1. INTRODUCTION TO PROJECT

PROJECT DESCRIPTION:

The introduction section provides an overview of the Hotel Management System project. It outlines the current challenges faced by the hospitality industry and emphasizes the need for a modernized approach to hotel management. This section introduces the HMS as a solution designed to address these challenges.

Furthermore, it briefly describes the context of the project, such as the state of the hospitality industry, emerging trends, and the competitive landscape. The section sets the stage for the reader, highlighting the significance and relevance of the HMS in the current market.

➢ **Streamlining Reservation Management**:

- The HMS aims to simplify the process of making and managing reservations, both for guests and hotel staff.

➢ **Efficient Room Inventory Management:**

- This objective involves optimizing the allocation of available rooms, minimizing overbooking, and ensuring a seamless guest experience.

➢ **Automating Billing and Invoicing Processes:**

- The system aims to reduce errors and save time by automating billing, invoicing, and payment transactions.

➢ **Enhancing Staff Management:**

- Efficient staff management is essential for providing excellent service. The HMS assists in staff allocation, task assignment, and performance analysis.

➢ **Providing Insightful Reporting and Analytics:**

- By generating reports on key performance indicators, the HMS empowers hotel management to make informed decisions and identify growth opportunities.

AIM OF THE PROJECT :

The primary aim of the Hotel Management System project is to develop an innovative and efficient software application that redefines the way hotels operate. This project aims to provide a user-friendly, centralized platform that automates various hotel functions, resulting in streamlined operations, improved guest services, and increased revenue.

PURPOSEOF THE PROJECT:

- **Enhance Efficiency:** The project seeks to automate and centralize hotel operations, reducing manual work, and improving efficiency in tasks such as reservation management, billing, staff coordination, and reporting.
- **Improve Guest Experiences:** By offering features like online reservations, real- time room availability checks, and simplified billing, the project aims to enhance the guest experience and satisfaction.
- **Increase Revenue:** Through optimized room allocation, accurate billing, and data-driven decision-making, the system intends to maximize revenue opportunities for the hotel.
- **Facilitate Staff Management:** The project helps in efficient staff allocation, task assignment, and performance analysis, leading to improved coordination among different hotel departments.
- **Provide Data-Driven Insights:** The comprehensive reporting and analytics feature provides valuable insights into hotel performance, allowing management to make informed decisions and identify areas for improvement.

2. MODULES OF THE PROJECT

MODULES USED IN THE PROJECT:

1. Reservation Module:

- createReservation(): This function allows guests to create new reservations, capturing guest information and room availability.
- updateReservation(): Enables guests to modify existing reservations as needed.
- cancelReservation(): Allows for the cancellation of reservations, making rooms available for new bookings.
- checkAvailability(): Provides real-time room availability checks for specific date ranges.

2. Room Module:

- addRoom(): Admins can add new rooms to the system's inventory, specifying room type, capacity, and amenities.
- removeRoom(): Handles the removal of rooms from the inventory when they are no longer available.
- getRoomDetails(): Retrieves detailed information about specific rooms.
- updateRoomDetails(): Allows updates to room information such as availability and pricing.

3. Billing Module:

- calculateBill(): Computes guest bills based on room charges and additional services.
- generateInvoice(): Creates detailed invoices for guests, including a breakdown of charges and payment instructions.
- processPayment(): Manages payment transactions and updates payment status.

4. Guest Module:

- addGuest(): Adds guest information to the database.
- updateGuestDetails(): Allows updates to guest details.
- retrieveGuestDetails(): Retrieves guest information and reservation history.
- searchGuests(): Enables searching for guests based on criteria like name or reservation ID.

5. Staff Module:

- addStaff(): Adds staff members to the system's staff database.
- removeStaff(): Handles staff removal when no longer employed.
- updateStaffDetails(): Updates staff information, including personal and work-related details.

- generateStaffReport(): Generates performance, attendance, and task assignment reports for staff members.

6. Reporting Module:

- generateOccupancyReport(): Provides room occupancy rates for specific time periods.
- generateRevenueReport(): Calculates and presents revenue statistics, including total revenue and revenue per available room.
- generateGuestSatisfactionReport(): Gathers guest feedback and compiles satisfaction reports.

These modules collectively provide the foundation for a comprehensive hotel management system.

3. HARDWARE & SOFTWARE REQUIREMENTS

SYSTEM REQUIREMENTS:

System requirement defines the hardware and software interface for the design, implement and debug project.

HARDWARE INTERFACE:

➢ Processor	: Intel Core i3 or equivalent
➢ Processor speed	: 2.0 GHz or higher
➢ RAM	: 4 GB or higher
➢ Hard disk	: 100 GB or higher

SOFTWARE INTERFACE:

➢ Programming Interface	: Visual C++ Compiler
➢ Visual Software	: Visual Studio Community 2022
➢ Database Software	: File I/O operations
➢ Documentation Tool	: Microsoft Office

4. ABOUT SOFTWARE TOOLS USED

C++ PROGRAMMING:

C++ programming serves as the foundation of the Hotel Management System. Its versatility, efficiency, and support for object-oriented programming make it an ideal choice for developing a complex system like HMS. C++ enables the creation of classes, data structures, and algorithms that are essential for the system's functionality.

C++ also offers the advantage of platform independence, allowing the HMS to run on various operating systems while maintaining consistent performance and functionality.

OOP (Object-Oriented Programming) is the best methodology or the paradigm to design a program using Classes and Objects. It simplifies the software development and maintenance by providing concepts such as **Inheritance** (defining class in terms of another class), **Polymorphism** (having many forms), **Data binding** (hiding background details) many more.

Object: Any entity that has state and behavior is known as an **Object**. It is an unique identity (runtime entity). In simple, **Object** is an instance of a **Class**.

Class: It is a group of similar objects. It may contain 1 or more objects. It is a template or a reference copy for object to manipulate.

A **class** is a way to bind the data & its associated functions together. It allows the data & function to be hidden, if necessary, from external use.

Generally, a class specification has two parts:

1. **Class Declaration** – It describes the type & scope of its members.
2. **Class function definition** – It describes how the class functions are implemented.

Features of OOP:

- ❑ Object-Oriented uses objects, class & its interactions to design applications and computer programs.
- ❑ OOP consists of a group of cooperating objects to exchange messages with real world applications.
- ❑ It is a type of high-level language that uses self-contained, modular instruction set (objects) for defining & manipulating aspects of system program.
- ❑ The inheritance allows an object to take functions of other objects that are connected functionally. They facilitate function & operator overloading.

FILE SYSTEM I/O FUNCTIONS:

File handling, implemented through C++, enables the system to store and retrieve data efficiently. It ensures that critical information, such as guest details, room availability, and billing records, is persistently stored and easily accessible.

File handling is used to store data permanently in a computer. Using file handling we can store our data in secondary memory (Hard disk).
How to achieve the File Handling
For achieving file handling, we need to follow the following steps:-
STEP 1 - Naming a file
STEP 2 - Opening a file
STEP 3 - Writing data into the file
STEP 4 - Reading data from the file
STEP 5 - Closing a file.

Streams in C++: -

We give input to the executing program and the execution program gives back the output. The sequence of bytes given as input to the executing program and the sequence of bytes that comes as output from the executing program are called stream. In other words, streams are nothing but the flow of data in a sequence.

The input and output operation between the executing program and the devices like keyboard and monitor are known as "console I/O operation". The input and output operation between the executing program and files are known as "disk I/O operation".

Classes for File stream operations: -

The I/O system of C++ contains a set of classes which define the file handling methods. These include ifstream, ofstream and fstream classes. These classes are derived from fstream and from the corresponding iostream class. These classes, designed to manage the disk files, are declared in fstream and therefore we must include this file in any program that uses files.

1. ios:-

- ios stands for input output stream.
- This class is the base class for other classes in this class hierarchy.

- This class contains the necessary facilities that are used by all the other derived classes for input and output operations.

2. istream: -

- istream stands for input stream.
- This class is derived from the class „ios‟.
- This class handle input stream.
- The extraction operator (>>) is overloaded in this class to handle input streams from files to the program execution.
- This class declares input functions such as get (), getline () and read ().

3. ostream: -

- ostream stands for output stream.
- This class is derived from the class „ios‟.
- This class handle output stream.
- The insertion operator (<<) is overloaded in this class to handle output streams to files from the program execution.
- This class declares output functions such as put () and write ().

4. streambuf: -

- This class contains a pointer which points to the buffer which is used to manage the input and output streams.

5. fstreambase: -

- This class provides operations common to the file streams. Serves as a base for fstream, ifstream and ofstream class.
- This class contains open () and close () function.

6. ifstream: -

- This class provides input operations.
- It contains open () function with default input mode.
- Inherits the functions get (), getline(), read(), seekg() and tellg() functions from the istream.

7. ofstream: -

- This class provides output operations.
- It contains open () function with default output mode.

- Inherits the functions put (), write (), seekp () and tellp () functions from the ostream.

8. fstream: -

- This class provides support for simultaneous input and output operations.
- Inherits all the functions from istream and ostream classes through iostream.

9. filebuf: -

- Its purpose is to set the file buffers to read and write.
- We can use file buffer member function to determine the length of the file.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.

- **ofstream:** Stream class to write on files
- **ifstream:** Stream class to read from files
- **fstream:** Stream class to both read and write from/to files.

ASCII ART FOR DESIGN:

ASCII art is a creative and visually appealing form of digital art that is constructed using characters from the ASCII character set. In this context, ASCII art is used to enhance the design and aesthetics of your report by incorporating text-based graphical elements. These ASCII art pieces are created through the arrangement of characters and symbols to depict shapes, patterns, or even recognizable objects, contributing to the visual appeal and thematic representation of your report. ASCII art serves as a unique and artistic way to engage readers and add a touch of creativity to your project.

5. SOURCE CODE

```
//================= HOTEL MANAGEMENT SYSTEM ================
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <numeric>
#include <iomanip>
#include <limits>
#include <windows.h>
#include <chrono>
#include <ctime>
#include <map>
//==============================================================
using namespace std;
map<string, string> loginCredentials;
// Function to load login credentials from a file
struct UserCredentials {
    string username;    string password;
};
//==============================================================
// Function to save user credentials to a file
void saveCredentials(const UserCredentials& credentials) {
    ofstream file("credentials.txt");
    if (file.is_open()) {
        file << credentials.username << " " << credentials.password << endl;
        file.close();
    }
}
//==============================================================
// Function to load user credentials from a file
UserCredentials loadCredentials() {
    UserCredentials credentials;
    ifstream file("credentials.txt");
    if (file.is_open()) {
        file >> credentials.username >> credentials.password;
        file.close();
    }
    return credentials;
}
class Introduction {
public:
    void intro();    void head();    void time();
};
//==============================================================
void Introduction::intro() {
    cout << "\033[1;33m";
    cout << setw(65) << "C++ PROJECT ON HOTEL MANAGEMENT SYSTEM" << endl
<< endl;
```

```
    cout << setw(55) << "MADE BY" << endl;
    cout << "\t\t=====================================================" << endl;
    cout << setw(55) << "FEROZA, NIDHI & SUHAS" << endl;
    cout << "\t\t=====================================================" << endl
<< endl;
}
void Introduction::head() {
    cout << "\t\t=====================================================" << endl;
    cout << "\t\t=====================================================" << endl;
    cout << setw(55) << "ELITE GROUP OF HOTELS" << endl;
    cout << "\t\t=====================================================" << endl;
    cout <<
"\t\t=====================================================" << endl;
}
//==============================================================
void Introduction::time() {
    int i = 0;    long long j;
    cout << "\n\n\n" << setw(55) << "Connecting to Server... " << endl;
    cout << setw(55) << "Syncing Data...." << endl;        cout << "\033[0m";
}
//==============================================================
// Reservation Module
struct ReservationData {
    int reservationID;    string guestName;    int roomNumber;    string checkInDate;
    string checkOutDate;
};
//==============================================================
class Reservation {
public:
    void createReservation();    void updateReservation();
    void cancelReservation();     void checkAvailability();
private:
    bool checkRoomAvailability(int roomNumber, const string& checkInDate, const string&
checkOutDate);
    bool reservationExists(int reservationID);
    bool saveReservationData(const ReservationData& reservationData);
    ReservationData retrieveReservationData(int reservationID);
    bool updateReservationData(const ReservationData& reservationData);
    bool cancelReservationData(int reservationID);
    bool isRoomAvailable(int roomNumber, const string& checkInDate, const string&
checkOutDate);
};
tm* getDateStruct(const string& dateStr) {
    tm* timeStruct = new tm();    istringstream iss(dateStr);
    iss >> get_time(timeStruct, "%Y-%m-%d");    return timeStruct;
}
vector<string> getDatesInRange(const string& checkInDate, const string& checkOutDate) {
    vector<string> datesInRange;
    // Convert check-in and check-out dates to std::chrono::time_point objects
    chrono::system_clock::time_point checkInTime =
chrono::system_clock::from_time_t(mktime(getDateStruct(checkInDate)));
    chrono::system_clock::time_point checkOutTime =
chrono::system_clock::from_time_t(mktime(getDateStruct(checkOutDate)));
    // Calculate the duration between check-in and check-out dates
    chrono::duration<double> duration = checkOutTime - checkInTime;
    int numDays = chrono::duration_cast<chrono::hours>(duration).count() / 24;
    // Generate the dates in the range
    for (int i = 0; i <= numDays; ++i) {
        time_t date = mktime(getDateStruct(checkInDate));
        date += i * 24 * 60 * 60;  // Add i days to the check-in date
        // Convert to std::tm using localtime_s
        tm timeStruct;         localtime_s(&timeStruct,  &date);
        char buffer[11]; // Buffer for formatted date (YYYY-MM-DD + '\0')
        strftime(buffer, sizeof(buffer), "%Y-%m-%d", &timeStruct);
        datesInRange.push_back(buffer);
    }
    return datesInRange;
}
//==================================================================
bool Reservation::checkRoomAvailability(int roomNumber, const string& checkInDate, const
string& checkOutDate) {
    // Read the room availability data from the file
    ifstream file("roomAvailability.txt");      string line;
    while (getline(file, line)) {
        istringstream iss(line);
        int room;
        string availability;
        iss >> room >> availability;
        if (room == roomNumber) {
            // Convert check-in and check-out dates to std::chrono::time_point objects
            chrono::system_clock::time_point checkInTime =
chrono::system_clock::from_time_t(mktime(getDateStruct(checkInDate)));
            chrono::system_clock::time_point checkOutTime =
chrono::system_clock::from_time_t(mktime(getDateStruct(checkOutDate)));
            // Iterate over the dates in the range and check availability
            for (const string& date : getDatesInRange(checkInDate, checkOutDate)) {
                // Find the position of the date in the availability string
                size_t pos = availability.find(date);
                if (pos != string::npos) {
                    // Check if the room is available on the specified date
                    if (availability[pos + 10] == '0') {
                        file.close();        return false; // Room is not available on the date
                    }
                }
            } break;
        }
    }
    file.close();     // Room is available for the entire duration
    return true;
}
//==================================================================
bool Reservation::reservationExists(int reservationID) {
    // Check if the reservation exists in the file
    ifstream file("reservations.txt");
    string line;
    while (getline(file, line)) {
        istringstream iss(line);        int id;        iss >> id;
        if (id == reservationID) { return true; }
    }
    return false;
}
//==================================================================
bool Reservation::saveReservationData(const ReservationData& reservationData) {
    // Save the reservation data to the file
    ofstream file("reservations.txt", ios::app);
    if (file.is_open()) {
        file << reservationData.reservationID << " "
            << reservationData.guestName << " "
            << reservationData.roomNumber << " "
            << reservationData.checkInDate << " "
            << reservationData.checkOutDate << endl;
        file.close();        return true;
    }
    return false;
}
//==================================================================
ReservationData Reservation::retrieveReservationData(int reservationID) {
    // Retrieve the reservation data from the file
    ReservationData reservationData;     reservationData.reservationID = -1;
    ifstream file("reservations.txt");     string line;
    while (getline(file, line)) {
        istringstream iss(line);
        int id;
        iss >> id;
        if (id == reservationID) {
            reservationData.reservationID = id;            iss >> reservationData.guestName;
            iss >> reservationData.roomNumber;            iss >> reservationData.checkInDate;
            iss >> reservationData.checkOutDate;            break;
        }
    }
    file.close();     return reservationData;
}
//==================================================================
bool Reservation::updateReservationData(const ReservationData& reservationData) {
    // Update the reservation data in the file
    ifstream inFile("reservations.txt");     ofstream outFile("temp.txt");
    if (inFile.is_open() && outFile.is_open()) {
        string line;
        while (getline(inFile, line)) {
            istringstream iss(line);          int id;          iss >> id;
            if (id != reservationData.reservationID) { outFile << line << endl; }
            else {
                outFile << reservationData.reservationID << " "
                    << reservationData.guestName << " "
                    << reservationData.roomNumber << " "
                    << reservationData.checkInDate << " "
                    << reservationData.checkOutDate << endl;
            }
        }
        inFile.close();        outFile.close();

        // Rename the temporary file to replace the original file
        remove("reservations.txt");        rename("temp.txt", "reservations.txt");   return true;
    }
    return false;
}
//==================================================================
bool Reservation::cancelReservationData(int reservationID) {
    // Delete the reservation data from the file
    ifstream inFile("reservations.txt");     ofstream outFile("temp.txt");
    if (inFile.is_open() && outFile.is_open()) {
        string line;        while (getline(inFile, line)) {
            istringstream iss(line);            int id;            iss >> id;
            if (id != reservationID) {
                outFile << line << endl;
            }
        }
        inFile.close();        outFile.close();

        // Rename the temporary file to replace the original file
        remove("reservations.txt");        rename("temp.txt", "reservations.txt");
        return true;
    }
    return false;
}
//==================================================================
bool Reservation::isRoomAvailable(int roomNumber, const string& checkInDate, const
string& checkOutDate) {
    // Check room availability
    bool isAvailable = checkRoomAvailability(roomNumber, checkInDate, checkOutDate);

    // Check existing reservations
    ifstream file("reservations.txt");     string line;
    while (getline(file, line)) {
        istringstream iss(line);        int id, room;        string inDate, outDate;
        iss >> id >> ws >> room >> ws >> inDate >> ws >> outDate;

        // Check if the room is already reserved for the specified dates
        if (room == roomNumber && ((inDate <= checkInDate && outDate >= checkInDate) ||
(inDate <= checkOutDate && outDate >= checkOutDate))) {
            isAvailable = false;            break;
        }
    }
    file.close();
    return isAvailable;
}
//==================================================================
void Reservation::createReservation() {
    // Get input from user and create a new reservation
    int reservationID;     string guestName;     int roomNumber;
    string checkInDate;     string checkOutDate;
    cout << setw(100) << "Enter reservation ID: ";      cin >> setw(100) >> reservationID;
    if (reservationExists(reservationID)) {
        cout << setw(100) << "Reservation ID already exists. Please enter a unique ID." << endl;
        return;
    }
    cout << setw(100) << "Enter guest name: ";      cin.ignore();
    getline(cin, guestName);
    cout << setw(100) << "Enter room number: ";      cin >> setw(100) >>roomNumber;
    cout << setw(100) << "Enter check-in date: ";      cin >> setw(100) >> checkInDate;
    cout << setw(100) << "Enter check-out date: ";      cin >> setw(100) >> checkOutDate;
    if (!isRoomAvailable(roomNumber, checkInDate, checkOutDate)) {
        cout << setw(100) << "Room not available for the specified dates. Please try again." <<
endl;        return;
    }
    ReservationData reservationData;
    reservationData.reservationID = reservationID;
    reservationData.guestName = guestName;
    reservationData.roomNumber = roomNumber;
    reservationData.checkInDate = checkInDate;
    reservationData.checkOutDate = checkOutDate;
    if (saveReservationData(reservationData)) {
        cout << setw(100) << "Reservation created successfully." << endl;
    }
    else { cout << setw(100) << "Failed to create reservation." << endl; }
}
//==================================================================
void Reservation::updateReservation() {
    // Get input from user and update an existing reservation
    int reservationID;     string guestName;     int roomNumber;
    string checkInDate;     string checkOutDate;
    cout << setw(100) << "Enter reservation ID to update: ";      cin >> reservationID;
    if (!reservationExists(reservationID)) {
        cout << setw(100) << "Reservation ID not found. Please enter a valid ID." << endl;
        return;
    }
    ReservationData reservationData = retrieveReservationData(reservationID);
    cout << setw(100) << "Enter guest name [" << reservationData.guestName << "]: ";
    cin.ignore();
    getline(cin, guestName);
    cout << setw(100) << "Enter room number [" << reservationData.roomNumber << "]: ";
    cin >> setw(100) >> roomNumber;
    cout << setw(100) << "Enter check-in date [" << reservationData.checkInDate << "]: ";
    cin >> setw(100) >> checkInDate;
    cout << setw(100) << "Enter check-out date [" << reservationData.checkOutDate << "]: ";
    cin >> setw(100) >> checkOutDate;
    if (!isRoomAvailable(roomNumber, checkInDate, checkOutDate)) {
        cout << setw(100) << "Room not available for the specified dates. Please try again." <<
endl;  return;
    }
    reservationData.guestName = guestName;
    reservationData.roomNumber = roomNumber;
    reservationData.checkInDate = checkInDate;
    reservationData.checkOutDate = checkOutDate;
    if (updateReservationData(reservationData)) {
        cout << setw(100) << "Reservation updated successfully." << endl;
    }
    else {
        cout << setw(100) << "Failed to update reservation." << endl;
    }
}
//==================================================================
void Reservation::cancelReservation() {
    // Get input from user and cancel an existing reservation
    int reservationID;
    cout << setw(100) << "Enter reservation ID to cancel: ";      cin >> reservationID;
    if (!reservationExists(reservationID)) {
        cout << setw(100) << "Reservation ID not found. Please enter a valid ID." << endl;
        return;
    }
    if (cancelReservationData(reservationID)) {
        cout << setw(100) << "Reservation canceled successfully." << endl;
    }
    else {
        cout << setw(100) << "Failed to cancel reservation." << endl;
    }
}
//==================================================================
void Reservation::checkAvailability() {
    // Get input from user and check room availability
    int roomNumber;     string checkInDate;     string checkOutDate;
    cout << setw(100) << "Enter room number: ";      cin >> setw(100) >>roomNumber;
    cout << setw(100) << "Enter check-in date: ";      cin >> setw(100) >> checkInDate;
    cout << setw(100) << "Enter check-out date: ";      cin >> setw(100) >> checkOutDate;
    if (isRoomAvailable(roomNumber, checkInDate, checkOutDate)) {
        cout << setw(100) << "Room is available for the specified dates." << endl;
    }
    else {
        cout << setw(100) << "Room is not available for the specified dates." << endl;
    }
}
//==================================================================
// Room Module
struct RoomData {
    int roomNumber;     string roomType;     double price;
};
class Room {
public:
    void addRoom();     void updateRoom();     void deleteRoom();     void listRooms();
private:
    bool roomExists(int roomNumber);
    bool saveRoomData(const RoomData& roomData);
    RoomData  retrieveRoomData(int  roomNumber);
    bool updateRoomData(const RoomData& roomData);
    bool deleteRoomData(int roomNumber);
};
//=====================================================================
bool Room::roomExists(int roomNumber) {
    ifstream file("rooms.txt");
    if (file.is_open()) {
        string line;
        while (getline(file, line)) {
            istringstream iss(line);            int number;            iss >> number;
            if (number == roomNumber) {
                file.close();               return true;
            }
        }
        file.close();
    }
    return false;
}
//=====================================================================
bool Room::saveRoomData(const RoomData& roomData) {
    ofstream file("rooms.txt", ios::app);
    if (file.is_open()) {
        file << roomData.roomNumber << " "  << roomData.roomType << " "
            << roomData.price << endl;
        file.close();       return true;
    }
    return false;
}
//=====================================================================
RoomData  Room::retrieveRoomData(int  roomNumber)  {
    RoomData roomData;     roomData.roomNumber = -1;     ifstream file("rooms.txt");
    if (file.is_open()) {
        string line;
        while (getline(file, line)) {
            istringstream iss(line);            int number;            iss >> number;
            if (number == roomNumber) {
                roomData.roomNumber = number;          iss >> roomData.roomType;
                iss >> roomData.price;     break;
            }
        }
        file.close();
    }
    return roomData;
}
//=====================================================================
bool  Room::updateRoomData(const  RoomData&  roomData)  {
    ifstream   inFile("rooms.txt");   ofstream   outFile("temp.txt");
    if (inFile.is_open() && outFile.is_open()) {
        string line;
        while  (getline(inFile,  line))  {
            istringstream iss(line);
            int number;            iss >> number;
            if (number != roomData.roomNumber) {
                outFile << line << endl;
            }
            else {
                outFile << roomData.roomNumber << " " << roomData.roomType << " "
                    << roomData.price << endl;
            }
        }
        inFile.close();        outFile.close();
        remove("rooms.txt");        rename("temp.txt",  "rooms.txt");        return true;
    }
    return false;
}
//==============================================================
bool Room::deleteRoomData(int  roomNumber) {
    ifstream inFile("rooms.txt");     ofstream outFile("temp.txt");
    if (inFile.is_open() && outFile.is_open()) {
        string line;
        while (getline(inFile, line)) {
            istringstream iss(line);            int number;            iss >> number;
            if (number != roomNumber) {
                outFile << line << endl;
            }
        }
        inFile.close();        outFile.close();
        remove("rooms.txt");        rename("temp.txt",  "rooms.txt");        return true;
    }
    return false;
}
//==============================================================
void Room::addRoom()  {
    int roomNumber;    string roomType;    double price;
    cout << setw(100) << "Enter room number: ";
    cin >> roomNumber;
    if (roomExists(roomNumber))  {
        cout << setw(100) << "Room number already exists. Please enter a unique number." <<
endl;       return;
    }
    cin.ignore(); // Ignore the newline character left in the input buffer
    cout << setw(100) << "Enter room type: ";      getline(cin, roomType);
    cout << setw(100) << "Enter room price: ";      cin >> price;
    RoomData roomData;     roomData.roomNumber = roomNumber;
    roomData.roomType = roomType;     roomData.price = price;
    if (saveRoomData(roomData))  {
        cout << setw(100) << "Room added successfully." << endl;
    }
    else {
        cout << setw(100) << "Failed to add room." << endl;
    }
}
//==============================================================
void Room::updateRoom() {
    int roomNumber;     string roomType;     double price;
    cout << setw(100) << "Enter room number to update: ";      cin >> roomNumber;
    if (!roomExists(roomNumber)) {
        cout << setw(100) << "Room number not found. Please enter a valid number." << endl;
        return;
    }
    RoomData roomData = retrieveRoomData(roomNumber);
    cin.ignore();  // Ignore the newline character left in the input buffer
    cout << setw(100) << "Enter room type [" << roomData.roomType << "]: ";
    getline(cin, roomType);
    cout << setw(100) << "Enter room price [" << roomData.price << "]: ";       cin >> price;
    roomData.roomType = roomType;     roomData.price = price;
    if (updateRoomData(roomData))  {
        cout << setw(100) << "Room updated successfully." << endl;
    }
    else {
        cout << setw(100) << "Failed to update room." << endl;
    }
}
//===========================================================================
void Room::deleteRoom() {
    int roomNumber;
    cout << setw(100) << "Enter room number to delete: ";      cin >> roomNumber;
    if (!roomExists(roomNumber)) {
        cout << setw(100) << "Room number not found. Please enter a valid number." << endl;
        return;
    }
    if (deleteRoomData(roomNumber))  {
        cout << setw(100) << "Room deleted successfully." << endl;
    }
    else {
        cout << setw(100) << "Failed to delete room." << endl;
    }
}
//===========================================================================
void Room::listRooms() {
    ifstream file("rooms.txt");
    if (file.is_open()) {
        string line;
        while (getline(file, line)) {
            cout << line << endl;
        }
        file.close();
    }
    else {
        cout << setw(100) << "Failed to open file." << endl;
    }
}
//===========================================================================
// Billing Module
class Billing {
```

```
public:
   void generateBill();     void displayBill();
public:
   double calculateBill(int reservationID);
   bool saveBillData(int reservationID, double billAmount);
   double retrieveBillData(int reservationID);
   string retrieveRoomCategory(int reservationID);
};
//=================================================================
void Billing::generateBill() {
   // Get input from user and generate bill for a reservation
   int reservationID;
   cout << setw(100) << "Enter reservation ID to generate bill: ";     cin >> reservationID;
   // Calculate the bill for the reservation
   double billAmount = calculateBill(reservationID);
   // Save the bill amount to a file or perform any other desired operation
   if (saveBillData(reservationID, billAmount)) {
      cout << setw(100) << "Bill generated successfully." << endl;
   }
   else {
      cout << setw(100) << "Failed to generate bill." << endl;
   }
}
//=================================================================
void Billing::displayBill() {
   // Get input from user and display bill for a reservation
   int reservationID;
   cout << setw(100) << "Enter reservation ID to display bill: ";     cin >> reservationID;
   // Retrieve the bill amount from the file or perform any other desired operation
   double billAmount = retrieveBillData(reservationID);
   if (billAmount >= 0.0) {
      cout << setw(100) << "Reservation ID: " << reservationID << endl;
      cout << setw(100) << "Bill Amount: $" << billAmount << endl;
   }
   else {
      cout << setw(100) << "Bill not found for the specified reservation ID." << endl;
   }
}
//=================================================================
double Billing::calculateBill(int reservationID) {
   // Calculate the bill amount for the reservation
   double baseRateAC = 2500.0; // Base rate for AC room
   double baseRateNonAC = 1500.0; // Base rate for non-AC room
   double additionalItemRate = 20.0; // Rate for additional items
   // Retrieve the room category for the reservation (e.g., from a database or other source)
   string roomCategory = retrieveRoomCategory(reservationID);
   double billAmount = 0.0;
   if (roomCategory == "AC") {
      billAmount += baseRateAC;
   }
   else if (roomCategory == "Non-AC") {
      billAmount += baseRateNonAC;
```

```
    }
    else {
        // Handle invalid room category
        cout << setw(100) << "Invalid room category." << endl;
        return -1.0;
    }

//=================================================================
    // Add additional items to the bill
    int numAdditionalItems;
    cout << setw(100) << "Enter the number of additional items: ";
    cin >> setw(100) >> numAdditionalItems;
    if (numAdditionalItems > 0) {
        billAmount += numAdditionalItems * additionalItemRate;
    }
    return billAmount;
}
//=================================================================
string Billing::retrieveRoomCategory(int reservationID) {
    if (reservationID % 2 == 0) {
        return "AC";
    }
    else {
        return "Non-AC";
    }
}
//=================================================================
bool Billing::saveBillData(int reservationID, double billAmount) {
    // Save the bill data to the file
    ofstream file("bills.txt", ios::app);
    if (file.is_open()) {
        file << reservationID << " " << billAmount << endl;
        file.close();
        return true;
    }
    return false;
}
//=================================================================
double Billing::retrieveBillData(int reservationID) {
    // Retrieve the bill amount from the file
    ifstream file("bills.txt");     string line;
    while (getline(file, line)) {
        istringstream iss(line);
        int id;        double amount;        iss >> id;        iss >> amount;
        if (id == reservationID) {
            file.close();            return amount;
        }
    }
    file.close();

    return -1.0; // Indicate that the bill was not found
}
```

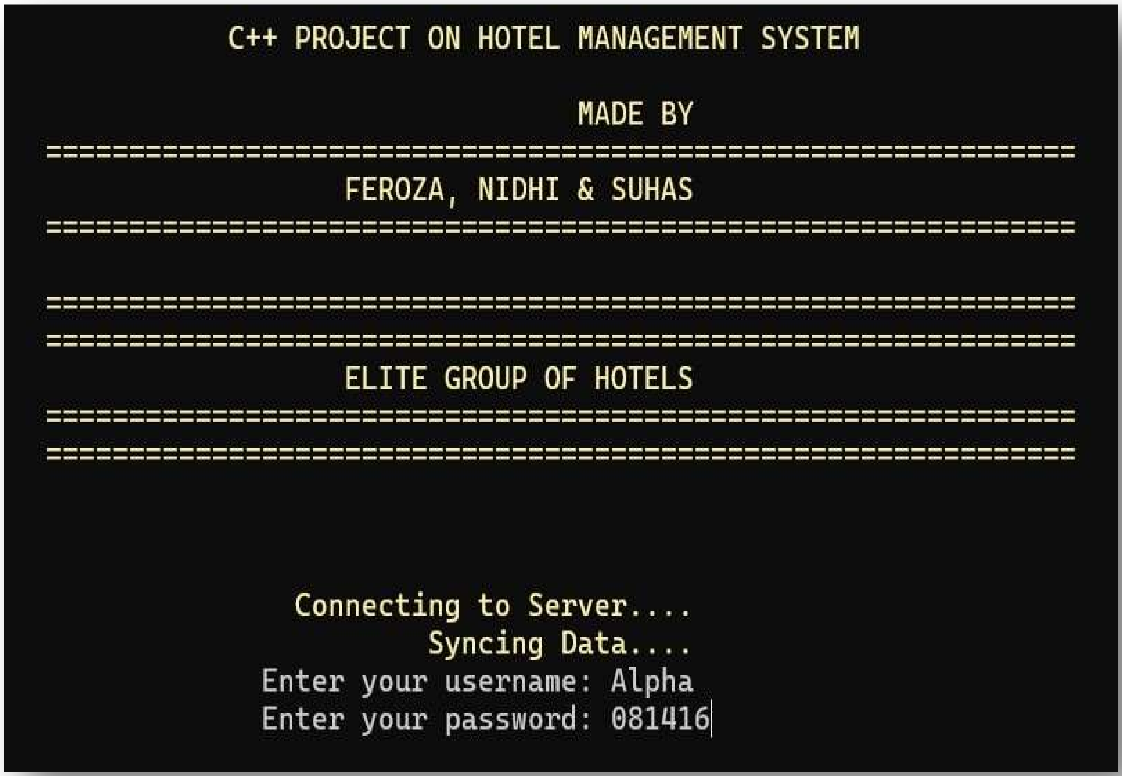
```
//===========================================================================
// Guest Module
struct GuestData {
    int guestID;     string guestName;     string guestAddress;
};
//===========================================================================
class Guest {
public:
    static void addGuest();     static void updateGuest();     static void deleteGuest();
    static void searchGuest();
private:
    static bool guestExists(int guestID);
    static bool saveGuestData(const GuestData& guestData);
    static GuestData retrieveGuestData(int guestID);
    static bool updateGuestData(const GuestData& guestData);
    static bool deleteGuestData(int guestID);
};
//===========================================================================
void Guest::addGuest() {
    GuestData guestData;
    // Prompt the user for guest details
    cout << setw(100) << "Enter guest ID: ";      cin >> guestData.guestID;
    if (guestExists(guestData.guestID)) {
        cout << setw(100) << "Guest already exists with the provided ID." << endl;
        return;
    }
    cout << setw(100) << "Enter guest name: ";      cin.ignore();
    getline(cin, guestData.guestName);
    cout << setw(100) << "Enter guest address: ";
    getline(cin, guestData.guestAddress);
    if (saveGuestData(guestData))  {
        cout << setw(100) << "Guest added successfully." << endl;
    }
    else {
        cout << setw(100) << "Failed to add guest." << endl;
    }
}
//===========================================================================
void Guest::updateGuest() {
    int guestID;
    cout << setw(100) << "Enter guest ID to update: ";      cin >> guestID;
    if (!guestExists(guestID)) {
        cout << setw(100) << "Guest does not exist with the provided ID." << endl;
        return;
    }
    GuestData guestData = retrieveGuestData(guestID);
    if (guestData.guestID == -1) {
        cout << setw(100) << "Failed to retrieve guest data." << endl;
        return;
    }
    cout <<setw(100) << "| Updating Guest      " << guestID << ":" << endl;
    cout <<setw(100) << "| 1. Guest Name      : " << guestData.guestName << endl;
```

```
        cout <<setw(100) << "| 2. Guest Address : " << guestData.guestAddress << endl;
        cout << setw(100) << "Enter the number of the field you want to update (1-2): ";
        int field;   cin >> field;
        cin.ignore(); // Ignore the newline character left in the input buffer
        switch (field) {
        case 1:
            cout << setw(100) << "Enter new guest name: ";
            getline(cin, guestData.guestName);       break;
        case 2:
            cout << setw(100) << "Enter new guest address: ";
            getline(cin, guestData.guestAddress);       break;
        default:
            cout << setw(100) << "Invalid field number." << endl;        return;
        }
        if (updateGuestData(guestData)) {
            cout << setw(100) << "Guest updated successfully." << endl;
        }
        else {
            cout << setw(100) << "Failed to update guest." << endl;
        }
    }
    //=================================================================
    void Guest::deleteGuest() {
        int guestID;
        cout << setw(100) << "Enter guest ID to delete: ";      cin >> guestID;
        if (!guestExists(guestID)) {
            cout << setw(100) << "Guest does not exist with the provided ID." << endl;
            return;
        }
        if (deleteGuestData(guestID)) {
            cout << setw(100) << "Guest deleted successfully." << endl;
        }
        else {
            cout << setw(100) << "Failed to delete guest." << endl;
        }
    }
    //=================================================================
    void Guest::searchGuest() {
        int guestID;
        cout << setw(100) << "Enter guest ID to search: ";      cin >> guestID;
        cin.ignore(); // Clear the newline character
        if (guestExists(guestID)) {
            GuestData guestData = retrieveGuestData(guestID);
            if (guestData.guestID == -1) {
                cout << setw(100) << "Failed to retrieve guest data." << endl;
            }
            else {
                cout << setw(100)<<" Guest Found       " << endl;
                cout << setw(100)<<"| Guest ID         :" << guestData.guestID << endl;
                cout << setw(100)<<"| Guest Name        :" << guestData.guestName << endl;
              /* cout << "                                        " << endl;
                cout << "| Guest Address     :" << guestData.guestAddress << endl;*/
            }
        }
        else {
            cout << setw(100)<< "Guest does not exist with the provided ID." << endl;
        }
    }
//=====================================================================
bool Guest::guestExists(int guestID) {
    ifstream file("guests.txt");     string line;
    while (getline(file, line)) {
        stringstream ss(line);        int id;        ss >> id;
        if (id == guestID) {
            file.close();            return true;
        }
    }
    file.close();     return false;
}
//=====================================================================
bool Guest::saveGuestData(const GuestData& guestData) {
    ofstream file("guests.txt", ios::app);
    if (!file) {
        return false;
    }
    file << guestData.guestID << " "  << guestData.guestName << " "
        << guestData.guestAddress << endl;
    file.close();
    return true;
}
//=====================================================================
GuestData  Guest::retrieveGuestData(int  guestID)  {
    GuestData  guestData;  guestData.guestID  =  -1;
    ifstream file("guests.txt");  string line;
    while (getline(file, line)) {
        stringstream ss(line);       int  id;       ss >> id;
        if (id == guestID) {
            guestData.guestID = id;
            getline(ss >> ws, guestData.guestName);
            getline(ss >> ws, guestData.guestAddress);
            file.close();            return guestData;
        }
    }
    file.close();     return guestData;
}
//=====================================================================
bool Guest::updateGuestData(const GuestData& guestData) {
    ifstream inputFile("guests.txt");     ofstream outputFile("temp.txt");
    if (!inputFile || !outputFile) {
        return false;
    }
    string line;
    while (getline(inputFile, line)) {
        stringstream ss(line);
        int id;
        ss >> id;
        if (id == guestData.guestID) {
            outputFile << guestData.guestID << " "
                << guestData.guestName << " "
                << guestData.guestAddress << endl;
        }
        else {
            outputFile << line << endl;
        }
    }
    inputFile.close();
    outputFile.close();
    remove("guests.txt");
    rename("temp.txt", "guests.txt");
    return true;
}
//=================================================================
bool Guest::deleteGuestData(int guestID) {
    ifstream  inputFile("guests.txt");
    ofstream outputFile("temp.txt");
    if (!inputFile || !outputFile) {
        return false;
    }
    string line;
    while (getline(inputFile, line)) {
        stringstream ss(line);
        int id;
        ss >> id;
        if (id != guestID) {
            outputFile << line << endl;
        }
    }
    inputFile.close();
    outputFile.close();

    remove("guests.txt");
    rename("temp.txt", "guests.txt");
    return true;
}
//=================================================================
// Staff Module
struct Staff {
    string name;
    int age;
    string designation;
    int salary;
};
// Function to add staff details
void addStaff(vector<Staff>& staffList) {
    Staff staff;
    cout << setw(100) << "Enter name: ";
    getline(cin >> ws, staff.name);
    cout << setw(100) << "Enter age: ";
    cin >> setw(100) >> staff.age;
    cout << setw(100) << "Enter designation: ";
    getline(cin >> ws, staff.designation);
    cout << setw(100) << "Enter salary: ";
    cin >> setw(100) >> staff.salary;
    staffList.push_back(staff);
    // Save the updated staff list to the file
    ofstream file("staff.txt");
    if (file.is_open()) {
        for (const auto& s : staffList) {
            file << s.name << "," << s.age << "," << s.designation << "," << s.salary << endl;
        }
        file.close();
        cout << setw(100) << "Staff added successfully." << endl;
    }
    else {
        cout << setw(100) << "Error: Unable to open file." << endl;
    }
}
//=====================================================================
// Function to display all staff details
void displayStaff(const vector<Staff>& staffList) {
    if (staffList.empty()) {
        cout << setw(100) << "No staff records found." << endl;
        return;
    }
    cout << "Staff Details:" << endl;
    for (const auto& staff : staffList) {
        cout << setw(100) << "| Name        :" << staff.name << endl;
        cout << setw(100) << "| Age         :" << staff.age << endl;
        cout << setw(100) << "| Designation:" << staff.designation << endl;
        cout << setw(100) << "| salary      :" << staff.salary << endl;
    }
}
//=====================================================================
// Function to find staff by name
int findStaffByName(const vector<Staff>& staffList, const string& name) {
    for (size_t i = 0; i < staffList.size(); ++i) {
        if (staffList[i].name == name) {
            return static_cast<int>(i);
        }
    }
    return -1;
}
//=====================================================================
void deleteStaff(vector<Staff>& staffList) {
    string nameToDelete;
    cout << setw(100) << "Enter the name of the staff to delete: ";
    getline(cin >> ws, nameToDelete);
    int index = findStaffByName(staffList, nameToDelete);
    if (index != -1) {
        staffList.erase(staffList.begin() + index);
        // Update the file after deletion
        ofstream file("staff.txt");
        if (file.is_open()) {
            for (const auto& staff : staffList) {
                file << staff.name << "," << staff.age << "," << staff.designation << endl;
            }
            file.close();
            cout << setw(100) << "Staff deleted successfully." << endl;
        }
        else {
            cout << setw(100) << "Error: Unable to open file." << endl;
        }
    }
    else {
        cout << setw(100) << "Staff not found." << endl;
    }
}
//==========================================================================
// Function to update staff details
void updateStaff(vector<Staff>& staffList) {
    string nameToUpdate;
    cout << setw(100) << "Enter the name of the staff to update: ";
    getline(cin >> ws, nameToUpdate);
    int index = findStaffByName(staffList, nameToUpdate);
    if (index != -1) {
        Staff& staff = staffList[index];
        cout << setw(100) << "Enter new age: ";
        cin >> staff.age;
        cout << setw(100) << "Enter new designation: ";
        getline(cin >> ws, staff.designation);
        // Update the file after modification
        ofstream file("staff.txt");
        if (file.is_open()) {
            for (const auto& s : staffList) {
                file << s.name << "," << s.age << "," << s.designation << endl;
            }
            file.close();
            cout << setw(100) << "Staff details updated successfully." << endl;
        }
        else {
            cout << setw(100) << "Error: Unable to open file." << endl;
        }
    }
    else {
        cout << setw(100) << "Staff not found." << endl;
    }
}
//==========================================================================
// Function to load staff data from file
void loadStaff(vector<Staff>& staffList) {
    ifstream file("staff.txt");
    if (file.is_open()) {
        string line;
        while (getline(file, line)) {
            Staff staff;
            size_t pos = line.find(",");
            staff.name = line.substr(0, pos);
            line.erase(0, pos + 1);
            pos = line.find(",");
            staff.age = stoi(line.substr(0, pos));
            line.erase(0, pos + 1);
            staff.designation = line;
            staffList.push_back(staff);
        }
        file.close();
    }
    else {
        cout << setw(100) << "Error: Unable to open file." << endl;
    }
}
//=================================================================
// Reporting Module
class Reporting {
public:
    void generateOccupancyReport(const vector<int>& occupancyData);
    void generateRevenueReport(const vector<double>& revenueData);
    void generateGuestSatisfactionReport(const vector<int>& satisfactionData);
};
void Reporting::generateOccupancyReport(const vector<int>& occupancyData) {
    // Calculate average occupancy rate
    double totalOccupancy = accumulate(occupancyData.begin(), occupancyData.end(), 0);
    double averageOccupancy = totalOccupancy / occupancyData.size();
    // Generate report data
    string reportData = "Occupancy Report:\n";
    reportData += " -----------------\n";
    reportData += "Total Occupancy: " + to_string(totalOccupancy) + "\n";
    reportData += "Average Occupancy Rate: " + to_string(averageOccupancy) + "\n";
    // Write report to file
    ofstream outputFile("occupancy_report.txt");
    if (outputFile.is_open()) {
        outputFile << reportData;
        outputFile.close();
        cout << setw(100) << "Occupancy report generated and saved to 'occupancy_report.txt'."
<< endl;
    }
    else {
        cout << setw(100) << "Unable to create occupancy report file." << endl;
    }
}
//=================================================================
void Reporting::generateRevenueReport(const vector<double>& revenueData) {
    // Calculate total revenue
    double totalRevenue = accumulate(revenueData.begin(), revenueData.end(), 0.0);
    // Generate report data
    string reportData = "Revenue Report:\n";
    reportData += " ----------------\n";
    reportData += "Total Revenue: " + to_string(totalRevenue) + "\n";
    // Write report to file
    ofstream outputFile("revenue_report.txt");
    if (outputFile.is_open()) {
        outputFile << reportData;
        outputFile.close();
        cout << setw(100) << "Revenue report generated and saved to 'revenue_report.txt'." <<
endl;
    }
    else {
        cout << setw(100) << "Unable to create revenue report file." << endl;
    }
}
//==============================================================================
void Reporting::generateGuestSatisfactionReport(const vector<int>& satisfactionData) {
    // Calculate average guest satisfaction score
    double averageSatisfaction = accumulate(satisfactionData.begin(), satisfactionData.end(),
0.0) / satisfactionData.size();
    // Generate report data
    string reportData = "Guest Satisfaction Report:\n";
    reportData += "________________________\n";
    reportData += "Average Guest Satisfaction: " + to_string(averageSatisfaction) + "\n";
    // Write report to file
    ofstream outputFile("guest_satisfaction_report.txt");
    if (outputFile.is_open()) {
        outputFile << reportData;
        outputFile.close();
        cout << setw(100) << "Guest satisfaction report generated and saved to
'guest_satisfaction_report.txt'." << endl;
    }
    else {
        cout << setw(100) << "Unable to create guest satisfaction report file." << endl;
    }
}
//==============================================================================
// File System
class FileSystem {
public:
    void saveDataToFile(const std::string& filename, const std::string& data);
    std::string readDataFromFile(const std::string& filename);
};
void FileSystem::saveDataToFile(const std::string& filename, const std::string& data) {
    std::ofstream file(filename);
    if (file.is_open()) {
        file << data;
        file.close();
        std::cout << "Data saved to " << filename << std::endl;
    }
    else {
        cout << "Unable to open file " << filename << std::endl;
    }
}
//======================================================================
std::string FileSystem::readDataFromFile(const std::string& filename) {
    std::ifstream file(filename);
    std::string data;
    if (file.is_open()) {
        std::string line;
        while (std::getline(file, line)) {
            data += line + "\n";
        }
        file.close();
        return data;
    }
    else {
        cout << "Unable to open file " << filename << std::endl;
        return "";
    }
}
//======================================================================
//MAIN FUNCTION OF THE PROGRAM
#include <iostream>
#include <string>
using namespace std;
int main() {
    Introduction s;
    s.intro();
    s.head();
    s.time();
    int choice;
    UserCredentials savedCredentials = loadCredentials();
    if (savedCredentials.username.empty() || savedCredentials.password.empty()) {
        cout << "No credentials found. Please set up your credentials." << endl;
        string newUsername, newPassword;
        cout << "Enter a new username: ";
        cin >> newUsername;
        cout << "Enter a new password: ";
        cin >> newPassword;
        UserCredentials newCredentials;
        newCredentials.username = newUsername;
        newCredentials.password = newPassword;
        saveCredentials(newCredentials);
        cout << setw(100) << "Credentials saved. You can now execute the program." << endl;
    }
    else {
        string enteredUsername, enteredPassword;
        cout << setw(100) << "Enter your username: ";
        cin >> setw(100) >> enteredUsername;
        cout << setw(100) << "Enter your password: ";
        cin >> setw(100) >>enteredPassword;
        if (enteredUsername == savedCredentials.username && enteredPassword ==
savedCredentials.password) {
            cout << "Login successful. You can access the program." << endl;
            do {
               // Clear screen
               cout << "\033[2J\033[H";
               // Set box width and height
               int boxWidth = 40;
               int boxHeight = 5;
               // Print floating window
               cout << "                                        " << endl;
               cout << "                                        " << endl;
               cout << "                                        " << endl;
               cout << "                                        " << endl;
               cout << "                                        " << endl;
               cout << setw(100) <<
"+========================================+" << endl;
               cout << setw(100) << "|           HOTEL MANAGEMENT           |" << endl;
               cout << setw(100) <<
"+========================================+" << endl;
               cout << setw(100) << "|                                      |" << endl;
               cout << setw(100) << "| Please select an option:             |" << endl;
               cout << setw(100) << "|                                      |" << endl;
               cout << setw(100) << "| 1. Reservation Management            |" << endl;
               cout << setw(100) << "| 2. Room Management                   |" << endl;
               cout << setw(100) << "| 3. Billing Management                |" << endl;
               cout << setw(100) << "| 4. Guest Management                  |" << endl;
               cout << setw(100) << "| 5. Staff Management                  |" << endl;
               cout << setw(100) << "| 6. Reporting                         |" << endl;
               cout << setw(100) << "| 7. Exit                              |" << endl;
               cout << setw(100) << "|                                      |" << endl;
               cout << setw(100) <<
"+========================================+"  <<  endl;
               cout << endl;
               cout << setw(100) << "Enter your choice (1-7): ";
               cin >> setw(20) >> choice;
               // Clear screen
               cout << "\033[2J\033[H";
               switch (choice) {
               case 1: {
                  Reservation reservation;
                  int reservationChoice;
                  do {
                     // Print Reservation Management menu
                     cout << setw(100) <<
"=============================================="  <<  endl;
                     cout << setw(100) << "   RESERVATION MANAGEMENT        " << endl;
                     cout << setw(100) <<
"==============================================" << endl;
                     cout << endl;
                     cout << setw(100) << "-------Please select an option-------- " << endl;
                     cout << setw(100) << "====================================="
```

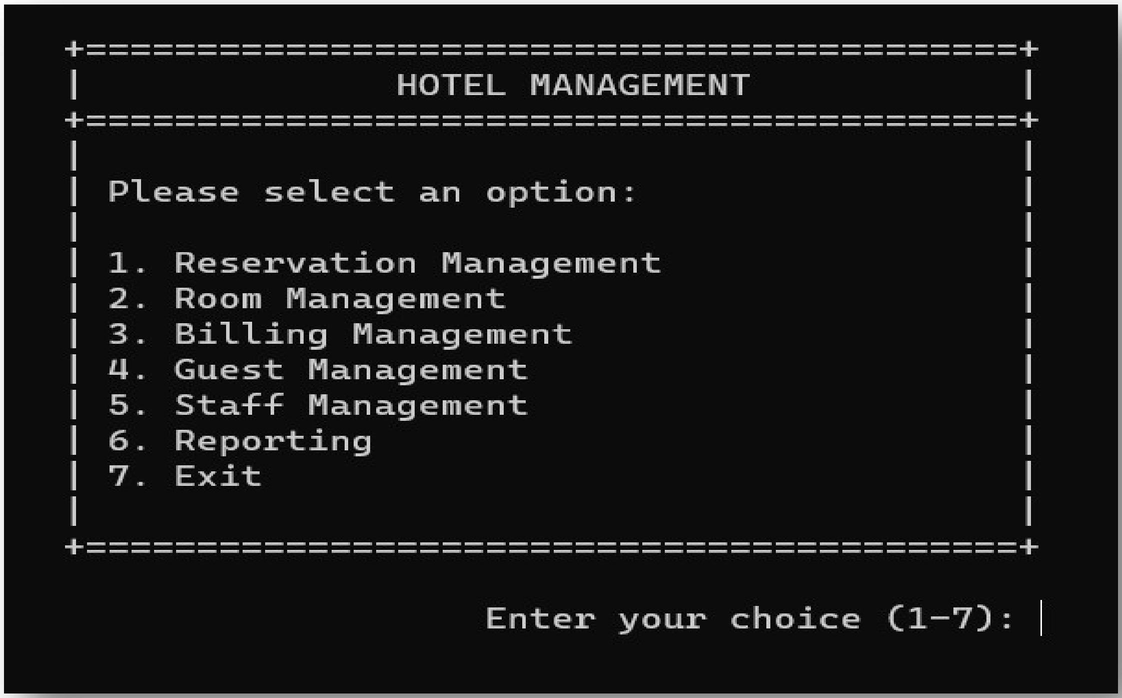
```
<< endl;
                cout << setw(70) << "|| 1. " << "Create Reservation        ||" << endl;
                cout << setw(70) << "|| 2. " << "Update Reservation        ||" << endl;
                cout << setw(70) << "|| 3. " << "Cancel Reservation        ||" << endl;
                cout << setw(70) << "|| 4. " << "Check Room Availability ||" << endl;
                cout << setw(70) << "|| 5. " << "Back to Main Menu         ||" << endl;
                cout << setw(100) << "======================================";
                cout << endl;
                cout << setw(100) << "Enter your choice (1-5): ";
                cin >> setw(100) >> reservationChoice;
                // Clear screen
                cout << "\033[2J\033[H";
                switch (reservationChoice) {
                case 1:  reservation.createReservation();      break;
                case 2: reservation.updateReservation(); break;
                case 3: reservation.cancelReservation(); break;
                case 4:  reservation.checkAvailability(); break;
                case 5: cout << setw(100) << "Returning to the main menu..." << endl; break;
                default:
                    cout << setw(100) << "OOPS! Invalid choice. Please try again." << endl;
                }
                // Pause program execution
                cout << setw(100) << "Press Enter to continue...";  cin.ignore();  cin.get();
                // Clear screen
                cout << "\033[2J\033[H";
            } while (reservationChoice != 5);     break;
        }
        case 2: {
            Room room;      int roomChoice;
            do {
                // Print Room Management menu
                cout << setw(100) << "================================" << endl;
                cout << setw(100) << "              ROOM MANAGEMENT                    " << endl;
                cout << setw(100) << "================================" << endl;
                cout << endl;
                cout << setw(100) << "Please select an option:" << endl;
                cout << setw(100) << "================================" << endl;
                cout << setw(75) << "|| 1. " << "Add Room           ||" << endl;
                cout << setw(75) << "|| 2. " << "Update Room        ||" << endl;
                cout << setw(75) << "|| 3. " << "Delete Room        ||" << endl;
                cout << setw(75) << "|| 4. " << "List Rooms         ||" << endl;
                cout << setw(75) << "|| 5. " << "Back to Main Menu ||" << endl;
                cout << setw(100) << "================================" << endl;
                cout << setw(100) << "Enter your choice (1-5): ";
                cin >> roomChoice;
                // Clear screen
                cout << "\033[2J\033[H";
                switch (roomChoice) {
                case 1:
                    room.addRoom();
                    break;
                case 2:
                    room.updateRoom();
                    break;
                case 3:
                    room.deleteRoom();
                    break;
                case 4:
                    room.listRooms();
                    break;
                case 5:
                    cout << setw(100) << "Returning to the main menu..." << endl;
                    break;
                default:
                    cout << setw(100) << "Oops! Invalid choice. Please try again." << endl;
                }
                // Pause program execution
                cout << setw(100) << "Press Enter to continue...";
                cin.ignore();
                cin.get();
                // Clear screen
                cout << "\033[2J\033[H";
            } while (roomChoice != 5);
            break;
        }
        case 3: {
            Billing  billing;
            int billingChoice;
            do {
                // Print Billing Management menu
                cout << setw(100) <<
"=============================================="  <<  endl;
                cout << setw(100) << "      BILLING MANAGEMENT             " << endl;
                cout << setw(100) <<
"==============================================" << endl;
                cout << endl;
                cout << setw(100) << "Please select an option:" << endl;
                cout << setw(100) << "==============================" << endl;
                cout << setw(75) << "| 1. " << "Generate Bill        |" << endl;
                cout << setw(75) << "| 2. " << "Display Bill         |" << endl;
                cout << setw(75) << "| 3. " << "Back to Main Menu    |" << endl;
                cout << setw(100) << "==============================" << endl;
                cout << endl;
                cout << setw(100) << "Enter your choice (1-3): ";
                cin >> setw(100) >> billingChoice;
                // Clear screen
                cout << "\033[2J\033[H";
                switch (billingChoice) {
                case 1:
                    billing.generateBill();
                    break;
                case 2:
                    billing.displayBill();
                    break;
                case 3:
                    cout << setw(100) << "Returning to the main menu..." << endl;
                    break;
                default:
                    cout << setw(100) << "Invalid choice. Please try again." << endl;
                }
                // Pause program execution
                cout << setw(100) << "Press Enter to continue...";
                cin.ignore();
                cin.get();
                // Clear screen
                cout << "\033[2J\033[H";
            } while (billingChoice != 3);
            break;
        }
        case 4: {
            int guestChoice;
            do {
                // Print Guest Management menu
                cout << setw(100) <<
"==============================================="  <<  endl;
                cout << setw(100) << "            GUEST  MANAGEMENT                    " << endl;
                cout << setw(100) <<
"===============================================" << endl;
                cout << endl;
                cout << setw(100) << "Please select an option:" << endl;
                cout << setw(100) << "                                  " << endl;
                cout << setw(100) << "================================" << endl;
                cout << setw(75) << "| 1. " << "Add Guest            |" << endl;
                cout << setw(75) << "| 2. " << "Update Guest         |" << endl;
                cout << setw(75) << "| 3. " << "Delete Guest         |" << endl;
                cout << setw(75) << "| 4. " << "Search Guest         |" << endl;
                cout << setw(75) << "| 5. " << "Back to Main Menu  |" << endl;
                cout << setw(100) << "================================" << endl;
                cout << endl;
                cout << setw(100) << "Enter your choice (1-5): ";
                cin >> setw(100) >> guestChoice;
                // Clear screen
                cout << "\033[2J\033[H";
                switch (guestChoice) {
                case 1:
                    Guest::addGuest();
                    break;
                case 2:
                    Guest::updateGuest();
                    break;
                case 3:
                    Guest::deleteGuest();
                    break;
                case 4:
                    Guest::searchGuest();
                    break;
                case 5:
                    cout << setw(100) << "Returning to the main menu..." << endl;
                    break;
                default:
                    cout << setw(100) << "Invalid choice. Please try again." << endl;
                    break;
                }
                // Pause program execution
                cout << setw(100) << "Press Enter to continue...";
                cin.ignore();
                cin.get();
                // Clear screen
                cout << "\033[2J\033[H";
            } while (guestChoice != 5);
            break;
        }
        case 5: {
            vector<Staff> staffList;
            int staffChoice;
            do {
                // Print Staff Management menu
                cout << setw(100) <<
"==============================================="  <<  endl;
                cout << setw(100) << "           STAFF MANAGEMENT                    " << endl;
                cout << setw(100) <<
"===============================================" << endl;
                cout << endl;
                cout << setw(100) << "Please select an option:" << endl;
                cout << setw(100) << "================================" << endl;
                cout << setw(75) << "| 1. " << "Add Staff            |" << endl;
                cout << setw(75) << "| 2. " << "Display Staff        |" << endl;
                cout << setw(75) << "| 3. " << "Delete Staff         |" << endl;
                cout << setw(75) << "| 4. " << "Update Staff         |" << endl;
                cout << setw(75) << "| 5. " << "Back to Main Menu  |" << endl;
                cout << setw(100) << "================================" << endl;
                cout << endl;
                cout << setw(100) << "Enter your choice (1-3): ";
                cin >> staffChoice;
                // Clear screen
                cout << "\033[2J\033[H";
                switch (staffChoice) {
                case 1:
                    addStaff(staffList);
                    break;
                case 2:
                    displayStaff(staffList);
                    break;
                case 3:
                    deleteStaff(staffList);
                    break;
                case 4:
                    updateStaff(staffList);
                    break;
                case 5:
                    cout << "Returning to the main menu..." << endl;
                    break;
                default:
                    cout << "Invalid choice. Please try again." << endl;
                    break;
                }
                // Pause program execution
                cout << setw(100) << "Press Enter to continue...";
                cin.ignore();                          cin.get();
                // Clear screen
                cout << "\033[2J\033[H";
            } while (staffChoice != 5);
            break;
        }
        case 6: {
            Reporting reporting;                    int reportingChoice;
            do {
                // Print Reporting menu
                cout << setw(100) <<
"=============================================" << endl;
                cout << setw(100) << "                REPORTING                  " << endl;
                cout << setw(100) <<
"=============================================" << endl;
                cout << endl;
                cout << setw(100) << "Please select an option:" << endl;
                cout << setw(100) << "                                " << endl;
                cout << setw(100) <<
"===============================================" << endl;
                cout << setw(60) << "| 1. " << "Generate Occupancy Report             |" << endl;
                cout << setw(60) << "| 2. " << "Generate Revenue Report               |" << endl;
                cout << setw(60) << "| 3. " << "Generate Guest Satisfaction Report |" << endl;
                cout << setw(60) << "| 4. " << "Back to Main Menu                     |" << endl;
                cout << setw(100) <<
"==============================================" << endl;
                cout << endl;
                cout << setw(100) << "Enter your choice (1-4): ";
                cin >> setw(100) >> reportingChoice;
                // Clear screen
                cout << "\033[2J\033[H";
                switch (reportingChoice) {
                case 1: {
                    vector<int> occupancyData;
                    // Populate occupancyData with actual data
                    reporting.generateOccupancyReport(occupancyData);
                    break;
                }
                case 2: {
                    vector<double> revenueData;
                    // Populate revenueData with actual data
                    reporting.generateRevenueReport(revenueData);
```

```
                        break;
                    }
                    case 3: {
                        vector<int> satisfactionData;
                        // Populate satisfactionData with actual data
                        reporting.generateGuestSatisfactionReport(satisfactionData);
                        break;
                    }
                    case 4:
                        cout << setw(100) << "Returning to the main menu..." << endl;
                        break;
                    default:
                        cout << setw(100) << "Invalid choice. Please try again." << endl;
                        break;
                    }
                    // Pause program execution
                    cout << setw(100) << "Press Enter to continue...";
                    cin.ignore();                    cin.get();
                    // Clear screen
                    cout << "\033[2J\033[H";
                } while (reportingChoice != 4);
                break;
            }
            case 7:
                cout << setw(100) << "Exiting the program..." << endl;
                break;
            default:
                cout << setw(100) << "Invalid choice. Please try again." << endl;
                break;
            }
            // Pause program execution
            cout << setw(100) << "Press Enter to continue...";
            cin.ignore();                cin.get();
            // Clear screen
            cout << "\033[2J\033[H";
        } while (choice != 7);
 //==============================================================
        //THANK YOU PAGE
        cout << "\033[1;32m";
cout<<setw(100)<<”>>========================================<<" << endl;
cout<<setw(100)<<”||### # #  # ### # #      # # # # #
||" << endl;
cout<<setw(100)<<”|| #  # # # # # # # #     # # # # # #
||" << endl;
cout<<setw(100)<<”|| #  ### ### # # ##       #  # # # #
||" << endl;
cout<<setw(100)<<”|| #  # # # # # # # #      #  # # # #
||" << endl;
cout<<setw(100)<<”|| #  # # # # # # # #      #   #  ###
||" << endl;
cout<<setw(100)<<”||
||" << endl;
cout<<setw(100)<<"||###   #   ##
||" << endl;
cout<<setw(100)<<"||#    # # # #
||" << endl;
cout<<setw(100)<<"||##   # # ##
||" << endl;
cout<<setw(100)<<"||#    # # # #
||" << endl;
cout<<setw(100)<<"||#     #  # #
||" << endl;
cout<<setw(100)<<"||
||" << endl;
cout<<setw(100)<<"|| ## ##  ### ### ### # #  #     ##      # # ### ###
# # ### ##    ## ### ### # #||" << endl;
cout<<setw(100)<<"||## #  #  # #  #  # # # # #             # # # #  #
# # #   # # #    #    #  # #||" << endl;
cout<<setw(100)<<"||  #   ##    #  # #  #   # # ###  #       # # # #  #
# # ##  ##    #    #    #    # ||" << endl;
cout<<setw(100)<<"||    # # #   #  # #  #   # # # #   #      # # # #  #
# # #   # #   #   #    #    # ||" << endl;
cout<<setw(100)<<"||##    # # ### # # ###   #   # # ##       ### # # ###
#  ### # # ##  ###   #    # ||" << endl;
cout<<setw(100)<<">>=======================================<<" << endl;
            cout << "\033[0m";
        }
        else {
            cout << setw(100) << "Login failed. Access denied." << endl;
        }
        return 0;
    }
}
//END OF THE PROGRAM
//=========================================================================
```

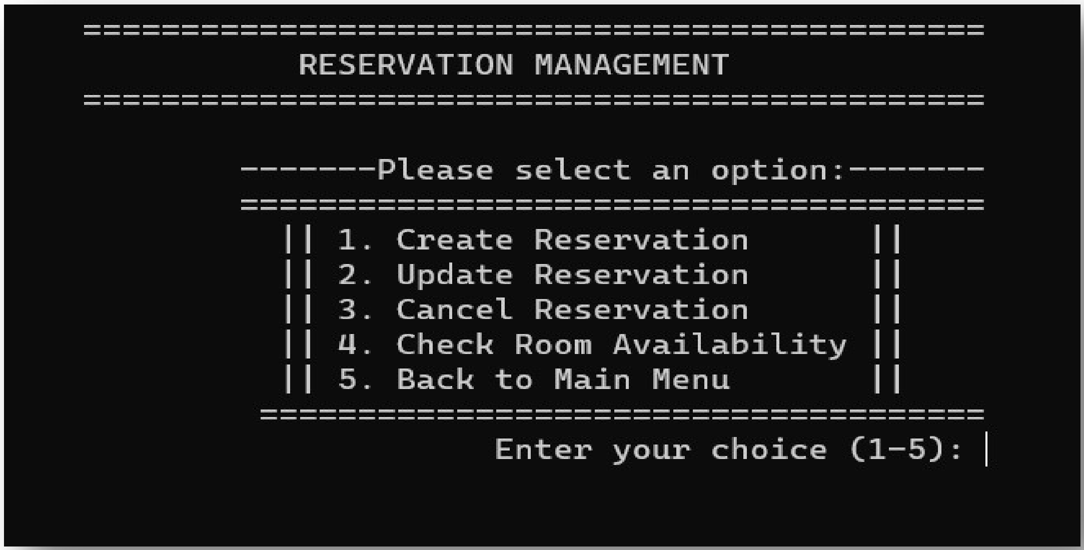
SCREEN SHOTS



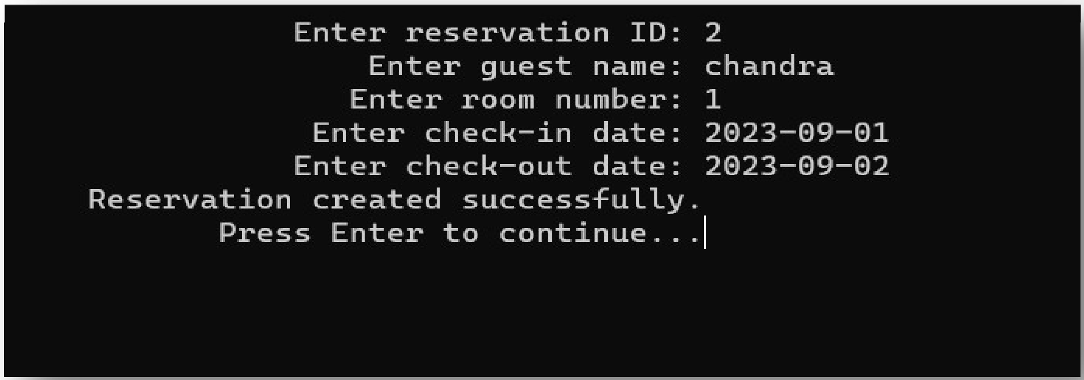
```
          C++ PROJECT ON HOTEL MANAGEMENT SYSTEM

                              MADE BY
==============================================================
                    FEROZA, NIDHI & SUHAS
==============================================================

==============================================================
==============================================================
                    ELITE GROUP OF HOTELS
==============================================================
==============================================================



                Connecting to Server....
                        Syncing Data....
              Enter your username: Alpha
              Enter your password: 081416
```



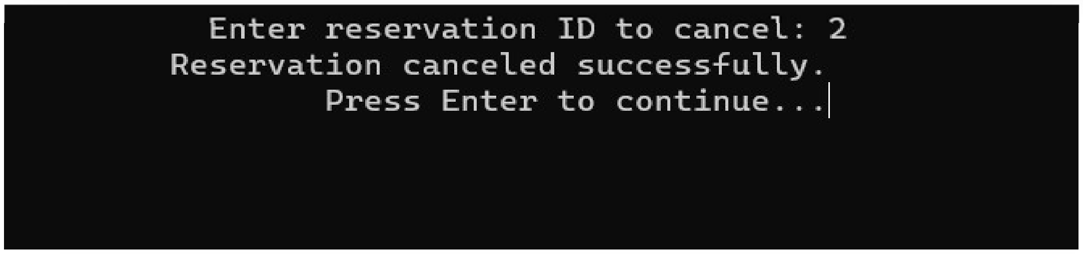
```
+============================================+
|              HOTEL MANAGEMENT              |
+============================================+
|                                            |
| Please select an option:                   |
|                                            |
| 1. Reservation Management                  |
| 2. Room Management                         |
| 3. Billing Management                      |
| 4. Guest Management                        |
| 5. Staff Management                        |
| 6. Reporting                               |
| 7. Exit                                    |
|                                            |
+============================================+

                   Enter your choice (1-7):
```



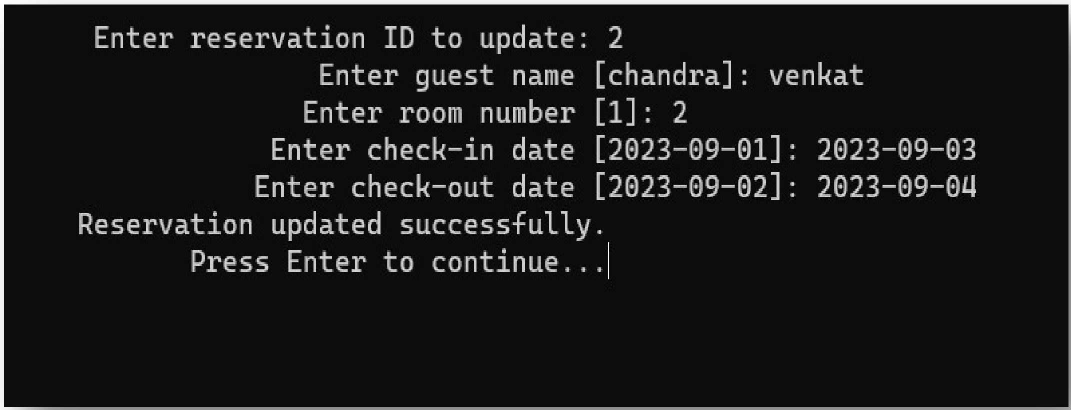
```
    ==============================================
               RESERVATION MANAGEMENT
    ==============================================

            -------Please select an option:-------
            ======================================
              || 1. Create Reservation      ||
              || 2. Update Reservation      ||
              || 3. Cancel Reservation      ||
              || 4. Check Room Availability ||
              || 5. Back to Main Menu       ||
             =====================================
                        Enter your choice (1-5):
```



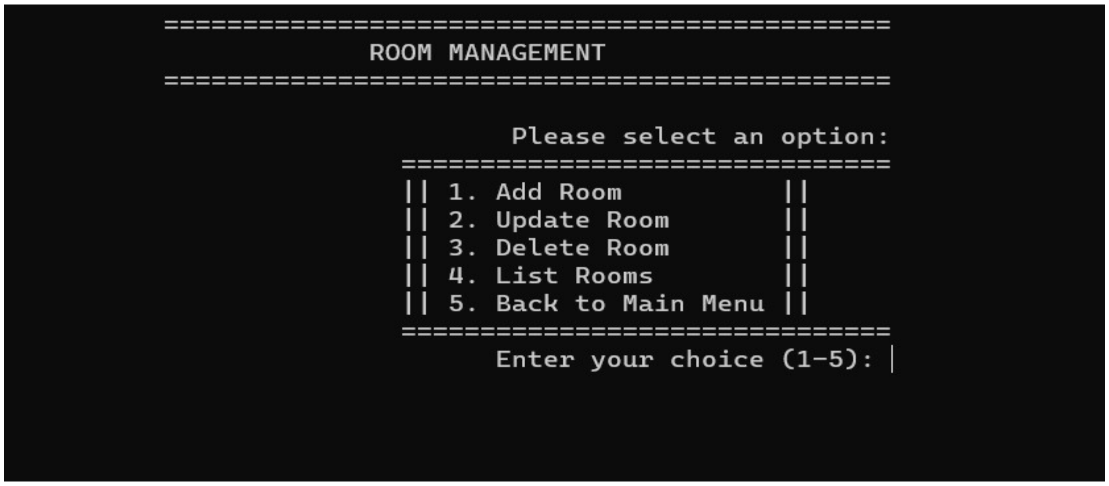
```
              Enter reservation ID: 2
                  Enter guest name: chandra
                 Enter room number: 1
               Enter check-in date: 2023-09-01
              Enter check-out date: 2023-09-02
    Reservation created successfully.
           Press Enter to continue...
```



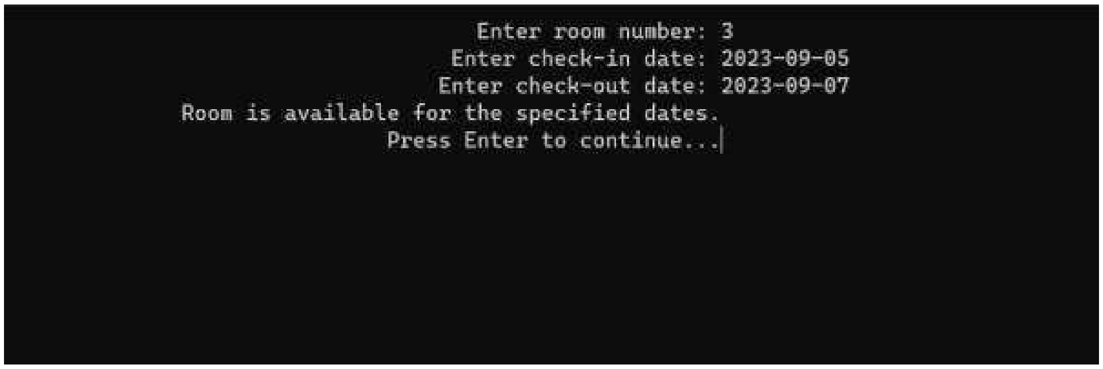
```
         Enter reservation ID to cancel: 2
       Reservation canceled successfully.
             Press Enter to continue...
```



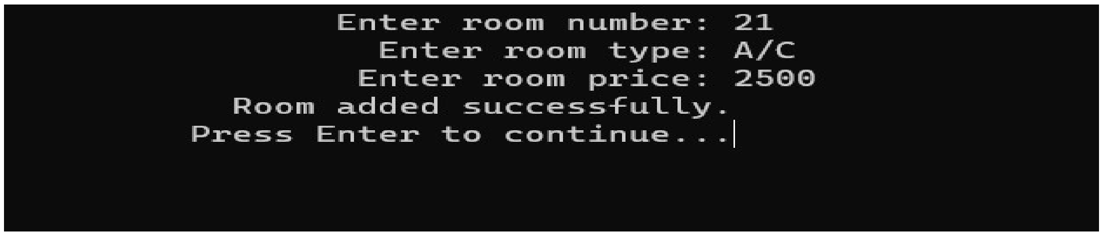
```
    Enter reservation ID to update: 2
                  Enter guest name [chandra]: venkat
                 Enter room number [1]: 2
               Enter check-in date [2023-09-01]: 2023-09-03
              Enter check-out date [2023-09-02]: 2023-09-04
   Reservation updated successfully.
          Press Enter to continue...
```



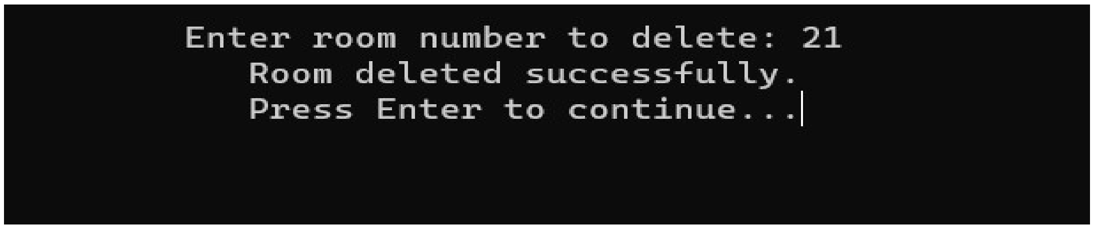
```
==================================================
               ROOM MANAGEMENT
==================================================

                       Please select an option:
                ==================================
                || 1. Add Room           ||
                || 2. Update Room        ||
                || 3. Delete Room        ||
                || 4. List Rooms         ||
                || 5. Back to Main Menu  ||
                ==================================
                      Enter your choice (1-5):
```



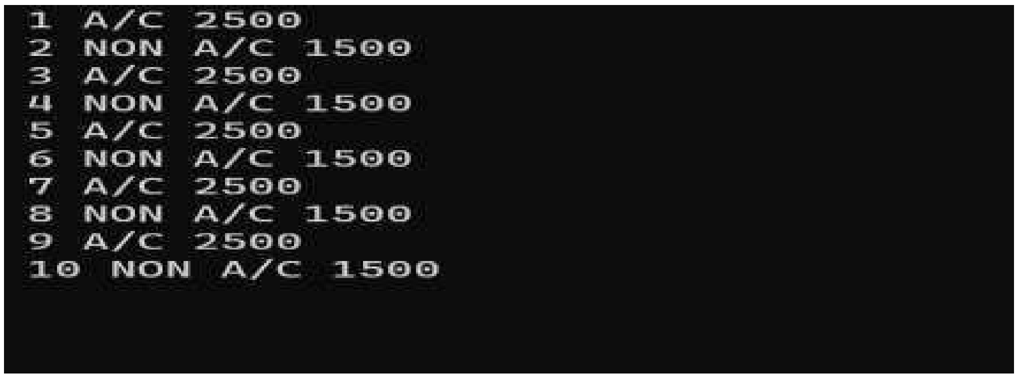
```
                        Enter room number: 3
                      Enter check-in date: 2023-09-05
                     Enter check-out date: 2023-09-07
Room is available for the specified dates.
                  Press Enter to continue...
```



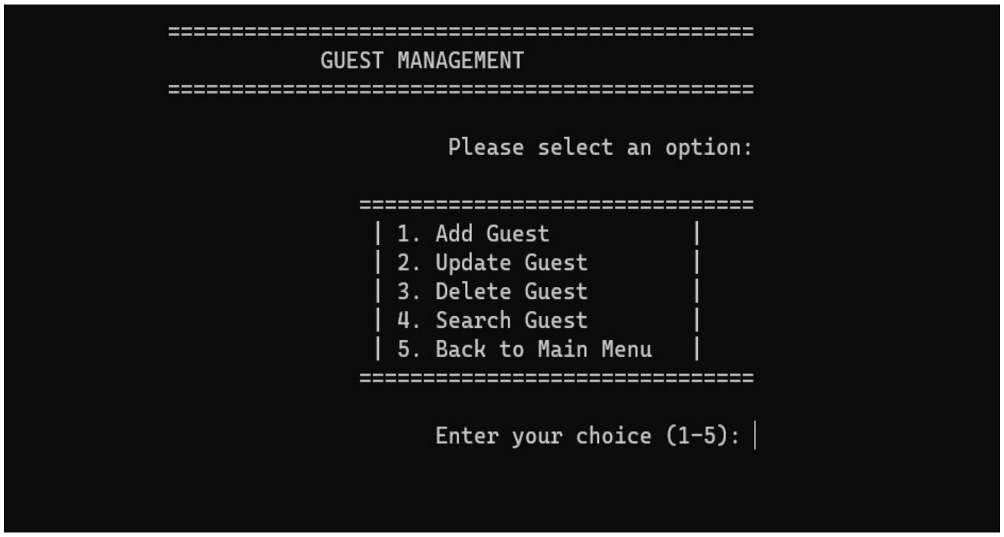
```
          Enter room number: 21
             Enter room type: A/C
           Enter room price: 2500
   Room added successfully.
Press Enter to continue...
```



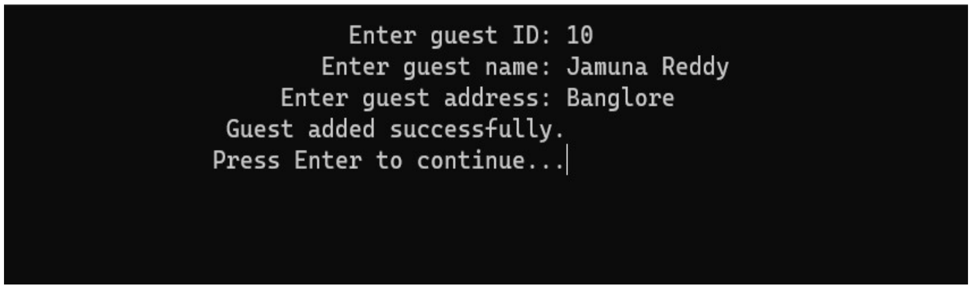
```
Enter room number to delete: 21
   Room deleted successfully.
   Press Enter to continue...
```



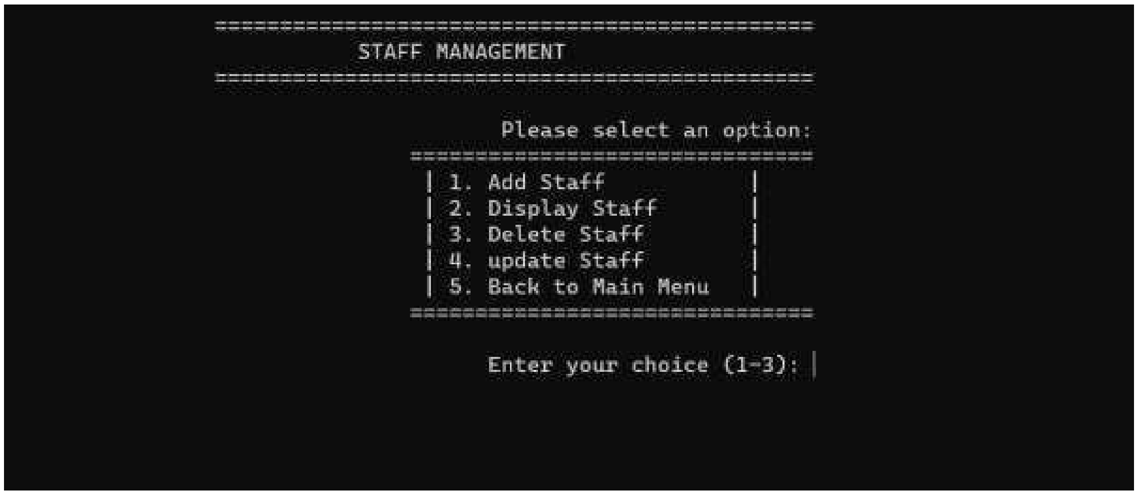
```
1 A/C 2500
2 NON A/C 1500
3 A/C 2500
4 NON A/C 1500
5 A/C 2500
6 NON A/C 1500
7 A/C 2500
8 NON A/C 1500
9 A/C 2500
10 NON A/C 1500
```



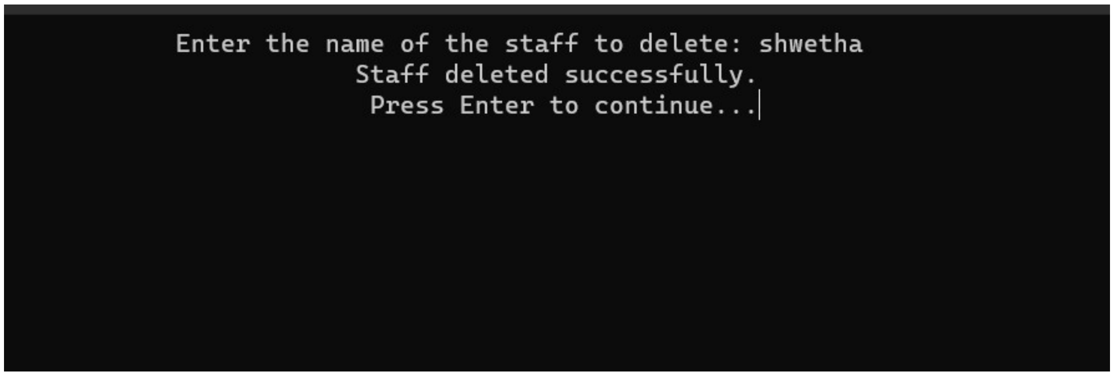
```
==============================================
            GUEST MANAGEMENT
==============================================

                      Please select an option:

               ===============================
                | 1. Add Guest             |
                | 2. Update Guest          |
                | 3. Delete Guest          |
                | 4. Search Guest          |
                | 5. Back to Main Menu     |
               ===============================

                     Enter your choice (1-5):
```



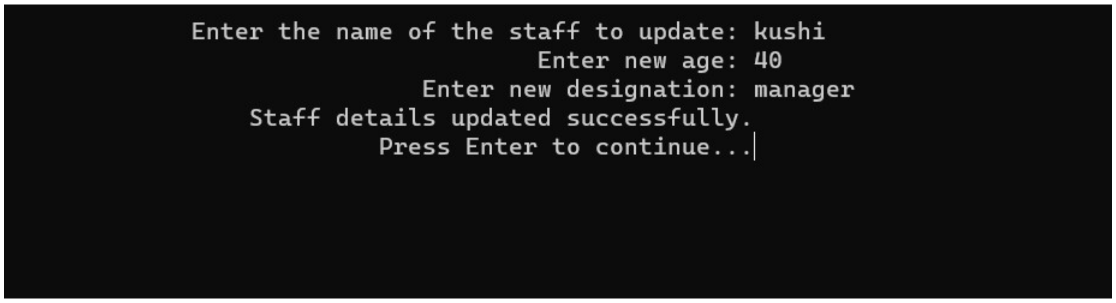
```
                Enter guest ID: 10
              Enter guest name: Jamuna Reddy
           Enter guest address: Banglore
       Guest added successfully.
      Press Enter to continue...
```



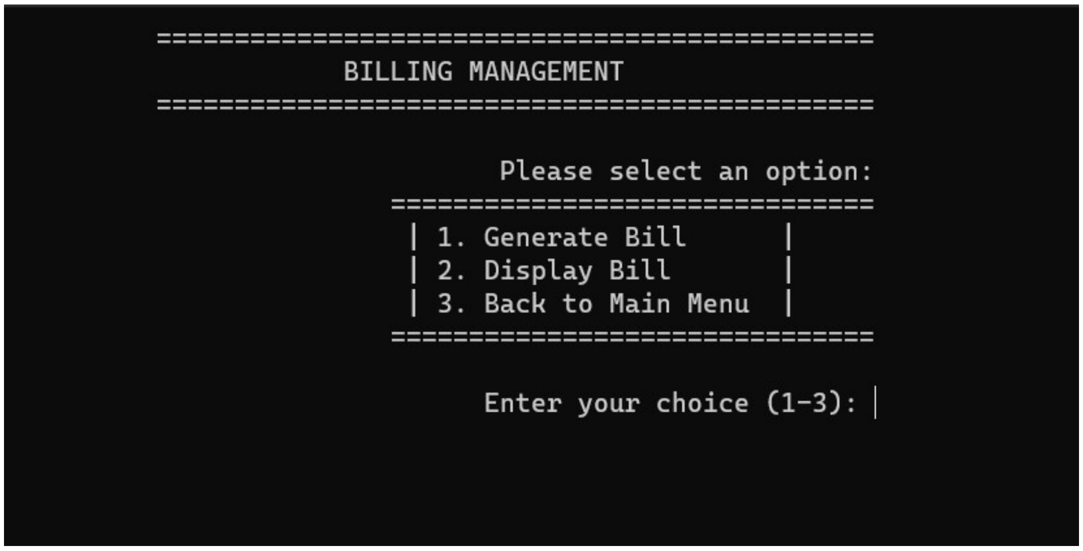
```
==============================================
             STAFF MANAGEMENT
==============================================

                      Please select an option:
               ===============================
                | 1. Add Staff             |
                | 2. Display Staff         |
                | 3. Delete Staff          |
                | 4. update Staff          |
                | 5. Back to Main Menu     |
               ===============================

                     Enter your choice (1-3):
```



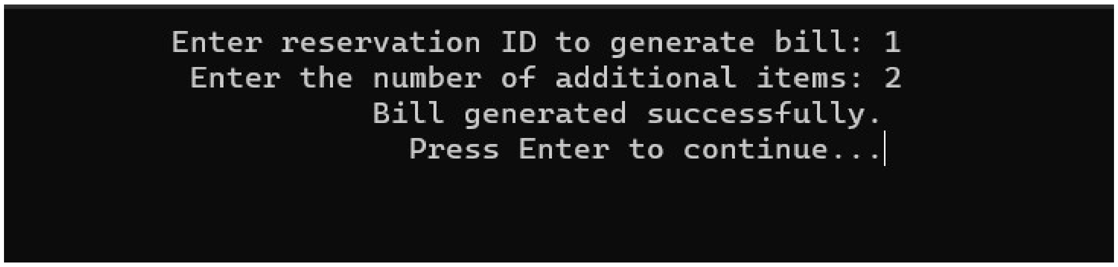
```
Enter the name of the staff to delete: shwetha
             Staff deleted successfully.
              Press Enter to continue...
```



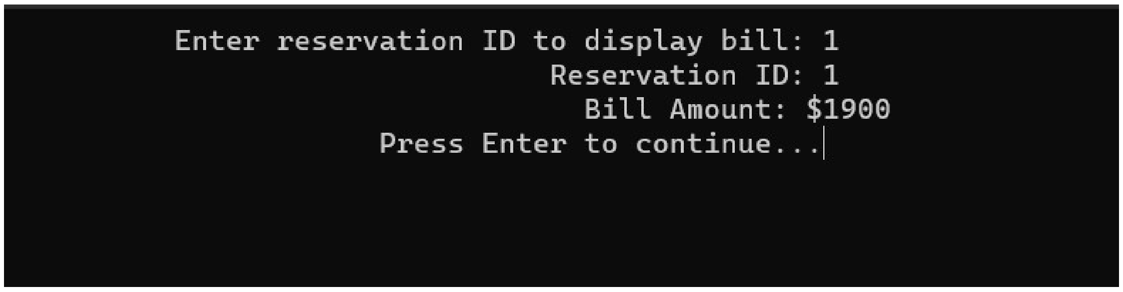
```
Enter the name of the staff to update: kushi
                        Enter new age: 40
                Enter new designation: manager
    Staff details updated successfully.
             Press Enter to continue...
```



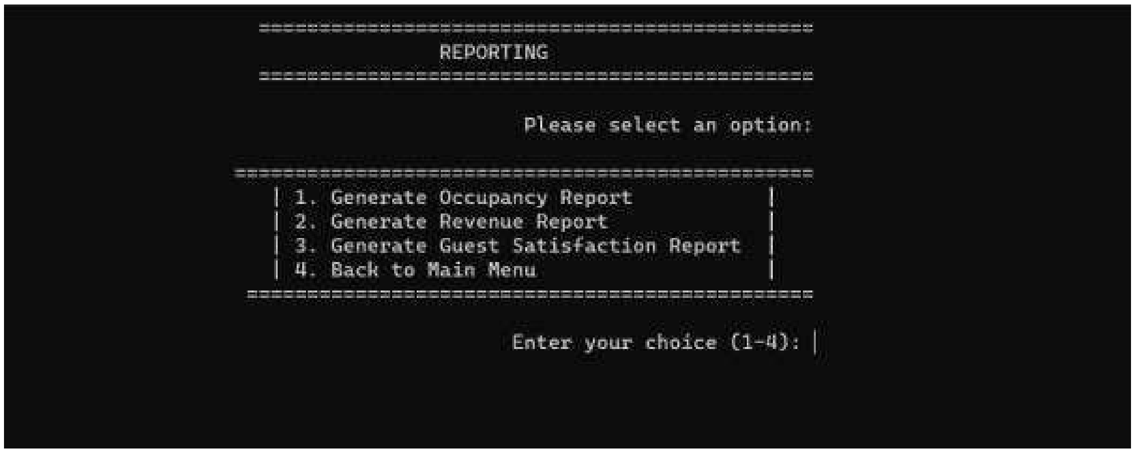
```
==================================================
            BILLING MANAGEMENT
==================================================

                       Please select an option:
                ==================================
                 | 1. Generate Bill         |
                 | 2. Display Bill          |
                 | 3. Back to Main Menu     |
                ==================================

                      Enter your choice (1-3):
```



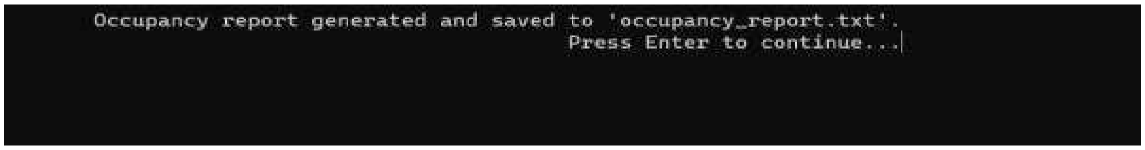
```
Enter reservation ID to generate bill: 1
 Enter the number of additional items: 2
           Bill generated successfully.
             Press Enter to continue...
```



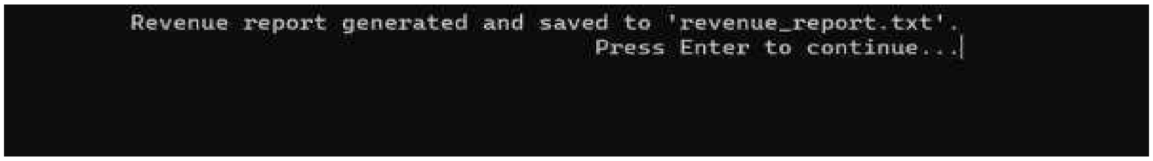
```
Enter reservation ID to display bill: 1
                     Reservation ID: 1
                       Bill Amount: $1900
           Press Enter to continue...
```



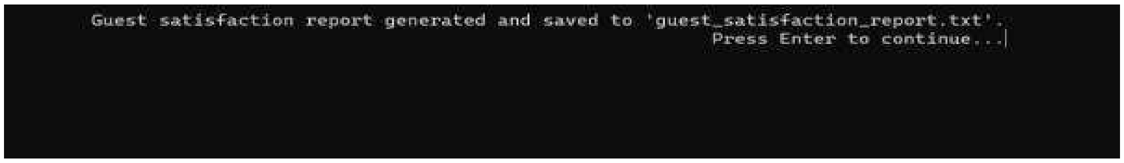
```
                    ==============================================
                                       REPORTING
                    ==============================================

                                            Please select an option:

                  ================================================
                     | 1. Generate Occupancy Report            |
                     | 2. Generate Revenue Report              |
                     | 3. Generate Guest Satisfaction Report   |
                     | 4. Back to Main Menu                    |
                   ===============================================

                                          Enter your choice (1-4):
```



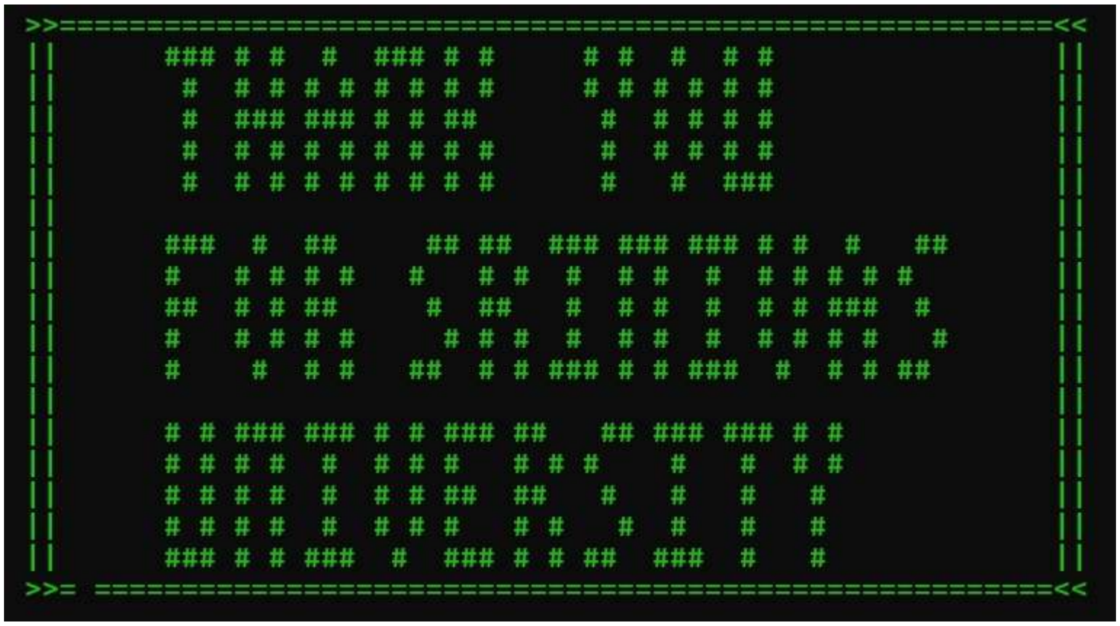
```
       Occupancy report generated and saved to 'occupancy_report.txt'.
                                     Press Enter to continue...
```



```
         Revenue report generated and saved to 'revenue_report.txt'.
                                     Press Enter to continue...
```



```
         Guest satisfaction report generated and saved to 'guest_satisfaction_report.txt'.
                                                      Press Enter to continue...
```



```
>>=========================================================<<
||      ### # #  #  ### # #     # #  #  # #                ||
||       #  # # # # # # # #     # # # # # #                ||
||       #  ### ### # # ##       #  # # # #                ||
||       #  # # # # # # # #      #  # # # #                ||
||       #  # # # # # # # #      #   #  ###                ||
||                                                         ||
||      ###  #  ##     ## ##  ### ### ### # #  #   ##      ||
||      #   # # # #   #   # #  #  # #  #  # # # # #        ||
||      ##  # # ##     #  ##   #  # #  #  # # ###  #       ||
||      #   # # # #     # # #  #  # #  #  # # # #   #      ||
||      #    #  # #   ##  # # ### # # ###  #  # # ##       ||
||                                                         ||
||      # # ### ### # # ### ##   ## ### ### # #            ||
||      # # # #  #  # # #   # # #    #   #  # #            ||
||      # # # #  #  # # ##  ##   #   #   #   #             ||
||      # # # #  #  # # #   # #   #  #   #   #             ||
||      ### # # ###  #  ### # # ##  ###  #   #             ||
>>= =======================================================<<
```

6. CONCLUSION

Hotel Management System represents a revolutionary solution for hotels seeking to modernize their operations and enhance guest experiences. By integrating and automating various functions, the HMS streamlines processes, improves efficiency, and reduces the risk of errors. It empowers hotel management to make informed.

FUTURE ENHANCEMENT

- **Online Booking:**

Implement an online booking system that allows guests to make reservations through the hotel's website or a dedicated mobile app.

Integrate real-time availability checks and secure payment processing for online bookings.

- **Mobile Check-In:**

Develop a mobile check-in feature that enables guests to check in and access their rooms using their smart-phones.

Implement digital key technology for a seamless and contactless check-in experience.

- **Integration with External Booking Platforms:**

Collaborate with popular external booking platforms (e.g., Booking.com, Expedia) to integrate the hotel's inventory and rates.

Ensure seamless synchronization of reservations and availability across all platforms.

- **Advanced Reporting and Analytics:**

Enhance the reporting and analytics module to provide more advanced insights into guest preferences, behavior, and market trends.

Implement predictive analytics to forecast occupancy rates and optimize pricing strategies.

- **Customer Relationship Management (CRM):**

Integrate a CRM system to manage guest relationships more effectively.

Collect and analyze guest data to provide personalized services and offers, improving guest loyalty.

- **Housekeeping Management:**

Develop a module for managing housekeeping operations, including room cleaning schedules and inventory tracking.

Optimize room turnaround times and maintenance schedules.

- **Restaurant and Event Management:**

Extend the system to include restaurant and event management functionalities.

Allow guests to make restaurant reservations and book event spaces, streamlining food and beverage operations.

- **Inventory Management:**

Implement an inventory management module for tracking and restocking hotel supplies, from toiletries to linens.

Optimize inventory levels to reduce waste and control costs.

- **Guest Feedback and Surveys:**

Create a system for collecting guest feedback and conducting surveys.

Use feedback data to identify areas for improvement and enhance overall guest satisfaction.

- **Security and Access Control:**

Enhance security measures with access control systems, including key card auditing and surveillance integration.

Ensure the safety and privacy of both guests and staff.

BIBILIOGRAPHY

BOOKS REFERRED:

1. "Let Us C++" – by Yeshwant5h Kanetkar, Kalyani Publications, 15th Edition, 2018.

2. "OOP's using C++" – by Rachhpal Singh, Kalyani Publishers, 5th Edition, 2010

3. "OOP using C++": A Simple Approach, by AM Padma Reddy, Nandi Publications, 2012

4. "File IO using C" – by Rashmi Eshwar, Kalyani Publications, 10th Edition, 2012

WEBSITES VISITED:

1. [Hotel Management System Overview]

(https://www.examplehoteltechwebsite.com/system-overview)

2. [OOP Concepts Explained Simple]

(https://www.tutorialpoint.com/oop_concepts-explained)

3. [File Structures - Made Simple]

(https://www.tutorialpoint.com/filestructures-basic_concepts)

*****