

Report



Group Members:

Brandon Schmidt
Christopher Stolo
Emily Williams
John Cole
Olajide Awoyera
Thomas Zack

Table of Contents

1	Overview – Foodies App	4
1.1	Team Member Contribution	4
2	Project Plan	6
2.1	Introduction	6
2.2	Roles/Responsibilities	6
2.3	WBS/Milestones.....	6
2.4	Software/System Requirement Specification (SRS).....	8
2.5	Project Schedule	9
3	User Guide	11
3.1	Project Overview.....	11
3.2	Purpose	11
3.3	Audience	11
3.4	Launch Instructions.....	11
3.5	Guide	12
4	Test Plan.....	30
4.1	Test Objectives.....	30
4.2	Test Assumptions.....	31
4.3	Approach.....	31
4.3.1	Roles and Expectations	31
4.3.2	Frequency.....	31
4.4	Component Functional Testing.....	31
4.4.1	GUI	31
4.4.2	Recipe.....	32
4.4.3	Pantry.....	33
4.5	Test Metrics	34
4.5.1	Evaluation Standardization	34
4.6	Defect Tracking and Reporting	34
5	Software Design.....	36
5.1	Statement of Goals	36
5.2	Functional Description	36

5.2.1	User Interface.....	37
5.2.2	Main Window:.....	37
5.2.3	Recipe Window:	38
6	Development History	40
6.1	Week 1	40
6.2	Week 2	41
6.3	Week 3	42
6.4	Week 4	43
6.5	Week 5	44
6.6	Week 6	45
6.7	Week 7	46
6.8	Week 8	47
7	Conclusions	49
7.1	Lessons Learned	49
7.2	Design Strengths	50
7.3	Limitations.....	50
7.3.1	Resource Management.....	50
7.3.2	Time Management.....	50
7.3.3	Personnel Management.....	50
7.4	Future Improvements	50
8	APPENDIX I – Test Log	52
9	APPENDIX II – Contribution Report.....	56
9.1	Week 1	56
9.2	Week 2	56
9.3	Week 3	56
9.4	Week 4	56
9.5	Week 5	57
9.6	Week 6	57
9.7	Week 7	57
9.8	Week 8	58

1 Overview – Foodies App

Welcome! The Foodies application is intended for those users who are looking to elevate how they currently manage their kitchen and food supply. For those who want to take control of their kitchen. For those who want to stop asking the questions, do we have that, can I make that, how do I make that, and where did I put that recipe book again.

Specifically, the Foodies App is the only pantry manager, ingredient comparison, recipe manager, and grocery list generator, all rolled into one cohesive application. The app leverages you're in stock ingredients and can provide suggestions from your recipes and help track what groceries are needed, just like a personal assistant.

Pantry Manager:

- Easily organize your kitchen into a pantry
- Set the names and quantities of your favorite ingredients, while the Foodies App checks and tracks your food inventory
- Easily add your favorite items from the pantry to your shopping list
- Easily search through your ingredients and search for recipes by ingredients

Recipe Manager:

- Easily search tasty recipes
- Links to the pantry manager and automatically suggests recipes which match your existing food inventory
- Add missing ingredients or your favorite recipes to your grocery list with a single click
- Upload your own recipes

Grocery List:

- Create your shopping list for groceries in seconds, include quantities and never forget an item

1.1 Team Member Contribution

The roles and responsibilities table in [section 2.2](#) outlines the assigned role and responsibilities for each team member, while the contribution report in [section 9](#) provides each week's contribution breakdown. Please reference those sections for more information.

Project Plan

Team Foodies

2 Project Plan

2.1 Introduction

This project's goal is to develop a single application that acts as a complete home kitchen aid. The application will provide the following key features:

- provides the user a recipe repository
- pantry inventory management system
- ability to compare current supply against any recipe in the repository
- generate a needed items list based on comparison
- a user-friendly graphical user interface (GUI)

2.2 Roles/Responsibilities

Member	Role	Responsibility
Brandon Schmidt	Developer/Tester	Responsible for the design and creation of the Recipe Repository component
Christopher Stolo	Developer/Tester	Responsible for the design and creation of the Pantry and comparator component
Emily Williams	Developer/Tester	Responsible for design and creation of the GUI component
John Cole	Documentation/Support/Tester	Responsible for the creation and review documentation alongside Olajide, as well as development support
Olajide Awoyera	Documentation/Support/Tester	Responsible for the creation and review documentation alongside John, as well as development support
Thomas Zack	Developer/Tester	Responsible for the design and creation of the Recipe Repository component alongside Brandon

2.3 WBS/Milestones

WBS	Deliverable	Due	Milestones	Status	Assigned	Start	End
1	Project Plan	03/29				03/22	03/29
1.1			Recipe Repository Decision	Done	Everyone		
1.2			Pantry Inventory Decision	Done	Everyone		
1.3			GUI Decision	Done	Everyone		
1.4			JDK Download	Done	Everyone		
1.5			Project Plan Inputs	Done	Everyone		
1.6			Document Prep	Done	Olajide, John		
2	User Guide & Test Plan	04/05				03/29	04/05
2.1			Repository Design	Done	Chris		

2.2			Pantry Design	Done	Brandon, Thomas		
2.3			GUI Design	Done	Emily		
2.4			User Guide & Test Plan inputs	Done	Everyone		
2.5			Document Prep	Done	Olajide, John		
3	Design	04/12				04/05	04/12
3.1			Design Documentation Inputs	Done	Everyone		
3.2			Document Prep	Done	Olajide, John		
4	Phase 1 Source	04/19				04/12	04/19
4.1			Initial Recipe Repository Functionality	Done	Brandon, Thomas		
			Initial Pantry Functionality	Done	Chris, John		
4.3			Initial GUI Functionality	Done	Emily, John		
5	Phase 2 Source	04/26				04/19	04/26
5.1			Comparator Functionality	Done	Chris, Brandon, Emily		
5.2			Update Documentation	Done	Olajide, John		
5.3			Recipe Repository Functionality	Done	Brandon, Thomas		
5.4			Pantry Functionality	Done	Chris, John		
5.5			GUI Functionality	Done	Emily, John		
6	Phase 3 Source	05/03				04/26	05/03
6.1			Bug Fixes and Finalization	Done	Everyone		
6.2			Program Demo	Done	Everyone		
5.3			Update Documentation	Done	Olajide, John		
7	Final	05/10			Everyone	05/03	05/10
7.1			Submit Final Product/Demo	Done	Everyone		
7.2			Update Documentation	Done	Olajide, John		

2.4 Software/System Requirement Specification (SRS)

Software Requirement Specification (SRS)
Functional Requirements (System):
Distributed Database: N/A Client/Server System: N/A Operating System: Windows, Mac OS, Linux Platform: Java 15 or newer Technologies: IDE that can run Java JDK/JRE; Text Files
Interface Requirements:
User Interfaces: Front-end Software - Java GUI Back-end Software - N/A Hardware Interfaces: No specific hardware requirements known of at this time. Software Interfaces: Operating Systems - We will recommend Windows, however, Mac Os and linux are supported. Our recommendation is for its user support and user friendliness. Java JRE - To implement the project we have chosen Java 16 language for its GUI support.
Design Constraints:
Based off our initial design concept, we have a couple design constraints. Users will need to use a .txt file when interfacing with this application. Users will need interface with a GUI; however, the application will need to be run through an IDE at first launch. Users IDE of choice will need to support Java JDK/JRE to launch the application.
Performance Requirements:
Speed of Response: N/A Throughput: N/A Execution Time: within 15 seconds of action request (this is dependent upon feature selected as some file sizes can be extremely large) Storage Capacity: As there is no database the storage capacity dependent on the user's computer drives space. We recommend that customers have at least 5 GB space for the application and associated documents.
Safety/Security Requirements:
Based on its current state this is not applicable. However, future consideration to security features would include authentication, authorization, encryption, and logging.

[illegible]

Users Guide and Test Plan

Team Foodies

3 User Guide

3.1 Project Overview

Specifically, the Foodies App is the only pantry manager, ingredient comparison, recipe manager, and grocery list generator, all rolled into one cohesive application. The app leverages you're in stock ingredients and can provide suggestions from your recipes and help track what groceries are needed, just like a personal assistant.

3.2 Purpose

The purpose of this document is to define the core elements of how a user will interact with the Foodies App, in addition to, the plan for testing the application and tracking defects.

3.3 Audience

The intended audience of this document is for project team members, development team, testers, and future users.

3.4 Launch Instructions

The following steps are to instruct users on how to launch the application:

STATUS: Pre-Production

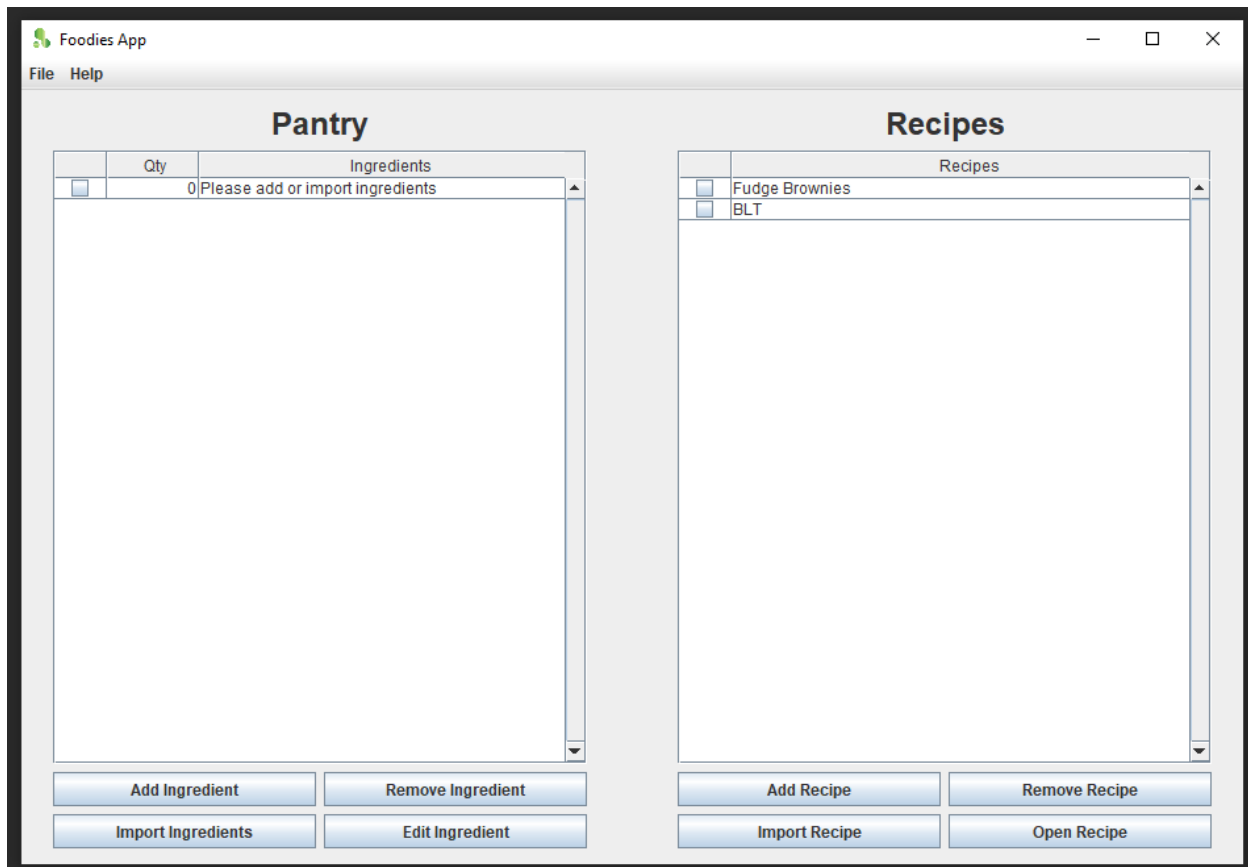
This status denotes that the application is not ready for production use and may have limited functionality. Below are the steps to instruct the user on how to start the application.

1. Download the zip files labeled Final and App Imports.
2. Extract all to a desired directory.
3. Through your preferred Integrated Development Environment (IDE) and create a new project from existing file(s). Select the source code folder by navigating through the file explorer. We recommend either IntelliJ Idea or Eclipse as IDE's.
4. Verify that the IDE has a set Java JDK installed. If not, please visit Oracles site to download a Java Development Kit (JDK) version 15 or higher. We recommend the Java 16.0.2 JDK files.
5. Once the JDK is downloaded follow the websites JDK setup instructions for proper installation.
6. Once the verified, within the IDE locate the run/launch function built into the IDE. Typically, this is indicated by a green play button or option that states "Run as".
7. If asked to select which file to compile, please compile/run this file: 'FoodAppGUI.java'.
8. This will launch the application on your desktop.

3.5 Guide

Initial Landing of the application.

User launches program from their desired IDE and will land on the GUI home. Upon first loading the application, the user will notice that the Pantry side will enable to the user to create a pantry from scratch or by importing the previously created pantry. On the Recipe side the application will automatically upload the recipe repository and display it on the right.



Pantry Management:

Pantry

	Qty	Ingredients
<input type="checkbox"/>	1	Ham
<input type="checkbox"/>	1	Lettuce
<input type="checkbox"/>	1	Bread
<input type="checkbox"/>	1	Tomato
<input type="checkbox"/>	3	Salt
<input type="checkbox"/>	1	Salted Butter
<input type="checkbox"/>	1	Sugar
<input type="checkbox"/>	1	Eggs
<input type="checkbox"/>	1	Vanilla Extract
<input type="checkbox"/>	1	Chocolate Chips
<input type="checkbox"/>	1	Coca Powder

Add Ingredient

Remove Ingredient

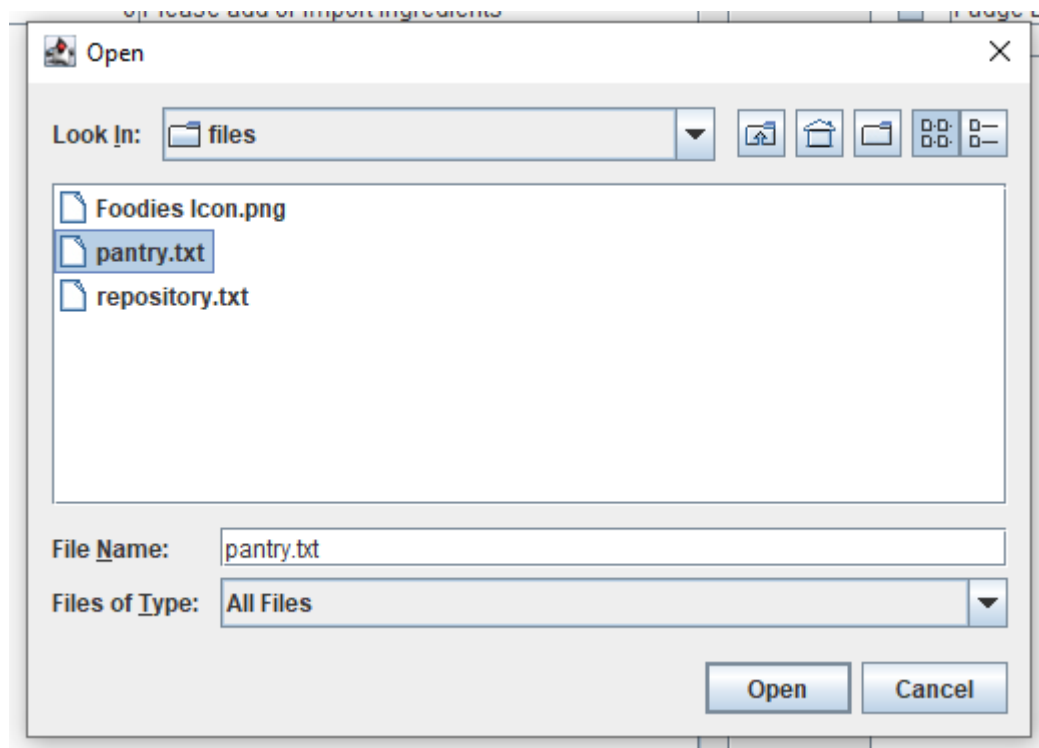
Import Ingredients

Edit Ingredient

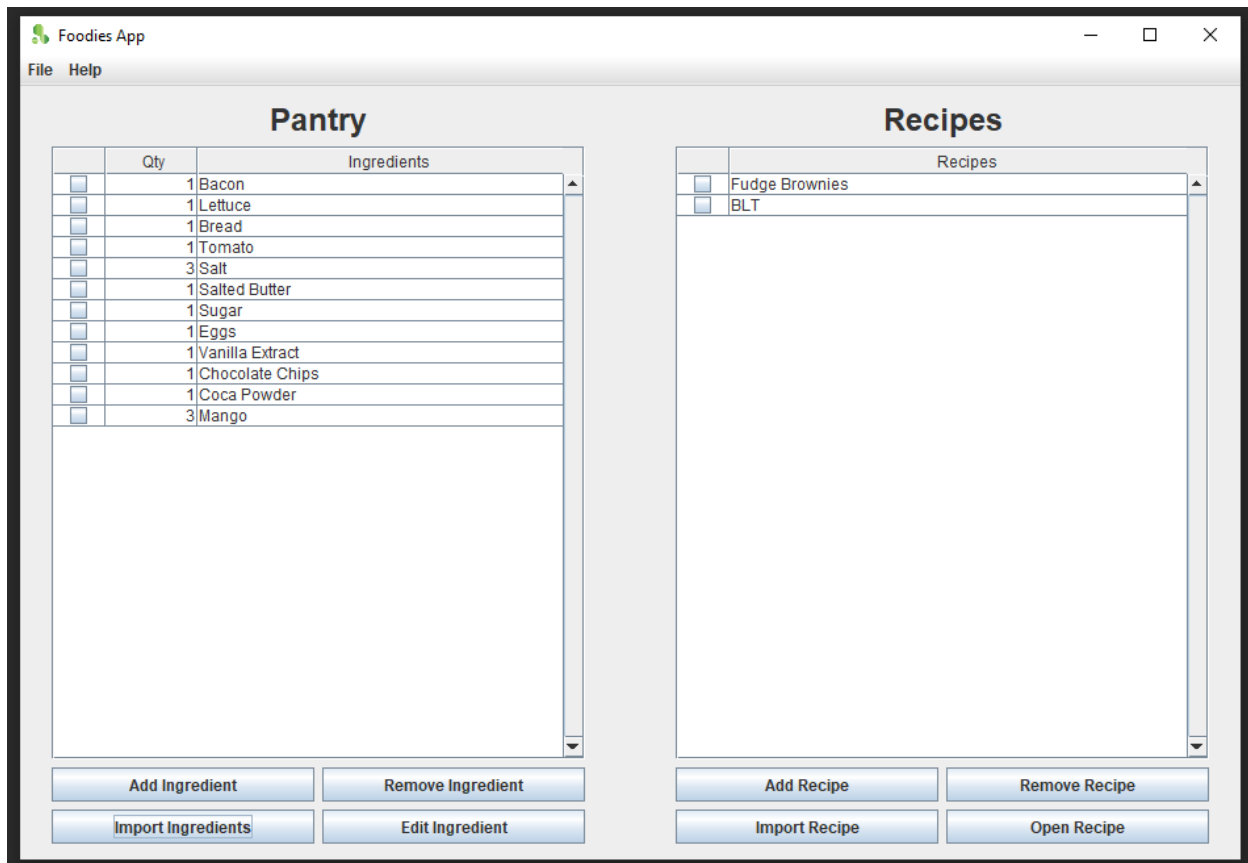
User selects 'Import Ingredients' button.

Import Ingredients

User navigates the files folder within the directory housing the application files and selects 'pantry.txt' file, in the files folder, then select open.



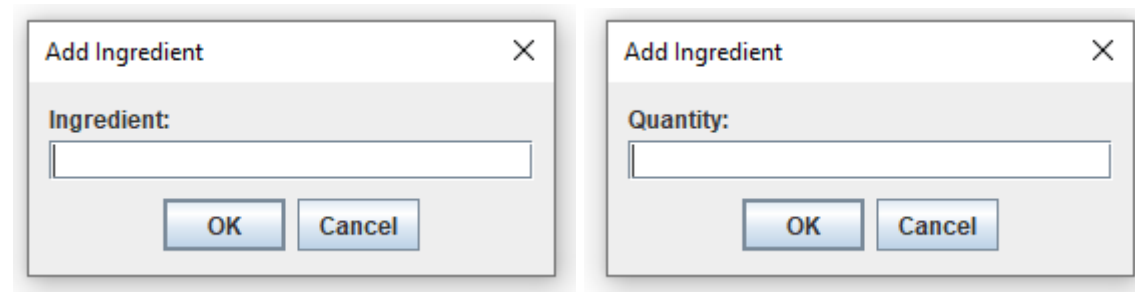
The table will be updated with the ingredients within the 'pantry.txt' file.



User can add ingredient to their pantry by selecting 'Add Ingredient' button.



User will be faced with two pop-up windows, in which, the user will be asked to input the ingredient name and quantity. After each, select okay.



The Pantry table will be automatically updated to reflect the new ingredients and the changes will be automatically saved to the 'pantry.txt' file. As an example, we will add Mint at the quantity of 4 to the pantry.

✕

Add Ingredient

Ingredient:

✕

Add Ingredient

Quantity:

<input type="checkbox"/>	1 Eggs
<input type="checkbox"/>	1 Vanilla Extract
<input type="checkbox"/>	1 Chocolate Chips
<input type="checkbox"/>	1 Coca Powder
<input type="checkbox"/>	3 Mango
<input type="checkbox"/>	4 Mint

pantry - Notepad

File Edit Format View Help

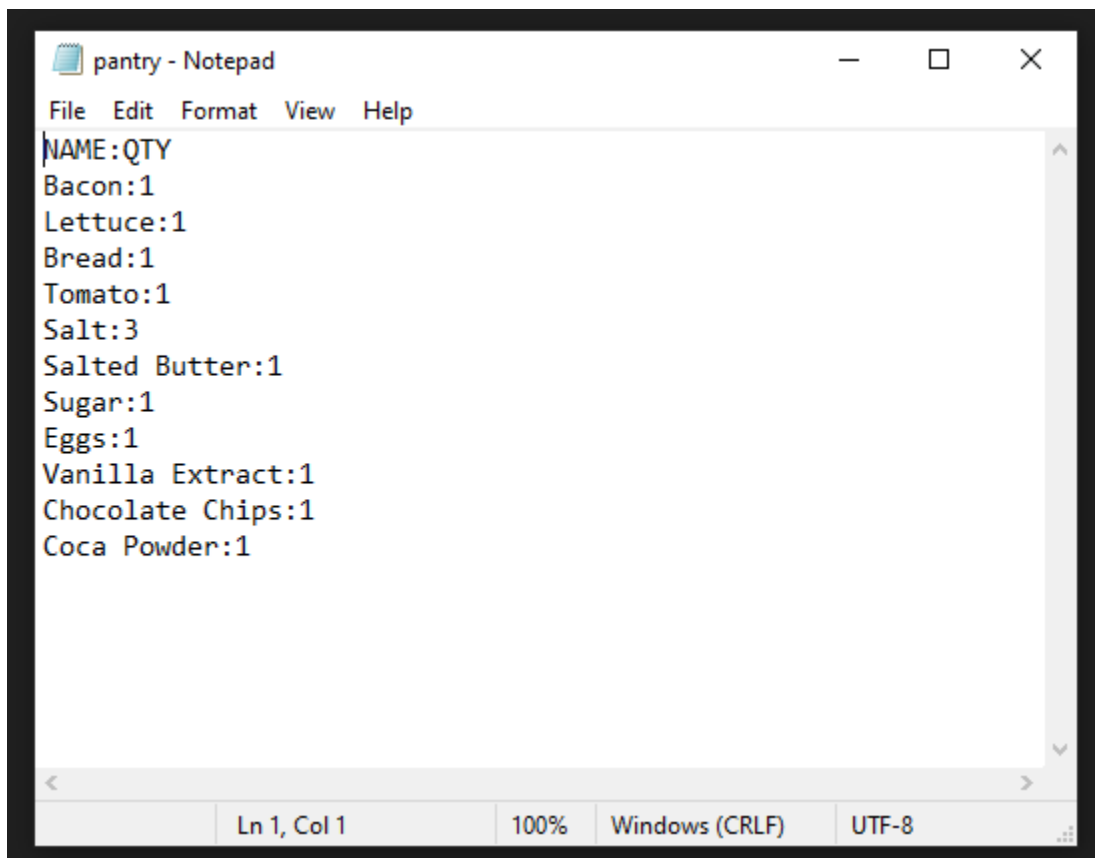
```
NAME:QTY
Bacon:1
Lettuce:1
Bread:1
Tomato:1
Salt:3
Salted Butter:1
Sugar:1
Eggs:1
Vanilla Extract:1
Chocolate Chips:1
Coca Powder:1
Mango:3
Mint:4
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

User can remove ingredient from their pantry by selecting the check box in the row of the desired ingredient(s) and then selecting the 'Remove Ingredient' button. Users can delete multiple ingredients at once. The table will be automatically updated to reflect the removal and the changes will be automatically saved to the 'pantry.txt' file.

<input type="checkbox"/>	1	Coca Powder
<input checked="" type="checkbox"/>	3	Mango
<input checked="" type="checkbox"/>	4	Mint

Remove Ingredient



User can edit the ingredient(s) in the table by selecting the 'Edit Ingredient' button. When this button is selected the Pantry becomes editable and the user can make the necessary adjustments followed by pressing enter. As an example, we will change 'Bacon' to 'Ham'. The user will notice that the table will remain editable until the changes are completed pressing enter and selecting the now 'Save Change' button. As a result, the changes will be automatically saved to the 'pantry.txt' file.

Edit Ingredient

Foodies App

File Help

Pantry

Qty	Ingredients
<input type="checkbox"/>	1 Bacon
<input type="checkbox"/>	1 Lettuce
<input type="checkbox"/>	1 Bread
<input type="checkbox"/>	1 Tomato
<input type="checkbox"/>	3 Salt
<input type="checkbox"/>	1 Salted Butter
<input type="checkbox"/>	1 Sugar
<input type="checkbox"/>	1 Eggs
<input type="checkbox"/>	1 Vanilla Extract
<input type="checkbox"/>	1 Chocolate Chips
<input type="checkbox"/>	1 Coca Powder

Add Ingredient

Remove Ingredient

Import Ingredients

Save Change

Recipes

Recipes	
<input type="checkbox"/>	Fudge Brownies
<input type="checkbox"/>	BLT

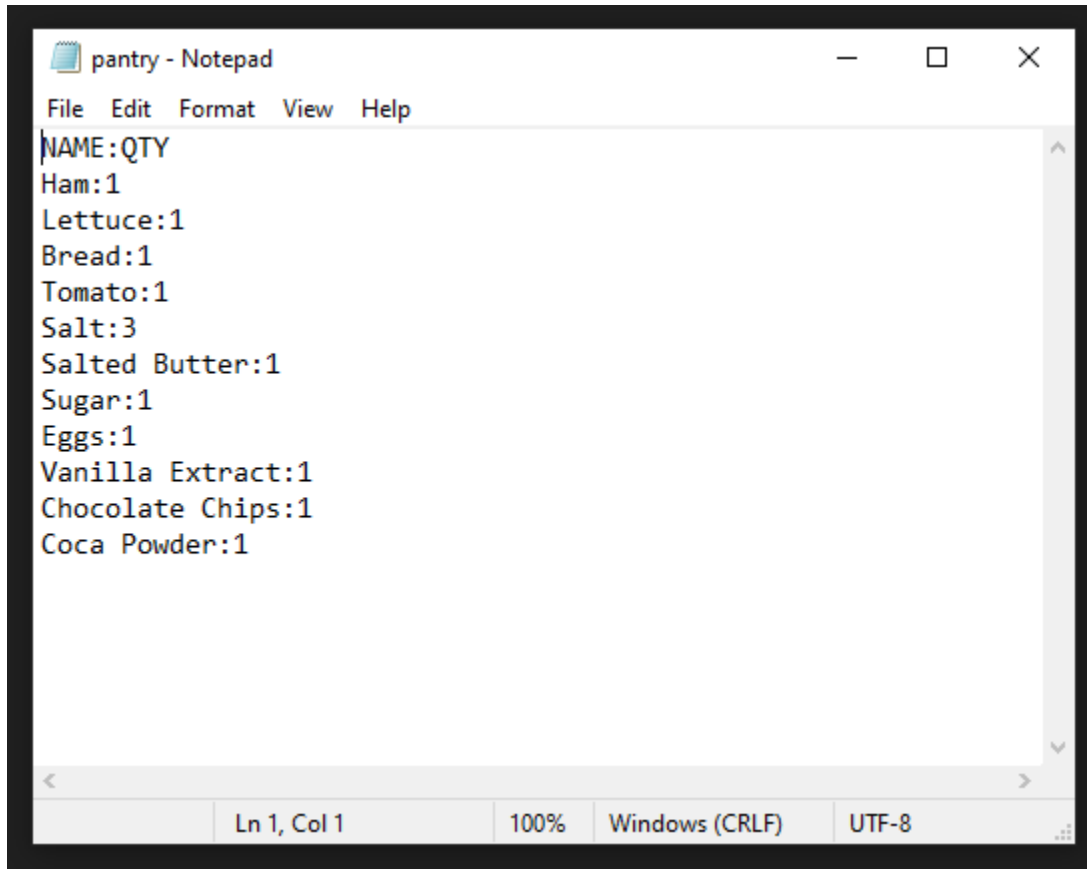
Add Recipe

Remove Recipe

Import Recipe

Open Recipe

Save Change



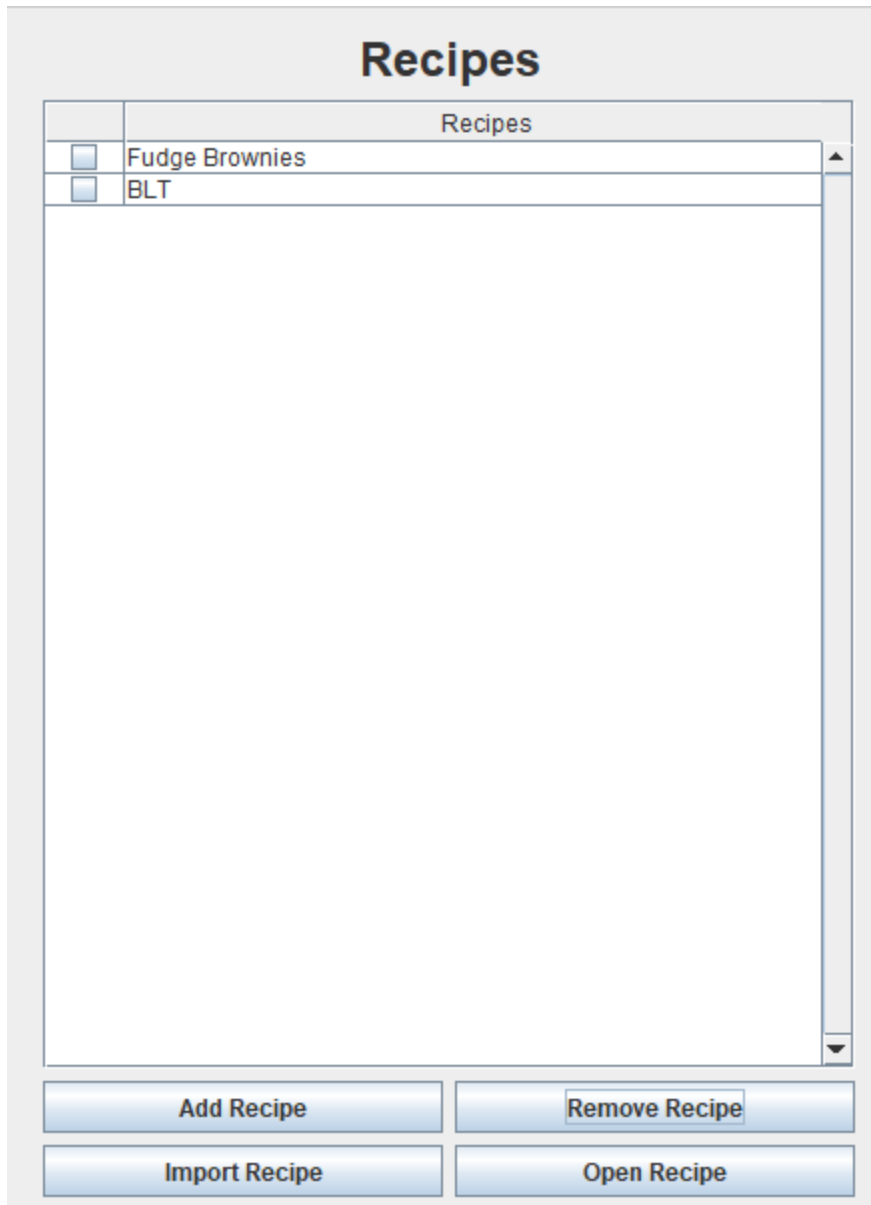
A screenshot of a Notepad window titled "pantry - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text inside the window is as follows:

```
NAME:QTY  
Ham:1  
Lettuce:1  
Bread:1  
Tomato:1  
Salt:3  
Salted Butter:1  
Sugar:1  
Eggs:1  
Vanilla Extract:1  
Chocolate Chips:1  
Coca Powder:1
```

The status bar at the bottom shows "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

Recipe Repository:

As mentioned previously, the user's recipe repository will be uploaded automatically from the 'repository.txt' file, in the files folder. However, the users can also import or create new recipes.




The screenshot shows a web application window titled "Recipes". Inside the window, there is a table with a header "Recipes" and two rows of data. The first row is "Fudge Brownies" and the second row is "BLT". Each row has a small square checkbox to its left. Below the table, there are four buttons arranged in a 2x2 grid: "Add Recipe", "Remove Recipe", "Import Recipe", and "Open Recipe".

	Recipes
<input type="checkbox"/>	Fudge Brownies
<input type="checkbox"/>	BLT

Add Recipe Remove Recipe

Import Recipe Open Recipe

User can import a recipe by following the instructions of the 'Recipe Form.xlsx' excel form located in the files folder.



Recipe Instructions for the Foodies App

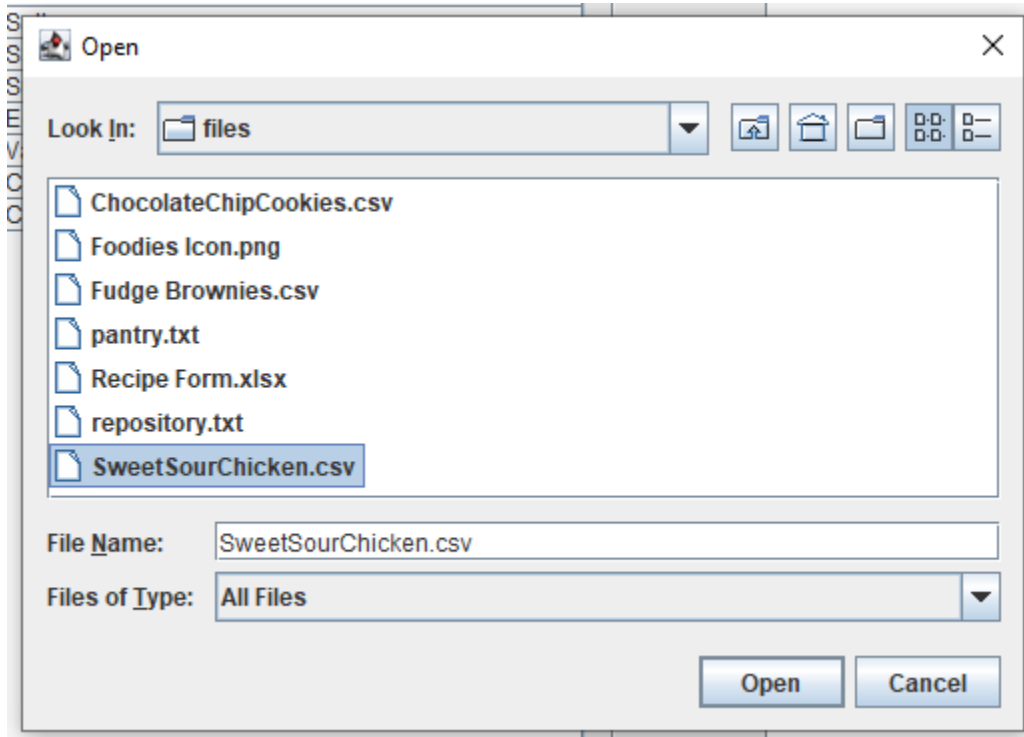
1. Add Recipe name to Line 2 of Recipe Sheet
2. Add ingredients and quantity needed to "Ingredients" section ensure Quantity is also filled out. It create more lines select last line and Right click Select "insert"
3. If Required Remove any Extra spaces between Ingredients and Directions.
4. Add directions as needed to properly complete recipe.
5. If Required Remove any Extra spaces following Directions.
6. Select file "Save As" Select "CSV(comma delimited)", enter name
7. CSV does not support multiple sheets so select CSV to save only the recipe as the active sheet.
8. As an example, we have included a Brownies recipe in the next tab. Please remember to save as a .csv file for that sheet.

Recipe	
Fudge Brownies	
Ingredients	
Salted Butter	1
Sugar	1
Eggs	1
Vanilla Extract	1
Choclate Chips	1
All-purpose Flour	1
Coca Powder	1
Salt	1
Directions	
Preheat oven to 350 degrees F. Line a metal 9x9 pan with parchment paper.	
Pour 10 tablespoons melted butter into a large mixing bowl.	
Whisk in 1 cup sugar by hand until smooth, 30 seconds.	
Add in 2 eggs and 2 tsp vanilla extract. Whisk 1 minute	
Whisk in 1/2 cup melted chocolate until combined and smooth.	
Use a rubber spatula to stir in 3/4 cup flour, 1/4 cup unsweetened cocoa powder, and 1/2 tsp. salt until just	
Pour into prepared pan and smooth out.	
Bake in the preheated oven for 30 minutes. Let cool in pan 30 minutes before slicing.	

Once the user has completed all the instructions, paying special attention to save the sheet as a '.csv' file type. The user can then import the file into the application to store in the repository by selecting the 'Import Recipe' button.

Import Recipe

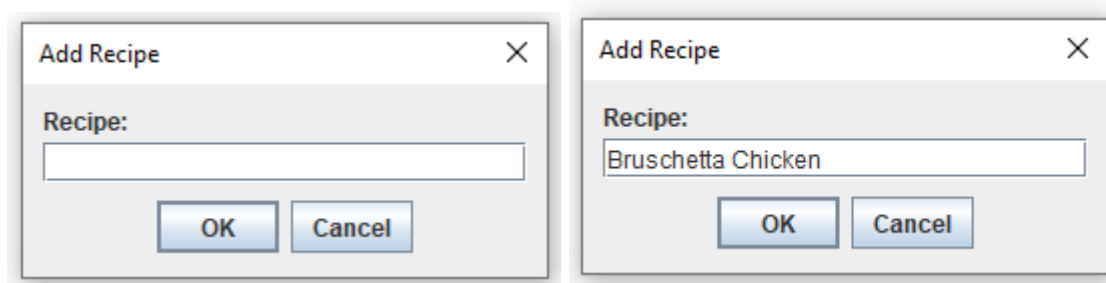
User navigates the files folder within the directory housing the application files and selects the recipe.csv file they would like to import, then selects open. As an example we have imported the 'SweetSourChicken.csv' as our recipe.



The table will be updated to reflect the addition of the desired recipe.

Recipes	
<input type="checkbox"/>	Fudge Brownies
<input type="checkbox"/>	BLT
<input type="checkbox"/>	Sweet & Sour Chicken

User can create recipe by selecting 'Add Recipe' button. This will prompt the user with a series of windows to complete each aspect of the recipe. As an example, we will add a recipe for Bruschetta Chicken to the recipe repository. On the left will be the prompt and, on the right, will be the example.



Add Ingredients ✕

Number of Ingredients:

OK Cancel

Add Ingredients ✕

Number of Ingredients:

OK Cancel

Add Ingredients ✕

Ingredient Name:

OK Cancel

Add Ingredients ✕

Ingredient Name:

OK Cancel

Add Ingredients ✕

Quantity:

OK Cancel

Add Ingredients ✕

Quantity:

OK Cancel

.... Add all the necessary ingredients with their quantity....

Add Steps ✕

Number of Steps:

OK Cancel

Add Steps ✕

Number of Steps:

OK Cancel

Add Step ✕

Step 1

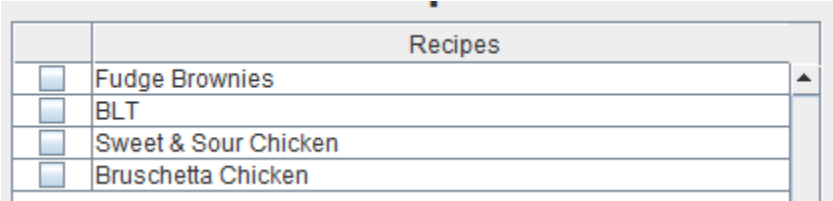
OK Cancel

Add Step ✕

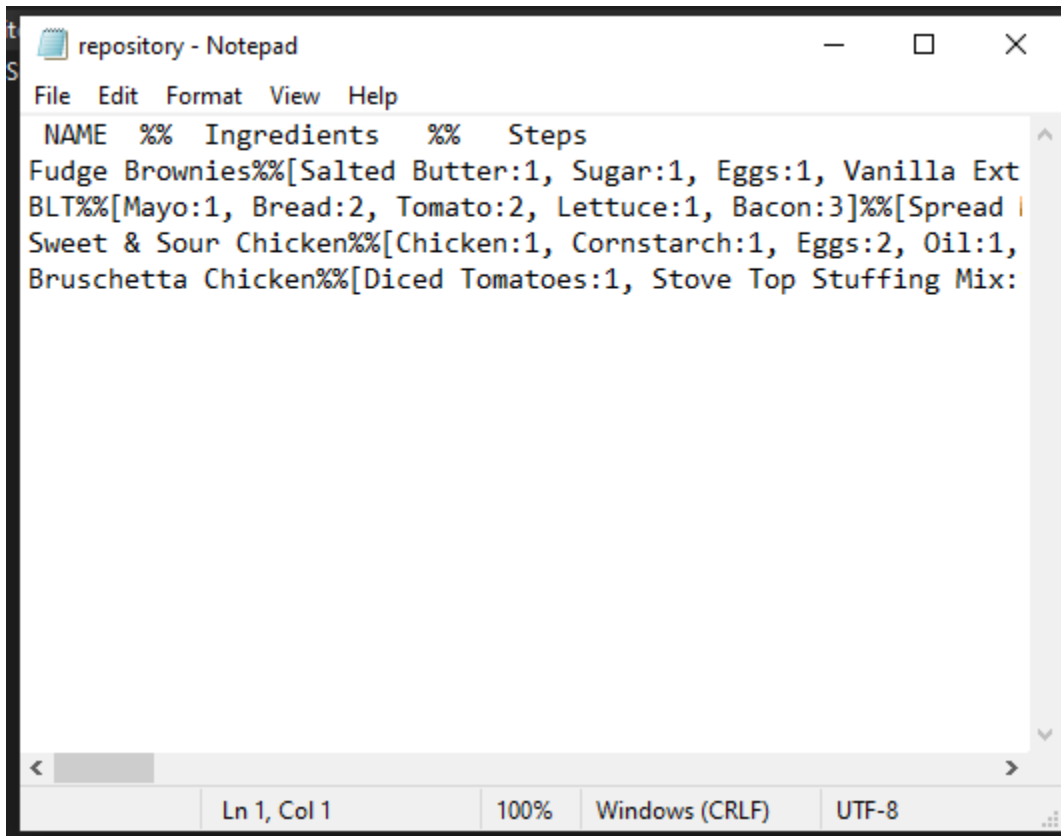
Step 1

OK Cancel

.... Add all the necessary steps to complete the instructions for making the recipe. Once completed, the recipe will automatically be updated to reflect in the recipes table and the change will be automatically saved to the 'repository.txt' file.

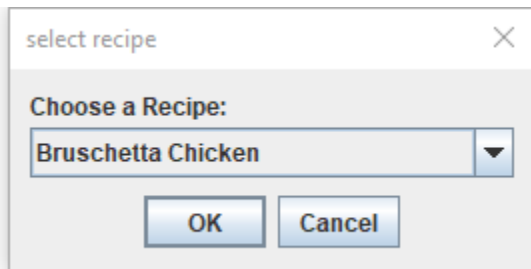
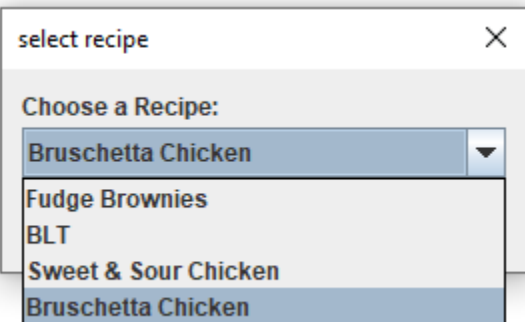
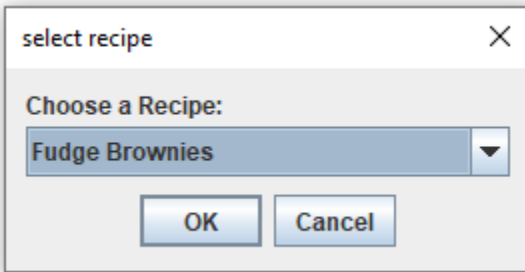


	Recipes
<input type="checkbox"/>	Fudge Brownies
<input type="checkbox"/>	BLT
<input type="checkbox"/>	Sweet & Sour Chicken
<input type="checkbox"/>	Bruschetta Chicken

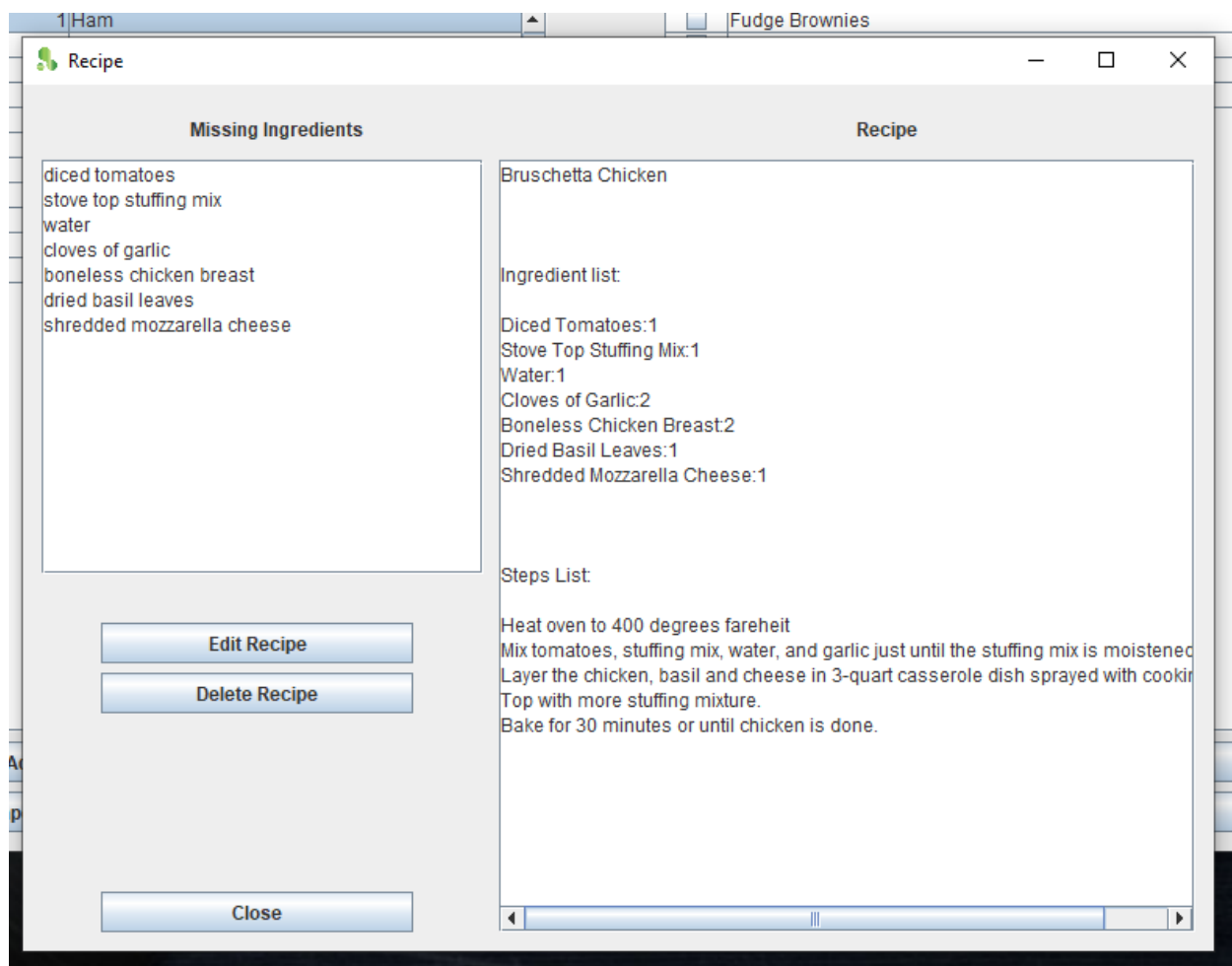


```
repository - Notepad
File Edit Format View Help
NAME Ingredients Steps
Fudge Brownies[Salted Butter:1, Sugar:1, Eggs:1, Vanilla Ext
BLT[Mayo:1, Bread:2, Tomato:2, Lettuce:1, Bacon:3][Spread
Sweet & Sour Chicken[Chicken:1, Cornstarch:1, Eggs:2, Oil:1,
Bruschetta Chicken[Diced Tomatoes:1, Stove Top Stuffing Mix:
```

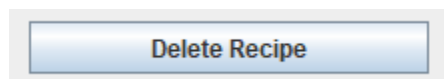
User can select the 'Open Recipe' button to view a recipe that is within their repository. The choose recipe window will appear with a drop-down list of the repositories recipes for the user to choose from. After desired recipe is chosen the user will then select ok. As an example, we will choose the Bruschetta Chicken recipe we just created to view.

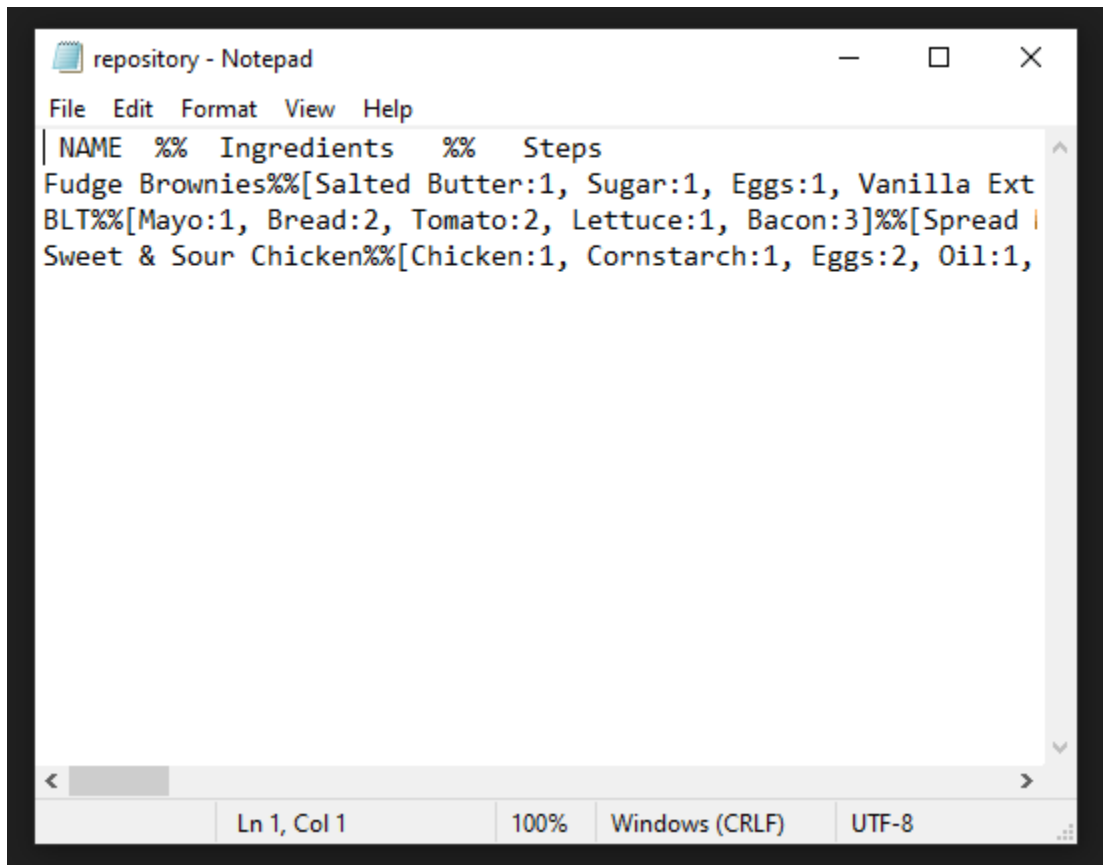


After the 'Ok' button is selected, the recipe window will appear. This window will contain three sections, the missing ingredients, recipe, and functionality buttons (Edit and Delete). The Missing Ingredients section compares the current pantry stock against the required ingredients needed to make the recipe and lists all the missing ingredients. To its right, the user will see the complete recipe as they filled out while adding the recipe.

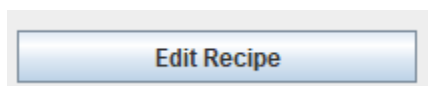


User can delete the recipe as well inside the Recipe window by selecting the 'Delete Recipe' button. This will automatically delete the recipe from the recipes table and from the 'repository.txt' file.

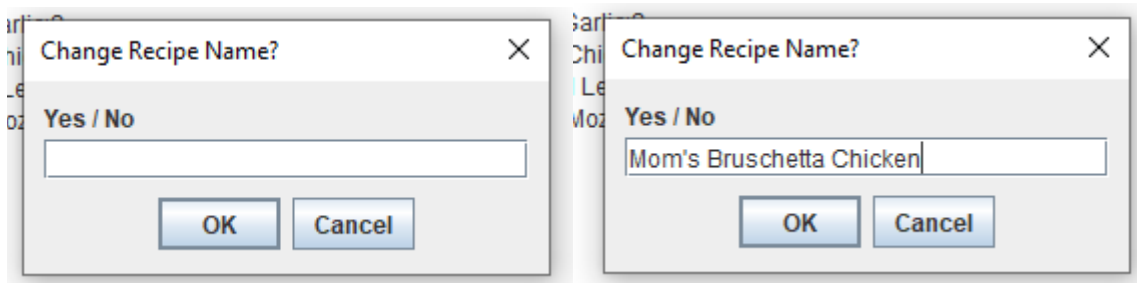


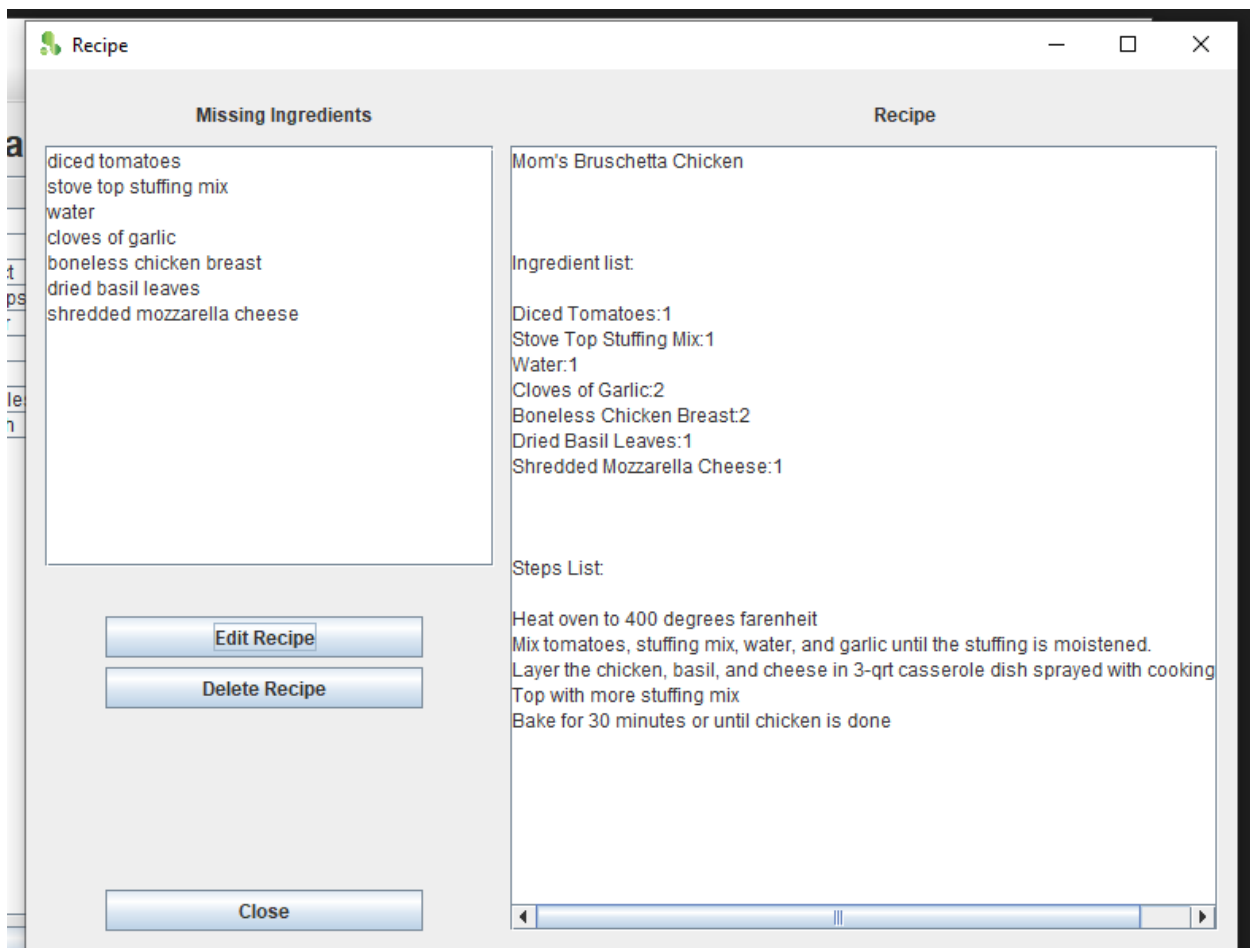
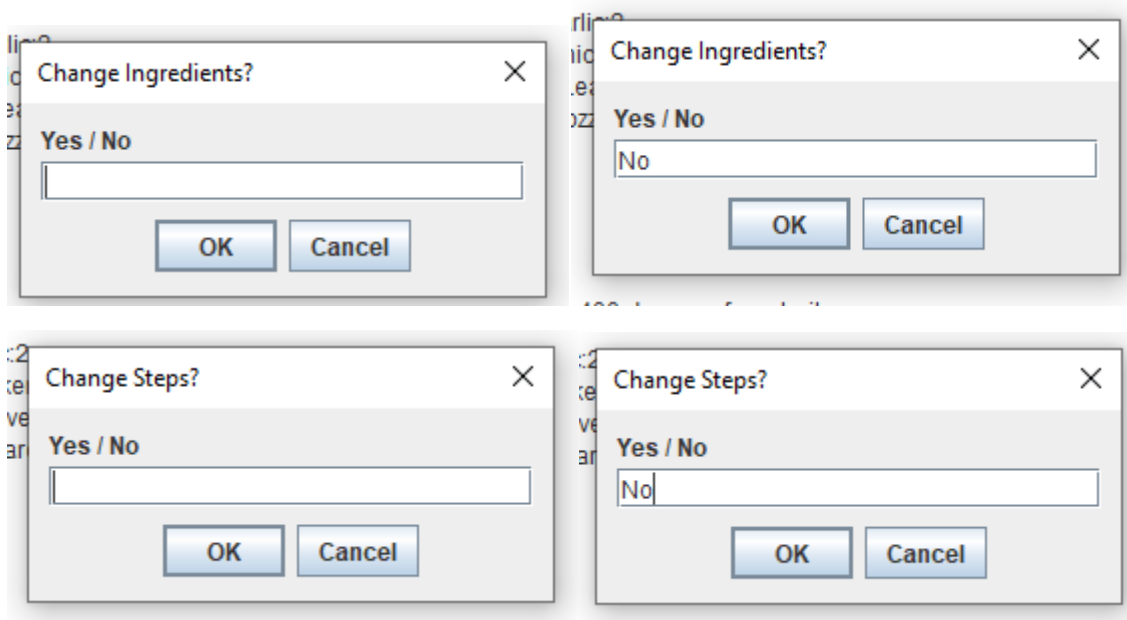


User can update/edit the recipe by selecting the 'Edit Recipe' button.



As before when adding a recipe, complete the steps to adjust/add the necessary changes via the prompts. Specifically there are 3 different prompts, to change the recipes name, to change the ingredients, and/or to change the steps. Should you wish to skip one option, simply respond 'No' when prompted. The recipe window, recipe table, and 'repository.txt' file will automatically be updated. As an example, I will change the recipe name for Bruschetta Chicken to Mom's Bruschetta Chicken.





Recipes

	Recipes
<input type="checkbox"/>	Fudge Brownies
<input type="checkbox"/>	BLT
<input type="checkbox"/>	Sweet & Sour Chicken
<input type="checkbox"/>	Mom's Bruschetta Chicken

```

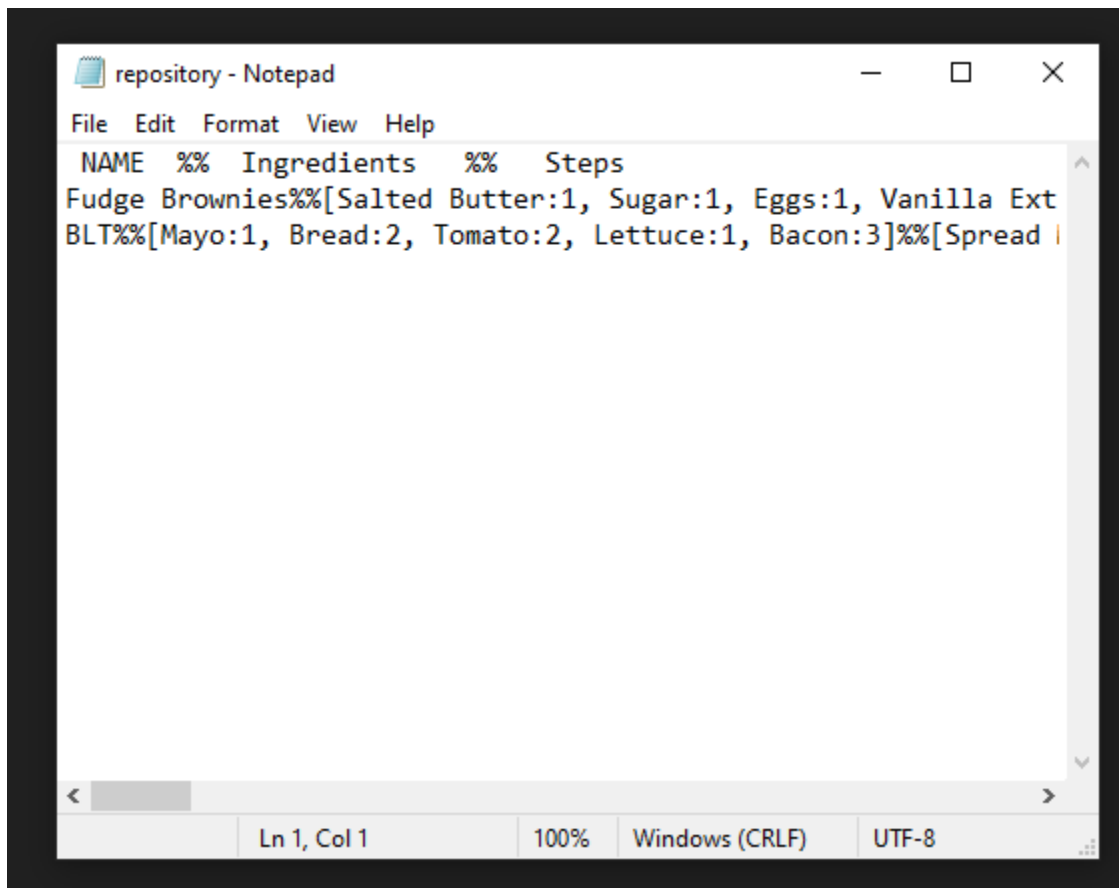
repository - Notepad
File Edit Format View Help
| NAME %% Ingredients %% Steps
Fudge Brownies%%[Salted Butter:1, Sugar:1, Eggs:1, Vanilla Ext
BLT%%[Mayo:1, Bread:2, Tomato:2, Lettuce:1, Bacon:3]%%[Spread
Sweet & Sour Chicken%%[Chicken:1, Cornstarch:1, Eggs:2, Oil:1,
Mom's Bruschetta Chicken%%[Diced Tomatoes:1, Stove Top Stuffin

```

User can delete recipe(s) by checking the boxes in the far-left column of the table and selecting the 'Remove Recipe' button. The recipe table and the 'repository.txt' file will automatically be updated. As an example, we will remove Mom's Bruschetta Chicken and Sweet & Sour Chicken.

	Recipes
<input type="checkbox"/>	Fudge Brownies
<input type="checkbox"/>	BLT
<input checked="" type="checkbox"/>	Mom's Bruschetta Chicken
<input checked="" type="checkbox"/>	Sweet & Sour Chicken

Recipes	
<input type="checkbox"/>	Fudge Brownies
<input type="checkbox"/>	BLT



User can minimize Foodies App by selecting the '-' in the top right-hand corner of the window.

User can exit the Foodies App by selecting the 'x' in the top right-hand corner of the window.

4 Test Plan

4.1 Test Objectives

The objectives of this test plan are as follows:

- Define the proper assumptions before testing
- Identify the approach to performing testing
- Define the functional testing for each component
- Standardize testing metrics

- Define the process for reporting defects

4.2 Test Assumptions

Production like data is required and must be available in the system/files prior to start of Functional Testing.

4.3 Approach

Our approach for testing the Foodies App is to focus on the component level of the application and derive the state of functionality within those components. The method to conduct testing will be cross-functional testing, due to the team size. Each member of the team will play a role in testing the Foodies application. Listed below are the development team members who worked to create each component and testers assigned to review each component. In addition, the frequency of reviews in accordance with the Foodies App Project Plan has been included.

4.3.1 Roles and Expectations

4.3.1.1 Development Team and Design Component

Component	Developers
GUI	Emily Williams, John Cole
Recipe	Brandon Schmidt, Thomas Zack
Pantry	Christopher Stolo, John Cole
Documentation	John Cole

4.3.1.2 Test Team and Assignments

Component	Testers
GUI	Brandon Schmidt, Thomas Zack
Recipe	Christopher Stolo, John Cole
Pantry	Emily Williams, Olijade Awoyera
Documentation	Olijade Awoyera

4.3.2 Frequency

Testing of the Foodie Apps components will occur at each milestone. The following testing period dates have been identified:

- Test/Review 1: April 19th, 2022 – April 26th, 2022
- Test/Review 2: April 27th, 2022 – May 3rd, 2022
- Test/Review 3: April 4th, 2022 – May 10th, 2022

4.4 Component Functional Testing

4.4.1 GUI

The test team will test the following functions of the GUI component contained within the following files:

- FoodAppGUI.java

- PantryBooleanTableModel.java
- RecipeBooleanTableModel.java
- RecipeWindow.java

Function	Test
Add Ingredient to Pantry	<ul style="list-style-type: none"> ▪ Tester will type ingredients into a text field and click the 'Add Ingredient' button to save the ingredients to a list.
Import Pantry from file	<ul style="list-style-type: none"> ▪ Tester will click the 'Import Ingredients' button to display a list of items from the Pantry file.
Removal Ingredient from Pantry	<ul style="list-style-type: none"> ▪ Tester will click the 'Remove Ingredient' button to display a list of items from the current pantry and select the ingredient they would like to delete.
Save Pantry to file	<ul style="list-style-type: none"> ▪ Tester will attempt to save pantry from just ingredients they've added without importing a pantry file. ▪ Tester will attempt to save to the pantry file they initial imported the pantry from.
Create a Recipe	<ul style="list-style-type: none"> ▪ Tester will click the 'Create Recipe' button to open a popup window ▪ Tester will input the recipe name, ingredients, and instructions into the correct fields in the popup window ▪ Tester will click the 'Save Recipe' button to add the recipe to the database
Search for a Recipe	<ul style="list-style-type: none"> ▪ Tester will input a word or phrase into the search bar and click the 'Search' button to find a recipe in the database ▪ Tester will select the desired recipe to display
Update a Recipe	<ul style="list-style-type: none"> ▪ Tester will select a recipe and click the 'Update Recipe' button ▪ Tester will make desired changes and click the 'Save Recipe' button to update the recipe in the database
Delete a Recipe	<ul style="list-style-type: none"> ▪ Tester will select a recipe and click the 'Delete Recipe' button ▪ Tester will search for the recipe to confirm that it was deleted
Minimize Application	<ul style="list-style-type: none"> ▪ Tester will click the '-' button to minimize the window
Close Application	<ul style="list-style-type: none"> ▪ Tester will click the 'x' button to exit the application ▪ Tester will choose the 'Exit' option in the menu to exit the application

4.4.2 Recipe

The test team will test the following functions of the recipe component contained within the following files:

- Recipe.java
- Repository.java

Function	Test
Create Recipes	<ul style="list-style-type: none"> ▪ Tester will manually enter recipe from developed test data into the command line. ▪ Tester will verify that the recipe was created and logged into the .txt file.

	<ul style="list-style-type: none"> ▪ Tester will repeat process with a new test data set while leaving the name of the recipe input blank. ▪ Tester will verify that the recipe was not created and not logged into the .txt file. ▪ Tester will repeat process with the previous data set and enter a name into the recipe input on this step. ▪ Tester will verify that the recipe was created and logged into the .txt file. ▪ Tester will move database file to a different folder and enter the recipe data test in. ▪ Tester will verify that the recipeDB.txt was created and that the recipe was entered into the file.
Delete Recipes	<ul style="list-style-type: none"> ▪ Tester will delete the first recipe created in the previous test. ▪ Tester will verify that the correct recipe was delete from database. ▪ Tester will attempt to delete a recipe not found in the recipe database. ▪ Tester will verify that an error message was received.
Search Recipes	<ul style="list-style-type: none"> ▪ Tester will search for the last created recipe in the Create Recipe test using the recipe name. ▪ Tester will verify that the search found the correct recipe, and all information given is correct from entry. ▪ Tester will search for a recipe that is not in the database. ▪ Tester will verify that the recipe was not found, and no erroneous recipe is displayed.
Edit Recipes	<ul style="list-style-type: none"> ▪ Tester will edit the last recipe's name in the Create Recipes Stage according to the test data. ▪ Tester will verify that the correct data was changed in the last recipe. ▪ Tester will edit an ingredient in the recipe from the previous step according to the test data. ▪ Tester will verify that the correct data was changed in the previous recipe.

4.4.3 Pantry

The test team will test the following functions of the pantry component contained within the following files:

- Ingredient.java
- Pantry.java

Function	Test
Add to Pantry	<ul style="list-style-type: none"> ▪ Tester will add the ingredient soup, beans, chicken, and salt to the pantry. ▪ Tester will verify that the data is accurate through the showPantry function.
Save to Pantry	<ul style="list-style-type: none"> ▪ Tester will save current ingredients. ▪ Tester will verify that the correct data is contained within each test file.

	<ul style="list-style-type: none"> Tester will verify that the data is accurate through the showPantry function.
Remove from Pantry	<ul style="list-style-type: none"> Tester will remove the ingredients soup and beans from the pantry. Tester will verify that the data is accurate through the showPantry function.
Import Food Files	<ul style="list-style-type: none"> Tester will save the pantry test file(s) to the desired directory. Tester will verify that the correct data is contained within each test file. Tester will import each file independently. Tester will verify that the data is accurate through the showPantry function.

4.5 Test Metrics

4.5.1 Evaluation Standardization

Severity	Impact
Critical	<ul style="list-style-type: none"> Event or bug is critical enough to crash the system, cause file corruption, or cause potential data loss. Event or bug causes an abnormal return to the operating system (crash or a system failure message appears). Event or bug causes the application to hang and requires re-booting the system.
High	<ul style="list-style-type: none"> Event or bug causes a lack of vital program functionality with workaround.
Medium	<ul style="list-style-type: none"> Event or Bug will degrade the quality of the System. However, there is a workaround for achieving the desired functionality. Event or bug prevents other areas of the product from being tested. However other areas can be independently tested.
Low	<ul style="list-style-type: none"> Event or bug is an insufficient or unclear error message, which has minimum to no impact on product use.

4.6 Defect Tracking and Reporting

Each function test provided above is designed to test a core aspect of each component. When tested, all outcomes will need to be logged and reported to the development team for mitigation. If an event or bug is identified, the tester will need to document the event, assign a severity, and take a screenshot of the error message received. This information must be reported in the test log found in Appendix I of this document. All events must be addressed priority, set by its severity, prior to the next Test/Review period.

Software Design

Team Foodies

5 Software Design

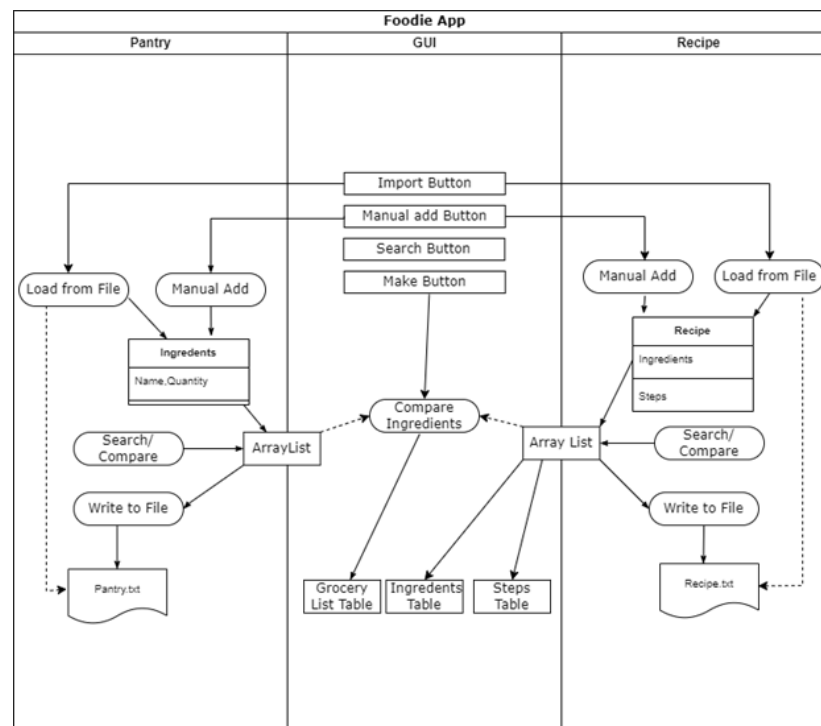
5.1 Statement of Goals

The Foodies App is designed to serve as a recipe repository and pantry management system. Specifically, the Foodies App is a pantry manager, recipe comparison, recipe manager, and grocery list, all-in-one application. The goal of this Software Design document is to outline the overall construction and identify the functionality provided throughout the Foodies Application.

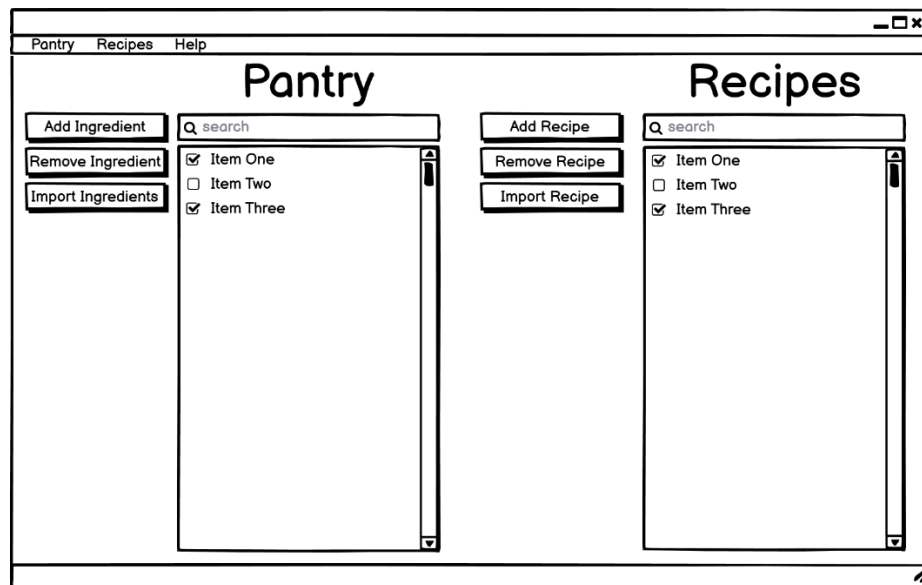
5.2 Functional Description

The Foodie App is designed to allow the user to input cooking items currently on hand. This list is stored locally and is loaded every time the app is started. Recipes can also be loaded via a manual entry process or scanned via text file. These recipes are also stored via text file and loaded every time the app is started.

Once running the user will have multiple functions to use. To make the app most effective that user will have to provide food items currently on hand. They will also be required to import or provide recipes to populate the database. At this point the user will be able to search previously stored recipes. Once selected the app will check the inventory of the Pantry and generate a list of items the user will need to purchase to make selected recipe. The Functional diagram below represents the different Java Classes that make up the Foodie Application.



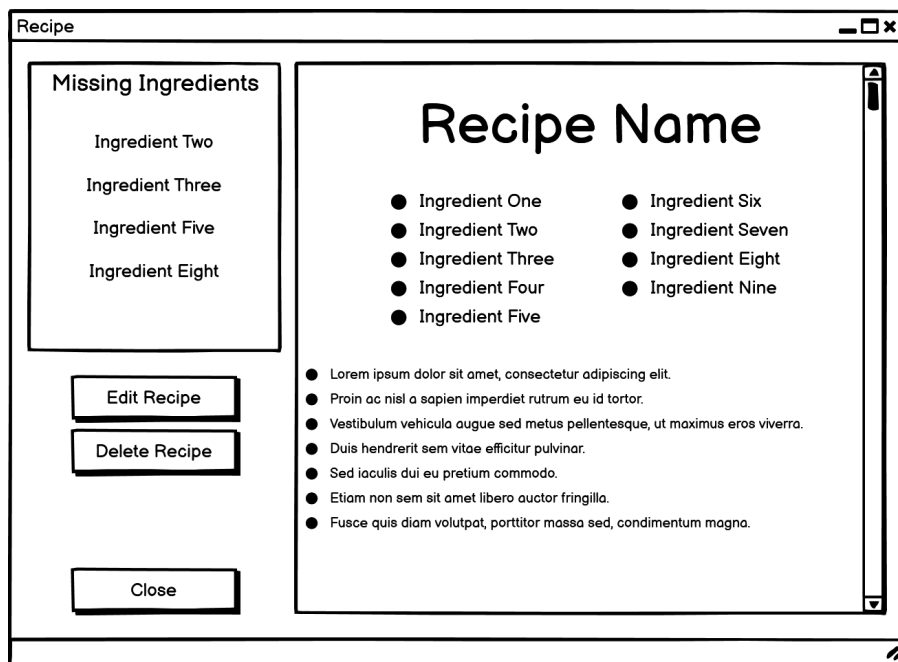
5.2.1 User Interface



5.2.2 Main Window:

- Navigation Bar
 - Pantry
 - Import Ingredients
 - Clear Pantry
 - Selecting this will display a popup window for the user to confirm
 - Recipes
 - Import Recipe
 - Clear Recipes
 - Selecting this will display a popup window for the user to confirm
 - Close (closes the program)
- Pantry
 - Buttons
 - Add Ingredient
 - Popup with a text field to add an ingredient. Multiple ingredients can be added with a comma delimiter
 - Remove Ingredient
 - Removes the selected ingredient(s) from the list
 - Import Ingredients
 - Displays a popup window to choose a text file
 - Search Bar – Search for ingredients in the list
 - Ingredients List
 - Scrolling window with a list of ingredients. Checkboxes to select ingredients.
- Recipes

- Buttons
 - Add Recipe
 - Popup with a text field to add a recipe
 - Remove Recipe
 - Removes the selected recipe(s) from the list
 - Import Ingredients
 - Displays a popup window to choose a text tile
- Search Bar – Search for recipes in the list
- Ingredients List
 - Scrolling window with a list of recipes. Checkboxes to select ingredients.
 - Double clicking a recipe opens it in a new window (see Recipe Window)



5.2.3 Recipe Window:

- Missing Ingredients
 - Displays the ingredients from the selected recipe that are missing from the pantry
 - Recipe frame
 - Scrolling window that displays the ingredients and instructions for the recipe
 - Buttons
 - Edit Recipe
 - Allows the user to edit the recipe in the recipe frame
 - Delete Recipe
 - Deletes the recipe after a confirmation popup and closes the recipe window
 - Close
 - Closes the recipe window

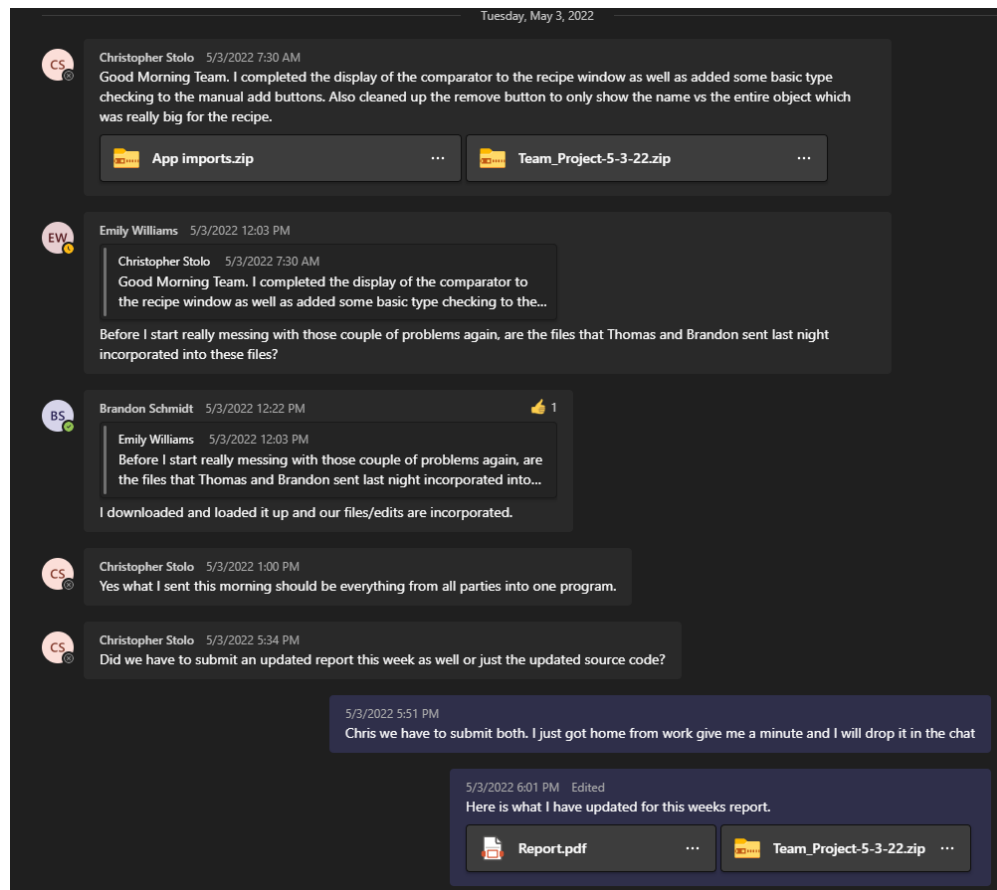
Development History

Team Foodies

6 Development History

6.1 Week 1

For week one, there wasn't any code development as it was spent forming the group, finalizing the application idea, and determining the logistics/administrative items for managing the project. Our group decided to call our application the Foodies App, due to the food basis of the application. The Foodies App was solidified as a pantry manager, recipe comparison, recipe manager, and grocery list generator, all rolled into one application. To bring this application to life, we choose to use Microsoft Teams as our primary communication platform. The group met every Thursday at 7:00 PM EST to discuss feedback from the previous week, the tasks we needed to accomplish over the next couple of days, and how we plan to achieve completion of those tasks. In between group meetings, our team used the chat feature to communicate effectively. As we all had different schedules the chat feature made it easy to accommodate those needs. Teams also enabled us to share our documentation and act as a repository when needed for our documentation. Another core reason we went with Microsoft Teams, was because of its easy-to-use mobile application and its seamless integration with the web and desktop application. For those that work full-time, this enabled us to stay in touch with the group while at work and away from our computers. The image below shows an example of that communication.

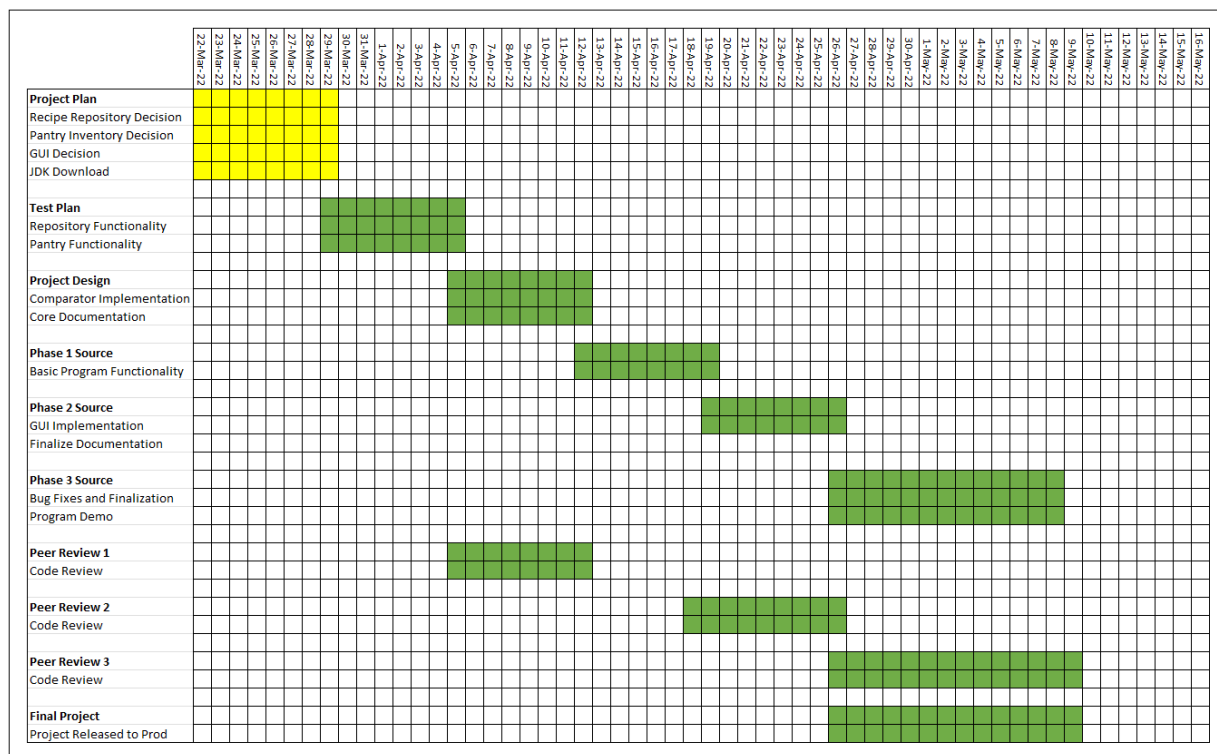


6.2 Week 2

During week 2 of the course, the team worked to create our Project Plan and Project Specifications for the Foodies App. First, we began generating the ideas of what the application would need to be successful and how we would further define that success. This let us dig deeper into breaking those success criteria into components, and then furthermore into specific elements. We identified four major elements for our application, pantry inventory, recipe repository, comparator, and a graphical user interface. With those established, the team focused on the big elephant in the room, accomplishing these success criteria in the remaining weeks of this course. As a group we walked back from the submission date of May 10th, 2022, to the start of week 2. With those chunks of time established it became an exercise of pinning high level tasks into that timeframe. Occasionally we needed to shuffle around the tasks to fit, what we thought would be a reasonable project schedule. With success and a schedule defined, we started to volunteer and/or assign tasks to team members. As those roles and responsibilities were ironed out, we went around the team for potential schedule conflicts to identify them as early as possible. This helped us have coverage and fill in the gaps as needed in the remaining weeks.

Week 2 Items:

- Create the Project Plan
- Create Project Schedule
- Create Contribution Log

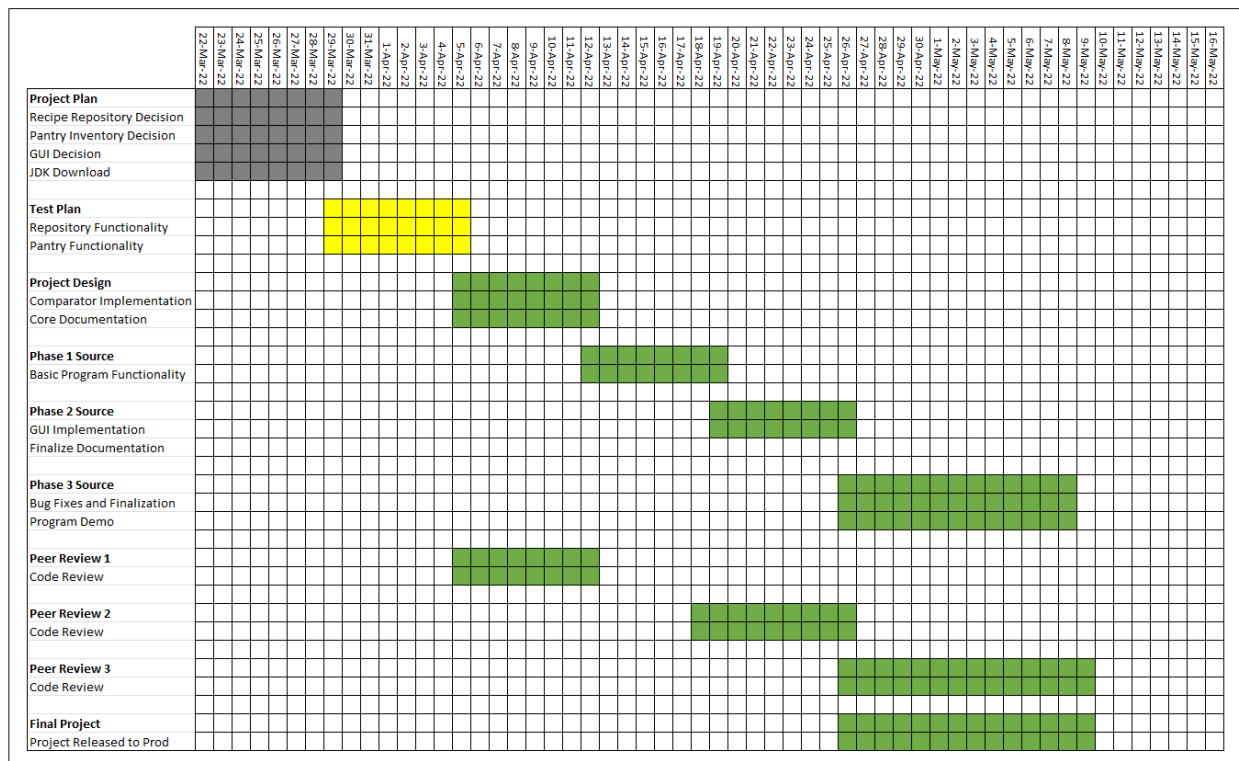


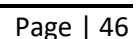
6.3 Week 3

During week 3 of the course, the team worked to User Guide, Test Plan, and Base Functionality (Terminal) for the Pantry and Recipe components. The team started to generate the questions of what we wanted our users to experience when using the Foodies App. This enabled us to start pairing experience and functionality requirements together. This would include a table style view that would display the information to the user and the functionality would revolve around these tables. Buttons would be included to enhance the users experience interacting with the tables. The main thing we wanted was so that the user could interact directly with the table, a natural and instinctive progression for a user. As these pairings developed further, we began our rough outline of organizing them into the flow we wanted our users to experience using the application in its totality. Taking it a step further, we identified specific tests to determine if those functions were working as intended. To do that we needed to define our test success criteria, when would test, and how we would test. This plan quickly formed into the create of the Test Plan. With a tentative roadmap laid out for the documentation, the group began the development of the initial code base.

Week 3 Items:

- Create the User Guide
- Create the Test Plan
- Create the Test Log
- Update Project Schedule
- Update Contribution Log
- Base Functionality (Terminal)





Conclusions

Team Foodies

7 Conclusions

7.1 Lessons Learned

This was a fun project overall. For most of the team, this was our first experience working with team members in the context of development. So, this was an excellent exercise to prepare us for what comes next as we enter the workforce as developers/IT professionals. Listed below are a few lessons learned throughout the construction of the Foodies App.

Start Small – Our group had an ambitious application idea from the start. It was based on three major components Pantry Manager, Recipe Manager, and Comparator. Initial we split the team and assigned the individuals to work on each component in parallel work streams. This was an executive decision that was made based on the timeframe we had to get the project done. However, when it came time to integrate each parallel workstream into each other, it became quite the process. This showed the team that it would have been better to work each component in queue format. Finalizing each component before moving to the next one.

Add Logging or Error Handling Early – The addition of logging or error handling was not something we included in our initial design for this application. This was deemed not necessary for the task at hand, but something as individuals we could add to the project after this course is over. However, when it became time to integrate each component, this logging and error handling would have been beneficial. Realizing this, we began to include error handling messages as pop-ups for the customer. This helped but will need to be taken another step further.

Understand the Code Elements – While our group had experience using JTable before, our design called for more experience than what was on the team initially. To achieve this, we decided to use TableModel's which extended the AbstractTableModel class. This was new to everyone, so it slowed things down while trying to research and understand the capabilities and requirements of the TableModel class. Since these were the center point of our application this was a big necessity to understand.

Change one thing at a time – Like every team we ran into some situations where it was difficult to track changes. Sometimes team members would make what they thought were small clean up changes only to find out that those changes caused bugs and errors elsewhere in the code. This made things extremely time-consuming doing comparison to track all the changes made.

Pseudo Code is important – The complexity of this project was a lot more than we initially anticipated. Some of it was due to the goals we set out to accomplish with our components and some of it was from a design perspective. As mentioned earlier, our components were built somewhat in silo's. The integration became increasing difficult for certain items. Mainly, the integration of the GUI. The initial code was all written to be tested in the terminal, this code had to be re-written to work with the GUI. This brought more complexity getting the GUI work and perform as anticipated. Often it was needed to write pseudo code to help keep us on track on what functionality needed to be accomplished.

7.2 Design Strengths

While the group is proud of the work contributed to the Foodies App, we would like to highlight the strengths of our design. Below we have listed our two major design strengths.

Built for the users – In the previous section, we mentioned that the TableModel Class was something new to the team and caused significant time to be allocated to understanding it. This effort was to provide the user with the most natural experience when faced with data presented in a table format. Instead of editing in another view or window, the user can simply select the ‘Edit Ingredient’ button and apply their changes directly in the table itself, then save the changes. This is something we would like to further build out as individuals after this course.

Meets the requirements – While it does not have all the bells and whistles yet, at its core, the Foodies App accomplishes the three base components we set out to achieve. That core functionality is the crux of our application and will enable us to adapt and apply new features going forward.

7.3 Limitations

Key limitations for this project will be discussed in this section that pertain to the development of the Foodies Application. These limitations have been broken down into three categories: Resource Management, Time Management, and Personnel Management.

7.3.1 Resource Management

This project had zero budget. Therefore, the application was designed with that in mind. The application needed to be able to function at a core level in any environment and with the assumption that it would have limited users. This budget prohibited the use of potential API’s that required payment. One section that would have benefited from the use of an API was the recipe portion. Being able to include recipe database hooks or for suggestions. Therefore, all design considerations for the development of this application needed to be limited to “free-tier” and open-source services only.

7.3.2 Time Management

Time management is a limitation for every project with a hard deadline that has no flexibility to it. In our case this is referring to the 8-week deadline of the course length. In a professional setting it is not uncommon to have a feature developed within an 8-week timeline, but not an entire application. This tight timeline added significant constraints to what features went into the application.

7.3.3 Personnel Management

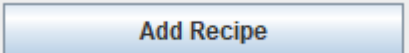
Unlike in a professional setting, our workforce was voluntary and was not dedicated to this project in a Full-Time capacity. While some individuals were students taking other classes and other individuals worked a full-time job, no one personnel resource was able to dedicate 40 hours a week to the contribution of this effort. The management of personnel required fluidity in trying to achieve the desired outcome. In a professional setting this would be negated through a dedicated development team.

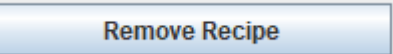
7.4 Future Improvements

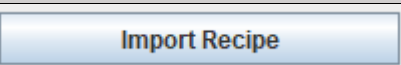
There are many enhancements that can be made for this application purely based on the nature of the application, however, below are a list of the prominent features that come to mind.

-
- Add additional fields for Ingredients/Food Items (i.e., Food Expiration Dates, calories, brands, nutrition values)
 - Add alerts for food that is about to expire
 - Add running low notification/alert
 - Add a barcode feature to scan in food items
 - Add profiles and household features
 - Add recipe filtering for specific diets (Keto, Gluten Free, Vegan, etc...)
 - Add recipe filtering for calorie ranges
 - Add grocery list generator
 - Add a share recipe feature (via text message or email)
 - Add a share grocery list (via text message or email)
 - Add Frequently bought suggestions when selecting grocery list
 - Add profiles
 - Add login feature
 - Add Authentication and authorization for profiles and sharing features
 - Add encryption for profile passwords
 - Add some sort of logging feature
 - Add a database presence
 - Add Info and Help pop-ups to better assist the users
 - Adjust the table to handle live edits versus toggling on and off with the Edit Ingredient button.

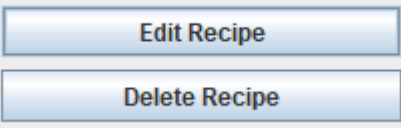
8 APPENDIX I – Test Log


Test/Review Period	Phase 1
Tester	Christopher Stolo
Component	Recipe Repository
Function	Add Recipe Button
Severity	High
Event/Bug Description	
The add recipe button does not work.	
Screenshot of Error/Error Code	
	

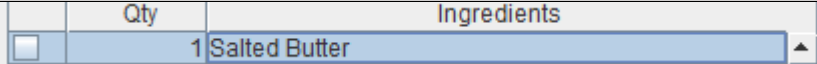
Test/Review Period	Phase 1
Tester	Christopher Stolo
Component	Recipe Repository
Function	Remove Recipe Button
Severity	High
Event/Bug Description	
The remove recipe button does not work.	
Screenshot of Error/Error Code	
	

Test/Review Period	Phase 1
Tester	Christopher Stolo
Component	Recipe Repository
Function	Add Recipe Button
Severity	High
Event/Bug Description	
The import recipe button does not work	
Screenshot of Error/Error Code	
	

Test/Review Period	Phase 1
Tester	John Cole
Component	Recipe Window
Function	Open Recipe Button
Severity	High
Event/Bug Description	
While the Open recipe button opens to the recipe window the functional buttons within the window do not work.	


Screenshot of Error/Error Code	
	

Test/Review Period	Phase 1
Tester	Brandon Schmidt
Component	Pantry Manager
Function	Buttons
Severity	High
Event/Bug Description	
Button Functionality does not work.	
Screenshot of Error/Error Code	
	

Test/Review Period	Phase 2
Tester	Thomas Zack
Component	Pantry Manager
Function	Edit Ingredient
Severity	Medium
Event/Bug Description	
I can remove and re-add an ingredient to make changes, however, I cannot select within the table to make my edits. The table is locked from editing.	
Screenshot of Error/Error Code	
	

Test/Review Period	Phase 2
Tester	Brandon Schmidt
Component	Pantry Manager
Function	Bulk Delete
Severity	Medium
Event/Bug Description	
I can delete through the drop-down list; however, I cannot bulk delete ingredients in Pantry table through the checkboxes.	
Screenshot of Error/Error Code	

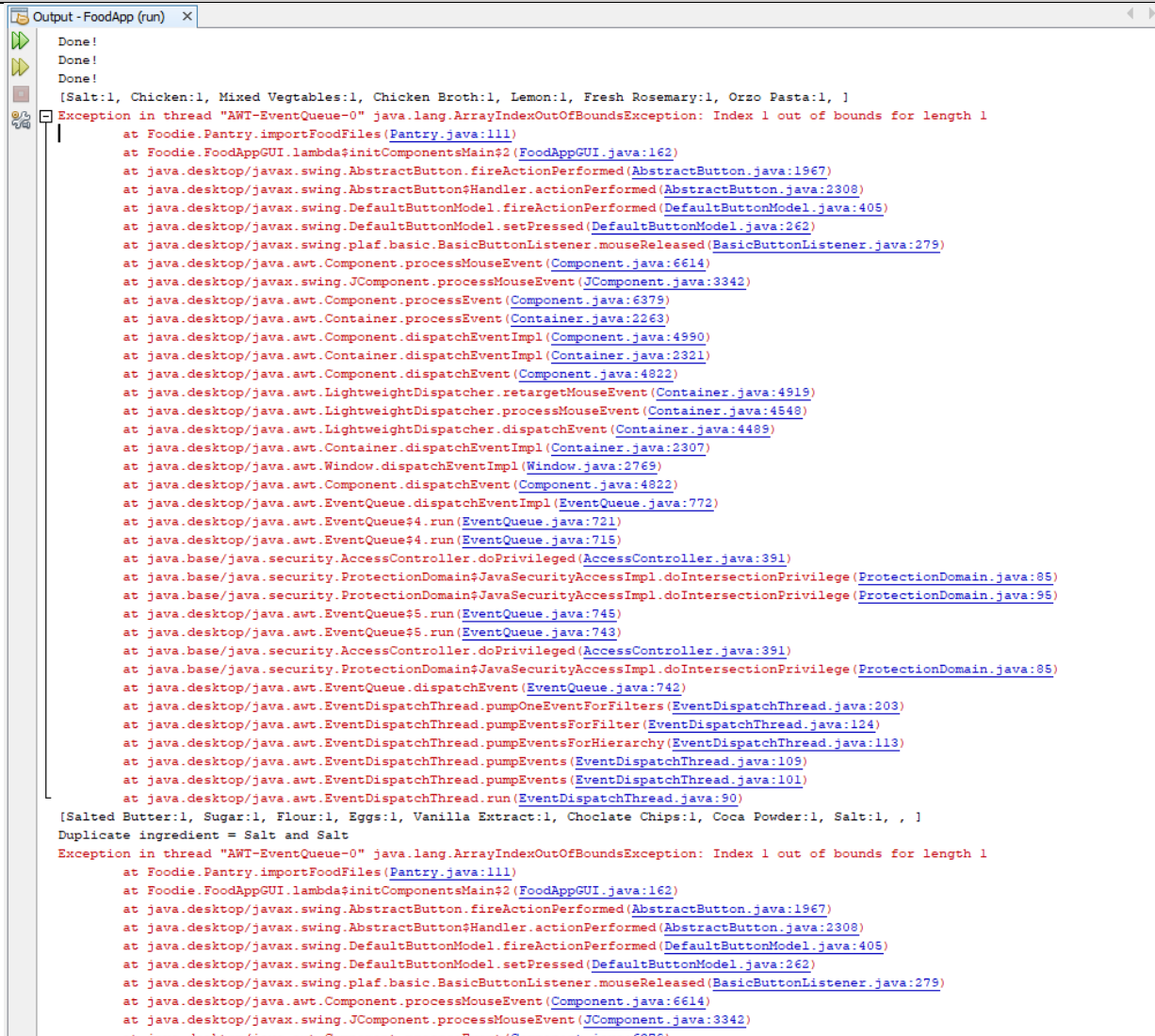
	Qty	Ingredients
<input checked="" type="checkbox"/>	1	Salted Butter
<input checked="" type="checkbox"/>	1	Eggs
<input checked="" type="checkbox"/>	1	Vanilla Extract
<input type="checkbox"/>	1	Chocolate Chips
<input type="checkbox"/>	1	Cocoa Powder
<input type="checkbox"/>	2	Salt

Test/Review Period	Phase 2
Tester	Emily Williams
Component	Recipe Repository
Function	Buttons
Severity	High
Event/Bug Description	
The buttons for the Recipe Repository do not function.	
Screenshot of Error/Error Code	
	

Test/Review Period	Phase 2
Tester	John Cole
Component	Recipe Repository
Function	Edit Recipe
Severity	Medium
Event/Bug Description	
If I select Edit Recipe then hit cancel. The application still runs but an exception is thrown.	
Screenshot of Error/Error Code	
<pre>Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException: Cannot invoke "String.isBlank()" because "recipeName" is null at Foodie.RecipeWindow\$.actionPerformed(RecipeWindow.java:88) <4 internal lines> at java.desktop/javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:279) <30 internal lines></pre>	

Test/Review Period	Phase 3
Tester	John Cole
Component	Recipe Repository
Function	Edit Recipe
Severity	Medium
Event/Bug Description	
Edits reflected in the repository.txt file and in the recipe table, but not in the recipe window. In addition, I receive the following error when attempting to refresh the recipe window.	
Screenshot of Error/Error Code	

```
Exception in thread "AWT-EventQueue-0" java.lang.ClassCastException: java.lang.String cannot be cast to class Foodie.Ingredient (java.lang.String is in module java.base of loader "bootstrap"; Foodie.Ingredient is in unnamed module of loader "app"
at Foodie.RecipeWindow.showRecipeIngredients(RecipeWindow.java:216)
at Foodie.RecipeWindow.initComponentRecipe(RecipeWindow.java:71)
at Foodie.RecipeWindow.<init>(RecipeWindow.java:65)
at Foodie.RecipePanel.lambda$initComponentsMain$2$($$Lambda$1000$)
at java.desktop/java.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
at java.desktop/java.awt.Component.processMouseEvent(Component.java:6614)
at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:4990)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2321)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2307)
at java.desktop/java.awt.Window.dispatchEventImpl(Window.java:2769)
at java.desktop/java.awt.Component.dispatchEvent(Component.java:4822)
at java.desktop/java.awt.EventQueue.dispatchEventImpl(EventQueue.java:772)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:721)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:715)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:391)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:85)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:95)
at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:745)
at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:743)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:391)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:85)
at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:742)
at java.desktop/java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:203)
at java.desktop/java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:124)
at java.desktop/java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:113)
at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:105)
at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:101)
at java.desktop/java.awt.EventDispatchThread.run(EventDispatchThread.java:90)
[Salt:1, Chicken:1, Mixed Veggies:1, Chicken Broth:1, Lemon:1, Fresh Rosemary:1, Orzo Pasta:1, ]
Duplicate ingredient = Salt and Salt
Exception in thread "AWT-EventQueue-0" java.lang.ArrayIndexOutOfBoundsException: Index 1 out of bounds for length 1
at Foodie.Pantry.importFoodFiles(Pantry.java:111)
at Foodie.FoodAppGUI.lambda$initComponentsMain$2$($$Lambda$1000$)
at java.desktop/java.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
at java.desktop/java.awt.Component.processMouseEvent(Component.java:6614)
at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:4990)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2321)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2307)
at java.desktop/java.awt.Window.dispatchEventImpl(Window.java:2769)
at java.desktop/java.awt.Component.dispatchEvent(Component.java:4822)
at java.desktop/java.awt.EventQueue.dispatchEventImpl(EventQueue.java:772)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:721)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:715)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:391)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:85)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:95)
at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:745)
at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:743)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:391)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:85)
at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:742)
at java.desktop/java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:203)
at java.desktop/java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:124)
at java.desktop/java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:113)
at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:105)
at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:101)
at java.desktop/java.awt.EventDispatchThread.run(EventDispatchThread.java:90)
```

Test/Review Period	Phase 3
Tester	Emily Williams
Component	Pantry Manager
Function	Import Pantry
Severity	High
Event/Bug Description	
Trying to modify ingredients in the table	
Screenshot of Error/Error Code	
	

9 APPENDIX II – Contribution Report

9.1 Week 1

Member	Contribution
Brandon Schmidt	Joined the team.
Christopher Stolo	Joined the team.
Emily Williams	Joined the team.
John Cole	Joined the team.
Olajide Awoyera	Joined the team.
Thomas Zack	Joined the team.

9.2 Week 2

Member	Contribution
Brandon Schmidt	Participated in team meeting and provided input to project decisions.
Christopher Stolo	Participated in team meeting and provided input to project decisions.
Emily Williams	Participated in team meeting and provided input to project decisions.
John Cole	Provided Software Requirement Specification input Participated in team meeting and provided input to project decisions.
Olajide Awoyera	Provided Work Breakdown Structure and Project Schedule Participated in team meeting and provided input to project decisions.
Thomas Zack	Participated in team meeting and provided input to project decisions.

9.3 Week 3

Member	Contribution
Brandon Schmidt	Participated in team meeting and provided inputs for Recipe portion User Guide and Test Plan.
Christopher Stolo	Participated in team meeting and provided inputs for Pantry portion User Guide and Test Plan.
Emily Williams	Participated in team meeting and provided inputs for GUI portion User Guide and Test Plan.
John Cole	Gathered inputs and created User Guide and Test Plan. Participated in team meeting and provided input to project decisions.
Olajide Awoyera	Reviewed User Guide and Test Plan. Participated in team meeting and provided input to project decisions.
Thomas Zack	Participated in team meeting.

9.4 Week 4

Member	Contribution
Brandon Schmidt	Participated in team meeting and provided inputs for Recipe portion of Design Document.
Christopher Stolo	Participated in team meeting and provided inputs for Pantry portion of Design Document.

Emily Williams	Participated in team meeting and provided inputs for GUI portion of Design Document.
John Cole	Created template for Report document based on feedback from professor, combining previous file submissions into a single document. Participated in team meeting and provided input to project decisions.
Olajide Awoyera	Gathered inputs and created Design Document. Participated in team meeting and provided input to project decisions.
Thomas Zack	Participated in team meeting.

9.5 Week 5

Member	Contribution
Brandon Schmidt	Reviewed code.
Christopher Stolo	Participated in team meeting and provided inputs for Pantry portion of Code.
Emily Williams	Participated in team meeting and provided inputs for GUI portion of Code.
John Cole	Participated in team meeting and updated documentation.
Olajide Awoyera	Participated in team meeting.
Thomas Zack	Participated in team meeting reviewed code.

9.6 Week 6

Member	Contribution
Brandon Schmidt	Provided inputs for Recipe portion of Code.
Christopher Stolo	Participated in team meeting and provided inputs for Pantry portion of Code.
Emily Williams	Participated in team meeting and worked with John on GUI integration.
John Cole	Participated in team meeting, updated documentation, and integrated Pantry functionality with GUI actionlisteners. Worked with Emily on GUI.
Olajide Awoyera	Traveling and notified the team ahead of time.
Thomas Zack	Participated in team meeting.

9.7 Week 7

Member	Contribution
Brandon Schmidt	Participated in team meeting and worked on finalizing Recipe component.
Christopher Stolo	Participated in team meeting and worked on finalizing Pantry component.
Emily Williams	Participated in team meeting and worked on finalizing GUI component.
John Cole	Participated in team meeting, updated documentation.
Olajide Awoyera	Participated in team meeting.
Thomas Zack	Participated in team meeting reviewed code.

9.8 Week 8

Member	Contribution
Brandon Schmidt	Participated in team meeting and finalized code.
Christopher Stolo	Participated in team meeting and finalized code.
Emily Williams	Participated in team meeting and finalized code.
John Cole	Participated in team meeting, updated documentation, and finalized code.
Olajide Awoyera	Participated in team meeting.
Thomas Zack	Participated in team meeting reviewed code.