**SIMATS SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**CHENNAI-602105**

**Hotel Management System **

**A CAPSTONE PROJECT REPORT**

***In partial fulfilment for the completion of course***

**DSA0199 – Object Oriented Programming with C++ using Encapsulation**

**Submitted by**

**Haritha. R (192210406)**

**Afrina begum. M (192210431)**

**Under the Supervision of**

**S Jayanthi**

**SEPTEMBER - 2024**

**ABSTRACT:**

This project involves the development of a Hotel Management System using C++, designed to streamline and automate various operations within a hotel. The system offers functionality for managing bookings, check-ins, check-outs, room availability, and billing. It allows administrators to manage customer details, room types, and services efficiently. The system is built with a user-friendly interface and includes key features like room allocation, reservation tracking, invoice generation, and customer data management. Object-oriented programming (OOP) principles such as classes, inheritance, and polymorphism are utilized to model real-world entities like rooms, customers, and staff. By providing a centralized solution, this system reduces manual effort, enhances operational efficiency, and ensures a seamless experience for both hotel staff and customers.

Key modules implemented in the system include:

1. Room Management – Handling room availability and pricing.

2. Customer Management – Storing customer details and reservation history.

3. Booking & Reservation – Facilitating room bookings and cancellations.

4. Billing System – Automating invoice generation and payment processing.

This C++-based solution offers scalability and flexibility, making it suitable for small to medium-sized hotels.

**Introduction:**

The hospitality industry is a dynamic and competitive sector where the ability to deliver exceptional service and maintain operational efficiency is key to success. Hotel management systems (HMS) play a critical role in achieving this by automating various hotel operations, from room reservations to billing and inventory control. As hotels evolve, the complexity of managing these processes increases, making it essential to adopt a robust and efficient system to handle daily activities seamlessly.

The traditional approach to hotel management often involves manual processes that are prone to errors, inefficiencies, and delays. Keeping track of room availability, handling multiple customer check-ins, and accurately generating bills for services are tasks that become increasingly difficult to manage manually, especially during peak seasons. A software-based solution can resolve these challenges by automating and simplifying routine tasks, allowing hotel staff to focus more on providing excellent customer service.

**Purpose of the Hotel Management System**

The purpose of developing a Hotel Management System (HMS) in C++ is to create a comprehensive tool that helps hotel administrators manage various operations in a streamlined and efficient manner. This system is designed to automate repetitive tasks, reduce the risk of human error, and improve service delivery. It also enhances the guest experience by ensuring smooth reservation processes, faster check-ins and check-outs, and accurate billing.

**Key Goals of the System:**

1. Operational Efficiency: The system automates booking, room allocation, billing, and other tasks, saving time and effort for hotel staff and management.
2. Data Accuracy: The system reduces manual errors in billing, room allocation, and customer data handling, ensuring accurate records at all times.
3. Customer Satisfaction: With faster check-in, check-out, and booking processes, the system provides a smooth and hassle-free experience for hotel guests.
4. Scalability: The system is designed to grow with the hotel, easily accommodating additional services such as restaurant billing, event management, or spa services.
5. Security: Customer data, including personal and financial information, is handled securely, minimizing risks of data breaches or loss.

**Functionalities**

* **Room Cancellation:**
  + Allow customers to cancel their booking and free up the room for future customers.
  + After cancellation, the room becomes available again.
* **Room Services:** 
  + Add the option for customers to request additional services (e.g., food, laundry, etc.) during their stay.
  + Charges for additional services can be added to the final bill.
  + Check-in and Check-out System: Include proper check-in and check-out mechanisms.
  + Track the time and date of check-in and check-out for each customer.

**Multiple Room Booking:**

* + Allow customers to book multiple rooms under the same name for group bookings.
  + Date-based Booking: Implement a feature to allow booking for specific dates to manage room availability better.
  + Use a date data structure to store the booking date and check if the room is available on those dates.
  + Discount System: Implement discount codes or loyalty programs for frequent customers.
  + For example, provide discounts for long-term stays or special promotions.
  + Payment System: Add various payment options (credit card, online payment) with a confirmation message.

**User Interface**

* Improved Menu System: Make the menu more detailed and intuitive, perhaps using a text-based UI (ASCII Art) or libraries for a more interactive terminal UI.
* Error Handling: Add proper input validation to prevent invalid data entry, such as wrong room numbers or non-numeric input.
* User Roles: Implement role-based access:
  + Admin: Can add/remove rooms, view all bookings, manage rooms.
  + Customer: Can only view available rooms and book rooms.
* Summary Reports: Provide summary options for the admin, like:
  + Total bookings made.
  + Rooms currently occupied.
  + Revenue generated.

**File Handling (Persistence)**

**File I/O for Data Storage:**

* + Save customer data and room availability status to a file (e.g., customers.txt, rooms.txt).
  + On program startup, load these files to restore previous data.

**Database Integration:**

For more advanced management, integrate a lightweight database like SQLite or MySQL to store customer, room, and transaction details. This will allow easy searching, updating, and deleting of records.

**Class Improvements**

* **Inheritance for Room Types:** Instead of using a roomType string, implement class inheritance to create subclasses like SingleRoom, DoubleRoom, and SuiteRoom with specific properties.
* **Dynamic Pricing:** Add dynamic pricing for different seasons (e.g., peak seasons with higher prices).
* **Additional Room Features:** Enhance the Room class with more details:
  + Number of beds, room size, view type (sea view, city view), etc.
* **Enhanced Customer Details:** Store additional customer information such as address, ID number, and nationality for record-keeping.

**Advanced Algorithms**

* **Search and Sorting**: Allow customers/admin to search for rooms based on various criteria (price, availability, room type). Use algorithms like binary search for efficient searching.
* **Queue for Waiting List:** If a customer wants to book a room that is unavailable, implement a waiting list (queue) where customers can be added if the room becomes available again.

**Reports & Analytics**

* **Revenue Reports:** Implement functionality to generate a report of total earnings over a period (daily, weekly, or monthly).
* **Occupancy Rate:** Track the percentage of rooms that are occupied versus available at any given time.
* **Customer History**: For returning customers, maintain a history of their stays, including room preferences and special requests.

**Security**

* **Password Protection for Admin:** Secure admin functions (e.g., adding/removing rooms, generating reports) using a password authentication system.
* **Data Encryption**: If using file storage, consider encrypting sensitive data like customer contact details.

**Multilingual Support**

* **Language Options:** Provide a multilingual interface to accommodate users from different regions (e.g., English, Spanish, French, etc.).

**Additional Rooms and Services**

* **Room Categories and Amenities:** Add detailed room amenities like air conditioning, Wi-Fi, mini-bar, etc. and let customers filter rooms based on amenities.
* **Special Rooms:** Allow for premium and luxury rooms with special pricing and features.
* **Service Package Upgrades:** Offer special packages for guests, such as honeymoon suites, family packages, etc.

**Mobile App or Web Interface (for later phases)**

* **Mobile App for Hotel Management:** Build an app or web interface for customers to book rooms remotely, view available rooms, and check their billing status.

**Additional User Interaction Features**

* **Feedback System:** After checkout, let customers leave feedback or a rating for the room and services provided.
* **Notifications:** Provide in-system notifications for customers about their check-in, check-out time, or special promotions.
* **Cancellation Policy and Refund System:** Add the option to refund customers based on a defined cancellation policy.

**Concurrency (Advanced)**

* Implement multithreading to handle concurrent requests from multiple customers (in a multi-user system).
* This can help simulate multiple bookings at the same time in larger hotels.

**Algorithm for Hotel Management System**

The algorithm outlines the steps involved in the functioning of the Hotel Management System, focusing on the core functionalities: displaying available rooms, booking rooms, and generating bills.

**Step 1: Initialize Hotel Data**

1. **Create Room objects** with attributes:
   * Room Number
   * Room Type
   * Price per night
   * Availability status (true/false)
2. Add these Room objects to a list (or array) of rooms.

**Step 2: Display Main Menu**

1. Display a list of available options:
   * 1: Display Available Rooms
   * 2: Book a Room
   * 3: Generate Bill
   * 4: Exit
2. Prompt the user to enter their choice.
3. Depending on the user's input, call the corresponding function:
   * If **1** is selected, call the displayAvailableRooms() function.
   * If **2** is selected, call the bookRoom() function.
   * If **3** is selected, call the generateBill() function.
   * If **4** is selected, exit the system.

**Step 3: Display Available Rooms (Function: displayAvailableRooms)**

1. **Input**: None
2. **Process**:
   * Loop through the list of rooms.
   * For each room, check if the room's isAvailable attribute is true.
   * If the room is available, display the room details (Room Number, Room Type, Price per Night).
3. **Output**: List of available rooms with their details.

**Step 4: Book a Room (Function: bookRoom)**

1. **Input**:
   * Customer's name.
   * Customer's contact information.
2. **Process**:
   * Display the available rooms by calling the displayAvailableRooms() function.
   * Prompt the user to enter the room number they want to book.
   * Search for the room with the entered room number in the list of rooms.
   * If the room is found and is available:
     + Set the room's isAvailable attribute to false (mark room as booked).
     + Store customer details (name, contact) and the room number in a customer object.
     + Add the customer object to the list of booked customers.
     + Display a booking confirmation message.
   * If the room is not available or the room number is invalid, display an error message.
3. **Output**: Confirmation of room booking or error message.

**Step 5: Generate Bill (Function: generateBill)**

1. **Input**:
   * Customer's name.
   * Number of nights stayed.
2. **Process**:
   * Search for the customer in the list of customers using the customer name.
   * If the customer is found:
     + Retrieve the room number the customer booked.
     + Calculate the total bill by multiplying the number of nights by the room's price per night.
     + Display the bill, including customer details, room type, number of nights stayed, and total cost.
   * If the customer is not found, display an error message.
3. **Output**: Display the total bill for the customer or an error message if the customer is not

found.

**Step 6: Exit the System**

1. **Input**: None
2. **Process**:
   * End the program.
3. **Output**: Display a message indicating the system is exiting.

**C++ code for hotel management system:**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Room {

public:

int roomNumber;

string roomType;

bool isAvailable;

double pricePerNight;

Room(int number, string type, double price) {

roomNumber = number;

roomType = type;

pricePerNight = price;

isAvailable = true; // Default all rooms as available

}

void displayRoomDetails() {

cout << "Room Number: " << roomNumber << "\n"

<< "Room Type: " << roomType << "\n"

<< "Price per Night: $" << pricePerNight << "\n"

<< "Availability: " << (isAvailable ? "Available" : "Not Available") << "\n";

}

};

class Customer {

public:

string name;

string contact;

int bookedRoomNumber;

Customer(string cname, string ccontact, int roomNum) {

name = cname;

contact = ccontact;

bookedRoomNumber = roomNum;

}

void displayCustomerDetails() {

cout << "Customer Name: " << name << "\n"

<< "Contact: " << contact << "\n"

<< "Booked Room Number: " << bookedRoomNumber << "\n";

}

};

class Hotel {

private:

vector<Room> rooms;

vector<Customer> customers;

public:

Hotel() {

// Initialize rooms

rooms.push\_back(Room(101, "Single", 100.0));

rooms.push\_back(Room(102, "Double", 150.0));

rooms.push\_back(Room(103, "Suite", 250.0));

}

void displayAvailableRooms() {

cout << "\n--- Available Rooms ---\n";

for (Room &room : rooms) {

if (room.isAvailable) {

room.displayRoomDetails();

cout << "------------------------\n";

}

}

}

void bookRoom() {

string customerName, customerContact;

int roomNumber;

cout << "\nEnter customer name: ";

cin >> customerName;

cout << "Enter contact: ";

cin >> customerContact;

displayAvailableRooms();

cout << "\nEnter the room number to book: ";

cin >> roomNumber;

bool found = false;

for (Room &room : rooms) {

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

room.isAvailable = false; // Book the room

customers.push\_back(Customer(customerName, customerContact, roomNumber));

cout << "Room " << roomNumber << " has been successfully booked!\n";

found = true;

break;

}

}

if (!found) {

cout << "Room not available or invalid room number.\n";

}

}

void generateBill() {

string customerName;

cout << "\nEnter customer name for billing: ";

cin >> customerName;

bool found = false;

for (Customer &cust : customers) {

if (cust.name == customerName) {

for (Room &room : rooms) {

if (room.roomNumber == cust.bookedRoomNumber) {

int nights;

cout << "Enter number of nights stayed: ";

cin >> nights;

double totalBill = nights \* room.pricePerNight;

cout << "\n--- Bill Details ---\n"

<< "Customer: " << cust.name << "\n"

<< "Room Number: " << cust.bookedRoomNumber << "\n"

<< "Room Type: " << room.roomType << "\n"

<< "Total Bill: $" << totalBill << "\n";

found = true;

break;

}

}

}

}

if (!found) {

cout << "Customer not found.\n";

}

}

};

int main() {

Hotel hotel;

int choice;

do {

cout << "\n--- Hotel Management System ---\n";

cout << "1. Display Available Rooms\n";

cout << "2. Book a Room\n";

cout << "3. Generate Bill\n";

cout << "4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

hotel.displayAvailableRooms();

break;

case 2:

hotel.bookRoom();

break;

case 3:

hotel.generateBill();

break;

case 4:

cout << "Exiting the system...\n";

break;

default:

cout << "Invalid choice. Please try again.\n";

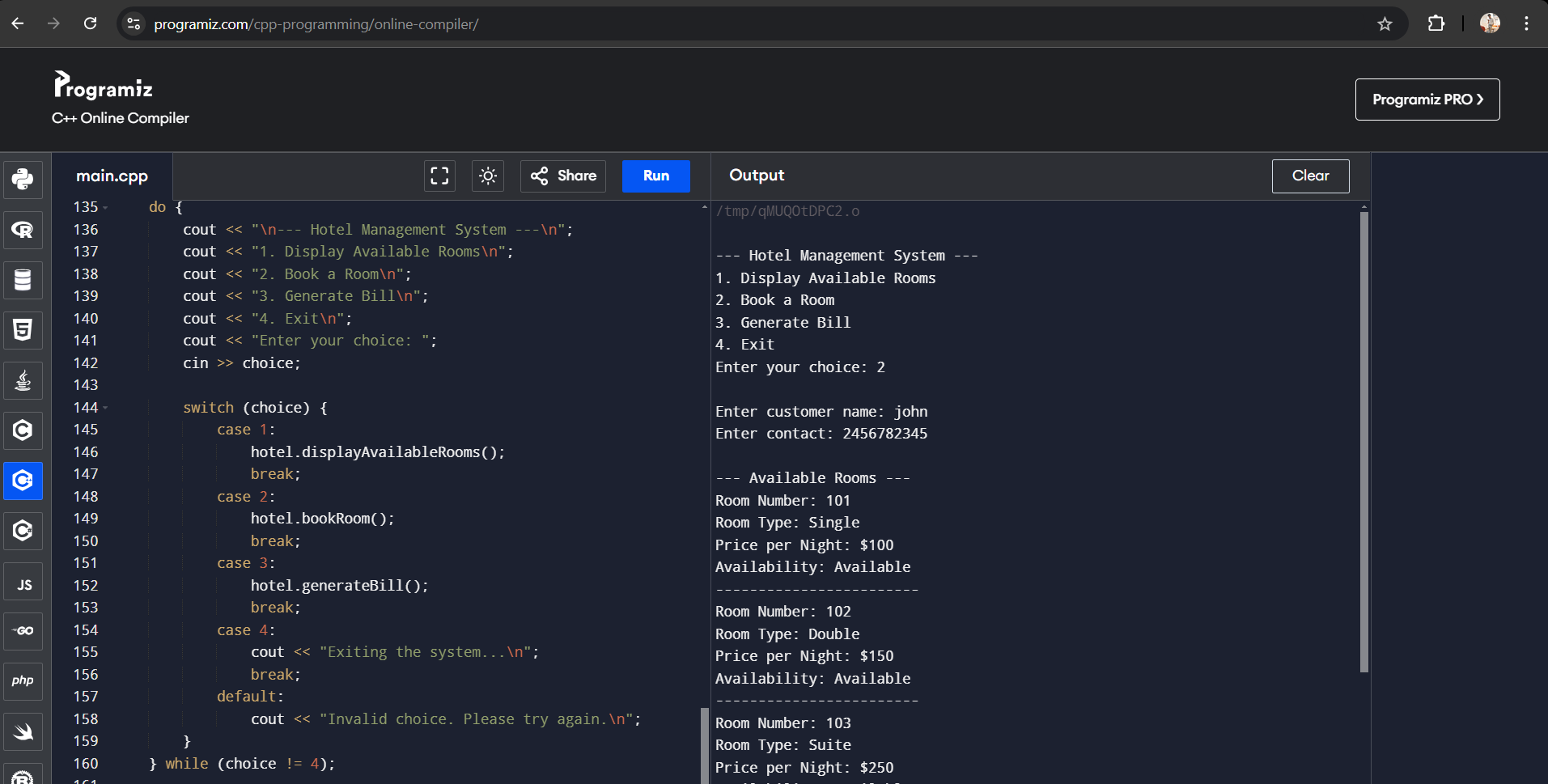
}

} while (choice != 4);

return 0;

}

**Output**

****

**Conclusion**

In conclusion, the development of a Hotel Management System using C++ aims to offer an effective solution to automate and simplify hotel operations. By reducing manual errors, improving operational efficiency, and enhancing customer satisfaction, the system serves as a valuable tool for modern hotels looking to stay competitive in the fast-paced hospitality industry. With a flexible design that supports future growth, this system is capable of accommodating the changing needs of any hotel, ensuring seamless operations and an improved guest experience.