Task 3

RECOMMENDATION SYSTEM

Code:

```
import java.util.*;


public class RecommendationSystem {


    // Sample data: user ratings for items

    private static final Map<String, Map<String, Double>> userRatings = new
HashMap<>();


    static {

        userRatings.put("User1", Map.of("Item1", 5.0, "Item2", 3.0, "Item3", 2.5));

        userRatings.put("User2", Map.of("Item1", 2.0, "Item2", 2.5, "Item3", 5.0,
"Item4", 2.0));

        userRatings.put("User3", Map.of("Item1", 2.5, "Item3", 4.0, "Item4", 4.5));

        userRatings.put("User4", Map.of("Item2", 3.5, "Item3", 4.0, "Item4", 5.0));

        userRatings.put("User5", Map.of("Item1", 3.0, "Item2", 4.0, "Item3", 2.0,
"Item4", 3.0));

    }


    public static void main(String[] args) {

        String user = "User5";

        List<String> recommendations = getRecommendations(user);

        System.out.println("Recommendations for " + user + ": " +
recommendations);

    }
```

```
    private static List<String> getRecommendations(String user) {
        Map<String, Double> userRatings =
RecommendationSystem.userRatings.get(user);
        Map<String, Double> totalScores = new HashMap<>();
        Map<String, Integer> similarityCounts = new HashMap<>();

        for (String otherUser : RecommendationSystem.userRatings.keySet()) {
            if (!otherUser.equals(user)) {
                double similarity = calculateSimilarity(userRatings,
RecommendationSystem.userRatings.get(otherUser));

                if (similarity > 0) {
                    for (Map.Entry<String, Double> entry :
RecommendationSystem.userRatings.get(otherUser).entrySet()) {
                        if (!userRatings.containsKey(entry.getKey())) {
                            totalScores.merge(entry.getKey(), similarity *
entry.getValue(), Double::sum);
                            similarityCounts.merge(entry.getKey(), 1, Integer::sum);
                        }
                    }
                }
            }
        }

        Map<String, Double> rankings = new HashMap<>();
        for (String item : totalScores.keySet()) {
            rankings.put(item, totalScores.get(item) / similarityCounts.get(item));
        }
```

```
        List<Map.Entry<String, Double>> sortedRankings = new
ArrayList<>(rankings.entrySet());
        sortedRankings.sort((a, b) -> Double.compare(b.getValue(), a.getValue()));

        List<String> recommendations = new ArrayList<>();
        for (Map.Entry<String, Double> entry : sortedRankings) {
            recommendations.add(entry.getKey());
        }

        return recommendations;
    }

    private static double calculateSimilarity(Map<String, Double> ratings1,
Map<String, Double> ratings2) {
        double sum = 0;
        int count = 0;

        for (String item : ratings1.keySet()) {
            if (ratings2.containsKey(item)) {
                sum += ratings1.get(item) * ratings2.get(item);
                count++;
            }
        }

        if (count == 0) return 0;
```

```
        double norm1 =
Math.sqrt(ratings1.values().stream().mapToDouble(Double::doubleValue).sum()
);
        double norm2 =
Math.sqrt(ratings2.values().stream().mapToDouble(Double::doubleValue).sum()
);


        return sum / (norm1 * norm2);
    }
}
```