**\*1.INTRODUCTION**

**Introduction to Hotel Management System**

A Hotel Management System is a comprehensive software solution designed to streamline and automate the operations of a hotel. This system helps manage various aspects such as room booking, check-in/check-out, billing, inventory, and guest services efficiently. The goal of the system is to improve the guest experience, optimize hotel operations, and reduce manual efforts.

In a competitive hospitality industry, maintaining high standards of service, efficient room management, and effective communication between departments are essential. The Hotel Management System achieves this by providing real-time data to hotel staff, allowing them to make informed decisions, thus improving overall productivity.

The system typically includes modules like:

* **Reservation Management**: Handles online and offline bookings.
* **Room Management**: Manages room availability, housekeeping, and maintenance.
* **Billing and Payment**: Facilitates accurate billing and different payment methods.
* **Guest Management**: Stores guest details, preferences, and history for personalized service.
* **Inventory Control**: Keeps track of the inventory of hotel supplies and services.

By implementing this system, hotels can ensure better organization, faster guest service, and a smoother operational flow, leading to improved customer satisfaction and profitability.

**\*2.Key Features of Hotel Management System:-**

^Here are the **key features** of a Hotel Management System:

### 1. ****Reservation and Booking Management****

* **Online & Offline Booking**: Accepts bookings from both online portals and walk-in guests.
* **Real-Time Room Availability**: Displays available rooms, ensuring up-to-date information.
* **Booking Modifications**: Allows guests or staff to change or cancel bookings easily.
* **Automated Confirmation**: Sends booking confirmations via email or SMS.

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

* **Quick Check-In and Check-Out**: Streamlines the process to ensure quick and efficient guest handling.
* **Digital Records**: Stores check-in/check-out history, reducing paperwork.
* **Room Assignment**: Automatically allocates rooms based on availability and guest preferences.

### 3. ****Billing and Payment Processing****

* **Automated Billing**: Generates detailed bills for room stays, food, and other services.
* **Multiple Payment Options**: Accepts various payment methods, including credit/debit cards, digital wallets, and cash.
* **Tax Calculation**: Automatically applies the correct taxes based on location.

### 4. ****Room and Housekeeping Management****

* **Room Status Updates**: Monitors and updates room status (occupied, vacant, under maintenance).
* **Housekeeping Scheduling**: Assigns housekeeping tasks, ensuring cleanliness and maintenance schedules are followed.
* **Maintenance Alerts**: Sends alerts for rooms that need repairs or special attention.

### 5. ****Guest Management****

* **Guest Profiles**: Stores guest information, including preferences and previous stays, to offer personalized services.
* **Loyalty Programs**: Manages reward points and special discounts for frequent guests.
* **Special Requests Handling**: Records and manages guest requests, like additional room amenities.

### 6. ****Inventory Management****

* **Stock Management**: Tracks and monitors the availability of hotel supplies (e.g., toiletries, food items).
* **Reorder Notifications**: Alerts staff when stock levels are low and need replenishment.
* **Vendor Management**: Maintains records of suppliers and purchase orders.

### 7. ****Reporting and Analytics****

* **Sales Reports**: Provides daily, weekly, or monthly sales data for rooms, services, and products.
* **Occupancy Reports**: Tracks room occupancy rates, helping management optimize pricing strategies.
* **Financial Reports**: Offers a summary of revenue, expenses, and profit margins.

### 8. ****Multi-Property Management****

* For hotel chains, this feature allows centralized management of multiple properties from a single interface.

### 9. ****User Access Control****

* **Role-Based Access**: Different access levels for managers, staff, and housekeeping to ensure data security.
* **Audit Logs**: Tracks user activity, improving accountability and security.

### 10. ****Customer Feedback and Complaint Management****

* Collects guest feedback after their stay to improve services.
* Logs and tracks complaints to ensure quick resolution.

These features together enhance the efficiency of hotel operations, leading to better guest experiences and increased profitability.

**\*3.Hardware Requirements :-**

 **Workstations/PCs**:

* **Processor**: Intel Core i5 or i7, 2.0 GHz or higher.
* **RAM**: 8 GB minimum (16 GB for smoother performance).
* **Storage**: 250 GB SSD.
* **Display**: 19-inch or larger LCD/LED screen.
* **Operating System**: Windows 10 or 11, Linux (Ubuntu or Fedora), or macOS.
* **Network Connectivity**: Wired (Ethernet) or Wireless (Wi-Fi) capable.

 **Laptops/Tablets** (for mobile staff):

* **Processor**: Intel Core i5 or i7, 1.6 GHz or higher.
* **RAM**: 8 GB minimum.
* **Storage**: 128 GB SSD.
* **Display**: 13-inch or larger.
* **Operating System**: Windows, Linux, or Android/iOS (if using tablets).

**\*4.Software Requirements :-**

 **Operating System**:

* Server: **Windows Server 2016+**, **Linux** (Ubuntu/CentOS).
* Client: **Windows 10/11**, **Linux**, or **macOS**.

 **Database**:

* **MySQL**, **PostgreSQL**, **SQL Server**, or **Oracle**.

 **Web Server**:

* **Apache**, **Nginx**, or **IIS**.

 **Programming Languages**:

* Backend: **PHP**, **Node.js**, **Python**, **Java , c**.
* Frontend: **HTML5**, **CSS3**, **JavaScript** (**React.js**, **Angular.js**, or **Vue.js**).

 **Middleware** (optional):

* **REST API** or **GraphQL** for system integrations

**\*5.Algorithm for Hotel Management System :-**

**Algorithm**:

1. Start the program.

2. Initialize rooms:

- Set each room as available.

- Assign room type (AC or Non-AC).

- Save room data to a file.

3. Display menu with options:

1. Display Room Status.

2. Book a Room.

3. Checkout and Generate Bill.

4. Unbook a Room.

5. Exit.

4. Process user input:

- If `1`: Show room status (available/occupied).

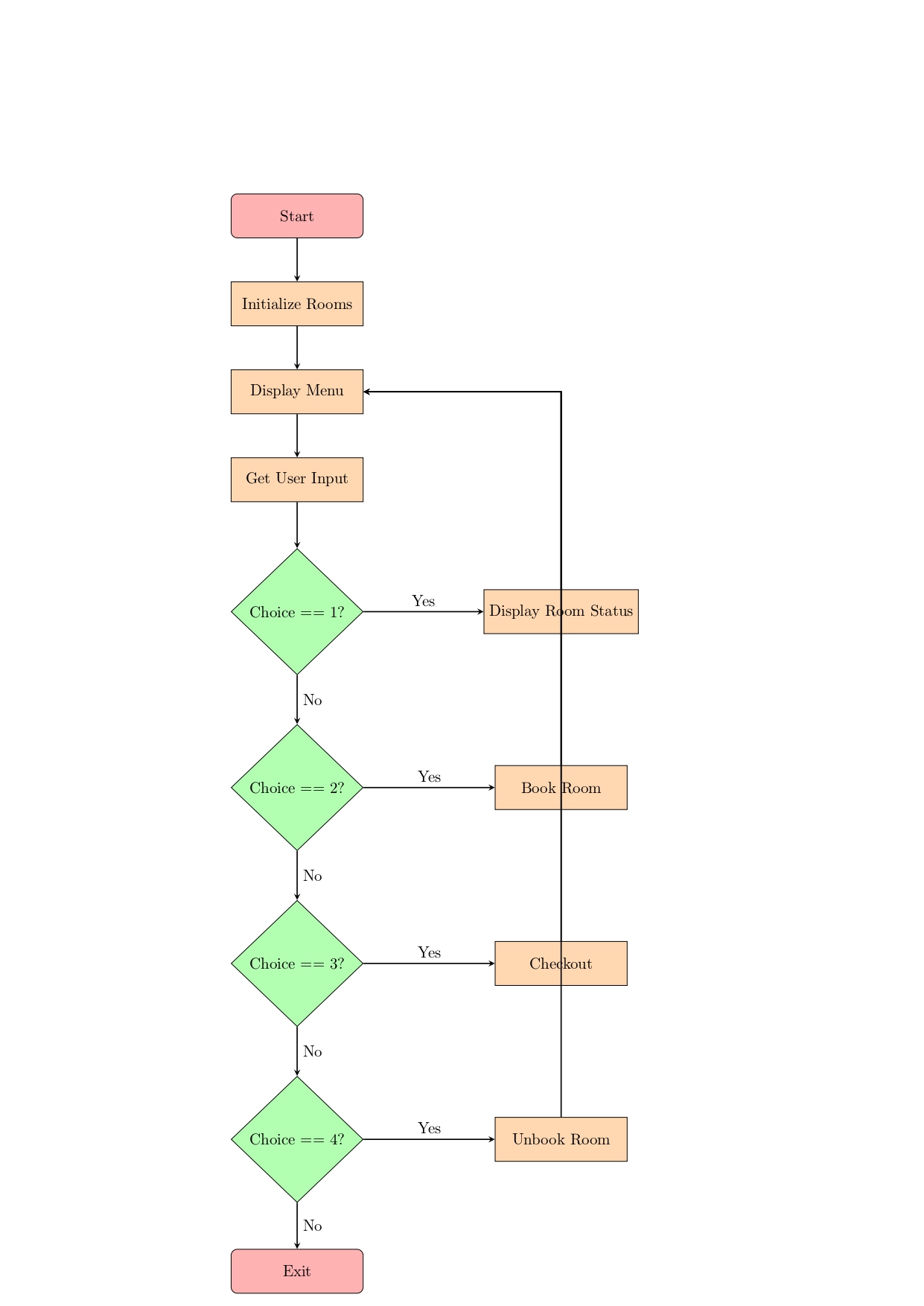
- If `2`: Book a room (ask for name, days, and assign room).

- If `3`: Checkout (calculate bill based on room type, free room).

- If `4`: Unbook a room (free the room).

- If `5`: Exit the program.

5. Repeat until user chooses to exit.

****

\*6.Flowchart :-

**\*7.Main code /program :-**

#include <stdio.h>

#include <string.h>

#define MAX\_ROOMS 5 // Total number of rooms available

#define MAX\_NAME\_LENGTH 50

#define AC\_ROOM\_RENT 1500 // Rent for AC room

#define NON\_AC\_ROOM\_RENT 1000 // Rent for Non-AC room

// Structure to store customer information

struct Customer {

char name[MAX\_NAME\_LENGTH];

int roomNumber;

int daysStayed;

int isOccupied; // 1 if room is occupied, 0 if not

char roomType[10]; // Type of room: AC or Non-AC

};

// Function to save room data to a file

void saveToFile(struct Customer hotel[]) {

FILE \*file = fopen("hotel\_data.txt", "w");

if (file == NULL) {

printf("Error opening file!\n");

return;

}

for (int i = 0; i < MAX\_ROOMS; i++) {

fprintf(file, "Room %d (%s): ", hotel[i].roomNumber, hotel[i].roomType);

if (hotel[i].isOccupied) {

fprintf(file, "Occupied by %s for %d days\n", hotel[i].name, hotel[i].daysStayed);

} else {

fprintf(file, "Available\n");

}

}

fclose(file);

}

// Initialize hotel with empty rooms

void initializeRooms(struct Customer hotel[]) {

for (int i = 0; i < MAX\_ROOMS; i++) {

hotel[i].roomNumber = i + 1;

hotel[i].isOccupied = 0; // All rooms are initially available

// Set room type based on room number

if (i < MAX\_ROOMS / 2) {

strcpy(hotel[i].roomType, "AC Room"); // First half are AC Rooms

} else {

strcpy(hotel[i].roomType, "Non-AC Room"); // Second half are Non-AC Rooms

}

}

// Save initial room data to file

saveToFile(hotel);

}

// Function to display all rooms and their status

void displayRooms(struct Customer hotel[]) {

printf("\n--- Room Status ---\n");

for (int i = 0; i < MAX\_ROOMS; i++) {

if (hotel[i].isOccupied) {

printf("Room %d (%s): Occupied by %s\n", hotel[i].roomNumber, hotel[i].roomType, hotel[i].name);

} else {

printf("Room %d (%s): Available\n", hotel[i].roomNumber, hotel[i].roomType);

}

}

}

// Function to book a room

void bookRoom(struct Customer hotel[]) {

int roomNumber, days;

char name[MAX\_NAME\_LENGTH];

printf("\nEnter Room Number to Book (1-%d): ", MAX\_ROOMS);

scanf("%d", &roomNumber);

// Check if room is available

if (roomNumber < 1 || roomNumber > MAX\_ROOMS) {

printf("Invalid room number!\n");

return;

}

if (hotel[roomNumber - 1].isOccupied) {

printf("Sorry, Room %d is already occupied.\n", roomNumber);

return;

}

// Booking process

printf("Enter Your Name: ");

scanf("%s", name);

printf("Enter Number of Days of Stay: ");

scanf("%d", &days);

// Assign room to the customer

strcpy(hotel[roomNumber - 1].name, name);

hotel[roomNumber - 1].daysStayed = days;

hotel[roomNumber - 1].isOccupied = 1;

printf("Room %d (%s) booked successfully for %s for %d days!\n", roomNumber, hotel[roomNumber - 1].roomType, name, days);

// Save updated room data to file

saveToFile(hotel);

}

// Function to checkout and calculate bill

void checkout(struct Customer hotel[]) {

int roomNumber;

int totalBill;

printf("\nEnter Room Number to Checkout: ");

scanf("%d", &roomNumber);

// Check if the room is valid and occupied

if (roomNumber < 1 || roomNumber > MAX\_ROOMS || !hotel[roomNumber - 1].isOccupied) {

printf("Invalid room number or the room is not occupied.\n");

return;

}

// Calculate total bill based on room type

if (strcmp(hotel[roomNumber - 1].roomType, "AC Room") == 0) {

totalBill = hotel[roomNumber - 1].daysStayed \* AC\_ROOM\_RENT;

} else {

totalBill = hotel[roomNumber - 1].daysStayed \* NON\_AC\_ROOM\_RENT;

}

printf("Checkout successful for %s. Total Bill: INR %d\n", hotel[roomNumber - 1].name, totalBill);

// Free up the room

hotel[roomNumber - 1].isOccupied = 0;

// Save updated room data to file

saveToFile(hotel);

}

// Function to unbook a room

void unbookRoom(struct Customer hotel[]) {

int roomNumber;

printf("\nEnter Room Number to Unbook: ");

scanf("%d", &roomNumber);

// Check if the room is valid and occupied

if (roomNumber < 1 || roomNumber > MAX\_ROOMS || !hotel[roomNumber - 1].isOccupied) {

printf("Invalid room number or the room is not occupied.\n");

return;

}

// Free up the room

hotel[roomNumber - 1].isOccupied = 0;

printf("Room %d (%s) unbooked successfully!\n", roomNumber, hotel[roomNumber - 1].roomType);

// Save updated room data to file

saveToFile(hotel);

}

int main() {

struct Customer hotel[MAX\_ROOMS];

int choice;

// Initialize rooms

initializeRooms(hotel);

while (1) {

printf("\n--- Hotel Management System ---\n");

printf("1. Display Room Status\n");

printf("2. Book a Room\n");

printf("3. Checkout and Generate Bill\n");

printf("4. Unbook a Room\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

displayRooms(hotel);

break;

case 2:

bookRoom(hotel);

break;

case 3:

checkout(hotel);

break;

case 4:

unbookRoom(hotel);

break;

case 5:

printf("Exiting Hotel Management System...\n");

return 0;

default:

printf("Invalid choice! Please try again.\n");

}

}

return 0;

}

**\*8.OUTPUT :-**

**1. Start the system**

--- Hotel Management System ---

1. Display Room Status

2. Book a Room

3. Checkout and Generate Bill

4. Unbook a Room

5. Exit

Enter your choice: 1

**2. Display room status (initially all rooms are available)**

--- Room Status ---

Room 1 (AC Room): Available

Room 2 (AC Room): Available

Room 3 (Non-AC Room): Available

Room 4 (Non-AC Room): Available

Room 5 (Non-AC Room): Available

**3. Book a room**

--- Hotel Management System ---

Enter your choice: 2

Enter Room Number to Book (1-5): 2

Enter Your Name: Karan Aher

Enter Number of Days of Stay: 3

Room 2 (AC Room) booked successfully for John for 3 days!

**4. Display room status (after booking Room 2)**

--- Hotel Management System ---

Enter your choice: 1

--- Room Status ---

Room 1 (AC Room): Available

Room 2 (AC Room): Occupied by Karan Aher

Room 3 (Non-AC Room): Available

Room 4 (Non-AC Room): Available

Room 5 (Non-AC Room): Available

**5. Checkout and generate bill for Room 2**

--- Hotel Management System ---

Enter your choice: 3

Enter Room Number to Checkout: 2

Checkout successful for Karan Aher. Total Bill: INR 4500

**6. Display room status (after checkout)**

--- Hotel Management System ---

Enter your choice: 1

--- Room Status ---

Room 1 (AC Room): Available

Room 2 (AC Room): Available

Room 3 (Non-AC Room): Available

Room 4 (Non-AC Room): Available

Room 5 (Non-AC Room): Available

**7. Exit the system**

--- Hotel Management System ---

Enter your choice: 5

Exiting Hotel Management System...

\*9.Conclusion :-

The provided **Hotel Management System** code offers a simple yet effective way to manage hotel room bookings, customer check-ins, and checkouts. It demonstrates key functionalities such as:

1. **Room Initialization**: All rooms are initialized with information on whether they are AC or Non-AC rooms, and they start as available.
2. **Booking Functionality**: Users can book a room by entering their name and the number of days they will stay. The system assigns the room and updates its status.
3. **Checkout and Billing**: When checking out, the system calculates the total bill based on the number of days stayed and the type of room (AC or Non-AC).
4. **Room Management**: The system allows users to view room availability, book, unbook, or checkout from rooms efficiently.
5. **File Handling**: The room status is saved to a file, ensuring that room data can be persisted and reviewed later, simulating a real-world hotel system where records are kept for auditing or management purposes.