**DSA THEORY Project Documentation**

1. **Introduction**

This project implements a Hotel Management System designed to streamline key operations such as room allocation, updating room details, and maintaining customer records. The system leverages file handling to ensure data persistence and efficient retrieval.

**2. Features Section**

**Features**:

* **Add Room**: Allocate a new room to a customer, including customer details and room information.
* **Update Room**: Modify details of an existing room allocation, such as changing customer details.
* **Search Room**: Find room information based on room number or customer name.
* **Delete Room**: Remove room allocation from the system records.
* **File Handling**: Save and load all room and customer details for future use, ensuring data persistence.

**3. Classes and Methods Section**

**Class: Hotel**

* **Attributes**:
  + roomNo: Holds the room number.
  + customerName: Stores the customer's name.
  + next: Points to the next node in the singly linked list.

**Methods**:

* addRoom(): Adds a new room to the system.
* updateRoom(): Updates the details of an allocated room.
* searchRoom(): Searches for a room using the room number or customer name.
* deleteRoom(): Deletes a specific room allocation.
* saveToFile(): Saves the current data to a file for future reference.
* loadFromFile(): Loads data from the file when the system restarts.

**4. File Handling Section (Detailed Explanation)**

**Suggested Addition**: File handling is used to save and retrieve hotel room data persistently. Data is stored in a text file in the following format:

css

Copy code

Room Number | Customer Name | Other Details

When the program starts, it loads this data, ensuring previously allocated rooms are not lost.

**5. How to Use Section**

**How to Use**:

1. Run the program.
2. Select one of the options from the menu:
   * 1 to Add Room.
   * 2 to Update Room.
   * 3 to Search Room.
   * 4 to Delete Room.
   * 5 to Save & Exit.
3. Follow the on-screen instructions to complete the selected operation.
4. Exit the program to ensure data is saved.

**6. Conclusion Section**

In conclusion, the Hotel Management System effectively handles basic hotel operations such as room allocation and data persistence. Future improvements could include integrating features like customer billing and a graphical user interface (GUI) to enhance user experience.

**CODE**

#include <iostream>

#include <string>

#include <fstream>

#include <ctime>

#include <cctype>

using namespace std;

class Hotel {

private:

    struct Node {

        int id;

        string name;

        string room\_type;

        string date;

        Node\* next;

    };

public:

    Node\* head = NULL;

    void insertRoom();

    void updateRoom();

    void searchRoom();

    void deleteRoom();

    void sortRoom();

    void showRoom();

    void saveToFile();

    void loadFromFile();

    void menu();

};

void Hotel::menu() {

    loadFromFile();

    string input;

    int choice;

    do {

        cout << "\nWelcome to the Hotel Management System" << endl;

        cout << "1. Allocate Room" << endl;

        cout << "2. Search Room" << endl;

        cout << "3. Update Room" << endl;

        cout << "4. Delete Room" << endl;

        cout << "5. Show Room Records" << endl;

        cout << "6. Exit" << endl;

        cout << "Enter Your Choice: ";

        getline(cin, input);

        if (input.length() == 1 && isdigit(input[0])) {

            choice = stoi(input);

        } else {

            cout << "Invalid input. Please enter a number between 1 and 6." << endl;

            continue;

        }

        switch (choice) {

            case 1:

                insertRoom();

                break;

            case 2:

                searchRoom();

                break;

            case 3:

                updateRoom();

                break;

            case 4:

                deleteRoom();

                break;

            case 5:

                sortRoom();

                showRoom();

                break;

            case 6:

                saveToFile();

                cout << "Exiting program. Goodbye!" << endl;

                return;

            default:

                cout << "Invalid command. Please try again." << endl;

        }

    } while (true);

}

void Hotel::insertRoom() {

    Node\* newNode = new Node;

    cout << "Enter Room ID: ";

    cin >> newNode->id;

    cin.ignore();

    cout << "Enter Customer Name: ";

    getline(cin, newNode->name);

    cout << "Enter Room Type (Single/Double/Twin): ";

    getline(cin, newNode->room\_type);

    time\_t now = time(0);

    tm\* ltm = localtime(&now);

    newNode->date = to\_string(1900 + ltm->tm\_year) + "-" +

                    to\_string(1 + ltm->tm\_mon) + "-" +

                    to\_string(ltm->tm\_mday);

    newNode->next = NULL;

    if (head == NULL) {

        head = newNode;

    } else {

        Node\* ptr = head;

        while (ptr->next != NULL) {

            ptr = ptr->next;

        }

        ptr->next = newNode;

    }

    cout << "Room Allocated Successfully on Date: " << newNode->date << endl;

}

void Hotel::searchRoom() {

    int id;

    if (head == NULL) {

        cout << "No Rooms Found!" << endl;

        return;

    }

    cout << "Enter Room ID: ";

    cin >> id;

    Node\* ptr = head;

    while (ptr != NULL) {

        if (id == ptr->id) {

            cout << "Room ID: " << ptr->id << endl;

            cout << "Customer Name: " << ptr->name << endl;

            cout << "Room Allocated Date: " << ptr->date << endl;

            cout << "Room Type: " << ptr->room\_type << endl;

            return;

        }

        ptr = ptr->next;

    }

    cout << "Room not found!" << endl;

}

void Hotel::updateRoom() {

    int id;

    if (head == NULL) {

        cout << "No Rooms Found!" << endl;

        return;

    }

    cout << "Enter Room ID to Update: ";

    cin >> id;

    cin.ignore();

    Node\* ptr = head;

    while (ptr != NULL) {

        if (id == ptr->id) {

            cout << "Enter New Room ID: ";

            cin >> ptr->id;

            cin.ignore();

            cout << "Enter New Customer Name: ";

            getline(cin, ptr->name);

            cout << "Enter New Room Type: ";

            getline(cin, ptr->room\_type);

            time\_t now = time(0);

            tm\* ltm = localtime(&now);

            ptr->date = to\_string(1900 + ltm->tm\_year) + "-" +

                        to\_string(1 + ltm->tm\_mon) + "-" +

                        to\_string(ltm->tm\_mday);

            cout << "Room Updated Successfully!" << endl;

            return;

        }

        ptr = ptr->next;

    }

    cout << "Room ID not found!" << endl;

}

void Hotel::deleteRoom() {

    int id;

    if (head == NULL) {

        cout << "No Rooms Found!" << endl;

        return;

    }

    cout << "Enter Room ID to Delete: ";

    cin >> id;

    if (id == head->id) {

        Node\* ptr = head;

        head = head->next;

        delete ptr;

        cout << "Room Deleted Successfully!" << endl;

        return;

    }

    Node\* pre = head;

    Node\* ptr = head->next;

    while (ptr != NULL) {

        if (id == ptr->id) {

            pre->next = ptr->next;

            delete ptr;

            cout << "Room Deleted Successfully!" << endl;

            return;

        }

        pre = ptr;

        ptr = ptr->next;

    }

    cout << "Room ID not found!" << endl;

}

void Hotel::sortRoom() {

    if (head == NULL) {

        cout << "No Rooms Found!" << endl;

        return;

    }

    Node\* i = head;

    while (i != NULL) {

        Node\* j = i->next;

        while (j != NULL) {

            if (i->id > j->id) {

                swap(i->id, j->id);

                swap(i->name, j->name);

                swap(i->date, j->date);

                swap(i->room\_type, j->room\_type);

            }

            j = j->next;

        }

        i = i->next;

    }

}

void Hotel::showRoom() {

    if (head == NULL) {

        cout << "No Rooms Found!" << endl;

        return;

    }

    Node\* ptr = head;

    while (ptr != NULL) {

        cout << "\nRoom ID: " << ptr->id;

        cout << "\nCustomer Name: " << ptr->name;

        cout << "\nAllocated Date: " << ptr->date;

        cout << "\nRoom Type: " << ptr->room\_type << endl;

        ptr = ptr->next;

    }

}

void Hotel::saveToFile() {

    ofstream file("newrooms.txt", ios::trunc);

    if (!file) {

        cout << "Error opening file for saving!" << endl;

        return;

    }

    Node\* ptr = head;

    while (ptr != NULL) {

        file << ptr->id << endl;

        file << ptr->name << endl;

        file << ptr->room\_type << endl;

        file << ptr->date << endl;

        ptr = ptr->next;

    }

    file.close();

    cout << "Data saved to file successfully!" << endl;

}

void Hotel::loadFromFile() {

    ifstream file("newrooms.txt");

    if (!file) {

        cout << "No previous records found." << endl;

        return;

    }

    while (true) {

        Node\* newNode = new Node;

        if (!(file >> newNode->id)) {

            delete newNode;

            break;

        }

        file.ignore();

        getline(file, newNode->name);

        getline(file, newNode->room\_type);

        getline(file, newNode->date);

        newNode->next = NULL;

        if (head == NULL) {

            head = newNode;

        } else {

            Node\* ptr = head;

            while (ptr->next != NULL) {

                ptr = ptr->next;

            }

            ptr->next = newNode;

        }

    }

    file.close();

    cout << "Data loaded from file successfully!" << endl;

}

int main() {

    Hotel h;

    h.menu();

    return 0;

}