

University of prince Mugrin

College of Computer and

cyber sciences



TastyTrack

Customized Recipe Finder

java project

Cs112

Instructor: Dr. Maha Abdulkhabeer

**Karemah baker (4310154)**

**Ferdows Adam (4313204)**

**Dania Talal (4310292)**

**Table of Contents**

[INTRODUCTION: 2](#_Toc166355726)

[detailed code analysis: 3](#_Toc166355727)

[ Class and Files 3](#_Toc166355728)

[1. HelloApplication (The Mine Class) 3](#_Toc166355729)

[2. Meal Class 3](#_Toc166355730)

[3. Recipes Class 4](#_Toc166355731)

[4. Three Files 4](#_Toc166355732)

[ The Main Methods 4](#_Toc166355733)

[1. Method (fromFileToHashMap) 4](#_Toc166355734)

[2. Method (getStringIntegerMap) 5](#_Toc166355735)

[3. Method (suggestionRecipes): 7](#_Toc166355736)

[Project Development Challenges 8](#_Toc166355737)

[1. showing all the recipes that contain these ingredients even if it just one. 8](#_Toc166355738)

[2. Reading the correct meal file when the user selects the meal’s type. 9](#_Toc166355739)

[4. Writing the recipes in the file in an appropriate way 9](#_Toc166355740)

[5. Dealing with the information in flexible way 10](#_Toc166355741)

[Future Prospects 11](#_Toc166355742)

[1. Integration with a Large Database 11](#_Toc166355743)

[2. Embedded Video Integration 11](#_Toc166355744)

[3. Nutritional information 11](#_Toc166355745)

[4. Social Sharing Features 12](#_Toc166355746)

[5. Personalized Recommendations 12](#_Toc166355747)

[program’s code 13](#_Toc166355748)

[ The main class 13](#_Toc166355749)

[ Recipes class 14](#_Toc166355751)

[ Meal class 15](#_Toc166355752)

[ files 16](#_Toc166355753)

[ Run the code 17](#_Toc166355754)

[Students work 18](#_Toc166355755)

[concloution 19](#_Toc166355756)

[references 20](#_Toc166355757)

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | | --- | | java project | | TastyTrack |     INTRODUCTION:  In a fast-paced and varied society, individuals frequently experience confusion and time waste while making even the most basic everyday choices. This occurs when they struggle to determine what aligns with their preferences and what options are accessible to them. This is particularly accurate when it comes to selecting meals because of the extensive range of options accessible. This uncertainty may prompt individuals to choose the easier and quicker option of consuming harmful fast food, which has an adverse effect on their long-term health and physical well-being. Therefore, there emerged a need for a resolution that streamlines the process of choosing healthy homemade meals to eliminate this confusion and reduce the dependence on eating out.  The goal of our Java project is to assist users in selecting appropriate recipes for their meals and preferences, considering the ingredients within their reach. The system allows users to input the items they want to use for cooking. By using three primary methods, the system generates the top five recipes that use the specified components. Additionally, it provides a hyperlink that provides instructions on how to prepare these meals. An interactive interface that uses JavaFX and all the requirements from the course (CS112) accomplishes all this. |

detailed code analysis:

## Class and Files

1. HelloApplication (The Mine Class)

The HelloApplication class is the main class for our project. It provides an interactive interface for users to input their meal preferences and available ingredients. It handles user interactions, processes input data, and displays relevant recipe suggestions, enhancing the user's ability to make informed meal choices using the ingredients they have on hand. This implementation leverages JavaFX for the GUI and integrates file reading based on the meal type that the user chooses then, creating a Meal object, and invoking the suggestionRecipes method to get the top 5 recipes containing the given ingredients.

1. Meal Class

The Meal class is the core component where all operations related to recipe suggestions take place based on user-provided ingredients. It stores recipes, matches them with available ingredients, sorts them by relevance, and returns the best matches. Additionally, it includes functionality to read recipes from a file and convert them into a usable format, making it the central operational component of the recipe recommendation system.

1. Recipes Class

Is a simple data structure class used to store information about individual recipes. It encapsulates the ingredients and the link associated with a recipe, providing a structured way to manage and access recipe data. This class is used by other parts of the system, such as the Meal class, to store, retrieve, and manipulate recipe information effectively.

1. Three Files

We created three associated files for each meal—breakfast, lunch, and dinner—that contain 20 suggested recipes in a specific syntax to ease the reading process and to differentiate each part of it, which are the recipe name, the ingredient, and the link.

## The Main Methods

1. **Method (fromFileToHashMap)**

- **Method Name and Signature:**

public static Map<String, Recipes> fromFileToHashMap(BufferedReader file) throws IOException.

-**Description:**

The fromFileToHashMap method reads data from a file represented by a BufferedReader object and constructs a HashMap containing recipe names as keys and Recipes objects as values.

- **Parameters:**

BufferedReader file: Represents the file to be read.

**- Return Value:**

Map<String, Recipes>: A HashMap where the keys are recipe names (as strings) and the values are Recipes objects containing ingredients and instructions.

- **Functionality:**

The method reads each line from the file, splits it into a list of strings based on the delimiter "--", and constructs a Recipes object using the elements of the list. It then adds this object to the HashMap with the recipe name as the key.

- **Exception Handling:**

The method declares that it throws IOException in case of any input/output errors encountered while reading the file.

2. **Method (getStringIntegerMap)**

- **Method Name and Signature:**

private Map<String, Integer> getStringIntegerMap(List<String> availableIngredients)

**- Description:**

The getStringIntegerMap method generates a mapping of recipes to their respective ingredient match counts based on the user's available ingredients. This method is important for determining which recipes best match the ingredients the user has on hand, facilitating the suggestion of the most relevant recipes.

**- Parameters:**

availableIngredients: A List<String> representing the ingredients available to the user for meal preparation.

**- Return Value:**

Map<String, Integer>: A map where each key-value pair represents a recipe name (as a string) and its corresponding ingredient match count (as an integer).

**- Functionality:**

* Initialize Map: An empty map(matchingRecipes) is initialized to store the recipe names and their ingredient match counts.
* Iterate Through Recipes: The method iterates through each entry in the choice map, which contains all available recipes.
* Extract Recipe Details: For each recipe entry, it extracts the recipe name, required ingredients, and recipe link.
* Match Counting: A counter variable match is initialized to zero to keep track of the number of matching ingredients.
* Check Ingredients: The method iterates through each ingredient in the availableIngredients list ,If the required ingredients of the recipe contain the current ingredient, the match counter is incremented.
* Store Results: After checking all ingredients, the method stores the recipe name, recipe link, and match count in the (matchingRecipes) map. The key is formatted as "{recipeName} for further information: {link}".
* Return Map: Finally, the method returns the (matchingRecipes) map, which contains the mapping of recipes to their ingredient match counts.

1. **Method (suggestionRecipes):**

**- Method Name and Signature:**

suggestionRecipes(List<String> availableIngredients)

**- Description:**

The suggestionRecipes method is responsible for generating a list of the top 5 recipe suggestions based on the user's available ingredients. It calculates the match count for each recipe and returns the names of the recipes with the highest match counts.

**- Parameters:**

availableIngredients: A List<String> representing the ingredients available to the user for meal preparation.

**- Return Value:**

List<String>: A list of the names of the top 5 recipes that best match the available ingredients, each followed by a link for further details.

- **Functionality**:

* Initialize List: An empty list (suggestedRecipes) is initialized to store the names of the top 5 suggested recipes.
* Generate Match Counts: The method calls getStringIntegerMap(availableIngredients) to generate a map of recipes to their ingredient match counts.
* Sort Recipes: The entries in the matchingRecipes map are converted into a list and sorted in descending order based on the match counts.
* Select Top Recipes: The top 5 recipes are selected from the sorted list: The method iterates over the first 5 entries of the sorted list and adds the recipe names to the suggestedRecipes list.
* Return List: Finally, the method returns the suggestedRecipes list containing the names of the top 5 suggested recipes along with links for further information.

Project Development Challenges

1. showing all the recipes that contain these ingredients even if it just one.

The challenge was how to make the program show specific recipes with the highest matching number of ingredients in descending order,not all recipes that contain it.

To solve this problem, we make the(getStringInteger)method that matching user’s ingredients with the available recipes in the file after reading it and then using (suggestionRecipes) method that invoke the (getStringInteger) method to be able to sort it in descending order, after that, using a for loop to the sorted recipes entries to add just 5 recipes in (suggestedRecipes) list.

1. Reading the correct meal file when the user selects the meal’s type.

The problem of reading the correct meal file based on the user's selection is solved by using a switch statement within the (suggestButton) event handler in the HelloApplication class. When the user selects a meal type (Breakfast, Lunch, or Dinner) from the combo box and clicks the (Suggest Recipes) button, the event handler performs the operations, one of them is entering the switch statement blook and read the file depending on the case which is the meal that is chosen by the user.

1. Writing the recipes in the file in an appropriate way

The challenge was to find a way to write the recipes in the files in a specific syntax to be able to show, in the end, only the name of the recipes with the link.

To solve this problem, we wrote the ingredients in this syntax ( recipe name -- ingredients(separated by commas) -- link) ,to be able to deal with each part of the recipe in a flexible way. and then by using the for loop in (getStringIntegerMap) with couple of statements , it returns ( (name + link) as string and match ingredient number as an integer ), however in the for loop in (suggestionRecepies) method it returns the name+link only after sorting.

1. Dealing with the information in flexible way

The challenge was in finding and learning about a new data type that gives us the flexibility and freedom in dealing with the recipes by sorting, rearrange, and adding them easily so, we use flexible data types like List<String>, Map<String, Recipes>, and Map<String, Integer> to efficiently handle information. List<String> manages collections of ingredients, allowing easy manipulation and iteration. Map<String, Recipes> stores recipe names and their associated Recipes objects, which contain ingredients and links. Map<String, Integer> maps recipe names to their ingredient match counts.These data types enable our application to process user input, manage recipe data, and provide relevant suggestions based on available ingredients effectively.

Future Prospects

As part of our ongoing commitment to enhancing the Meal Suggestion Application, we have identified several areas for future development and improvement. These plans aim to further enrich the user experience, expand the application's functionality, and ensure its continued relevance in meeting the evolving needs of our users. Below, we outline the potential future plans:

1. Integration with a Large Database

With access to a larger dataset, the application will be able to perform more robust ingredient matching and recipe recommendation algorithms. This means that users will receive more accurate and relevant suggestions based on the ingredients they have on hand, as well as their individual tastes and dietary requirements.

1. Embedded Video Integration

Enhance user engagement by integrating video tutorials or cooking demonstrations directly within the application. Users can visually follow along with cooking instructions, making the recipe preparation process more interactive and accessible, especially for novice cooks.

1. Nutritional information

Enhance recipe listings with detailed nutritional information, including calorie counts, macronutrient breakdowns, and allergen warnings. This empowers users to make informed dietary choices and promotes healthier eating habits by providing transparent nutritional data for each recipe.

1. Social Sharing Features

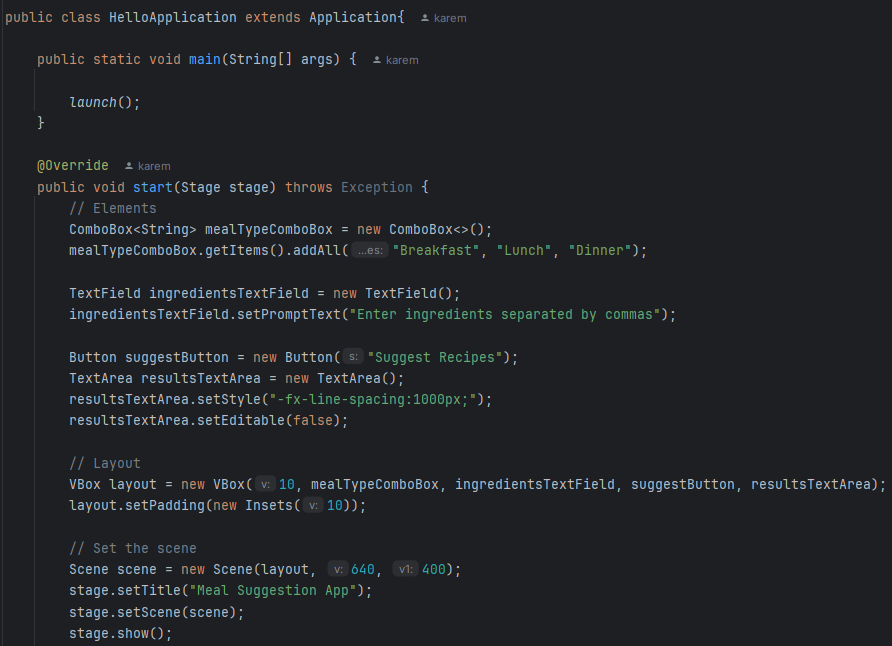
Integrate social media sharing functionality, allowing users to easily share their favorite recipes, cooking tips, and culinary creations with friends and followers on platforms like Instagram, and Pinterest. This fosters a sense of community and encourages user engagement and interaction within the application.

1. Personalized Recommendations

Implement machine learning algorithms or user profiling techniques to analyze user preferences, dietary restrictions, and cooking habits. Based on this data, provide personalized recipe recommendations tailored to each user's individual tastes, dietary requirements, and cultural preferences.

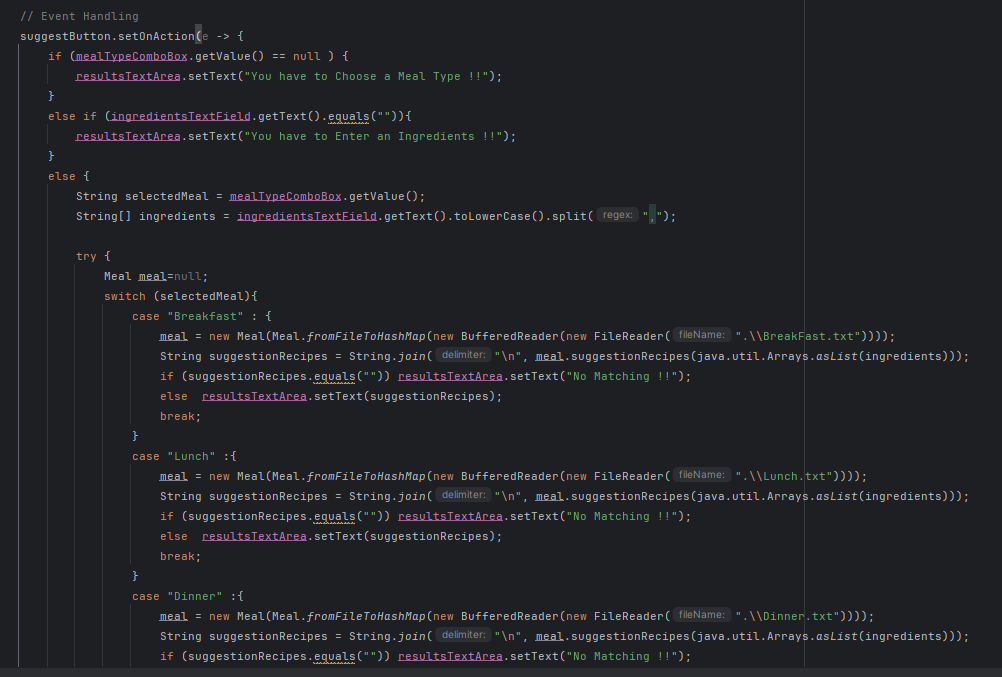
program’s code

The main class



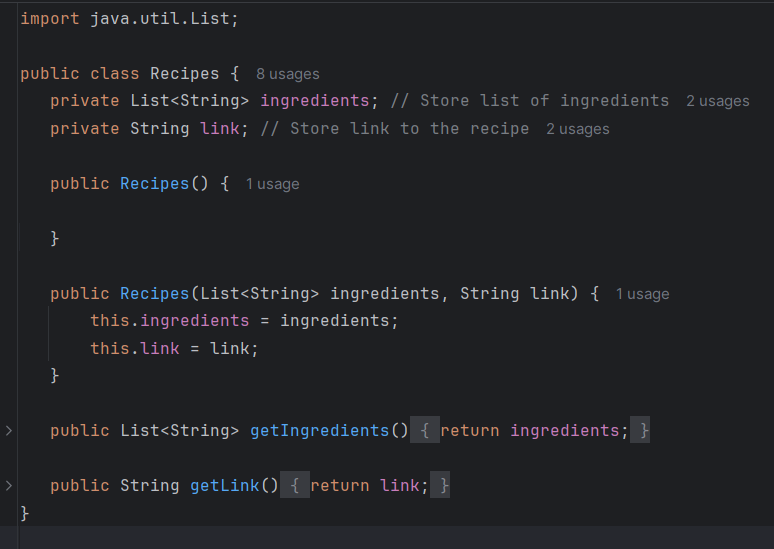
JavaFX

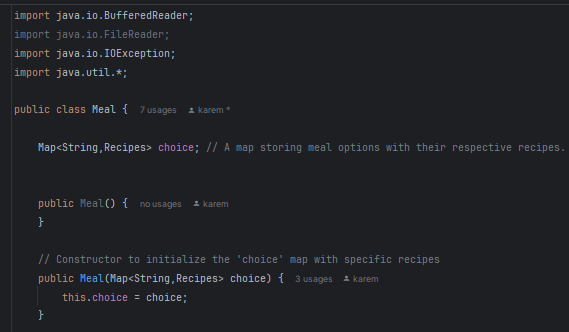
Imported classes.

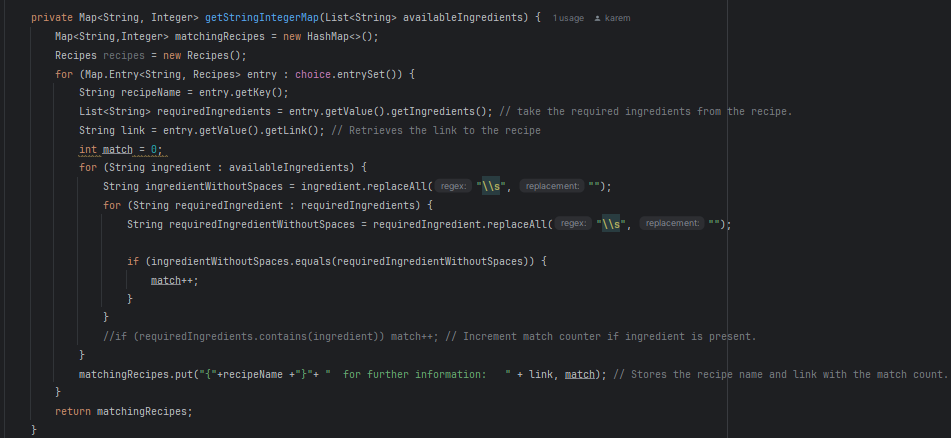
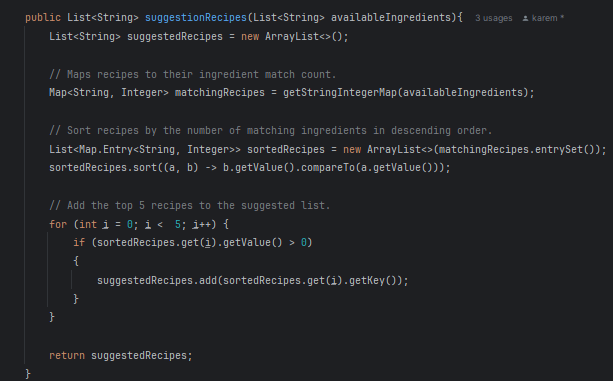


Starting the program

Recipes class



Meal class

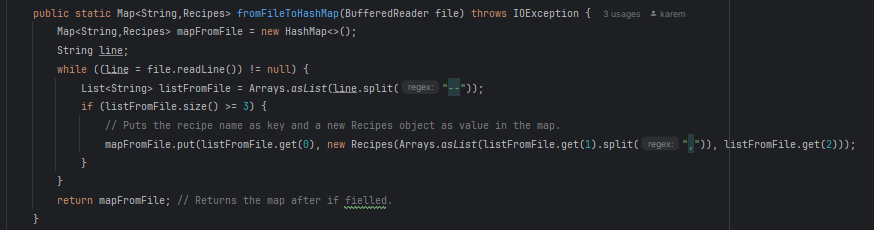
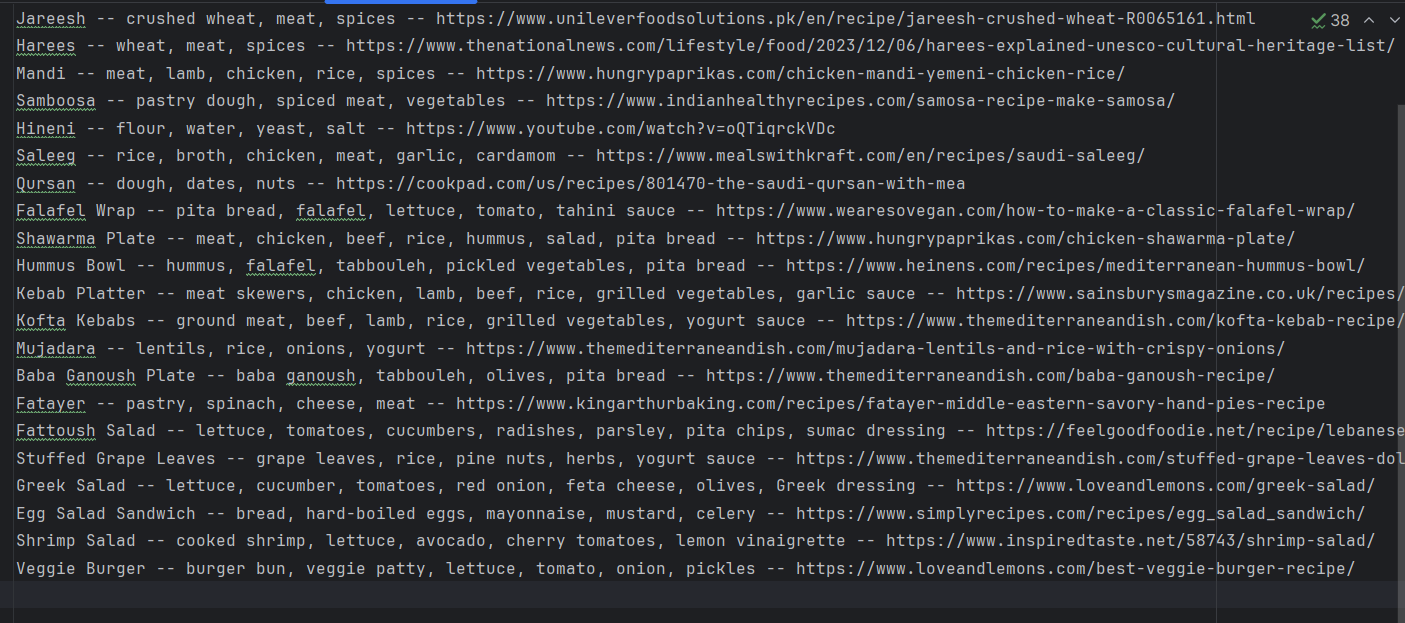
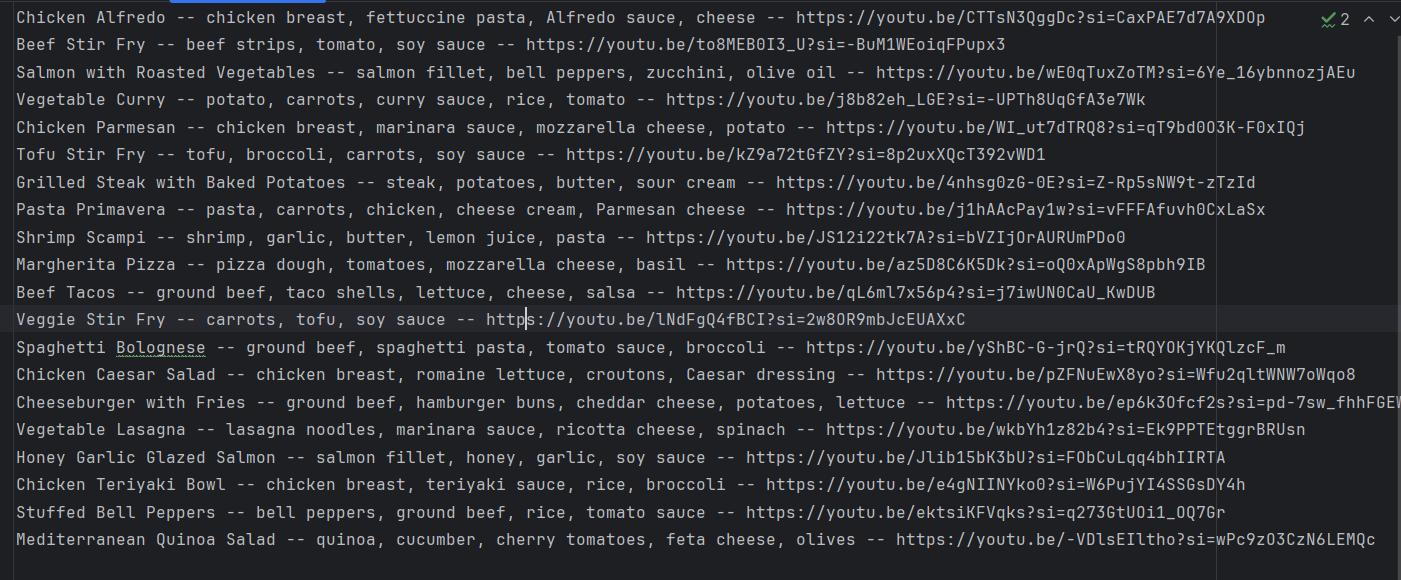
****

The beginning of the class

getStringIntegerMap method

sugestionRecipes method

A screen shot of a computer

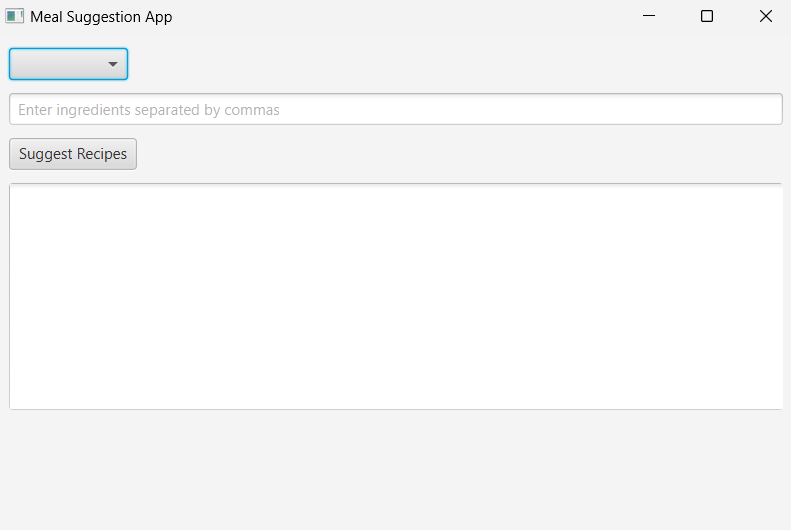
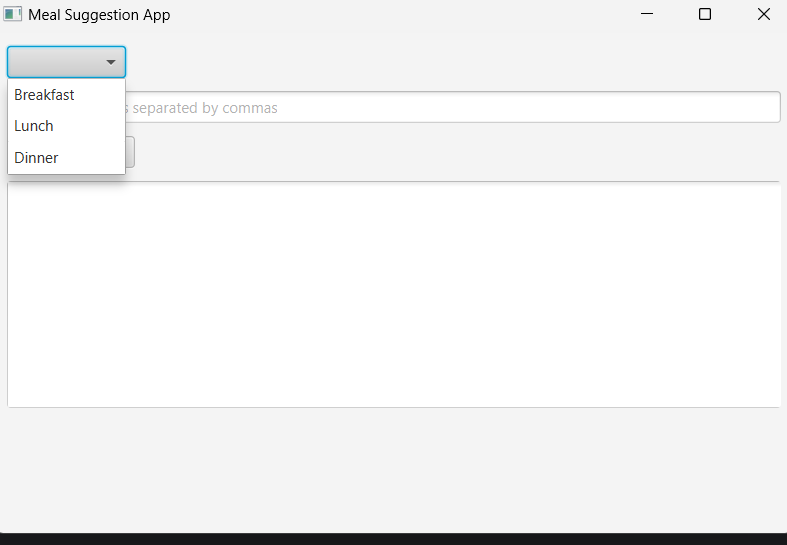
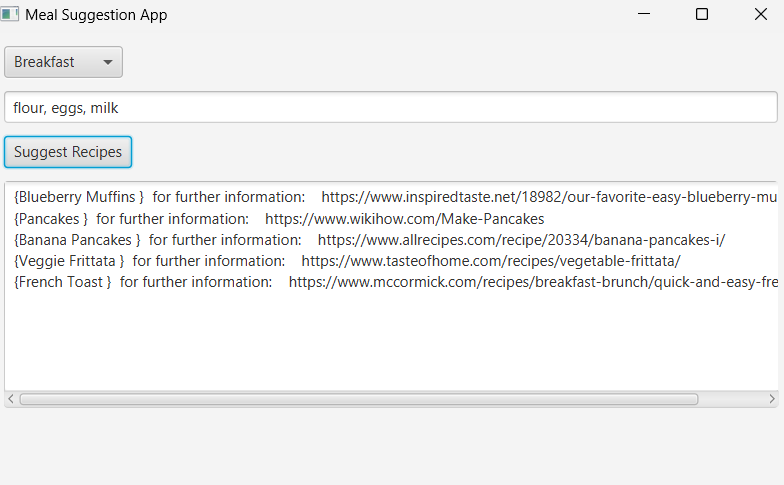
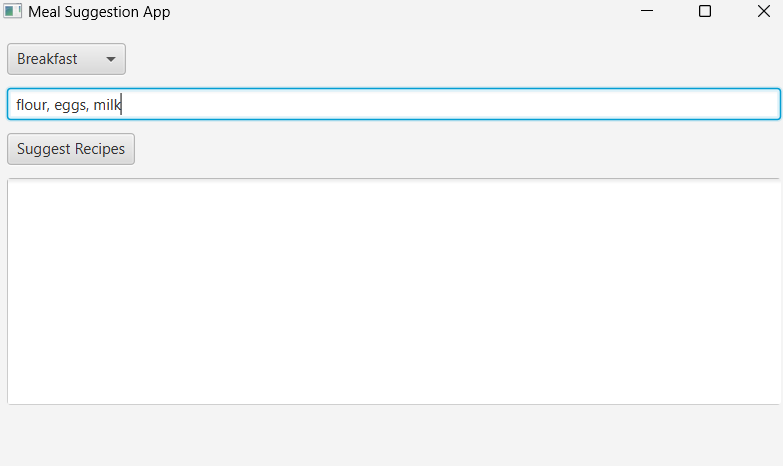
Description automatically generatedfiles

Breakfast recipes

Dinner recipes

lunch recipes.

fromFileToHashMap method

Run the code

Writing ingredients

Press suggest recipes

Choosing the meal

The main window

Students work

|  |  |
| --- | --- |
| Fardows Adam | Worked on method (fromFiletoHashMap) + breakfast file +group working in the main class and GUI interface +adding recipe class |
| karemah baker | Worked on method (suggestionRecepies) + Lunch file +group working in the main class and GUI interface |
| Dania Talal | Worked on method (getStringIntegerMap) + dinner file +group working in the main class and GUI interface |

concloution

Our Java project addresses the problem of unhealthy fast-food choices by offering a quick and easy way to select healthy homemade recipes. Using flexible data types like List<String> for ingredients and Map<String, Recipes> for storing recipes, the system allows users to input available ingredients and receive the top five recipe suggestions. The Meal class matches ingredients and generates suggestions, while the Recipes class stores recipe details. An interactive JavaFX interface makes it simple for users to access and prepare nutritious meals, promoting healthier eating habits efficiently, and reducing the confusion of multiple meal choices that sometimes lead us to unhealthy options.

references

* <https://openjfx.io/openjfx-docs/>
  + <https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/>
  + <https://www.geeksforgeeks.org/list-interface-java-examples/>
  + <https://sentry.io/answers/read-file-java/>
  + <https://www.bbcgoodfoodme.com/>