**TASK 4**

**RECOMMENDATION SYSTEM**

**Create a simple recommendation system that suggests items to**

**users based on their preferences. You can use techniques like**

**collaborative filtering or content-based filtering to recommend**

**movies, books, or products to users.**

**CODE:-**

import java.util.\*;

public class RecommendationSystem {

// Dummy data for movies and users' preferences

static Map<String, List<String>> userPreferences = new HashMap<>();

static Map<String, List<String>> movieGenres = new HashMap<>();

public static void main(String[] args) {

// Initialize some dummy data for user preferences

initializeData();

// Recommend movies for a specific user

String targetUser = "Alice";

System.out.println("Movies recommended for " + targetUser + " using collaborative filtering: ");

List<String> collabRecommendations = recommendUsingCollaborativeFiltering(targetUser);

System.out.println(collabRecommendations);

System.out.println("\nMovies recommended for " + targetUser + " using content-based filtering: ");

List<String> contentRecommendations = recommendUsingContentBasedFiltering(targetUser);

System.out.println(contentRecommendations);

}

// Collaborative Filtering: Recommend movies based on similar users' preferences

public static List<String> recommendUsingCollaborativeFiltering(String targetUser) {

List<String> targetUserMovies = userPreferences.get(targetUser);

Map<String, Integer> recommendationScores = new HashMap<>();

// Compare the target user's preferences with other users

for (String user : userPreferences.keySet()) {

if (!user.equals(targetUser)) {

List<String> otherUserMovies = userPreferences.get(user);

int similarityScore = getSimilarityScore(targetUserMovies, otherUserMovies);

// Add movies from similar users that the target user hasn't seen

for (String movie : otherUserMovies) {

if (!targetUserMovies.contains(movie)) {

recommendationScores.put(movie, recommendationScores.getOrDefault(movie, 0) + similarityScore);

}

}

}

}

// Sort recommendations based on scores

return getTopRecommendations(recommendationScores);

}

// Content-Based Filtering: Recommend movies based on the genres of movies the user has liked

public static List<String> recommendUsingContentBasedFiltering(String targetUser) {

List<String> targetUserMovies = userPreferences.get(targetUser);

Set<String> likedGenres = new HashSet<>();

// Identify genres of movies the user has liked

for (String movie : targetUserMovies) {

likedGenres.addAll(movieGenres.get(movie));

}

Map<String, Integer> recommendationScores = new HashMap<>();

// Recommend movies with similar genres

for (String movie : movieGenres.keySet()) {

if (!targetUserMovies.contains(movie)) {

int genreMatchCount = getGenreMatchCount(likedGenres, movieGenres.get(movie));

recommendationScores.put(movie, genreMatchCount);

}

}

return getTopRecommendations(recommendationScores);

}

// Get similarity score between two users' movie lists (Collaborative Filtering)

private static int getSimilarityScore(List<String> movies1, List<String> movies2) {

int commonCount = 0;

for (String movie : movies1) {

if (movies2.contains(movie)) {

commonCount++;

}

}

return commonCount;

}

// Count how many genres match between liked genres and the current movie (Content-Based Filtering)

private static int getGenreMatchCount(Set<String> likedGenres, List<String> movieGenres) {

int matchCount = 0;

for (String genre : movieGenres) {

if (likedGenres.contains(genre)) {

matchCount++;

}

}

return matchCount;

}

// Get top recommendations based on recommendation scores

private static List<String> getTopRecommendations(Map<String, Integer> recommendationScores) {

List<Map.Entry<String, Integer>> sortedRecommendations = new ArrayList<>(recommendationScores.entrySet());

sortedRecommendations.sort((e1, e2) -> e2.getValue() - e1.getValue());

List<String> topRecommendations = new ArrayList<>();

for (Map.Entry<String, Integer> entry : sortedRecommendations) {

if (entry.getValue() > 0) {

topRecommendations.add(entry.getKey());

}

}

return topRecommendations;

}

// Initialize dummy data for users and movie genres

private static void initializeData() {

// User preferences: users and the movies they have watched/liked

userPreferences.put("Alice", Arrays.asList("Movie1", "Movie3", "Movie5"));

userPreferences.put("Bob", Arrays.asList("Movie2", "Movie4", "Movie5"));

userPreferences.put("Charlie", Arrays.asList("Movie1", "Movie4", "Movie6"));

userPreferences.put("Dave", Arrays.asList("Movie3", "Movie6", "Movie7"));

// Movie genres: movies and their associated genres

movieGenres.put("Movie1", Arrays.asList("Action", "Adventure"));

movieGenres.put("Movie2", Arrays.asList("Action", "Sci-Fi"));

movieGenres.put("Movie3", Arrays.asList("Comedy", "Drama"));

movieGenres.put("Movie4", Arrays.asList("Sci-Fi", "Adventure"));

movieGenres.put("Movie5", Arrays.asList("Action", "Adventure"));

movieGenres.put("Movie6", Arrays.asList("Drama", "Romance"));

movieGenres.put("Movie7", Arrays.asList("Comedy", "Romance"));

}

}

**OUTPUT**:-

