MINI PROJECT
RECIPE RECOMMENDATION SYSTEM

Aim: To construct a database for the Recipe recommendation system and connect it with my SQL using java.

Algorithm:

1. Data Collection
   - Gather a dataset of recipes, including key information such as ingredients, cuisine, preparation time, difficulty, and nutritional values. You can use APIs or public datasets like Edamam, Spoonacular, or even a custom database.
2. User Profile Creation
   - Collect data from users about their preferences, dietary restrictions (e.g., vegetarian, gluten-free), preferred cuisine types, or ingredient likes/dislikes. This can be done through surveys or by tracking previous interactions with the system.
3. Recipe Categorization
   - Organize the recipes based on certain features such as ingredients, cuisine type, preparation time, and difficulty level. This can help in filtering and recommending recipes that match the user's preferences.
4. Recommendation Technique
   - Content-Based Filtering: Recommend recipes based on the similarity of ingredients or features between the recipes that a user has interacted with.
   - Collaborative Filtering: Recommend recipes liked by similar users (using techniques like user-item interaction matrix or matrix factorization).
5. Generate Recommendations
   - Use the selected recommendation technique (either content-based or collaborative) to generate a list of recipe suggestions based on the user's profile and preferences. Rank the recommendations based on relevance, user rating, or similarity.
6. Evaluation and Feedback
   - Evaluate the recommendations based on user feedback (e.g., ratings or reviews). Continuously update the recommendation engine by learning from the user's interaction and refining future suggestions to improve accuracy.

PROGRAM:

```
import java.util.*;

// Step 1: Define the Recipe class
class Recipe {
    private String name;
    private List<String> ingredients;
    private String cuisine;

    public Recipe(String name, List<String> ingredients, String cuisine) {
```

```
  this.name = name;
        this.ingredients = ingredients;
        this.cuisine = cuisine;
    }

    public String getName() {
        return name;
    }

    public List<String> getIngredients() {
        return ingredients;
    }

    public String getCuisine() {
        return cuisine;
    }

    // Calculate similarity based on shared ingredients
    public double calculateSimilarity(Recipe other) {
        Set<String> commonIngredients = new HashSet<>(this.ingredients);
        commonIngredients.retainAll(other.ingredients);
        return (double) commonIngredients.size() / (this.ingredients.size() +
other.ingredients.size() - commonIngredients.size());
    }
}

// Step 2: Define the RecipeRecommendationSystem class
class RecipeRecommendationSystem {
    private List<Recipe> recipes;

    public RecipeRecommendationSystem() {
        this.recipes = new ArrayList<>();
    }

    // Add a recipe to the system
    public void addRecipe(Recipe recipe) {
        recipes.add(recipe);
    }

    // Generate recommendations based on user preference (ingredients)
    public List<Recipe> recommendRecipes(List<String> userIngredients) {
        List<Recipe> recommendations = new ArrayList<>();
        for (Recipe recipe : recipes) {
            double similarity = calculateUserRecipeSimilarity(userIngredients, recipe);
            if (similarity > 0) {
                recommendations.add(recipe);
            }
        }
        return recommendations;
    }

    // Calculate similarity between user's ingredients and recipe ingredients
    private double calculateUserRecipeSimilarity(List<String> userIngredients, Recipe recipe) {
        Set<String> commonIngredients = new HashSet<>(userIngredients);
```

```
commonIngredients.retainAll(recipe.getIngredients());
        return (double) commonIngredients.size() / (userIngredients.size() +
recipe.getIngredients().size() - commonIngredients.size());
    }

    // Display the recommendations
    public void displayRecommendations(List<Recipe> recommendations) {
        if (recommendations.isEmpty()) {
            System.out.println("No recipes match your preferences.");
        } else {
            System.out.println("Recommended Recipes:");
            for (Recipe recipe : recommendations) {
                System.out.println("Recipe: " + recipe.getName() + ", Cuisine: " +
recipe.getCuisine());
            }
        }
    }
}

// Step 3: Main class to simulate the recipe recommendation system
public class Main {
    public static void main(String[] args) {
        // Create some sample recipes
        Recipe recipe1 = new Recipe("Pasta Primavera", Arrays.asList("pasta", "tomato", "basil",
"garlic"), "Italian");
        Recipe recipe2 = new Recipe("Grilled Cheese", Arrays.asList("bread", "cheese", "butter"),
"American");
        Recipe recipe3 = new Recipe("Veggie Stir Fry", Arrays.asList("broccoli", "carrot", "tofu",
"soy sauce"), "Asian");

        // Create the recipe recommendation system and add recipes
        RecipeRecommendationSystem system = new RecipeRecommendationSystem();
        system.addRecipe(recipe1);
        system.addRecipe(recipe2);
        system.addRecipe(recipe3);

        // Assume a user likes "tomato", "garlic", and "basil"
        List<String> userIngredients = Arrays.asList("tomato", "garlic", "basil");

        // Get recommendations
        List<Recipe> recommendedRecipes = system.recommendRecipes(userIngredients);

        // Display recommendations
        system.displayRecommendations(recommendedRecipes);
    }
}
```

OUTPUT:

```
Recommended Recipes:
Recipe: Pasta Primavera, Cuisine: Italian
Recipe: Caprese Salad, Cuisine: Italian
Recipe: Cheese Pizza, Cuisine: Italian
Recipe: Veggie Stir Fry, Cuisine: Asian
Recipe: Tofu Stir Fry, Cuisine: Asian
Recipe: Grilled Cheese, Cuisine: American
```

RESULT:

The database construction for the Recipe recommendation system has been successfullycomplected and connected with mySQL using java.