//MOVIE RECOMMENDATION BACKEND   
#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

// Define structures for user, movie, and user rating

struct User {

    int id;

    string username;

    vector<string> genre\_preferences;

};

struct Movie {

    int id;

    string title;

    string genre;

    int release\_year;

};

struct UserRating {

    int user\_id;

    int movie\_id;

    int rating;

};

class MovieRecommendationSystem {

private:

    vector<User> users;

    vector<Movie> movies;

    vector<UserRating> userRatings;

public:

    // register a new user

    void registerUser(int id, string username, const vector<string>& genre\_preferences) {

        users.push\_back({id, username, genre\_preferences});

    }

    //add a new movie

    void addMovie(int id, string title, string genre, int release\_year) {

        movies.push\_back({id, title, genre, release\_year});

    }

    //rate a movie

    void rateMovie(int user\_id, int movie\_id, int rating) {

        userRatings.push\_back({user\_id, movie\_id, rating});

    }

    // recommend movies based on user's choice

    vector<Movie> recommendMovies(int user\_id) {

        vector<Movie> recommendedMovies;

        // Getting registered users preference

        vector<string> genre\_preferences = users[user\_id - 1].genre\_preferences;

        // Recommend movies with matching genres

        for (const auto& movie : movies) {

            if (find(genre\_preferences.begin(), genre\_preferences.end(), movie.genre) != genre\_preferences.end()) {

                recommendedMovies.push\_back(movie);

            }

        }

        // Recommend movies similar to those highly rated by the user

        vector<int> ratedMovies;

        for (const auto& rating : userRatings) {

            if (rating.user\_id == user\_id)

                ratedMovies.push\_back(rating.movie\_id);

        }

        for (const auto& rating : userRatings) {

            if (rating.rating >= 4 && find(ratedMovies.begin(), ratedMovies.end(), rating.movie\_id) == ratedMovies.end()) {

                recommendedMovies.push\_back(movies[rating.movie\_id - 1]);

            }

        }

        return recommendedMovies;

    }

};

int main() {

    MovieRecommendationSystem system;

    // User Registration

    system.registerUser(1, "Puniti", {"comedy", "action"}); // (user\_id, name , genre preference)

    system.registerUser(2, "Aastha", {"drama", "thriller"});

    // Add movies

    system.addMovie(1, "12th Fail", "drama", 2023); // (movie\_id, title, genre, year\_released)

    system.addMovie(2, "The Dark Knight", "action", 2008);

    system.addMovie(3, "Inception", "action", 2010);

    system.addMovie(4, "Pulp Fiction", "crime", 1994);

    system.addMovie(5,"pk","comedy",2014);

    system.addMovie(6,"mr & mrs mahi","drama",2024);

    // Rate movies

    system.rateMovie(1, 1, 5); // puniti rates "12th Fail" with 5 stars

    system.rateMovie(2, 2, 3); // aastha rates "The Dark Knight" with 3 stars

    system.rateMovie(2, 5, 4); // aastha rates "pk" with 5 stars

    system.rateMovie(1, 3, 5); // puniti rates "Inception" with 5 stars

    system.rateMovie(2, 4, 4); // aastha rates "Pulp Fiction" with 4 stars

    system.rateMovie(2, 5, 5); // puniti rates "pk" with 5 stars

    system.rateMovie(2, 6, 3); // aastha rates "mr & mrs mahi" with 3 stars

    // Recommend movies for user

    cout << "Recommended movies for Puniti:" << endl;

    vector<Movie> PunitiRecommendations = system.recommendMovies(1);

    for (const auto& movie : punitiRecommendations) {

        cout << "- " << movie.title << " (" << movie.release\_year << "), Genre: " << movie.genre << endl;

    }

    cout << "\nRecommended movies for Aastha:" << endl;

    vector<Movie> aasthaRecommendations = system.recommendMovies(2);

    for (const auto& movie : AasthaRecommendations) {

        cout << "- " << movie.title << " (" << movie.release\_year << "), Genre: " << movie.genre << endl;

    }

    return 0;

}