**Recipe Management System Project Details**

**Project Overview:**

The **Recipe Management System** is a Java-based application designed to help users store, search, view, and delete their recipes. It allows users to manage recipes by adding, updating, and searching for recipes based on their names or ingredients. This project offers a simple user interface using the console, making it easy to understand and operate.

**Core Features:**

1. **Add Recipe**: Allows users to input details of a new recipe including the name, ingredients, steps, and cooking time.
2. **View All Recipes**: Displays a list of all saved recipes along with their ingredients, cooking time, and preparation steps.
3. **Search Recipes by Name**: Enables users to search for a recipe by its name.
4. **Search Recipes by Ingredient**: Allows users to search for recipes that include a specific ingredient.
5. **Delete Recipe**: Users can delete a recipe by specifying its name.
6. **Exit**: The program will exit when the user selects this option.

**How the System Works:**

1. **Recipe Class**:
   * This class contains the basic structure of a recipe. It holds attributes like:
     + name: Name of the recipe.
     + ingredients: A list of ingredients used in the recipe.
     + steps: The steps required to prepare the recipe.
     + cookingTime: The time required to cook the recipe in minutes.
   * The class also includes a method displayRecipe() to print the details of the recipe.
2. **RecipeManager Class**:
   * This is the main class where all interactions happen.
   * It uses an ArrayList to store the list of recipes.
   * Users can add, view, search, and delete recipes using the options provided in the console menu.
   * The program provides a simple text-based menu with options like "Add Recipe," "View All Recipes," "Search Recipe by Name," and so on.
   * All recipes are stored in memory (i.e., no files are saved in this version of the project).
3. **User Input Handling**:
   * The program uses a **Scanner** object to take input from the user.
   * The user is prompted to input the name, ingredients, cooking time, and steps when adding a new recipe.
   * Users can also search for recipes by name or ingredient and delete them by entering the recipe name.

**Functional Flow:**

1. **Menu Display**: The program continuously displays a menu with options like Add, View, Search, Delete, and Exit.
2. **Add Recipe**: On choosing "Add Recipe," the user is asked to input:
   * Name of the recipe.
   * Ingredients (comma-separated).
   * Cooking time in minutes.
   * Cooking steps.
3. **View All Recipes**: Displays all the recipes stored in the recipes list.
4. **Search by Name**: The user is prompted to search for a recipe by its name, and the program will display the matching recipe (if found).
5. **Search by Ingredient**: The user can search for recipes that contain a specific ingredient.
6. **Delete Recipe**: The user enters the name of the recipe to be deleted, and it is removed from the list.
7. **Exit**: The program exits when the user selects the Exit option.

**Project Enhancements (Optional for Advanced Users):**

* **File I/O**: Instead of keeping recipes in memory, you can implement **file storage** (e.g., .txt or .dat files) so the recipes are saved between program runs.
* **Recipe Categorization**: Add categories for recipes like "Breakfast," "Lunch," "Dinner," and allow the user to filter by category.
* **Graphical User Interface (GUI)**: Implement a GUI using **Swing** or **JavaFX** to make the project more interactive.
* **Sorting Recipes**: Sort the recipes by cooking time or name.
* **Data Validation**: Add input validation for ingredients, cooking time, etc.