendly Input Collection: 15](#_Toc182907224)

[3. Error Messages for Invalid Choices: 15](#_Toc182907225)

[Limitations and Opportunities for Improvement: 15](#_Toc182907226)

[1. Input Validation: 15](#_Toc182907227)

[2. Persistent Data Storage: 15](#_Toc182907228)

[3. Error Handling: 15](#_Toc182907229)

[4. Scalability Issues: 15](#_Toc182907230)

[5. Sensitive Data Security: 16](#_Toc182907231)

[Future Enhancements: 16](#_Toc182907232)

[1. Advanced Features: 16](#_Toc182907233)

[2. Optimized User Experience: 16](#_Toc182907234)

[3. Improved Code Structure: 16](#_Toc182907235)

[4. Data Analytics and Reporting: 16](#_Toc182907236)

[5. Multi-User Support: 16](#_Toc182907237)

[Future Work 17](#_Toc182907238)

[1. Advanced Features 17](#_Toc182907239)

[2. Enhanced Data Management 17](#_Toc182907240)

[3. Improved Security 18](#_Toc182907241)

[4. User Interface and Experience 18](#_Toc182907242)

[5. Scalability 18](#_Toc182907243)

[6. Analytics and Reporting 19](#_Toc182907244)

[References 20](#_Toc182907245)

**Abstract**

- The hotel booking system is designed to streamline the reservation process for customers and enhance operational efficiency for hotel administrators.

- Features include the ability to choose room types, specify guest details, and book stays for varying durations, with options for multiple rooms based on the number of guests.

- The system securely collects and manages personal information such as phone numbers, identification details, and addresses to maintain accurate records.

- A user-friendly interface ensures seamless interactions, reducing errors and minimizing manual effort for hotel staff.

- Additional functionalities include personalized customer offers based on unique IDs, ensuring tailored services for frequent guests.

- The project showcases modern programming practices and database management techniques to provide a scalable, efficient, and adaptable solution for hotels of different sizes and customer bases.

- The system aims to improve customer satisfaction and operational reliability in the hospitality industry.

Statement of Contribution:

In the development of the Hotel Management System, our team collaboratively designed and implemented the various functionalities that make the system efficient and user-friendly.

## 1. Neha Kujur:

Neha took the lead in developing the Customer class, focusing on the management of customer information, including handling loyalty points and tracking booking details.

She ensured that the customer-related features such as adding guests, updating loyalty points, and managing booking dates were efficiently handled.

## 2. Ayush Kumar:

Ayush was responsible for implementing the Room and DeluxeRoom classes, focusing on managing room attributes, availability, and calculating the billing system for both standard and deluxe rooms.

He also worked on the polymorphic behavior of rooms, ensuring that both room types (Standard and Deluxe) had their own rate calculation logic.

## 3. Shivangi Dubey:

Shivangi contributed to the implementation of the Hotel class, ensuring that room booking logic was accurate, including room availability checks, booking validation, and bill calculation.

She was responsible for integrating customer interactions with room availability, booking, and display functionalities, ensuring smooth operation of the system.

### 4. Jagrat Patel:

Jagrat played a crucial role in implementing the user interface logic in the main function, providing an intuitive experience for users to interact with the system.

He focused on handling user input, menu navigation, and managing different options for room booking, viewing rooms, and customer offers. Additionally, he integrated the error-handling logic to ensure smooth system operation.

Together, our team successfully built a modular and functional hotel management system, providing a smooth user experience while managing room bookings, customer data, and offers effectively.

# List of Tables

**Table 1 – Customer Information**

| **S.No** | **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | Customer\_ID | Varchar(50) | Primary Key | Contains a unique ID for each customer. |
| 2 | Name | Varchar(50) | - | Stores the name of the customer. |
| 3 | Phone\_Number | Varchar(15) | - | Stores the contact number of the customer. |
| 4 | Aadhar\_Number | Varchar(12) | - | Stores the Aadhar card number of the customer. |
| 5 | Address | Text | - | Stores the address of the customer. |

**Table 2 – Room Details**

| **S.No** | **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | Room\_ID | Varchar(50) | Primary Key | Contains a unique ID for each room. |
| 2 | Room\_Type | Varchar(20) | - | Indicates the type of the room (Standard/Deluxe). |
| 3 | Availability | Boolean | - | Indicates if the room is available. |
| 4 | Price\_per\_Night | Float | - | Stores the price per night for the room. |

**Table 3 – Booking Details**

| **S.No** | **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | Booking\_ID | Varchar(50) | Primary Key | Contains a unique ID for each booking. |
| 2 | Customer\_ID | Varchar(50) | Foreign Key | Links to the Customer\_ID in the Customer table. |
| 3 | Room\_ID | Varchar(50) | Foreign Key | Links to the Room\_ID in the Room table. |
| 4 | Guests | Integer | - | Stores the number of guests. |
| 5 | Rooms\_Needed | Integer | - | Indicates the number of rooms booked. |
| 6 | Nights | Integer | - | Indicates the number of nights booked. |
| 7 | Check\_in\_Date | Date | - | Stores the check-in date. |
| 8 | Check\_out\_Date | Date | - | Stores the check-out date. |

**Table 4 – Guest Names**

| **S.No** | **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | Guest\_ID | Varchar(50) | Primary Key | Contains a unique ID for each guest. |
| 2 | Booking\_ID | Varchar(50) | Foreign Key | Links to the Booking\_ID in the Booking table. |
| 3 | Guest\_Name | Varchar(50) | - | Stores the name of each guest. |

**Table 5 – Offers and Discounts**

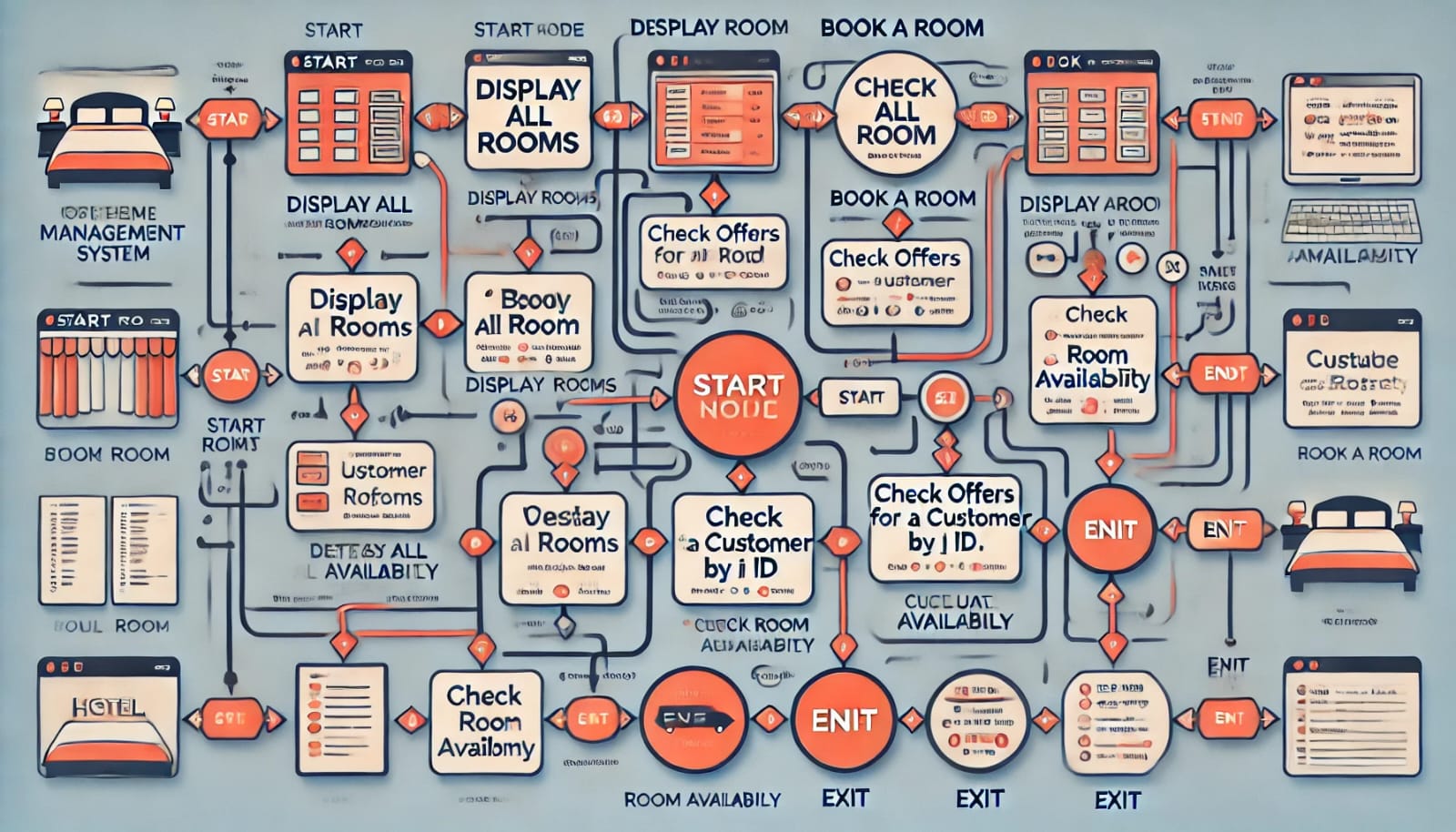
| **S.No** | **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | Offer\_ID | Varchar(50) | Primary Key | Contains a unique ID for each offer. |
| 2 | Customer\_ID | Varchar(50) | Foreign Key | Links to the Customer\_ID in the Customer table. |
| 3 | Discount\_Percent | Float | - | Stores the percentage of the discount. |
| 4 | Offer\_Description | Text | - | Describes the offer. |
| 5 | Validity\_Period | Date | - | Stores the expiration date of the offer. |

**Table 6 – Transaction Records**

| **S.No** | **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| 1 | Transaction\_ID | Varchar(50) | Primary Key | Contains a unique ID for each transaction. |
| 2 | Booking\_ID | Varchar(50) | Foreign Key | Links to the Booking\_ID in the Booking table. |
| 3 | Payment\_Amount | Float | - | Stores the total amount of the transaction. |
| 4 | Payment\_Method | Varchar(20) | - | Indicates the mode of payment (e.g., Card, Cash). |
| 5 | Payment\_Date | Date | - | Stores the date of payment. |

# List of Figures

Figure1:Functional flow



# Introduction

## 1. Introduction The Hotel Booking System (HBS) is an essential software application designed to automate and streamline various administrative and operational tasks within the hospitality industry. By enabling efficient management of bookings, guest information, billing, and other essential services, the system enhances the overall efficiency and accuracy of hotel operations. Implementing such a system significantly improves service quality, minimizes human errors, and reduces manual workload for hotel staff.

## Developed in C++ and utilizing object-oriented programming (OOP) principles, the Hotel Booking System offers a scalable, modular, and user-friendly approach to hotel administration. Key features include guest data management, room allocation, billing, and booking history. Additionally, features like error handling and a clean user interface ensure reliable performance and ease of use. This report explores the design and implementation of the HBS, highlighting its advantages in modernizing hotel operations.

## 1.1 Objectives of the Hotel Booking System The primary goal of the HBS is to provide a comprehensive digital solution for managing hotel operations. By replacing traditional methods like spreadsheets and manual logs, the system enables hotels to maintain accurate and up-to-date records while offering enhanced guest experiences. It ensures:

## Simplified room and guest management.

## Streamlined booking and billing processes.

## Reduced manual effort and operational inefficiencies.

## Enhanced accuracy in data management.

## 1.2 Scope The Hotel Booking System is designed as a command-line application suitable for small- to medium-sized hotels. It provides essential features required for managing daily operations without the need for complex database integration or graphical user interfaces. With C++ vectors and dynamic data structures, the system ensures scalability as hotel operations grow.

## 1.3 Goals

## To enable efficient addition, update, and deletion of room and guest records.

## To ensure data integrity by validating inputs and preventing duplicate entries.

## To create a user-friendly interface, allowing hotel staff to operate the system with minimal technical expertise.

## To develop an extensible and modular codebase that can be expanded in future iterations with advanced features like online booking or analytics.

# Methodology

## 1. User Input and Room Booking

Purpose: To gather necessary details from the user for booking a room.

Steps:

## 1.11Prompt the User:

## Ask for the type of room (roomType) and the number of guests (numGuests).

## 1.12. Calculate Rooms Needed:

If numGuests > 2, ask the user to specify the number of rooms (roomsNeeded).

Otherwise, set roomsNeeded to 1.

### 1.13. Gather Additional Information:

Collect details like the number of nights (nights), phone number (phone), Aadhar card number (aadhar), and address (address).

Use getline() for the address to allow multi-word input.

### 1.14. Capture Guest Names:

Use a loop to input names for all the guests and store them in a list (guestNames).

### 1.15. Book the Room:

Call the hotel.bookRoom() method, passing all the gathered information as parameters.

## 2. Displaying Customer Details

Purpose: To view all booked customers in the system.

Steps:

Use the hotel.displayAllCustomers() function to display details of all customers.

This feature enables the user to monitor current bookings.

## 3. Checking Offers for a Specific Customer

Purpose: To retrieve and display offers applicable to a specific customer based on their ID.

Steps:

Prompt the user for a customer ID (customerId).

Use the hotel.checkCustomerOffersById(customerId) method to display relevant offers.

## 4. Exit Option

Purpose: To allow the user to exit the program gracefully.

Steps:

If the user selects option '5', the program prints a message and terminates.

## 5. Error Handling

Purpose: To handle invalid inputs gracefully.

Steps:

If the user enters an invalid menu option, display an error message and prompt them to try again.

# Discussion

## 1. Functionality Overview

Room Booking: The system efficiently collects all relevant details, such as room type, number of guests, nights, and personal details. It accommodates edge cases (like multiple rooms for larger groups).

Customer Management: By incorporating features like displayAllCustomers() and checkCustomerOffersById(), the system provides user-friendly management of customer data.

Error Handling: Basic validation ensures the program can handle invalid menu choices gracefully.

## 2. Strengths

### 2.1. Modular Approach:

Different functionalities (e.g., booking, displaying customer details, and checking offers) are logically separated. This makes the code easier to understand and extend.

### 2.2. Scalability:

The system allows for the addition of more features, such as cancellation policies or loyalty programs, without major code restructuring.

### 2.3. Data Handling:

Collecting guest details in a loop (guestNames.push\_back(guestName)) ensures flexibility in handling variable guest counts.

Using getline() for address input avoids truncation issues caused by spaces in strings.

## 3. Limitations and Suggestions for Improvement

### 3.1. Validation of Inputs:

Issue: Inputs like roomType, numGuests, and phone are not validated. For instance:

roomType can only be 0 or 1, but no checks ensure valid input.

numGuests or roomsNeeded can be negative.

Improvement: Add input validation to restrict invalid or unrealistic values.

### 3.2. Data Storage:

Issue: The data is not saved persistently; once the program terminates, all customer data is lost.

Improvement: Implement a database (e.g., SQLite) or file-based storage system to save and retrieve bookings.

### 3.3. Error Handling:

Issue: The program might fail if invalid input is entered (e.g., letters for numGuests).

Improvement: Use exception handling (e.g., try-catch) to manage runtime errors and ensure input is of the correct type.

### 3.4. User Experience:

Issue: The menu-driven approach works but could be improved for usability.

Improvement: Add clear instructions, confirmation messages, and formatting for output to make the program more user-friendly.

### 3.5. Security Concerns:

Issue: Sensitive data like Aadhar numbers and phone numbers are handled as plain text.

Improvement: Encrypt sensitive data before storage and implement access control.

## 4. Future Enhancements

Add Features:

Room availability tracking based on dates.

Dynamic pricing based on demand or seasonal trends.

Integration with a payment gateway for online transactions.

Optimize Code:

Abstract repetitive operations (e.g., input gathering) into reusable functions.

Use object-oriented principles to encapsulate data and methods related to rooms, customers, and bookings.

Interface Upgrade:

Replace the text-based menu with a graphical user interface (GUI) or a web interface to make the system more accessible.

# Concluding Remarks

The code effectively demonstrates the foundational structure of a hotel management system with a clear focus on essential operations such as room booking, customer data management, and offer checking. However, as with any initial implementation, there are multiple aspects that can be enhanced to improve functionality, efficiency, and scalability. Below are the detailed points summarizing its strengths, limitations, and potential areas for growth:

## Strengths:

### 1. Core Functionality:

The system successfully implements essential features like booking rooms, capturing guest details, and managing multiple guests efficiently.

Modular design with separate operations for booking, displaying customer data, and checking offers ensures maintainability.

### 2. User-Friendly Input Collection:

Features like loops for guest name input and getline() for capturing multi-word addresses make the program intuitive for users.

### 3. Error Messages for Invalid Choices:

The program includes basic error handling to notify users of invalid menu options, promoting better user interaction.

## Limitations and Opportunities for Improvement:

### 1. Input Validation:

The lack of checks on user input (e.g., roomType, numGuests, phone, and aadhar) can lead to invalid or inconsistent data. Adding validation rules will improve data integrity.

### 2. Persistent Data Storage:

Currently, all data is lost when the program exits. Integrating persistent storage (e.g., database or file systems) would allow for long-term data retention.

### 3. Error Handling:

Input type mismatches (e.g., entering letters for numGuests) can cause crashes. Incorporating exception handling will make the system more robust.

### 4. Scalability Issues:

While the code works well for basic scenarios, it lacks room availability tracking, which limits its scalability for large hotel operations.

### 5. Sensitive Data Security:

Sensitive information like phone numbers and Aadhar details are not encrypted, which poses a privacy risk. Security measures, including encryption, are necessary for real-world use.

## Future Enhancements:

### 1. Advanced Features:

Add features like room availability tracking, dynamic pricing based on demand, seasonal offers, and cancellation policies.

Enable integration with payment gateways for secure online transactions.

### 2. Optimized User Experience:

Replace the text-based interface with a graphical user interface (GUI) or web application for better usability and accessibility.

Include clear instructions, input confirmations, and better-formatted output to make the program more professional.

### 3. Improved Code Structure:

Encapsulate repetitive code into reusable functions and classes for better modularity.

Follow object-oriented principles to represent rooms, customers, and bookings as separate entities.

### 4. Data Analytics and Reporting:

Add features to generate reports on occupancy rates, revenue, and customer demographics for better decision-making.

### 5. Multi-User Support:

Extend the system to support multiple staff members accessing the system simultaneously with role-based access control.

# Future Work

To enhance the functionality, scalability, and user experience of the hotel management system, the following future work can be implemented:

## 1. Advanced Features

**Room Availability Tracking:**

Develop a system to track room availability in real-time, including check-in and check-out dates, to prevent overbooking.

Include filters for room preferences such as type (Standard or Deluxe) and amenities.

**Dynamic Pricing:**

Implement dynamic pricing based on demand, seasonal trends, or special events to maximize revenue.

Introduce discounts or promotional offers for long stays or loyalty customers.

**Cancellation Policies:**

Add functionality to cancel bookings with automated refund calculations based on the hotel's cancellation policies.

**Customer Loyalty Program:**

Introduce a loyalty program that rewards repeat customers with discounts or free services.

## 2. Enhanced Data Management

**Persistent Data Storage:**

Integrate a database (e.g., MySQL, SQLite) to store customer and booking details securely.

Enable data retrieval and reporting for better management and analytics.

**Data Backup and Recovery:**

Implement an automated backup system to ensure data safety in case of system failures.

**Data Validation:**

Include robust validation rules to ensure inputs are accurate, such as validating phone numbers, Aadhar card details, and numerical fields.

## 3. Improved Security

**Data Encryption:**

Encrypt sensitive customer information (e.g., phone numbers, Aadhar numbers) to ensure compliance with data protection laws.

**Authentication and Authorization:**

Implement login functionality with role-based access control (e.g., admin and staff roles) to restrict unauthorized access.

**Audit Logs:**

Maintain logs of all user activities to enhance system accountability and traceability.

## 4. User Interface and Experience

**Graphical User Interface (GUI):**

Replace the text-based menu system with a GUI using libraries like Qt (C++) or frameworks like PyQt or JavaFX.

**Web or Mobile Application:**

Develop a web-based or mobile-friendly version to make the system accessible to customers and staff from anywhere.

**Multi-Language Support**

Include multi-language options to cater to users from different regions.

## 5. Scalability

**Multi-User Environment:**

Implement multi-user support to allow simultaneous access by multiple staff members.

Introduce session management to track user actions.

**Integration with Third-Party Systems:**

Integrate with third-party platforms such as payment gateways for secure transactions and travel aggregators for bookings.

**Cloud Deployment:**

Deploy the system on the cloud to support large-scale operations and remote access.

## 6. Analytics and Reporting

**Booking Insights:**

Generate reports on occupancy rates, revenue trends, and popular room types for business analytics.

**Customer Feedback System:**

Incorporate a feedback module to collect and analyze customer satisfaction metrics.

**Predictive Analytics:**

Use machine learning to predict customer preferences, peak booking times, and revenue trends.

# References

Book References

1. Sommerville, Ian. Software Engineering (10th Edition). Pearson, 2015.

2. Pressman, Roger S. Software Engineering: A Practitioner's Approach. McGraw-Hill Education, 2014.

3. Lafore, Robert. Object-Oriented Programming in C++ (4th Edition). Sams Publishing, 2002.

Web-Based References

1. GeeksforGeeks. "C++ Object-Oriented Programming Concepts."

https://www.geeksforgeeks.org