**HOTEL MANAGEMENT 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

Hyndhavi Chakka (AP23110010993)

Lokeshwar Madhira (AP23110010722)

Kaushik Mosa (AP23110010965)

Pravallika Attipatla (AP23110010520)



Under the Guidance of

**MARY CH**

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[November,2004]



**CERTIFICATE**

***This is to certify that the Project report entitled “HOTEL MANAGEMENT SYSTEM” is being submitted by*** *Hyndhavi Chakka (AP23110010993),Lokeshwar Madhira (AP23110010722) , Kaushik Mosa (AP23110010965) , Pravallika Attipatla (AP23110010520)*  ***students of Department of Computer Science and Engineering, SRM University,in partial fulfillment of the requirement for the degree of “B.Tech(CSE)” carried out by her/hisduring the academic year 2024-2025.***

***Signature of the Lab In charge***

*MARY CH*

*Assistant Professor-Ad Hoc*

*Department of CSE*

**TABLE OF CONTENTS**

**Name of the content Page. No**

List of figures 4

List of tables 4

Abstract 5

1. Introduction 8

2. Module organization

2.1. Module overview 9

2.2. Module description 10

2.3. Module implementation 11-16

3. Performance Evaluation 17-19

4. Conclusion 20

Appendix

A. Sample code 21-23

B. Sample screenshots 24-26

**List of figures**

|  |  |
| --- | --- |
| **Name** | **Page No** |
| Figure 1 | 25 |
| Figure 2 | 26 |
| Figure 3 | 26 |
| Figure 4 | 27 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Name** | **Page** |
| **1** | Table 1 | 8 |
| **2** | Table 2 | 10 |

**ABSRACT**

The Hotel Management System project is designed to automate and optimize essential hotel operations, including room booking, guest check-in/check-out, billing, staff management, event coordination, and feedback analysis. With a modular structure, this system uses a range of algorithms—such as linear search for room availability, sorting for organizing bookings, queue handling for check-ins, and basic sentiment analysis for feedback—to ensure smooth and efficient service. Key classes, including Room, Guest, Staff, Event, and Reservation, manage different aspects of hotel operations, emphasizing object-oriented principles like encapsulation and modularity. This setup allows the system to handle peak loads, reduce errors, and improve guest satisfaction, making it a practical, user-friendly solution for hotel management.

**Attributes:**

Class: HotelManagementSystem

rooms: List of Room objects in the hotel.

guests: List of Guest objects currently checked in.

staff: List of Staff objects working at the hotel.

reservations: List of Reservation objects representing future bookings.

Class: Room

roomNumber: Unique number identifying the room.

type: Room type (e.g., "Single", "Double").

availability: Boolean indicating if the room is available.

Class: Guest

name: Guest's full name.

currentRoom: Room object representing the guest's current room.

billing: Billing object for managing charges.

Class: Staff

name: Staff member's name.

role: Staff role (e.g., "Receptionist")

Class: Reservation

guest: Guest associated with the reservation.

room: Room reserved.

checkInDate: Date of check-in.

checkOutDate: Date of check-out.

Class: Billing

charges: Dictionary of items and costs.

totalAmount: Total amount due.

**Methods**

Class: HotelManagementSystem

bookRoom(guest, room, checkInDate, checkOutDate): Books a room for a guest.

checkIn(guest): Checks the guest into a room.

checkOut(guest): Processes guest check-out and finalizes billing.

Class: Room

setAvailablity(status): Updates room availability.

Class: Guest

addCharge(item, amount): Adds a charge to the billing.

viewBill(): Displays the guest’s bill.

Class: Staff

assignShift(date, shift): Assigns a work shift to the staff member.

Class: Reservation

cancelReservation(): Cancels the reservation.

Class: Billing

addCharge(item, cost): Adds an item and its cost to the charges.

calculateTotal(): Calculates the total amount due.

CHAPTER1

INTRODUCTION

**1.1 Brief of your project**

The Hotel Management System is a comprehensive project designed to automate and optimize various hotel operations using algorithms. It is structured into multiple modules that handle critical tasks such as Room Booking and Reservation, Check-In/Check-Out, Billing and Payment, Staff Management, Event and Conference Management, and Feedback and Review. The system is intended to improve efficiency, reduce errors, and enhance guest satisfaction.

In this system, guests can book rooms through a reservation module that uses search and sorting algorithms to offer options based on availability and preferences. The check-in/check-out process is managed using queue structures to handle guest flow effectively during peak times. Billing is automated through simple calculations, ensuring accurate invoices. The staff management module uses greedy algorithms for optimized scheduling, while the event and conference management module leverages binary search and sorting to avoid scheduling conflicts. The feedback module analyzes guest reviews using basic sentiment analysis to identify areas for improvement.

This project incorporates essential object-oriented programming concepts such as encapsulation, modularity, and algorithm design. By addressing real-world challenges, the system demonstrates how algorithms can be applied to enhance operational efficiency in the hospitality industry.

**1.2 Software Requirements**

|  |  |
| --- | --- |
| **Operating system** | **Windows** |
| s/w | C++ |
| ram | 8GB |
| data structure | Greedy algorithm, Linear Search, Binary search algorithms |
| Processor | Intel I3 |

Table 1: Requirements

CHAPTER 2

MODULE ORGANISATION

2.1 Module Overview

The Hotel Management System project is organized into various classes to ensure modularity:

Room Class: Manages individual room details, such as room number, price, and availability status.

Booking Class: Handles the booking process, including storing guest details and associating guests with specific rooms.

CheckIn Class: Manages the check-in process for guests, recording arrival details and updating room status.

CheckOut Class: Manages the check-out process, calculating total charges based on room stay duration and updating room availability.

Billing Class: Calculates and generates billing information for guests, including room charges, taxes, and additional services.

HotelManagement Class: Controls the overall system flow, initializing room data, managing bookings, check-ins, and check-outs, and ensuring smooth interactions among modules.

2.2 Module Description

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **Function** | **Algorithms Used** | **Purpose** |
| **Room Booking and Reservation** | Manages room reservations, checks availability, and organizes room options for guests. | Linear Search, Sorting (Bubble Sort) | Enables guests to reserve rooms and ensures efficient availability checks and room organization. |
| **Check-In/Check-Out** | Handles guest check-ins and check-outs, updates room statuses, and maintains guest records. | Queue (FIFO) | Manages guest arrivals and departures efficiently by processing them in a first-come-first-served order. |
| **Billing and Payment** | Calculates room rates, service charges, taxes, generates invoices, and processes payments. | Simple Calculation, Linear Search | Automates bill calculation and payment processing to ensure fast, accurate transactions. |
| **Staff Management** | Tracks staff schedules, attendance, and assigns tasks. | Greedy Algorithm | Optimizes staff scheduling and task allocation based on availability, skill, and workload. |
| **Event and Conference Management** | Manages event space bookings (conference rooms, banquet halls), and schedules events. | Binary Search, Merge Sort | Ensures event spaces are booked efficiently and avoids conflicts by organizing events by date and time. |
| **Feedback and Review** | Collects and analyzes guest feedback to assess service quality and identify improvement areas. | Sentiment Analysis (Keyword-based, optional NLP) | Gathers and analyzes feedback to improve hotel services and provide insights for management decisions. |

Table 2: Module description

2.3 Module Implementation

1. **Room Booking and Reservation**

Check Room Availability

Yes No

Book Room (Store Guest

Details, Assign Room)

Confirm Reservation

2. **Check-In/Check-Out**

Verify Reservation

Check-In: Update Room

Status to "Occupied”

Check-out Process ? No

Check-Out: Update Room

Status to "Available"

**3. Billing and Payment**

Calculate Room Charges

Add Additional Charges

(Room Service, etc.)

Generate Bill & Display

Process Payment

4. Staff Management

Input Staff Details

(Name, Role, Shift)

Assign or Update Staff

Schedule

Display Staff Details

5. Event and Conference Management

Check Event Space Availability

Reserve Event Space

(Enter Event Details)

Confirm Reservation

6. Feedback and Review

Ask Guest for Feedback

Record Rating & Comments

Store Feedback

CHAPTER 3

PERFORMANCE EVALUATION

1. Algorithm Efficiency

**Overall Time Complexity:** The hotel management system operates efficiently with algorithms suited to small-to-moderate data sets, given the typical scale of a hotel. Core functions, such as checking room availability, booking events, and calculating bills, run in O(n log n) or O(n) time for efficient, real-time responses.

**Overall Space Complexity:** Since the data structures (such as reservation lists, schedules, and guest records) are moderate in size, the system maintains O(n) space complexity, supporting efficient memory use.

**Generalized Time Complexity:** For larger hotels or chains, complexity could scale based on data structure enhancements, for example:

**Room availability:** O(n log n) for search efficiency with sorted lists.

**Staff management and event scheduling:** O(n log n) to handle larger shifts or bookings.

For larger hotels, space complexity may rise to O(n), depending on the number of records managed for bookings, events, and staff.

2. System Usability

**User Interactions:** A straightforward command-line or potential graphical interface provides users with intuitive prompts for making reservations, checking in, and processing payments. Prompts guide users through required inputs, making operations simple and efficient.

**System Flow:** The system’s modular design, with clearly defined modules for reservation, billing, and feedback, ensures a smooth flow between processes. Guests experience minimal wait time due to efficient algorithms, enhancing the service experience.

**Feedback Mechanism:** Immediate feedback on room availability, billing confirmations, and guest satisfaction comments provides a positive user experience, allowing both guests and staff to see results in real-time.

3. Adaptability and Scalability

**Scalability:** The design allows for easy expansion, with the ability to add new modules for advanced hotel functions like loyalty programs, dynamic pricing, or integrations with external booking platforms. The modular design supports scalability for larger hotel chains.

**Extensibility:** The project can be extended with a GUI for enhanced user experience, mobile compatibility for remote access, or machine learning algorithms for dynamic pricing and personalized service recommendations.

4. Testing and Reliability

**Robustness:** The system includes validations for room availability, input verification for billing, and checks for valid event scheduling, improving reliability and preventing data conflicts.

**Error Handling:** Error messages and feedback direct users to correct issues with inputs (e.g., invalid booking dates or incorrect billing details), improving reliability and ensuring smooth, consistent operation.

CHAPTER 4

CONCLUSION

This Hotel Management System project is a robust and modular software solution designed to streamline hotel operations, improving both guest and staff experiences. By employing key classes (such as Room, Booking, Billing, Staff, Event, and Feedback), the project achieves modularity, with each class responsible for a specific part of the hotel management process. Essential features such as room booking, check-in/check-out, billing, and event scheduling are effectively implemented, leading to efficient time and space usage due to the organized structure.

The system prioritizes clarity and usability by guiding staff and guests through each interaction, ensuring smooth room assignments, accurate billing, and clear feedback collection. The integration of automated checks, such as room availability and billing validation, enhances reliability while reducing operational errors. This project also emphasizes maintainability, as each component’s singular responsibility simplifies any future upgrades or modifications.

The architecture is highly adaptable for expansion, with potential for adding features such as loyalty programs, dynamic pricing, and mobile support. Additionally, the modular design supports the integration of advanced functionalities, such as AI-driven recommendations and real-time analytics. A future enhancement could include a graphical user interface (GUI) to improve accessibility, along with a scoring or loyalty tracking system to enhance guest engagement over multiple stays.

Overall, this Hotel Management System demonstrates core concepts in data structures and object-oriented programming, offering a comprehensive and

efficient solution to hotel management challenges while leaving room for scalable growth and future innovations.

**APPENDIX**

**A. Sample code:**

#include <iostream>

#include <vector>

#include <string>

// Room class to manage each room's availability and price

class Room {

public:

int roomNumber;

bool isAvailable;

double price;

Room(int number, double price) : roomNumber(number), price(price), isAvailable(true) {}

};

// Hotel class to manage booking and billing operations

class Hotel {

private:

std::vector<Room> rooms;

public:

Hotel(int numRooms, double pricePerRoom) {

for (int i = 1; i <= numRooms; i++) {

rooms.push\_back(Room(i, pricePerRoom));

}

}

void displayAvailableRooms() {

std::cout << "\nAvailable Rooms:\n";

for (const auto &room : rooms) {

if (room.isAvailable) {

std::cout << "Room " << room.roomNumber << " - $" << room.price << " per night\n";

}

}

}

void bookRoom(const std::string &guestName) {

for (auto &room : rooms) {

if (room.isAvailable) {

room.isAvailable = false;

std::cout << guestName << " booked Room " << room.roomNumber << " successfully!\n";

return;

}

}

std::cout << "No rooms available.\n";

}

void checkoutRoom(int roomNumber) {

for (auto &room : rooms) {

if (room.roomNumber == roomNumber && !room.isAvailable) {

room.isAvailable = true;

std::cout << "Room " << room.roomNumber << " is now checked out and available.\n";

return;

}

}

std::cout << "Room not found or already available.\n";

}

};

// Main function to demonstrate user input handling

int main() {

Hotel hotel(5, 100.0); // Initialize hotel with 5 rooms at $100 per night

int choice;

std::cout << "Welcome to Simple Hotel Management System!\n";

do {

std::cout << "\nOptions:\n1. Display Available Rooms\n2. Book a Room\n3. Checkout Room\n4. Exit\n";

std::cout << "Choose an option: ";

std::cin >> choice;

if (choice == 1) {

hotel.displayAvailableRooms();

} else if (choice == 2) {

std::string guestName;

std::cout << "Enter guest name: ";

std::cin >> guestName;

hotel.bookRoom(guestName);

} else if (choice == 3) {

int roomNumber;

std::cout << "Enter room number to checkout: ";

std::cin >> roomNumber;

hotel.checkoutRoom(roomNumber);

} else if (choice == 4) {

std::cout << "Exiting the system. Thank you!\n";

} else {

std::cout << "Invalid option. Please try again.\n";

}

} while (choice != 4);

return 0;

}

**SAMPLE OUTPUTS**







