#include <iostream>

#include <string>

#include <vector>

#include <map>

#include <cstdlib>

#include <ctime>

using namespace std;

// Utility functions for generating random flight details

string generateRandomTime() {

    int hour = rand() % 24;

    int minute = rand() % 60;

    char buffer[6];

    snprintf(buffer, sizeof(buffer), "%02d:%02d", hour, minute);

    return string(buffer);

}

string generateRandomDate() {

    int day = rand() % 28 + 1;  // Days between 1 and 28

    int month = rand() % 12 + 1; // Months between 1 and 12

    int year = rand() % 5 + 2025; // Years between 2025 and 2029

    char buffer[11];

    snprintf(buffer, sizeof(buffer), "%02d/%02d/%04d", day, month, year);

    return string(buffer);

}

class Flight {

public:

    string flightID;

    string flightName;

    string origin;

    string destination;

    string departureTime;

    string arrivalTime;

    string date;

    int distance;

    Flight(string id, string name, string ori, string dest, string dep, string arr, string dt, int dist)

        : flightID(id), flightName(name), origin(ori), destination(dest), departureTime(dep), arrivalTime(arr), date(dt), distance(dist) {}

    void displayDetails() {

        cout << "Flight ID: " << flightID << endl;

        cout << "Flight Name: " << flightName << endl;

        cout << "Origin: " << origin << endl;

        cout << "Destination: " << destination << endl;

        cout << "Departure Time: " << departureTime << endl;

        cout << "Arrival Time: " << arrivalTime << endl;

        cout << "Date: " << date << endl;

        cout << "Distance: " << distance << " km" << endl;

    }

};

class User {

public:

    string name;

    string username;

    string mail;

    string nationality;

    string address;

    string password;

    string securityCode;

    int loyaltyPoints;

    vector<string> bookedFlights;

    User(string n, string u, string m, string nat, string addr, string pass, string sec)

        : name(n), username(u), mail(m), nationality(nat), address(addr), password(pass), securityCode(sec), loyaltyPoints(0) {}

    void displayAccountDetails() {

        cout << "Name: " << name << endl;

        cout << "Email: " << mail << endl;

        cout << "Address: " << address << endl;

        cout << "Loyalty Points: " << loyaltyPoints << endl;

        if (!bookedFlights.empty()) {

            cout << "Booked Flights: \n";

            for (const string &flightID : bookedFlights) {

                cout << "  - " << flightID << endl;

            }

        } else {

            cout << "No flights booked yet.\n";

        }

    }

};

class Booking {

public:

    string bookingID;

    string flightID;

    string userID;

    string ticketClass;

    double price;

    Booking(string bID, string fID, string uID, string tClass, double p)

        : bookingID(bID), flightID(fID), userID(uID), ticketClass(tClass), price(p) {}

    void displayBookingDetails() {

        cout << "Booking ID: " << bookingID << endl;

        cout << "Flight ID: " << flightID << endl;

        cout << "User ID: " << userID << endl;

        cout << "Ticket Class: " << ticketClass << endl;

        cout << "Price: " << price << endl;

    }

};

class AirlineManagementSystem {

private:

    vector<User> users;

    vector<Flight> flights;

    vector<Booking> bookings;

    User \*currentUser = nullptr;

    void generateFlights() {

        flights.push\_back(Flight("F001", "SkyHigh", "New York", "Los Angeles", generateRandomTime(), generateRandomTime(), generateRandomDate(), 4500));

        flights.push\_back(Flight("F002", "AirElite", "London", "Paris", generateRandomTime(), generateRandomTime(), generateRandomDate(), 340));

        flights.push\_back(Flight("F003", "FlySafe", "Tokyo", "Seoul", generateRandomTime(), generateRandomTime(), generateRandomDate(), 1150));

        flights.push\_back(Flight("F004", "Oceanic", "Sydney", "Melbourne", generateRandomTime(), generateRandomTime(), generateRandomDate(), 800));

        flights.push\_back(Flight("F005", "GlobalAir", "Dubai", "New Delhi", generateRandomTime(), generateRandomTime(), generateRandomDate(), 2200));

        flights.push\_back(Flight("F006", "JetSpeed", "San Francisco", "Chicago", generateRandomTime(), generateRandomTime(), generateRandomDate(), 3000));

        flights.push\_back(Flight("F007", "AirNova", "Mumbai", "Singapore", generateRandomTime(), generateRandomTime(), generateRandomDate(), 4000));

        flights.push\_back(Flight("F008", "SkyLux", "Berlin", "Madrid", generateRandomTime(), generateRandomTime(), generateRandomDate(), 1850));

        flights.push\_back(Flight("F009", "AirPrime", "Rome", "Athens", generateRandomTime(), generateRandomTime(), generateRandomDate(), 1200));

        flights.push\_back(Flight("F010", "CloudFly", "Cairo", "Istanbul", generateRandomTime(), generateRandomTime(), generateRandomDate(), 1600));

        flights.push\_back(Flight("F011", "QuickJet", "Toronto", "Vancouver", generateRandomTime(), generateRandomTime(), generateRandomDate(), 3300));

        flights.push\_back(Flight("F012", "AirVoyager", "Bangkok", "Kuala Lumpur", generateRandomTime(), generateRandomTime(), generateRandomDate(), 1500));

        flights.push\_back(Flight("F013", "StarFly", "Cape Town", "Johannesburg", generateRandomTime(), generateRandomTime(), generateRandomDate(), 1400));

        flights.push\_back(Flight("F014", "PolarAir", "Moscow", "Saint Petersburg", generateRandomTime(), generateRandomTime(), generateRandomDate(), 700));

        flights.push\_back(Flight("F015", "Falcon", "Beijing", "Shanghai", generateRandomTime(), generateRandomTime(), generateRandomDate(), 1200));

        flights.push\_back(Flight("F016", "Sunshine", "Miami", "Orlando", generateRandomTime(), generateRandomTime(), generateRandomDate(), 400));

        flights.push\_back(Flight("F017", "NorthStar", "Oslo", "Stockholm", generateRandomTime(), generateRandomTime(), generateRandomDate(), 500));

        flights.push\_back(Flight("F018", "JetStream", "Lisbon", "Porto", generateRandomTime(), generateRandomTime(), generateRandomDate(), 600));

        flights.push\_back(Flight("F019", "AirWave", "Havana", "Santiago", generateRandomTime(), generateRandomTime(), generateRandomDate(), 850));

        flights.push\_back(Flight("F020", "EagleFly", "Buenos Aires", "Sao Paulo", generateRandomTime(), generateRandomTime(), generateRandomDate(), 1900));

    }

    Flight \*searchFlightByRoute(const string &origin, const string &destination) {

        for (auto &flight : flights) {

            if (flight.origin == origin && flight.destination == destination) {

                return &flight;

            }

        }

        return nullptr;

    }

    Flight \*searchFlight(string flightID) {

        for (auto &flight : flights) {

            if (flight.flightID == flightID) {

                return &flight;

            }

        }

        return nullptr;

    }

public:

    AirlineManagementSystem() {

        srand(time(0));

        generateFlights();

    }

    void signUp() {

        string name, username, mail, nationality, address, password, securityCode;

        cout << "Enter Name: ";

        cin.ignore();

        getline(cin, name);

        cout << "Enter Username: ";

        cin >> username;

        cout << "Enter Mail: ";

        cin >> mail;

        cout << "Enter Nationality: ";

        cin.ignore();

        getline(cin, nationality);

        cout << "Enter Address: ";

        getline(cin, address);

        cout << "Enter Password: ";

        cin >> password;

        cout << "Enter Security Code: ";

        cin >> securityCode;

        users.emplace\_back(name, username, mail, nationality, address, password, securityCode);

        cout << "Sign-up successful!\n";

    }

    void login() {

        string identifier, password;

        cout << "Enter Username or Mail: ";

        cin >> identifier;

        cout << "Enter Password: ";

        cin >> password;

        for (auto &user : users) {

            if ((user.username == identifier || user.mail == identifier) && user.password == password) {

                currentUser = &user;

                cout << "Login successful!\n";

                return;

            }

        }

        char choice;

        cout << "Invalid password. Do you want to reset it? (y/n): ";

        cin >> choice;

        if (choice == 'y') {

            string securityCode, newPassword;

            cout << "Enter Security Code: ";

            cin >> securityCode;

            for (auto &user : users) {

                if ((user.username == identifier || user.mail == identifier) && user.securityCode == securityCode) {

                    cout << "Enter New Password: ";

                    cin >> newPassword;

                    user.password = newPassword;

                    cout << "Password reset successful!\n";

                    return;

                }

            }

            cout << "Invalid Security Code.\n";

        } else {

            cout << "Returning to menu.\n";

        }

    }

    void viewFlights() {

        cout << "Available Flights:\n";

        for (auto &flight : flights) {

            flight.displayDetails();

            cout << "\n";

        }

    }

    void searchFlights() {

        string origin, destination;

        cout << "Enter Origin: ";

        cin.ignore();

        getline(cin, origin);

        cout << "Enter Destination: ";

        getline(cin, destination);

        Flight \*flight = searchFlightByRoute(origin, destination);

        if (flight) {

            cout << "Flight found:\n";

            flight->displayDetails();

        } else {

            cout << "No flights available for the given route.\n";

        }

    }

    void bookFlight() {

        if (!currentUser) {

            cout << "Please login first to book a flight.\n";

            return;

        }

        string flightID, ticketClass;

        cout << "Enter Flight ID: ";

        cin >> flightID;

        Flight \*flight = searchFlight(flightID);

        if (!flight) {

            cout << "Invalid Flight ID.\n";

            return;

        }

        cout << "Select Ticket Class (Economy/Business/First): ";

        cin >> ticketClass;

        double basePrice = flight->distance \* 0.1;

        if (ticketClass == "Economy") basePrice \*= 1.0;

        else if (ticketClass == "Business") basePrice \*= 1.5;

        else if (ticketClass == "First") basePrice \*= 2.0;

        else {

            cout << "Invalid class selection.\n";

            return;

        }

        double discount = (currentUser->loyaltyPoints >= 100) ? basePrice \* 0.1 : 0;

        double finalPrice = basePrice - discount;

        cout << "Ticket Price: $" << basePrice << endl;

        cout << "Loyalty Discount: $" << discount << endl;

        cout << "Final Price: $" << finalPrice << endl;

        char confirm;

        cout << "Confirm booking? (y/n): ";

        cin >> confirm;

        if (confirm == 'y') {

            bookings.emplace\_back("B" + to\_string(bookings.size() + 1), flightID, currentUser->username, ticketClass, finalPrice);

            currentUser->loyaltyPoints += 10;

            currentUser->bookedFlights.push\_back(flightID);

            cout << "Booking successful! Loyalty points added.\n";

        } else {

            cout << "Booking cancelled.\n";

        }

    }

        void accountDetails() {

        if (!currentUser) {

            cout << "Please login first to view account details.\n";

            return;

        }

        cout << "\nAccount Details:\n";

        currentUser->displayAccountDetails();

    }

    void helpline() {

        cout << "\nHelpline Center:\n";

        cout << "1. For flight booking issues, contact: booking\_support@example.com\n";

        cout << "2. For baggage-related issues, contact: baggage\_support@example.com\n";

        cout << "3. For general inquiries, contact: info@example.com\n";

    }

    void menu() {

        int choice;

        do {

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

            cout << "1. Sign Up\n";

            cout << "2. Login\n";

            cout << "3. View Flights\n";

            cout << "4. Search Flights\n";

            cout << "5. Book a Flight\n";

            cout << "6. View Account Details\n";

            cout << "7. Helpline\n";

            cout << "8. Exit\n";

            cout << "Enter your choice: ";

            cin >> choice;

            switch (choice) {

                case 1:

                    signUp();

                    break;

                case 2:

                    login();

                    break;

                case 3:

                    viewFlights();

                    break;

                case 4:

                    searchFlights();

                    break;

                case 5:

                    bookFlight();

                    break;

                case 6:

                    accountDetails();

                    break;

                case 7:

                    helpline();

                    break;

                case 8:

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

                    break;

                default:

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

            }

        } while (choice != 8);

    }

};

int main() {

    AirlineManagementSystem system;

    system.menu();

    return 0;

}