

Project Proposal

Phase 1

1.1 Project Background/Overview:

Introduction about the company, the context

In 2022, the average US consumer spent roughly 10% of their income on food (Economic Research Service, 2023), and stats have shown that inflation has greatly increased food prices in recent years compared to the inflation rate reported from 2002 to 2021. Notably, in 2021 a third of USD spent on food was spent on eating out or takeaway, and again rose dramatically with “food-away-from-home spending [accounting] for 56 percent of total food expenditures in 2022” (Economic Research Service, 2023). This number also does not account for the reported amount spent on microwavable meals, which in 2021 the national average spent on just frozen prepared meals was over 150 USD (Wunsch, N.G., 2022). Even when cooking, “the average household wastes 31.9% of the food it buys” (Yu et al. 2020). The goal of our company, *MHDR Software*, is to simplify the process of making home-cooked meals for our users and create new methods for users to reduce their food waste. This will help consumers reduce the portion of their income spent on food and food services.

1.2 Problem Description:

Briefly describe the problem and the need to solve it

For many people, cooking can be an overwhelming task, particularly for those who are on a tight budget. It can also be challenging to keep track of what ingredients you currently have in your kitchen, the quantity of those ingredients, and their expiration dates. Furthermore, with so many different methods of finding new recipes, whether that be from websites, physical cookbooks, or word-of-mouth, it can be difficult to keep track of all your favorite meals. Our proposed app, *Personal Cookbook* aims to alleviate some of the mental and financial stress of cooking by consolidating the user’s desired recipes on one easy-to-use app.

1.3 Project Scope:

Describe the aims, benefits and the outcome of the solution developed

Through the *Personal Cookbook* app, users will be able to create a personalized online database of their favorite recipes. *Personal Cookbook* users will also be able to enter the foods they currently have in their kitchen to easily determine what ingredients they need for any given recipe. Additionally, this app will help users easily browse through their favorite meals, preserve family recipes, and reduce the complexity of meal planning. Overall, the *Personal Cookbook* app will be beneficial by helping users save time, money, reduce food waste, and plan out balanced, nutritious meals.

One of the desired outcomes of our app is to allow users to build a *Personal Cookbook Library* and *Personal Cookbook User Pantry*. These features will allow users to add, modify, or remove a desired recipe/food. Additionally, users will have the option to select the meals they

would like to make and automatically generate a grocery list based on the ingredients they do not already have in their kitchen. Finally, the user's recipes in the *Personal Cookbook* app will be searchable by specific ingredients. This may be particularly useful for users who have a large quantity of one ingredient or are trying to use up an item before it goes bad.

1.4 Project Objectives:

Describe the predetermined results or achievements

Our *Personal Cookbook* app will be built using Python and will require the PyQt5 module to build the graphical user interface. Users will begin interacting with our app by creating their *Personal Cookbook Library* and *Personal Cookbook User Pantry*. To create their library, users will be prompted to enter the title, ingredients, quantity required for each ingredient, instructions, and a brief description for a given recipe. Once the user has submitted their recipe, they will be able to browse through their library, make any adjustments to the recipe, add additional notes, or remove a recipe as desired. Each new recipe will default to private unless the user explicitly selects to make it available to other users in the *Personal Cookbook* community. Similarly, users will also be able to build and edit their *Personal Cookbook User Pantry* by adding the name and corresponding quantities of foods they currently have available in their kitchens. Unlike the cookbook library, all entries within a user's pantry will remain private. To find new recipes, users will be able to browse the *Personal Cookbook Public Library* to add community created recipes to their personal library.

Upon selecting a recipe(s) to cook, the *Personal Cookbook User Pantry* will be updated to reflect the new remaining quantity based on each recipe's required ingredients. Additionally, a grocery list will automatically be created for any ingredient(s) the user does not already have. If any food in the user's pantry is used up after making a recipe, these items will automatically be removed from the *Personal Cookbook User Pantry*. Users will also be able to provide ingredient(s) that they want to cook with and view a list of recipes from their Library that require those ingredient(s). For each recipe that the user selects to prepare, they will have the option to adjust the serving yield as well as the units of measurement used in a recipe as desired.

These objectives are summarized in the following list:

1. Build and browse through an editable *Personal Cookbook Library*
2. Build and browse through an editable *Personal Cookbook User Pantry*
3. Publish recipes so that they are available to other users
4. Select desired recipes to make
5. Update the *Personal Cookbook User Pantry* after a recipe has been selected
6. Automatically generate a grocery list based off the recipe(s) selected and food the user currently has in their kitchen
7. Search for recipes containing specific ingredient(s)
8. Easily adjust the number of servings per recipe
9. Change units of measurement within a recipe

Citations:

Economic Research Service. (2023, Jul 19). *Food Prices and Spending*. U.S. Department of Agriculture. <https://www.consumerfinance.gov/consumer-tools/managing-someone-elses-money/>

Wunsch, N.G. (2022, Oct 7). *Average annual expenditure on frozen prepared foods per consumer unit in the United States from 2007 to 2021 (in U.S. dollars)**. Statista. [U.S. expenditure on frozen prepared foods 2007- 2021 | Statista](#)

Yu, Y. and Jaenicke, E.C. (2020), Estimating Food Waste as Household Production Inefficiency. *Amer. J. Agr. Econ.*,102: 525-547. <https://doi.org/10.1002/ajae.12036>

Phase 2

2.1 Project Background/Overview:

Introduction about the company, the context

WBS Items	Units/Hrs	Cost/Unit/Hr.	Subtotals	WBS Level 2 Totals	% of Total
1. Project Management				\$180,000.00	16
Project Manager	720	\$100.00	\$72,000.00		
Software Development Team Manager	720	\$75.00	\$54,000.00		
IT and Security Team Manager	720	\$75.00	\$54,000.00		
2. Hardware				\$34,000.00	3
Servers	3	\$3,000.00	\$9,000.00		
Programming Hardware (Laptops, Monitors, etc.)	10	\$2,500.00	\$25,000.00		
3. Software					
Licensed Software (GitHub Enterprise, Oracle, Microsoft Office Suite, etc.)	100	\$200.00	\$20,000.00	\$529,040.00	46
Back-End App Development (Software Engineers)	2,880	\$54.00	\$155,520.00		
Back-End App Development (Independent Contractors)	1,440	\$45.00	\$64,800.00		
Front-End App Development (Software Engineers)	2,880	\$54.00	\$155,520.00		
Front-End App Development (Independent Contractors)	720	\$45.00	\$32,400.00		
Website Domain Name	1	\$500.00	\$500.00		
SSL Certificate and Instalation	1	\$300.00	\$300.00		
Website Security	1	\$100,000.00	\$100,000.00		
4. Testing (Additioanl 10% of Hardware and Software)				\$56,304.00	5
5. Training and Support				\$204,400.00	18
Training Cost	100	\$100.00	\$10,000.00		
IT and Security Team	1,440	\$75.00	\$108,000.00		
Software and Training Support	1,440	\$60.00	\$86,400.00		
6. Reserves (15% of Estimate)				\$150,560.00	13
Total Estimate				\$1,154,304.00	

Table 1: The table above shows the estimated budget required to complete our application. The total estimated budget for this project is \$ 1,154,304.00. Most of this budget is anticipated to be spent on the hardware and software requirements. For this project, our company anticipates needing 1 project manager, 2 department managers, 11 software engineers, 2 IT and security specialists, and 2 training and support specialists. To help ensure our team does not run out of money while creating this app, a 15% reserve has been added to the total estimated budget.

2.2 Project Background/Overview:

Introduction about the company, the context

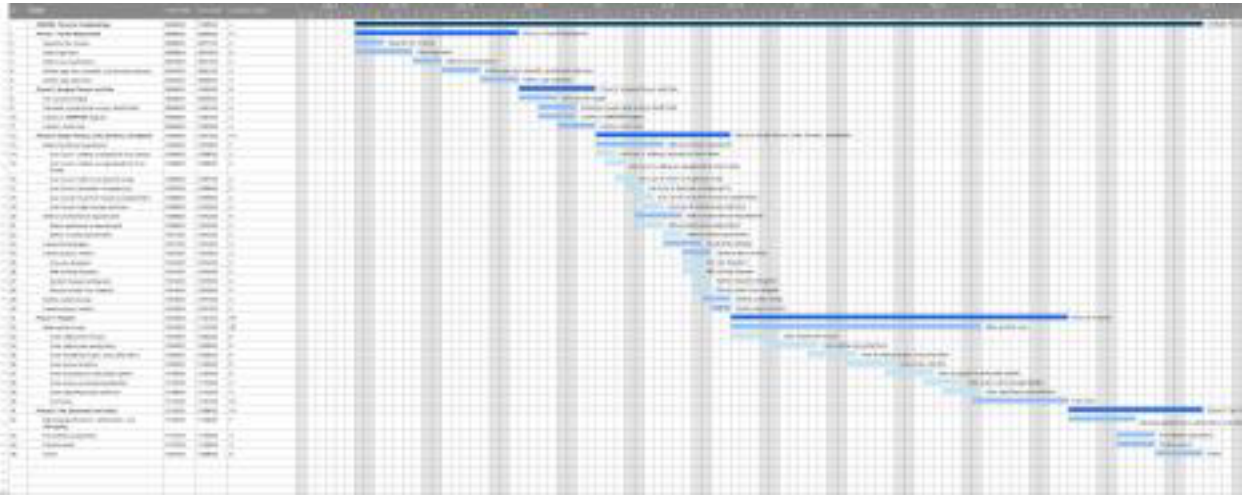


Figure 1: The figure above depicts a Gantt Chart. This figure shows the various tasks our group needs to complete to create our Personal Cookbook app, the estimated amount of time required, and the start and end date for each task.

* A larger version of this chart is also available in the *PersonalCookbookGantt.pdf* or *PersonalCookbookGantt.png*. *

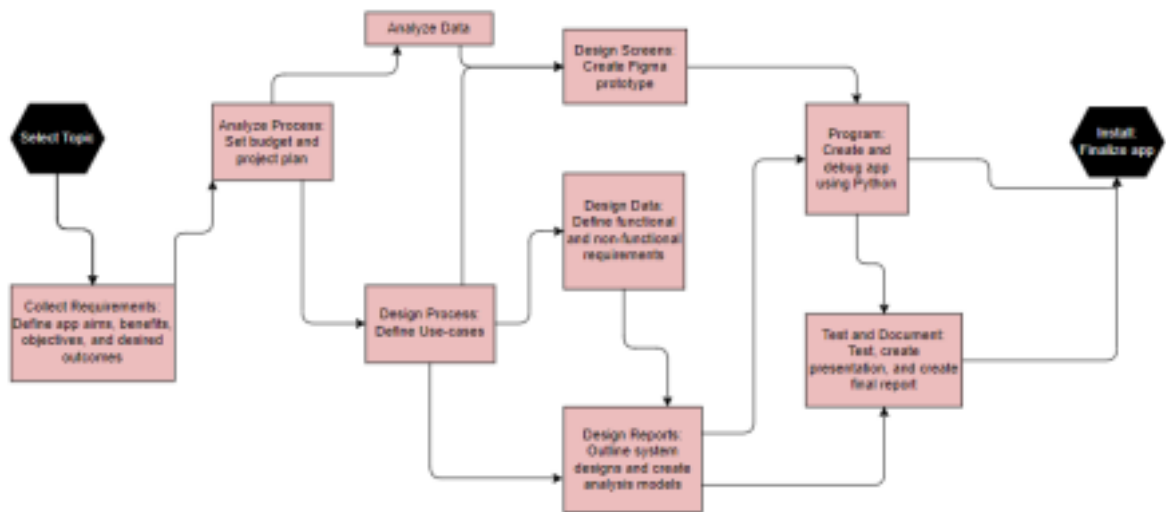


Figure 2: The figure above depicts an ADM/PDM diagram. This diagram is used to determine the schedule needed for our project and show what actions must be completed to begin another action.



Figure 3: The figure above depicts a Mind Map of our Personal Cookbook app. This diagram is organized into 4 categories with each of the features related to that category branching off.

Phase 3

Use Case Diagram

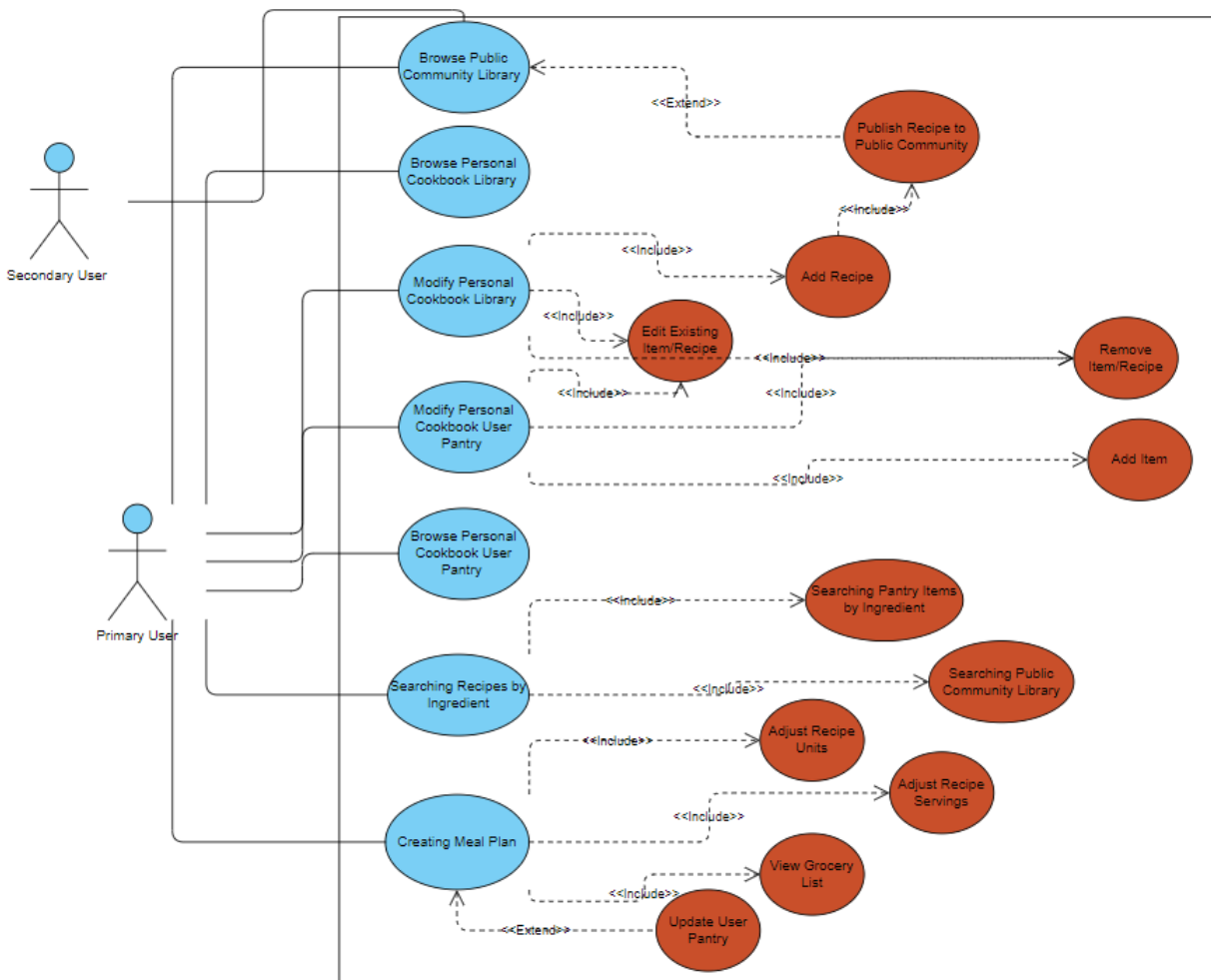


Figure 4: The figure above depicts the use case diagram for our Personal Cookbook application. This diagram shows the actors and the various system behaviors they can interact with.

Complete User Case Description

Use Case 1

Use case name		Building a <i>Personal Cookbook Library</i>
Scenario (purpose) Brief description		Users will use our app to store all their favorite recipes by entering the recipe title, ingredients, quantity of those ingredients, cooking instructions, and a brief description. Users will also be able to browse, modify, or remove any recipe from their library.
Triggering event		A user adding, viewing, modifying, or removing a recipe in their <i>Personal Cookbook Library</i> .
Actors		Primary user, secondary user
Related use cases		Logging into personal account
Priority		High
Stakeholder		Primary user, secondary user, <i>Personal Cookbook</i> team
Pre-conditions		The user must be logged into their account
Post-conditions		The user will be able to view their <i>Personal Cookbook Library</i> and/or see the change they have made (add, browse, modify, remove recipes).
Flow of activities		
S#	Actor Action	System Response

1	The user navigates to the <i>Personal Cookbook Library</i> page and selects to a) add a new recipe; b) browse current recipes; c) modify an existing recipe; or d) remove a recipe.	Depending on the action selected by the user, (add, browse, modify, or remove), the corresponding page opens.
2	If the user selects to add a recipe, they will fill out the recipe title, ingredients, quantity, instructions, and recipe description. Additionally, the user checks a box to make their recipe public or remain private.	The new recipe is added to the user's <i>Personal Cookbook Library</i> . If the user specifies to make their recipe public, it will also be published to the <i>Public Community Library</i> .
3	If the user selects to browse their recipes, they will be able to see the title and description of all their recipes.	The user can scroll through a list of each recipe (title and description) in their library.
4	1.1 If the user selects to modify an existing recipe, they will enter the name of the recipe, the type of information they wish to change, and the new value.	The recipe that the user has selected to modify will reflect the change(s) specified.

Use Case 2

Use case name		Building a <i>Personal Cookbook User Pantry</i>
Scenario (purpose) Brief description		Users can choose to track the foods and their quantities of items they currently have in their kitchen using our app. This includes adding, browsing, modifying, and removing any food from the <i>Personal Cookbook User Pantry</i> .
Triggering event		A user adding, viewing, modifying, or removing a food in their <i>Personal Cookbook User Pantry</i> .
Actors		Primary user
Related use cases		Logging into personal account
Priority		High
Stakeholder		Primary user, <i>Personal Cookbook</i> team
Pre-conditions		The user must be logged into their personal account
Post-conditions		The user will be able to view their <i>Personal Cookbook User Pantry</i> and/or see the change they have made (add, browse, modify, remove item).
Flow of activities		
S#	Actor Action	System Response
1	The user navigates to <i>the Personal Cookbook User Pantry</i> page and selects to a) add a new food; b) browse current	Depending on which action is selected (add, browse, modify, remove) the corresponding page opens.

	items in pantry; c) modify an existing item; or d) remove an item.	
2	If the user selects to add an item, they will fill out the item name title, quantity, and expiration date (optional).	The new item is added to the <i>user's Personal Cookbook User Pantry</i> .
3	If the user selects to browse through their pantry, they will be able to see the names of all the foods in their pantry, the quantity of each and the expiration date (if applicable).	The user is presented with a list of each item in their pantry that they can scroll through.
4	1.2 If the user selects to modify an existing food, they will enter the name of the food, the information they wish to change, and the new value.	The item that the user has selected to modify will reflect the change(s) specified.

Use Case 3

Use case name		Creating a Meal Plan
Scenario (purpose) Brief description		Users can choose to create a meal plan of the recipes they plan to make and automatically generate a grocery list based on the ingredients they do not currently have listed in their <i>Personal Cookbook User Pantry</i> . Users will also be able to modify the size and units used in each selected recipe.
Triggering event		A user selects the recipes they wish to make, units they want each recipe quantity to be displayed as, and number of servings for each recipe.
Actors		Primary user
Related use cases		Logging into personal account, building a <i>Personal Cookbook User Pantry</i> , building a <i>Personal Cookbook Library</i> .
Priority		Low
Stakeholder		Primary user, <i>Personal Cookbook</i> team
Pre-conditions		The user must have logged into their account and added at least 1 recipe into their <i>Personal Cookbook Library</i> .
Post-conditions		After a user builds their meal plan, a grocery list will automatically be generated, and the <i>Personal Cookbook User Pantry</i> will be updated.
Flow of activities		
S#	Actor Action	System Response

1	The user browses through their <i>Personal Cookbook Library</i> and selects the recipes they plan to make.	Each recipe selected is added to a list for future calculations.
2	When each recipe is added to the meal plan, the user will specify if they want to adjust the recipe serving size or units of measurement.	Depending on the user's responses, the ingredient quantities and units of measurement will be adjusted.
3	The user submits their meal plan.	The ingredients and amounts required for each recipe are subtracted from the ingredients the user already has in their pantry and a grocery list of all the foods they do not have is generated.
Exception conditions	1.3 1.1 This use case will not be available to the user if they are not logged into their personal account.	1.2 The user must have at least 1 recipe in their <i>Personal Cookbook Library</i> .

Use Case 4

Use case name		Searching For Recipes
Scenario (purpose) Brief description		Users can choose to browse or search through their personal or public recipes by ingredients required.
Triggering event		User selects to search for recipes by ingredient(s).
Actors		Primary user
Related use cases		Logging into personal account, building a <i>Personal Cookbook User Pantry</i> , building a <i>Personal Cookbook Library</i>
Priority		Medium
Stakeholder		Primary user, <i>Personal Cookbook</i> team
Pre-conditions		The user must have logged into their account and added at least 1 recipe to their <i>Personal Cookbook Library</i> .
Post-conditions		All recipes containing the ingredient(s) the user has specified in either the <i>Public Library</i> or the user's <i>Personal Cookbook Library</i> will be displayed.
Flow of activities		
S#	Actor Action	System Response
1	The user selects to search for recipes in either their personal or the community	All recipes containing that ingredient in the library specified will be listed.

	library by ingredient and enters at least 1 ingredient.	
2	The user selects to browse through their <i>Personal Cookbook Library</i> and search for recipe(s) by name.	If the recipe exists, the user will be able to view it.
3	The user selects to browse through the <i>Community Library</i> and search for recipe(s) by name.	If the recipe exists, the user will be able to view it.
Exception conditions	<p>1.1 This use case will not be available to the user if they are not logged into their personal account.</p> <p>1.2 If a recipe does not exist, it will not be visible to the user.</p>	1.3 If there are no recipes that use the ingredient entered, no recipes will be returned.

Complete Class Diagram

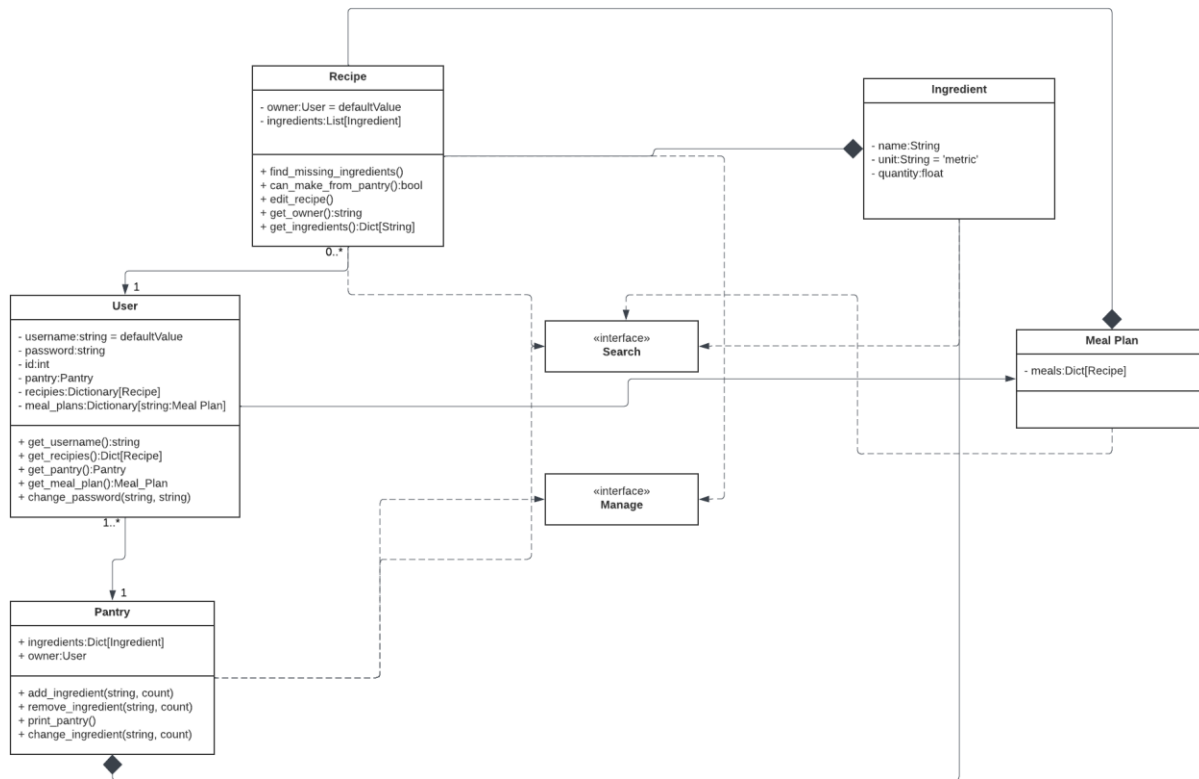


Figure 5: The figure above depicts the Complete Class Diagram for our Personal Cookbook application. This diagram shows the planned classes and their interactions with each other and the system.

Link to LucidChart: https://lucid.app/lucidchart/1a6c5359-020e-4a65-b71e-58d058d46796/edit?viewport_loc=-2138%2C565%2C2779%2C1226%2C0_0&invitationId=inv_453b7839-1308-40ca-b209-15cda02cb1d3

Complete Sequence Diagrams

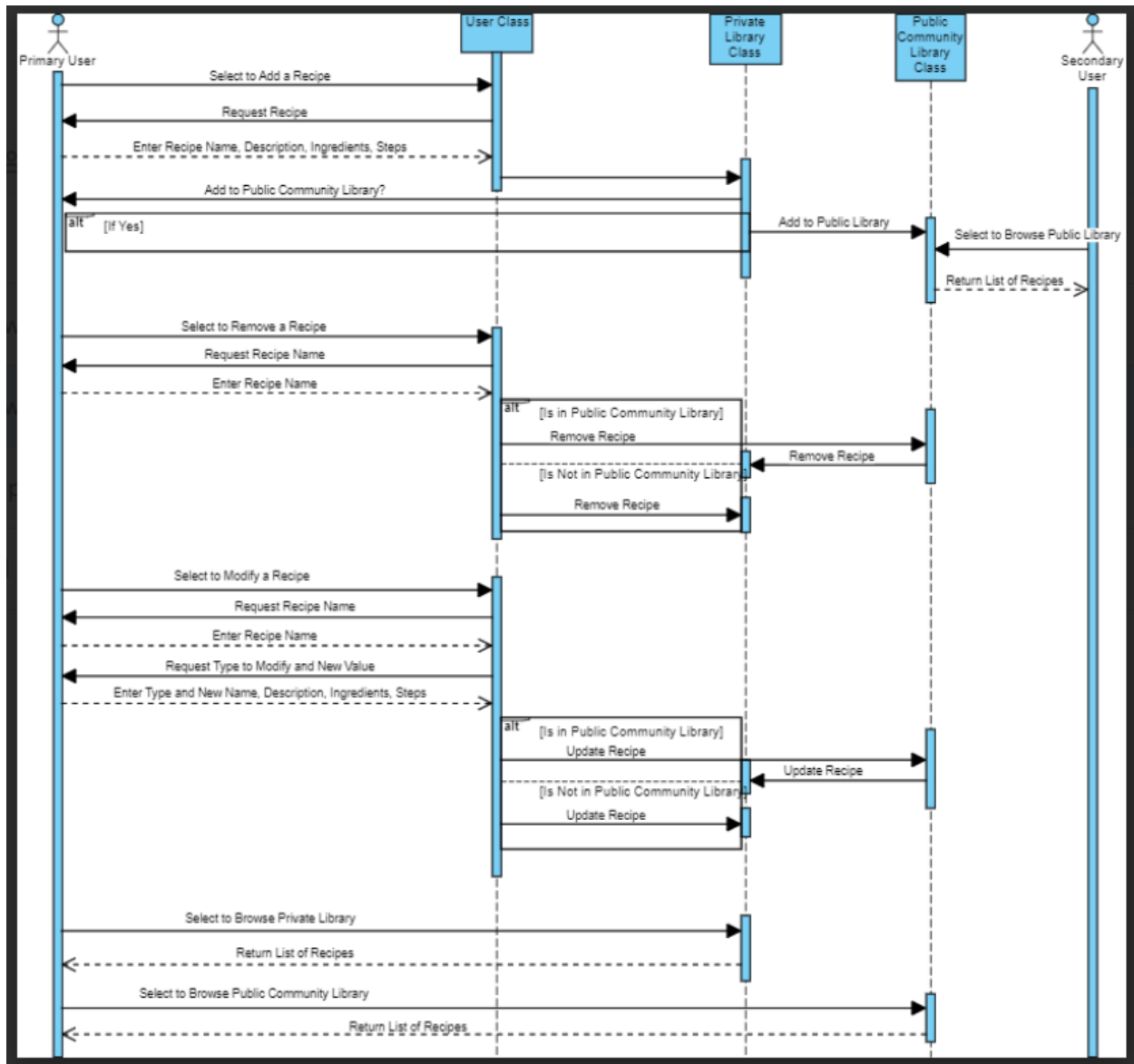


Figure 6: This figure depicts the system sequence diagram for the primary user to add, modify, remove, or browse items in either the public or private recipe libraries. Additionally, the system sequence for a secondary user to browse the public community library is also shown.

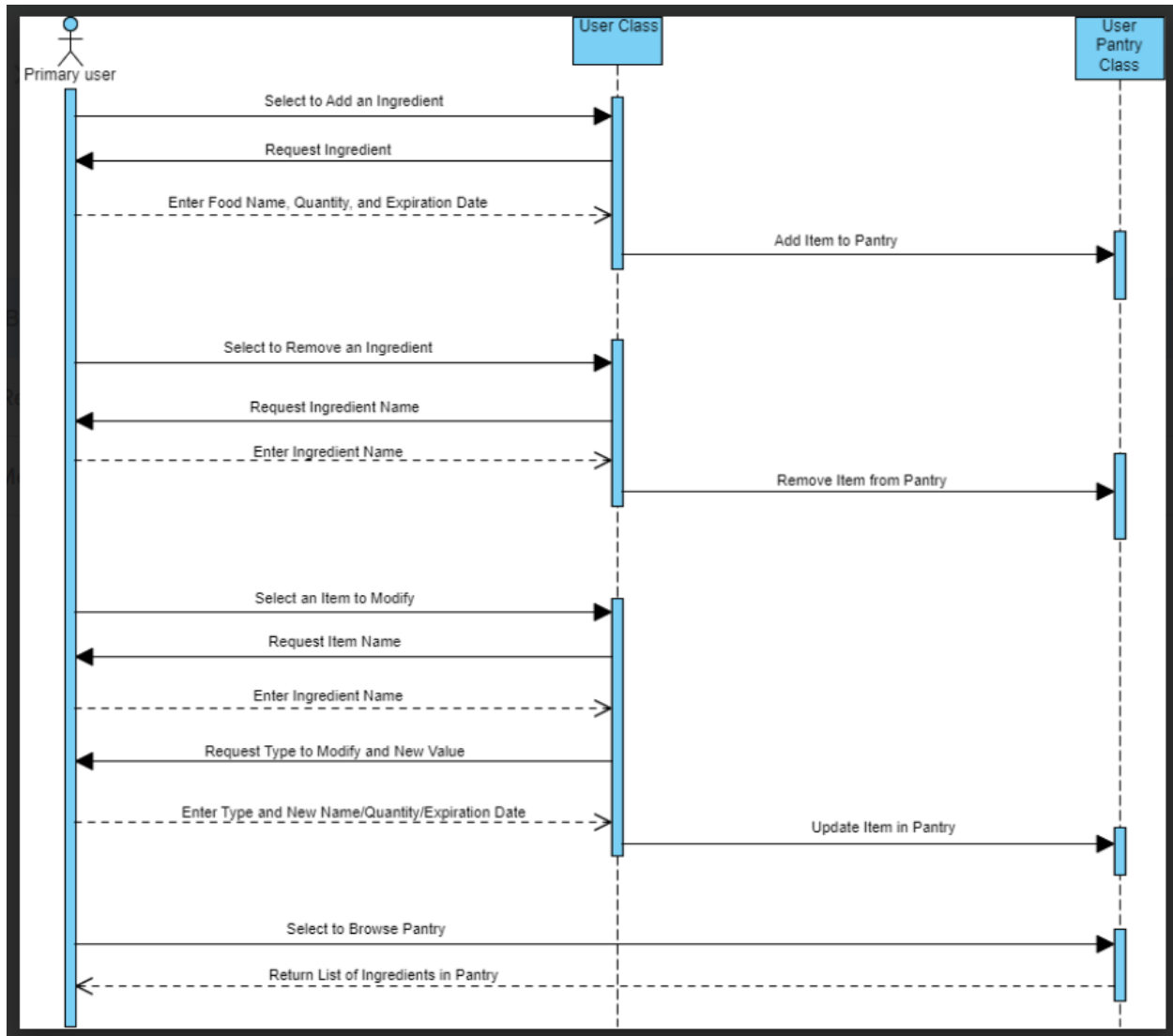


Figure 7: This figure depicts the system sequence diagram for the primary user to add, modify, remove, or browse items in their private user pantry.

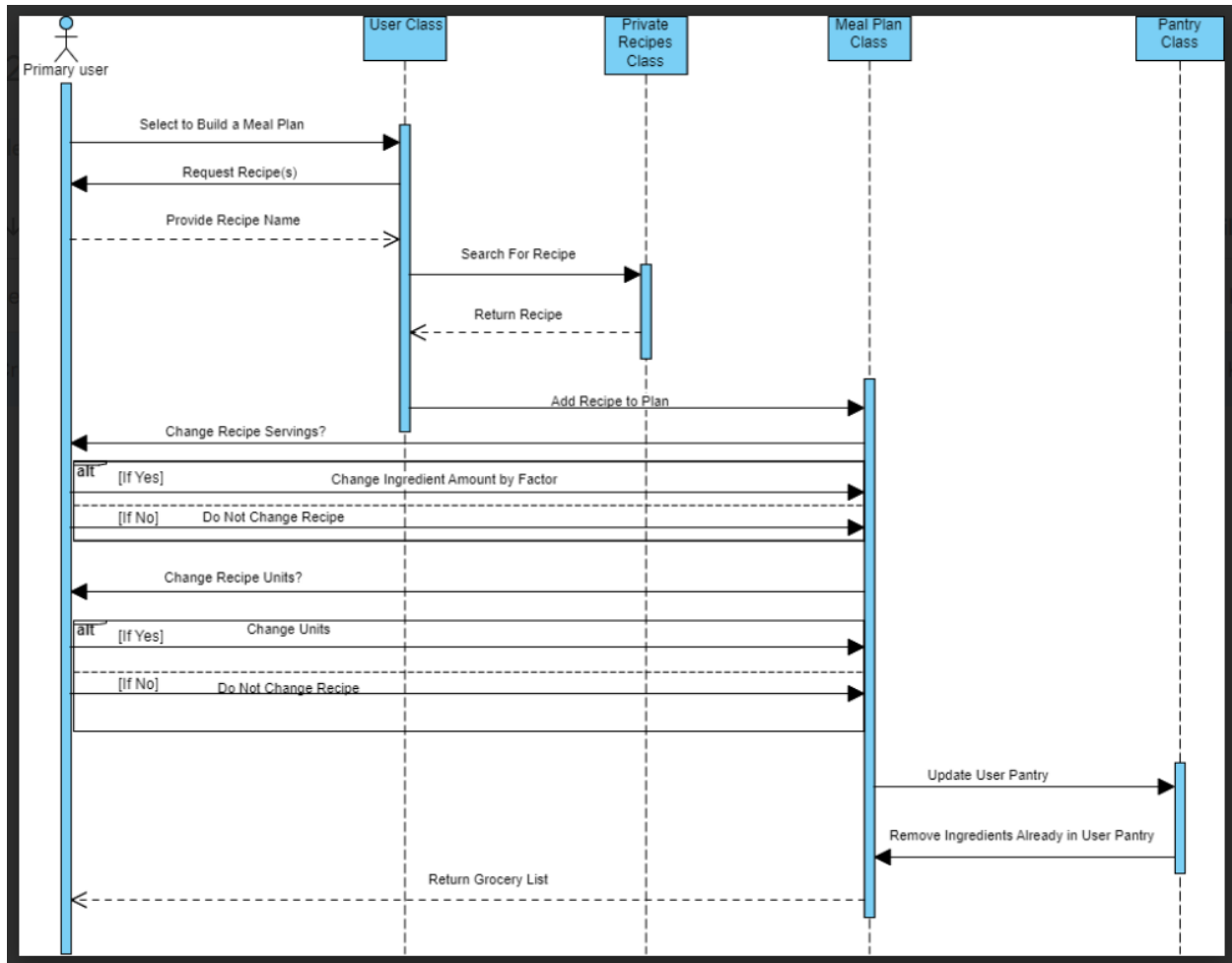


Figure 8: This figure depicts the system sequence diagram for the primary user to create a meal plan and get a shopping list based on the recipes they plan to make and the ingredients they currently have in their pantry.

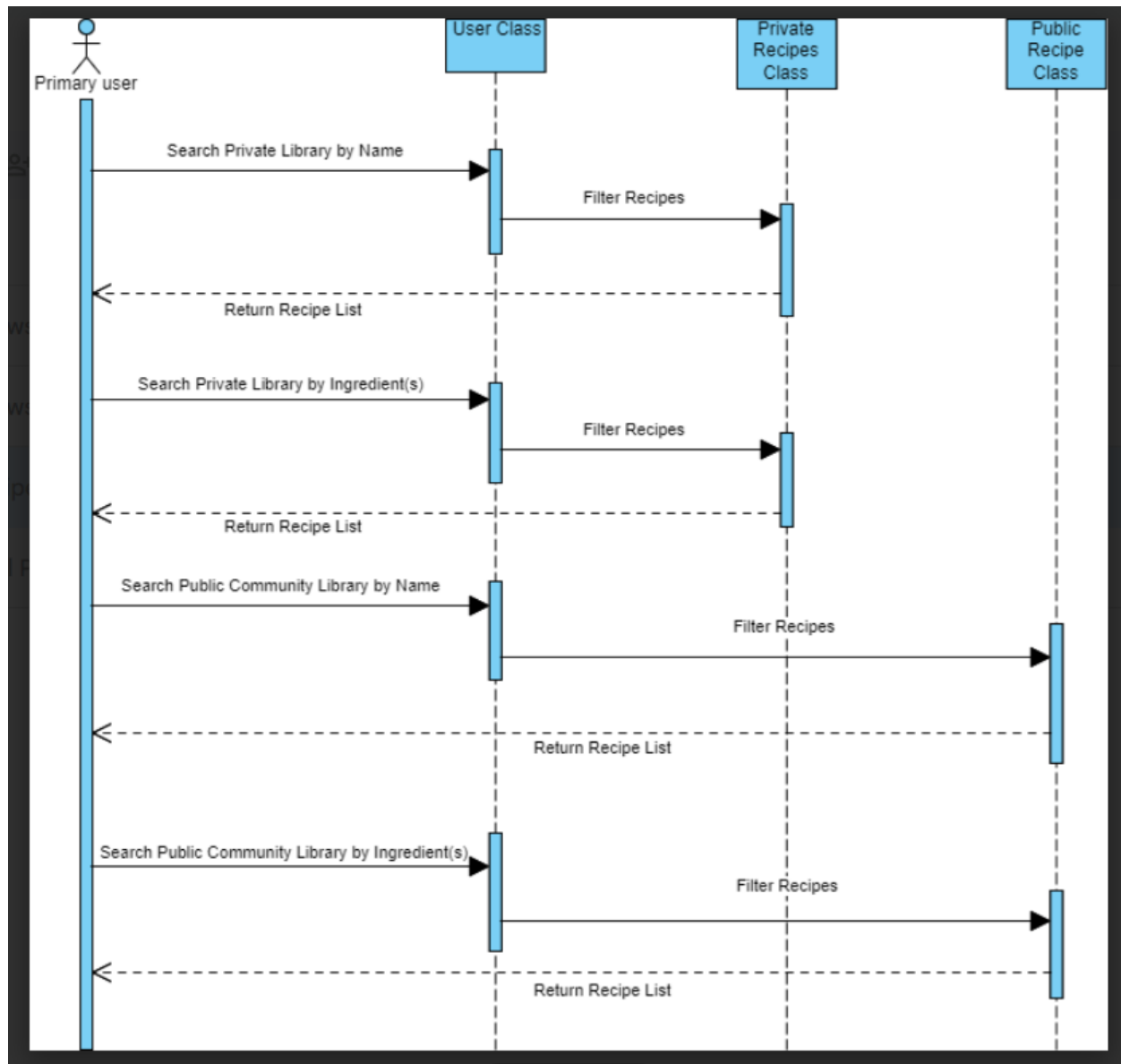


Figure 9: This figure depicts the system sequence diagram for the primary user to search recipes either in their private user library or the public community library by either recipe name or ingredient(s).

Mock Up Prototype

Personal Cookbook Home Page



Figure 10: *Personal Cookbook home screen, with 5 options to the user: access the personal library, the community library, the user's personal pantry, search for recipes, and create a meal plan.*

Personal/Community Library

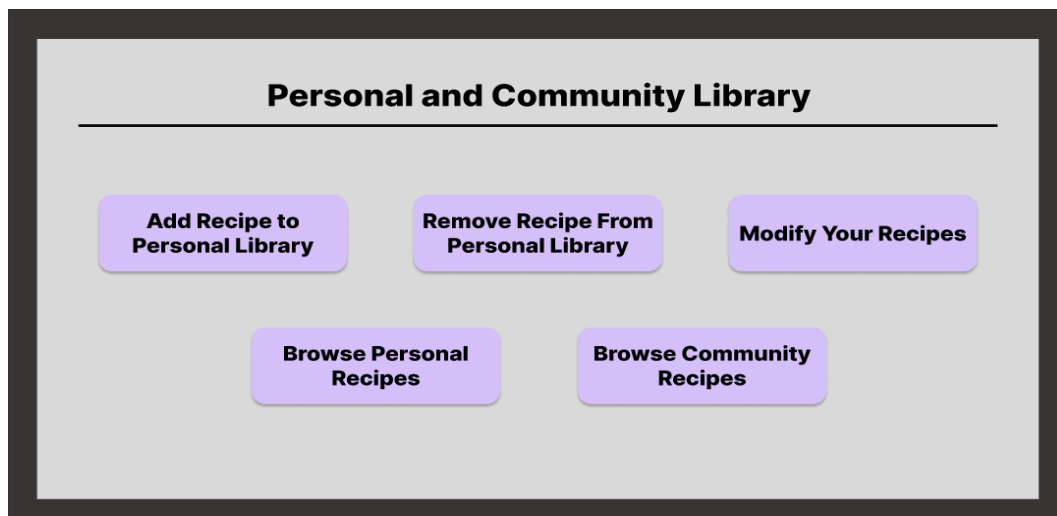
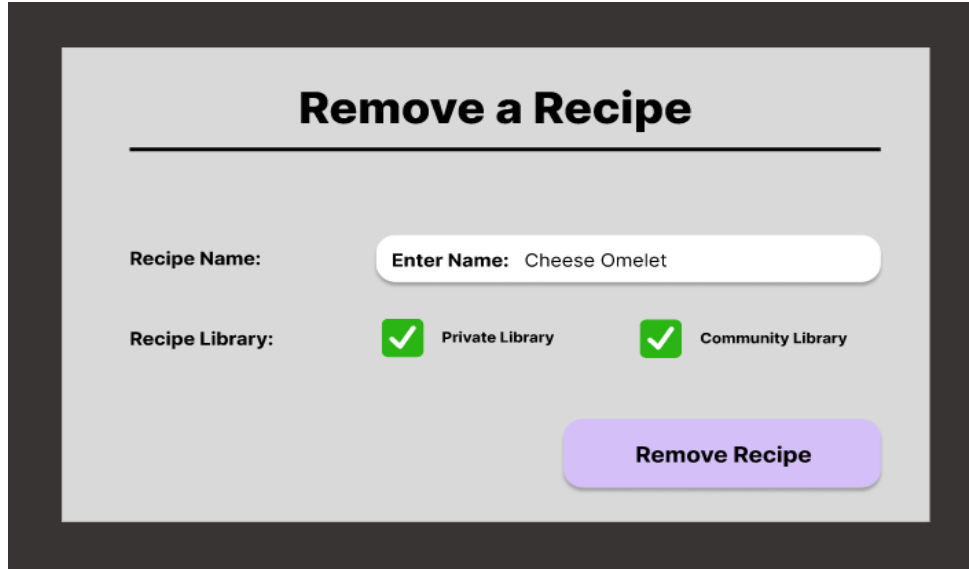


Figure 11: *Personal Cookbook Personal and Community Library, with 5 options to the user: add, remove, or modify a recipe to personal library, and browse personal or community recipes.*

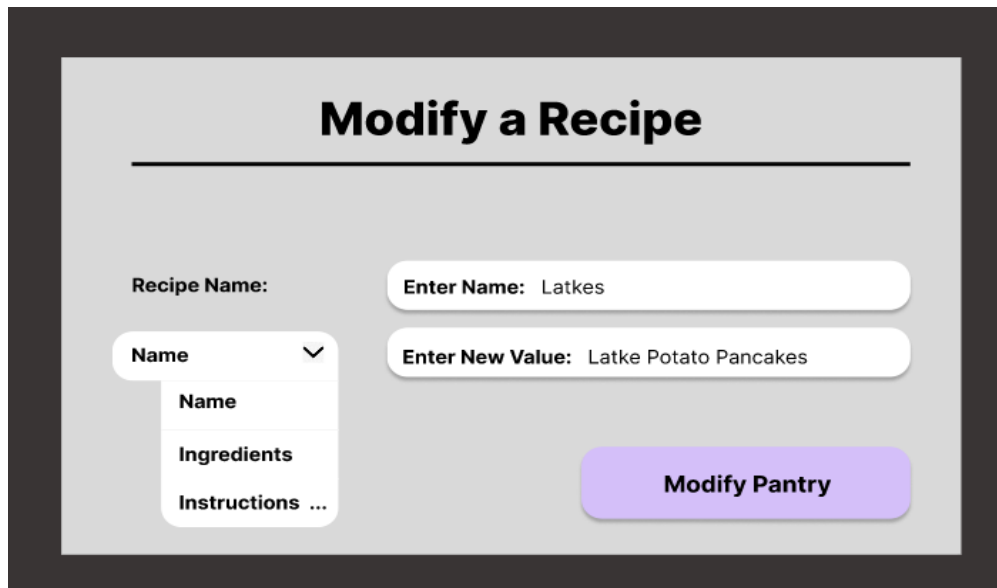
Remove a Personal Recipe



The form is titled "Remove a Recipe" in bold black text, underlined. It contains two input fields. The first is labeled "Recipe Name:" and has a value of "Enter Name: Cheese Omelet". The second is labeled "Recipe Library:" and has two checkboxes, both of which are checked: "Private Library" and "Community Library". A purple button labeled "Remove Recipe" is located at the bottom right of the form.

Figure 12: Remove Recipe prompts the user for the recipe name and where to remove it from (private library and/or the community library).

Modify a Personal Recipe



The form is titled "Modify a Recipe" in bold black text, underlined. It contains two input fields. The first is labeled "Recipe Name:" and has a value of "Enter Name: Latkes". The second is labeled "Enter New Value:" and has a value of "Latke Potato Pancakes". To the left of the second input field is a dropdown menu with the label "Name" and a downward arrow. The dropdown menu is open, showing three options: "Name", "Ingredients", and "Instructions ...". A purple button labeled "Modify Pantry" is located at the bottom right of the form.

Figure 13: Users can freely modify their recipes to change the name, ingredients, or instructions.

Browse Personal Recipes

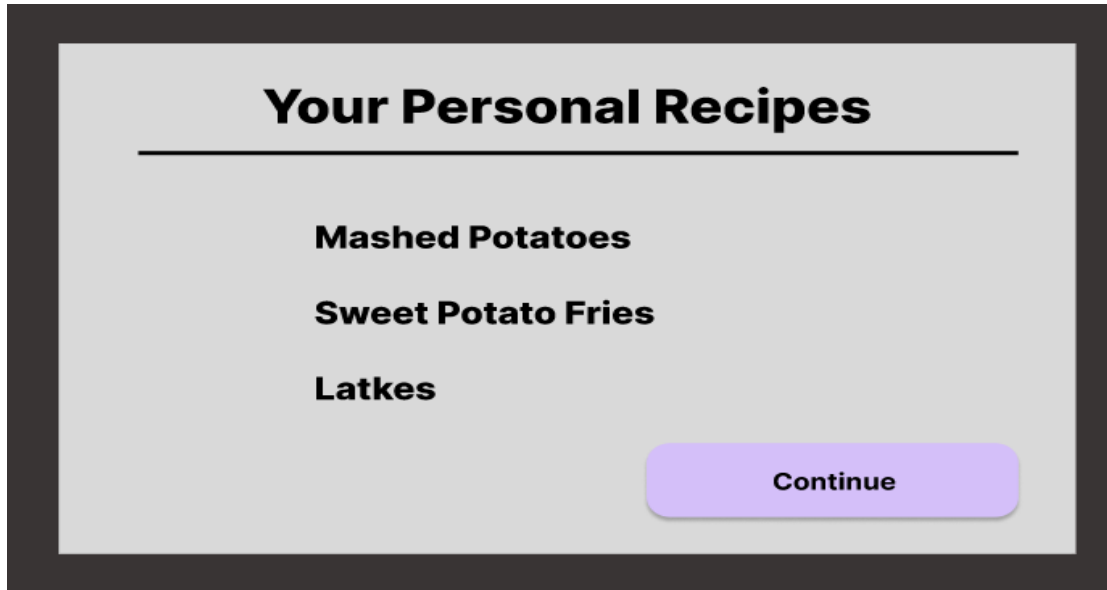


Figure 14: Browsing personal recipes returns the user a list of current recipes in their private library.

Browse Public Recipes

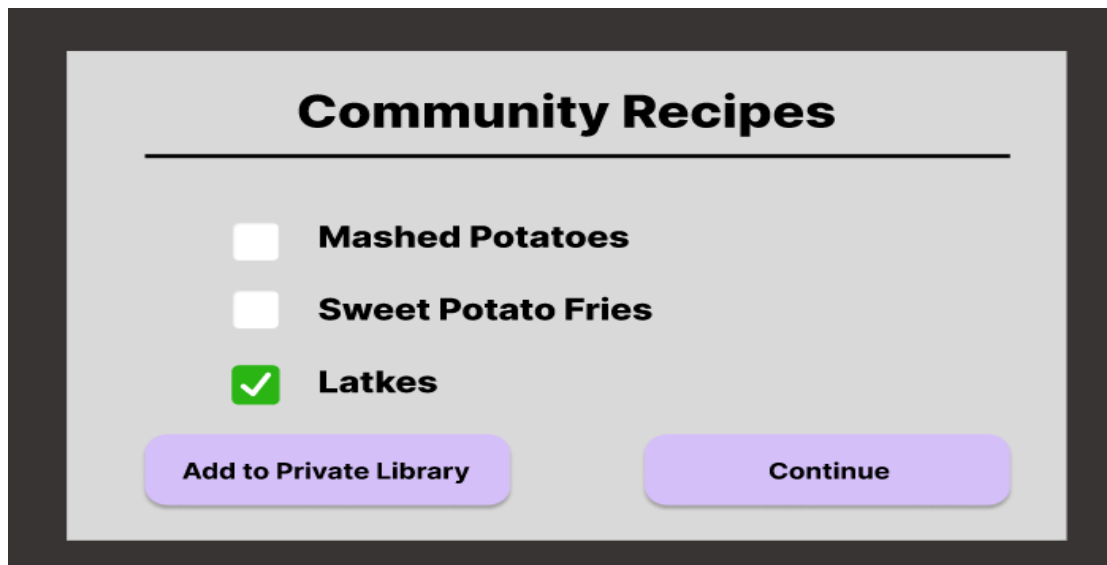
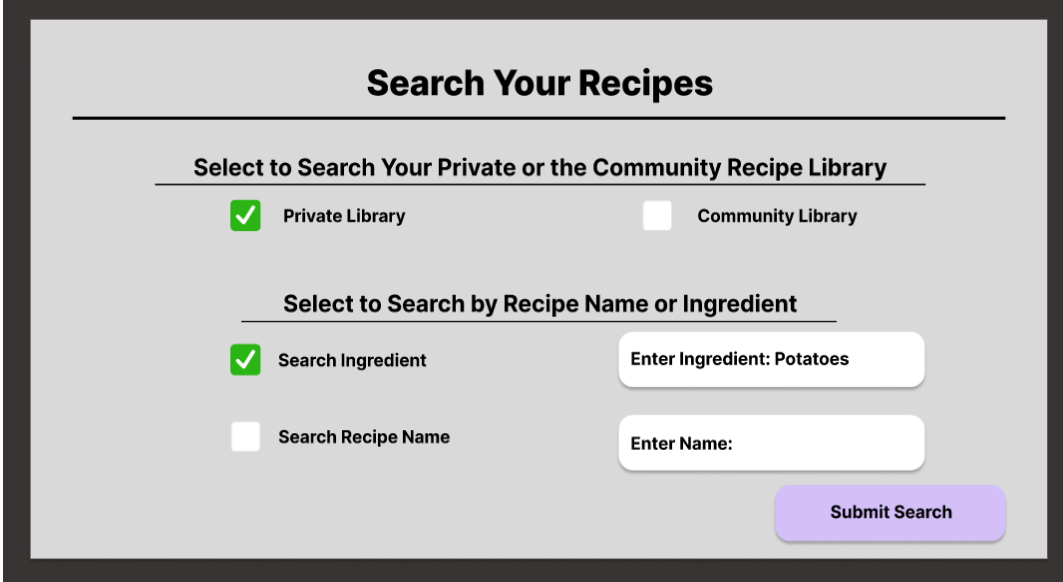


Figure 15: Browsing community recipes gives a list of community recipes and allows the user to check whichever ones they wish to add to their private library.

Search Recipes By Ingredient Or Name



Search Your Recipes

Select to Search Your Private or the Community Recipe Library

☒ Private Library ☐ Community Library

Select to Search by Recipe Name or Ingredient

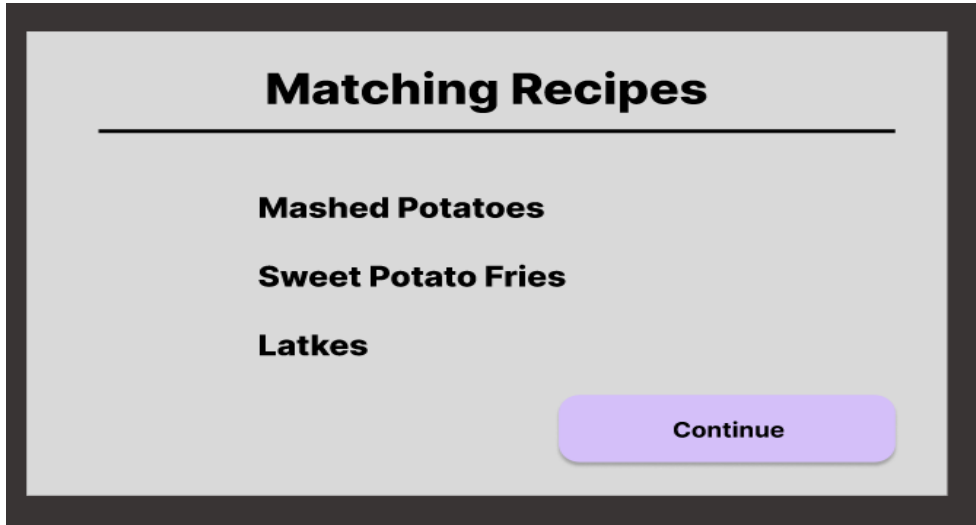
☒ Search Ingredient

☐ Search Recipe Name

Submit Search

Figure 16: Users can search their recipes in both the private and community libraries, either by ingredient or recipe name.

Found Recipes Matching Search



Matching Recipes

Mashed Potatoes

Sweet Potato Fries

Latkes

Continue

Figure 17: After completing a search, Personal Cookbook will return to the user a list of all recipes with matching information (ingredients or name). Seen above, it returns all recipes that use potatoes.

Personal Pantry Options Page

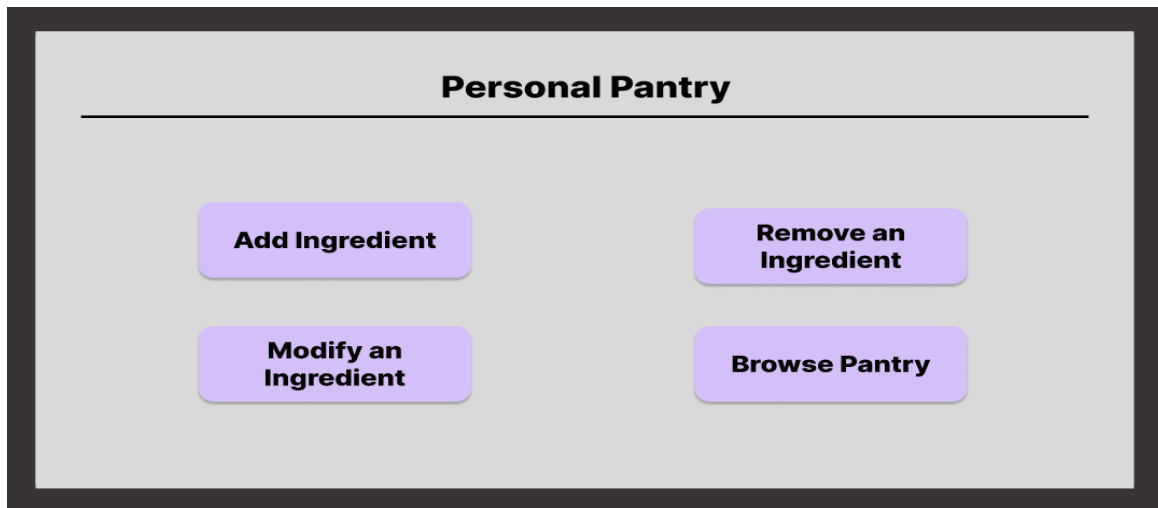


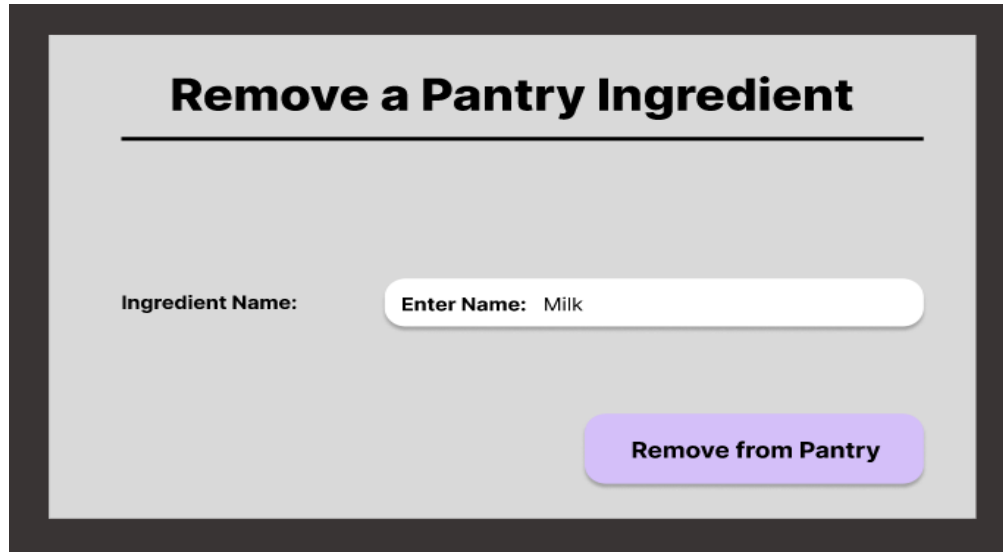
Figure 18: *Personal Cookbook offers each user their own pantry to record their current ingredients they have on hand. Shown above are their options to add, remove, or modify ingredients and to browse their current pantry.*

Add a New Pantry Ingredient

A screenshot of a web form titled "Add a New Pantry Ingredient". Below the title is a horizontal line. The form contains three input fields: "Ingredient Name:" with the value "Milk", "Ingredient Quantity:" with the value "1 Cup", and "Expiration Date:" with a green checkmark icon and the value "12/04/2023". Below these fields is a purple button labeled "Add to Pantry".

Figure 19: *Adding a new ingredient to their pantry prompt the user for the name, quantity, and expiration date so all relevant information is easily accessible when browsing.*

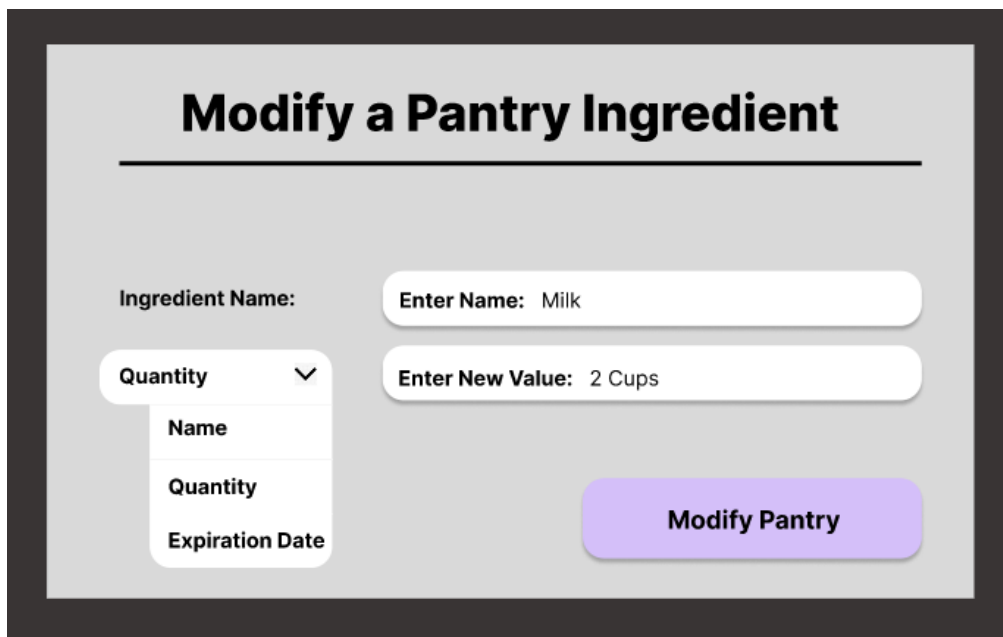
Remove a Pantry Ingredient



The interface for removing a pantry ingredient features a title "Remove a Pantry Ingredient" at the top. Below the title is a label "Ingredient Name:" followed by a text input field containing "Enter Name: Milk". At the bottom right is a purple button labeled "Remove from Pantry".

Figure 20: Removing an ingredient prompts the user for the name and automatically removes the reference to it.

Modify a Pantry Item



The interface for modifying a pantry item features a title "Modify a Pantry Ingredient" at the top. Below the title are two input fields: "Enter Name: Milk" and "Enter New Value: 2 Cups". To the left of these fields is a dropdown menu with "Quantity" selected. A menu is open showing options: "Name", "Quantity", and "Expiration Date". At the bottom right is a purple button labeled "Modify Pantry".

Figure 21: Modifying an ingredient prompts the user to select the name of a current ingredient and change the amount, name, or expiration date.

Browse Pantry Ingredients

Your Pantry Ingredients		
Ingredient	Quantity	Exp. Date
Milk	2 Cups	12/04/2023
Eggs	12 Eggs	11/12/2023
Butter	2 Tbls	
Potatoes	12	
Flour	1 Cup	
Cream Cheese	4 Tbls	11/30/2023
Continue		

Figure 22: Browsing returns to the user a table of all ingredients with their quantities and expiration dates.

Adding a Recipe to the Meal Plan

Create Meal Plan		
Enter One of the Recipes in Your Private Library to Make		
Recipe Name:	Enter Name: Mashed Potatoes	
Desired Serving Size:	Enter Serving Size: 4 Servings	
Desired Units:	<input type="checkbox"/> Metric Units	<input checked="" type="checkbox"/> Imperial Units
Submit Search		

Figure 23: Personal Cookbook offers a meal plan creation screen, having the user select their recipe and serving size (with a check for units). If the recipe is successfully added to the meal plan, it will move to Figure 24.

Success Message When a Recipe is Added to the Meal Plan

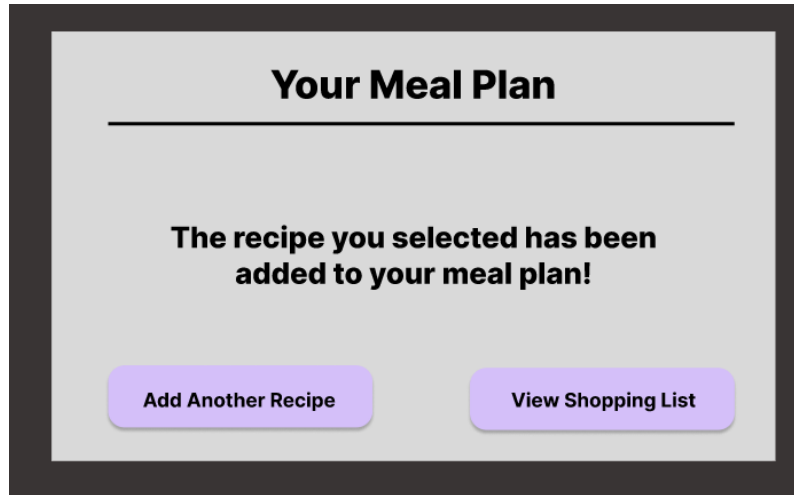


Figure 24: A successful meal plan will be added for the user and prompt to either view the automatically updated shopping cart or to add another recipe.

Browse Shopping List

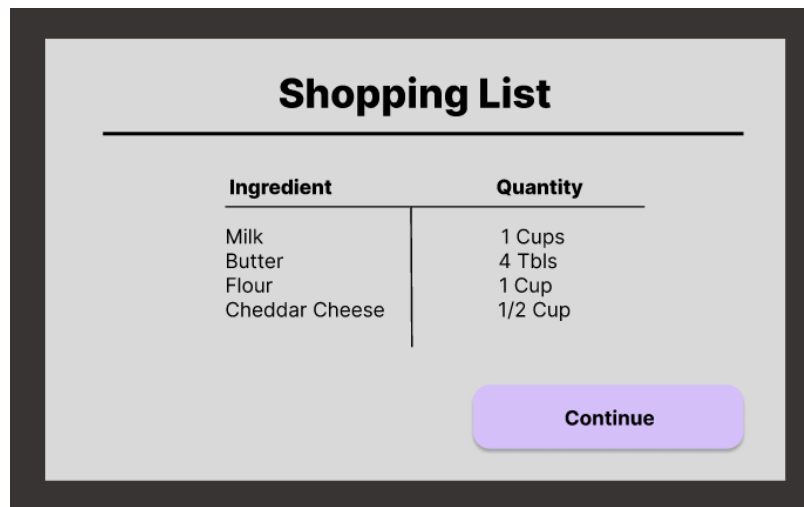


Figure 25: If the user looks into the shopping list of a newly added meal plan item, they will be returned a list of ingredients with their quantities in the desired units from figure 24.

Figma Link: <https://www.figma.com/file/Ad6aHcGQ1KS2m15Vb561rS/CSI5220--Final-Project?type=design&node-id=0%3A1&mode=design&t=47Mat7JnEJFO8pD3-1>

Phase 4

Step 1 : System Design Component Diagram

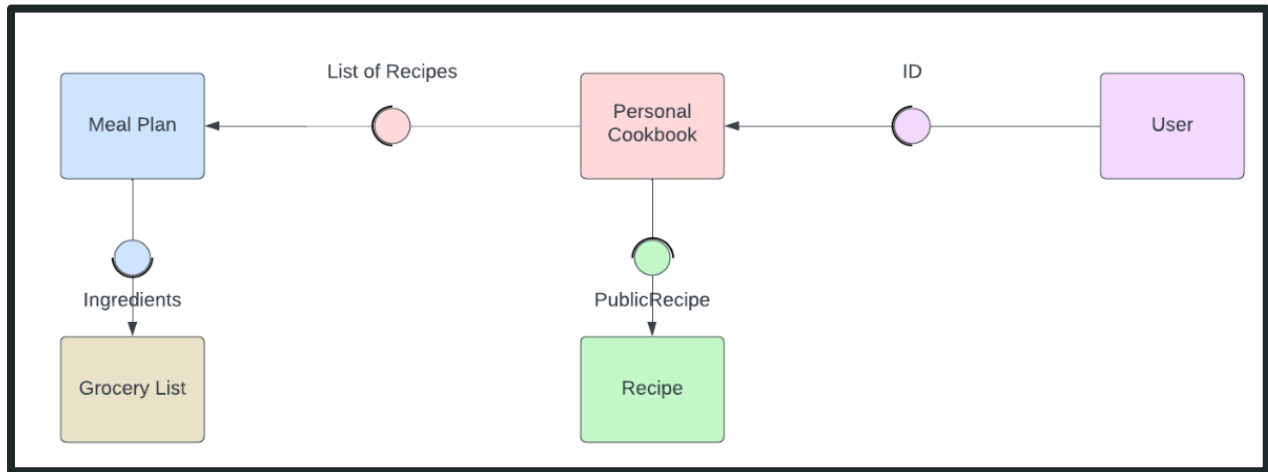


Figure 26: The figure above depicts a component diagram for our Personal Cookbook app. This diagram shows the different components of our system, their dependencies, organization, and the relationships between them.

Deployment Diagram

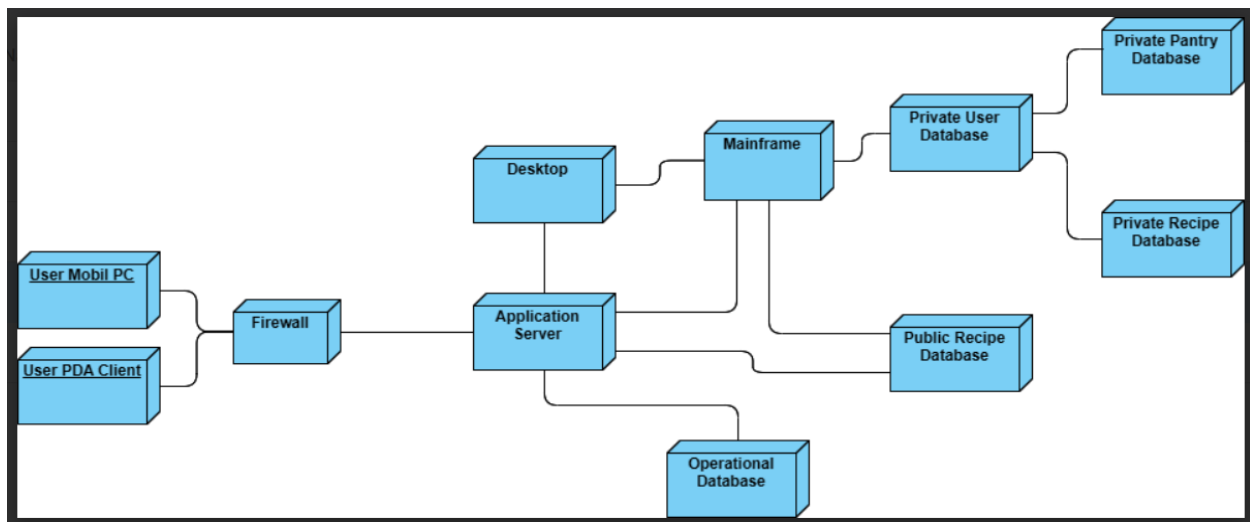


Figure 27: This deployment diagram shows the links between the hardware and software required for our program. One important thing to note from this diagram is how individual user data is separated from public data. Users are only able to access data from their own Private Recipe Database or Pantry, but all users have access to the Public Recipe Database.

Sequence Diagrams

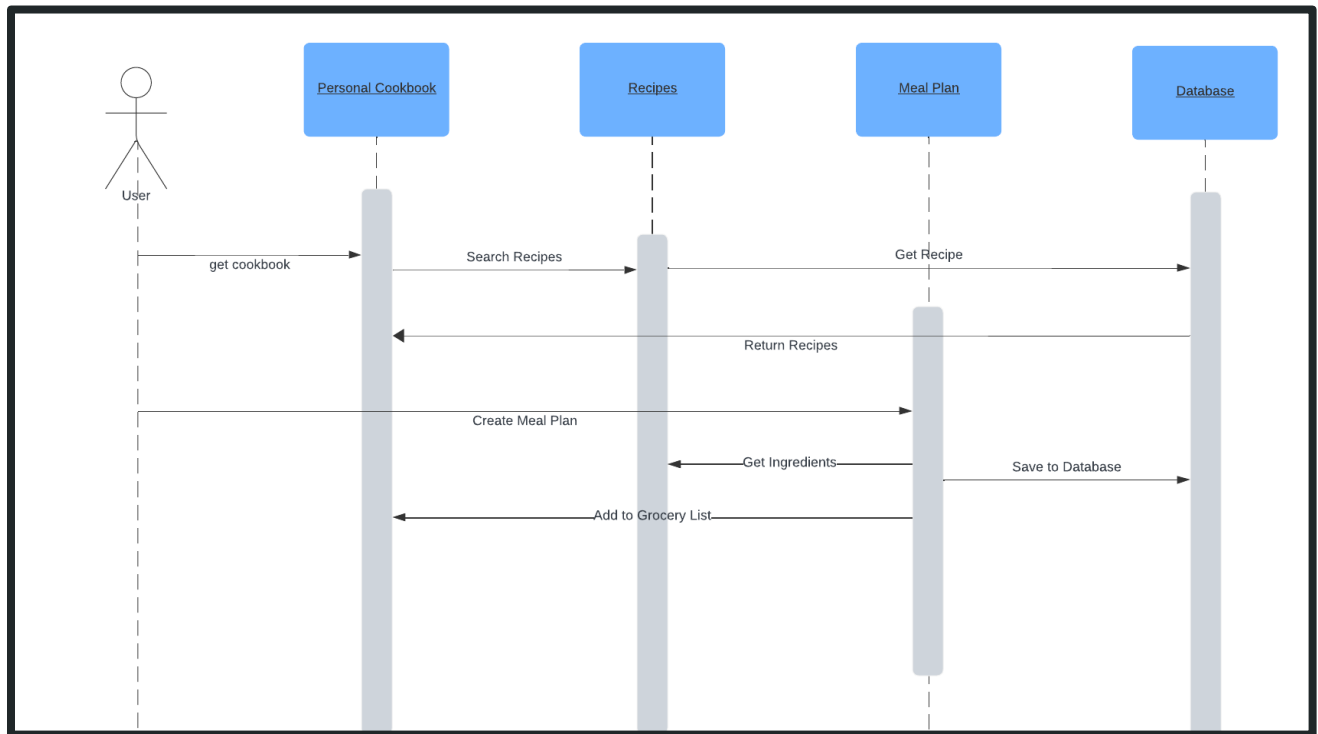


Figure 28: The figure above shows the sequence diagram for a user to search through either their Personal or the Public Recipe Library and begin creating a meal plan. A sequence diagram for each of the major use cases has been created (see appendix) to show the classes and methods required to carry out an event.

Package Diagram

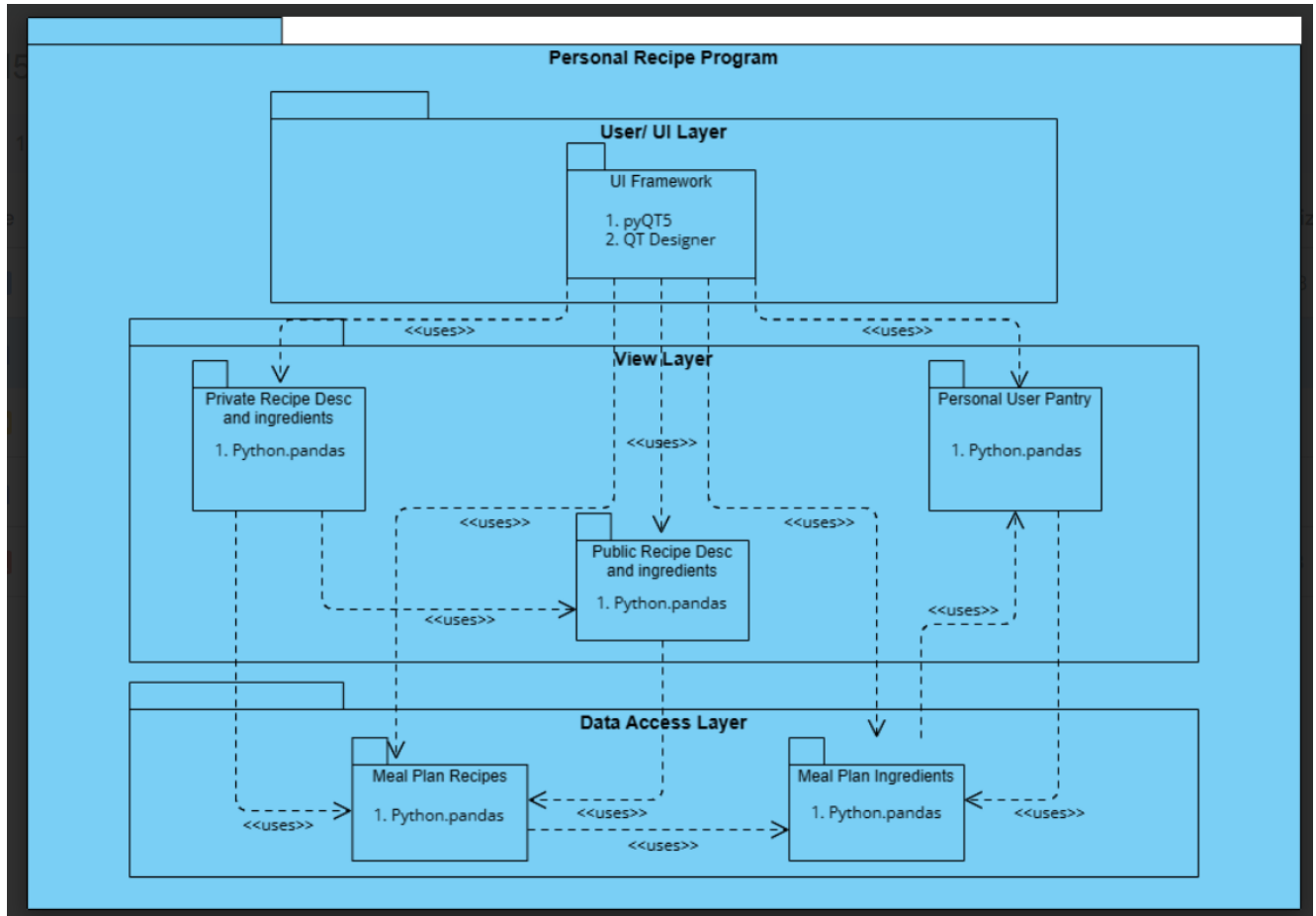


Figure 29: The figure above displays the Package diagram for our program. This diagram is broken down into three different layers: UI layer, view layer, and data access layer. Our app was built using Python3. The main packages required in our UI layer were pyQT5 and QTDesigner. To create the data frames needed within the view layer and data access layer, we used Python.pandas.

Step 2 Coding and Testing

Technical Details and Project Components

A. Database Design and Implementation

The Personal Cookbook app was created using the programming language Python. This app stores data describing recipes (title, instructions, descriptions, serving size) and the ingredients required to make them (ingredient name, quantity, unit of measurement) using pandas tables. Additionally, the Personal Cookbook app also stores data on a user's pantry items (ingredient name, quantity, unit of measurement). To build the database and connect these tables, a 1-many relationship between the recipe tables and ingredient tables was used. In the recipe tables (1 class), the recipe title was used as the primary key. In the ingredients tables (many class), the name of each recipe an ingredient belonged to was used as the foreign key.

The pantry table can be related to the meal plan recipe ingredients table through a 1-many relationship. This allows meal plan ingredient quantities to be compared to pantry item quantities where the ingredients and pantry item names match. For the pantry table, the item name is the primary key. For the meal plan ingredient table, the ingredient name is the foreign key.

To implement the user interface, QTdesigner was used. We chose to use this software because it allowed us to easily develop the layout of each UI screen. For any recipe library, ingredient, or pantry table the user can select to add, remove, modify or browse the table through the UI. This was done to help users easily understand what can and cannot be accomplished and provide symmetry throughout the app.

B. Algorithms and Libraries Used

Pandas was used to simplify data storage and manipulation. This library is built off of the NumPy library and allows us to build tables, access, and modify data. As the number of users for this app grows, we will also be able to use spreadsheets or SQL tables to populate the existing pandas tables. The versatility of this library will help to streamline future updates to the app.

The algorithms used in the Personal Cookbook app are those contained within the pandas and NumPy library. Specifically, the search algorithms within these libraries were used to efficiently locate any entries within a table matching a set of given conditions. These search algorithms were used in each update and remove function.

C. Source Code Snippets

```
import pandas as pd

class AllIng:
    def __init__(self, recipes=[], items=[], quantity=[], units=[], unitType=[]):
        self.RecipeName = recipes
        self.ItemName = items
        self.ItemQty = quantity
        self.Units = units
        self.UnitType = unitType

# creating a pandas table
allIngPrivate = AllIng()
allIngPrivate = pd.DataFrame(allIngPrivate.__dict__)
```

Code Snippet 1: The code above contains an example of how we created one of the pandas tables required for our app. In this example the class *AllIng* was used to specify the column names needed and the function *pd.DataFrame()* was used to convert the dictionary of column names into the necessary table. The particular class was used to create the meal plan, private recipe, and public recipe ingredient tables.

```
def addPrivateRecipe(title, instructions, servings, desc, ispublic):
    # checking if a recipe with that name already exists
    if (privateRecipes["RecipeName"] == title).any():
        print("This recipe is already in your library. Please choose another name.")
    else:
        # appending a new recipe to the table
        privateRecipes.loc[-1] = [title, instructions, servings, desc, ispublic]
        privateRecipes.index = privateRecipes.index + 1
        if ispublic == True:
            publicRecipes.loc[-1] = [title, instructions, servings, desc]
            publicRecipes.index = publicRecipes.index + 1
```

Code Snippet 2: The function above was used in our app to add new recipes to the user's private recipe library. This function takes all of the parameters necessary to populate one row of the private recipe table and will automatically add the recipe to the community library if the user selects to make their recipe public. To prevent duplicate recipes, this function only accepts unique recipe names.

```

def removePublicRecipe(recipeName):
    # checking to see if the recipe given is publicly available
    if (
        privateRecipes[
            (privateRecipes["RecipeName"] == recipeName) & (privateRecipes["isPublic"] == True)]).empty == False:
        # removing the recipe and updating table index
        privateRecipes.loc[privateRecipes["RecipeName"] == recipeName, "isPublic"] = False
        publicRecipes.drop(publicRecipes[publicRecipes["RecipeName"] == recipeName].index, inplace=True)

Denise Rauschendorfer *
def removePrivateRecipe(recipeName):
    # removing the recipe and updating table index
    removePublicRecipe(recipeName)
    privateRecipes.drop(privateRecipes[privateRecipes["RecipeName"] == recipeName].index, inplace=True)

```

Code Snippet 3: The functions shown above allow a user to remove either one of their personal recipes or a recipe that they previously made publicly available. The first function, removing a user's public recipe, checks to see if the recipe name provided is publicly available. When true, the `isPublic` status is changed to false in the user's private recipe table, the recipe is removed from the public database, and the table index is updated. Similarly, the second function, removing a private user recipe, uses the first function to remove the specified recipe from both libraries.

```

Denise Rauschendorfer +1
def searchByIngName(table, itemName):
    foundRecipes = table[table["ItemName"] == itemName]
    return foundRecipes

```

Code Snippet 4: When called, this function returns a table containing all of the recipes that use a given ingredient from a specified table.

```

def viewAll(table):
    table = table.sort_index()
    return table

```

Code Snippet 5: The function above is used to sort and view all of the entries in a given table. This function was used to display tables in the browse table UIs.

```

def updateRecipe(type, recipeName, newValue):
    # finding entry to update
    update = privateRecipes["RecipeName"] == recipeName
    privateRecipes.loc[update, type] = newValue
    # updating value if in both the public and private library when applicable
    if (
        privateRecipes[
            (privateRecipes["RecipeName"] == recipeName) & (privateRecipes["isPublic"] == True)]
        .empty == False:
        publicRecipes.loc[update, type] = newValue
    if (type == "isPublic") and (newValue == True):
        df = privateRecipes[privateRecipes["RecipeName"] == recipeName]
        publicRecipes.loc[-1] = [df.iat[0, 0], df.iat[0, 1], df.iat[0, 2], df.iat[0, 3]]
        publicRecipes.index = publicRecipes.index + 1

```

Code Snippet 6: The function above shows how a recipe value can be modified. This function uses the provided recipe name to determine what table row to modify and will update both private and public recipes when applicable.


```

def changePantryUnitType(itemName):
    # reset index and find all matching pantry items
    pantry.reset_index(drop=True, inplace=True)
    currentType = pantry.loc[pantry["ItemName"] == itemName, "UnitType"].values
    # change imperial units to metric
    if currentType == "Imperial":
        update = pantry["ItemName"] == itemName
        pantry.loc[update, "UnitType"] = "Metric"
        x = pantry.loc[update, "Units"].values.tolist()
        pantry.loc[update, "ItemQty"] = pantry.loc[update, "ItemQty"] * unitValsDict[x[0]]
        pantry.loc[update, "Units"] = unitNameDict[x[0]]
    # change metric units to imperial
    elif currentType == "Metric":
        update = pantry["ItemName"] == itemName
        pantry.loc[update, "UnitType"] = "Imperial"
        x = pantry.loc[update, "Units"].values.tolist()
        pantry.loc[update, "ItemQty"] = pantry.loc[update, "ItemQty"] * unitValsDict[x[0]]
        pantry.loc[update, "Units"] = unitNameDict[x[0]]
        x = pantry.loc[update, "Units"].values.tolist()
        y = pantry.loc[update, "ItemQty"].values.tolist()
        # simplifying units from tsp to Tbsp and oz to cups when applicable
        if (x[0] == "tsp") and (y[0] % 3 == 0):
            y[0] /= 3
            pantry.loc[update, "ItemQty"] = y[0]
            pantry.loc[update, "Units"] = "Tbsp"
        if (x[0] == "oz (liquid)") and (y[0] % 8 == 0):
            y[0] /= 8
            pantry.loc[update, "ItemQty"] = y[0]
            pantry.loc[update, "Units"] = "cups (liquid)"

```

Code Snippet 7: The function above can be called to change the unit type of a pantry item. This function updates the quantity and listed units of measurement for an item.

```

class Ui_MainWindow(object):
     Denise Rauschendorfer *
    def setupUiHomePage(self, MainWindow):
        # specifying UI screen size
        MainWindow.resize(800, 370)
        # personalizing UI screen
        MainWindow.setStyleSheet("background-color: #123456;")
        MainWindow.setWindowIcon(QtGui.QIcon('chef.png'))
        # creating UI screen
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        # adding a frame to the UI screen
        self.frame = QtWidgets.QFrame(self.centralwidget)
        self.frame.setGeometry(QtCore.QRect(20, 20, 751, 331))
        self.frame.setStyleSheet("background-color: white;")
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)

```

```

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUiHomePage(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

```

Code Snippet 8: The class above creates the UI home page for the Personal Cookbook app. In this class, the screen size, color, icon, are specified. This class also adds a frame layout. The second block of code was used to create an instance and execute the UI screen.

D. Software Interfaces

Home Page



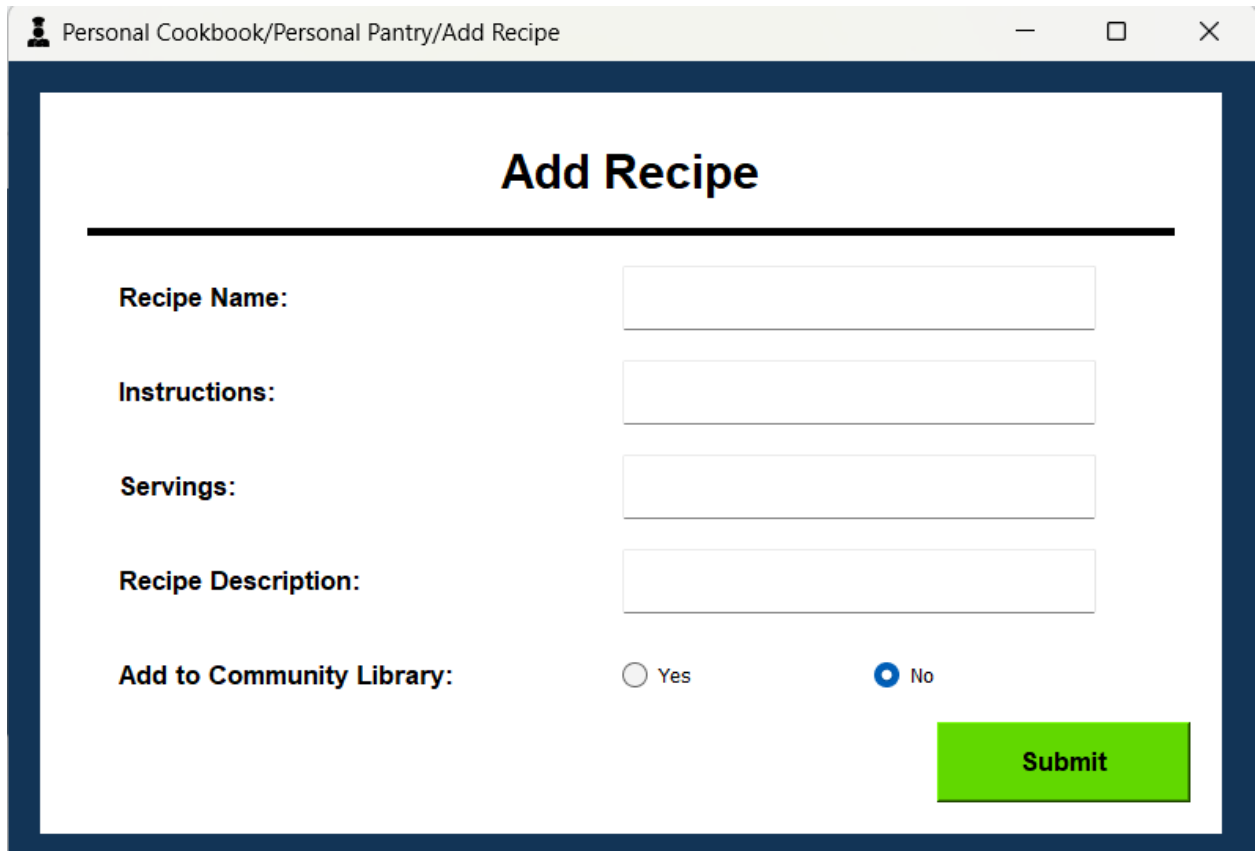
UI 1: This UI is the first screen the user sees when they launch the Personal Cookbook app,

Personal and Community Recipe Library Functionality



UI 2: This UI summarizes the operations a user can select to perform on either the community or their private recipe library. A similar screen is used to display the operations available for pantry items.

Add Recipe to Private or Community Library



The screenshot shows a web browser window with the title bar "Personal Cookbook/Personal Pantry/Add Recipe". The browser window contains a form titled "Add Recipe" with a dark blue border. The form has a white background and a horizontal line separating the title from the input fields. There are four text input fields labeled "Recipe Name:", "Instructions:", "Servings:", and "Recipe Description:". Below these fields is a radio button group for "Add to Community Library:" with options "Yes" and "No". The "No" option is selected. A green "Submit" button is located at the bottom right of the form.

Personal Cookbook/Personal Pantry/Add Recipe

Add Recipe

Recipe Name:

Instructions:

Servings:

Recipe Description:

Add to Community Library: ☐ Yes ☒ No

Submit

UI 3: The UI above allows a user to enter information about a new recipe and specify whether they would like to make the recipe publicly available or not. When this action is successfully completed, the user will be notified. A similar UI is available for the user to add a recipe ingredient or pantry item.

Remove a Recipe From Either the Public or Private Library

The screenshot shows a web browser window with the title 'Personal Cookbook/Recipe Library/ Remove'. The main content area has a dark blue border and a white background. At the top, the heading 'Remove a Recipe' is centered in a bold, black font, followed by a horizontal line. Below this, there are two labels: 'Recipe Name:' and 'Recipe Library:'. The 'Recipe Name:' label is followed by a white text input field with a blue border. The 'Recipe Library:' label is followed by two radio button options: 'Private Library' (which is selected, indicated by a blue dot) and 'Community Library' (which is unselected, indicated by a grey dot). To the right of these options is a green rectangular button with the word 'Submit' in white text.

***UI 4:** This UI is used to remove a recipe from the specified library. If the user selects to remove a recipe from their private library it will also be removed from the community recipe library. When this action is successfully completed, the user will be notified (UI 5). A similar UI is also available for the user to remove a recipe ingredient or pantry item.*

Example Notification After Removing a Recipe



***UI 5:** The message above shows an example of how the user will be notified when they successfully remove a recipe. For this particular example the recipe "Grilled cheese" was removed from both the private and public community library. A similar type of notification will appear anytime a user attempts to perform an action.*

Modify a Personal User Recipe

Personal Cookbook/Recipe Library/ Modify

Modify a Recipe

Recipe Name:

Recipe Name ▼
 Recipe Name
 Instructions
 Servings
 Description

Submit

UI 6: Users can access this UI when they want to change the recipe name, instruction, servings, or description of a recipe in their private library. Any changes made to a private recipe will also be reflected in the public community library when applicable. A similar UI is also displayed when a user wants to modify a recipe ingredient or pantry item. When this action is successfully completed, the user will be notified.

Browsing All Private Recipe Ingredients

Personal Cookbook/Recipe Library/ Ingredients/ Browse

Browse Your Recipe Ingredients

	Recipe Name	Ingredient	Quantity	Units	Unit Type
1	Grilled cheese	bread	2	Slice	Other
2	Grilled cheese	butter	1	Tbsp	Imperial
3	Grilled cheese	tomato	1	Each	Other
4	omelette	eggs	2	Each	Other

UI 7: The UI above shows how the private recipe ingredients table will be displayed in the app. A similar UI will appear anytime the user selects to browse a table or view searched items.

Search Recipes by Name or Ingredient

Personal Cookbook/ Search Recipes

Search Recipes

Search Your Private or the Community Recipe

☒ Private Library ☐ Community Library

Search by Recipe Name or Ingredient

Recipe Name

Recipe Name
Ingredient Name

Submit

***UI 8:** Users can select to search for a recipe by name or ingredient in either their private library or the public community library through this UI. Once the user submits a search, a UI similar to the one depicted above (UI 7) will appear.*

Testing and Evaluating Code

A. Language Justification

We developed the entirety of PersonalCookbook in Python for 4 central reasons:

1. Mutual familiarity with the language meant minimal learning required on either party's end that would have taken time away from development of the system itself.
2. Python is a language that focuses on readability, meaning understanding the code added to the system would require significantly less effort and time than most other languages where, unless the developer focuses heavily on readability, there will be more struggle to quickly catch up to any new developments the other party creates.
3. Python's verbose functions and tools make rapid prototyping and testing far easier to implement, especially early on. This means it is easy to develop the base of the system and test viability without needing to commit to a lengthy process.
4. Python has many incredibly useful tools to take advantage of, namely the Pandas and PyQt5 libraries. These allowed for easy data management and UI design

respectively and with prior experience using PyQt5 it also negated the learning time for the library.

B. Key Test Cases

To minimize user error, we included 7 major test cases throughout our app. These include:

1. Preventing duplicate recipes from being entered into a given recipe library.
2. Preventing duplicate ingredients or pantry items are entered into the corresponding table.
3. Preventing non-numeric values from being entered for any quantity.
4. Ensuring all ingredient and pantry item quantities are updated when a user changes the units of measurement used.
5. Ensuring only unit type matches the units of measurement provided for each ingredient and pantry item.
6. Ensuring all recipe ingredients are updated when a recipe is updated.
7. Ensuring all public and private tables are updated when a user adds, removes, or modifies one of their personal recipes that they have selected to make publicly available.

Summary of Test Cases per UI

1. Add Pantry Item UI
 - Verify pantry item is not already in the user's pantry table
 - Verify the item quantity entered is numeric
 - Verify the units selected match the unit type selected (imperial, metric, or other)
2. Add Recipe UI
 - Verify the recipe name entered is unique
 - Verify the serving quantity entered is numeric
 - Ensure that all necessary public and private tables are updated
3. Add Recipe Ingredient UI
 - Verify the recipe entered exists
 - Verify the ingredient entered is not already listed for the given recipe
 - Verify the quantity entered is numeric
 - Ensure ingredient quantities are updated when the unit of measurement is changed
 - Ensure that all necessary public and private tables are updated
4. Modify Pantry Item UI
 - Verify the pantry item entered exists
 - Verify that item quantity is numeric (when selected)
5. Modify Recipes UI
 - Verify the recipe entered exists

- Verify the new serving quantity entered is numeric (when selected)
 - Verify all listed ingredients for a recipe are updated when the recipe name changes
 - Ensure that all necessary public and private tables are updated
6. Modify Recipe Ingredients UI
 - Verify the recipe entered exists
 - Verify the ingredient entered exists for the given recipe
 - Verify the new item quantity is numeric (when selected)
 - Ensure that all necessary public and private tables are updated
 7. Remove Pantry Items UI
 - Verify the pantry item entered exists
 8. Remove Recipes UI
 - Verify the recipe entered exists
 - Ensure that all necessary public and private tables are updated
 9. Remove Recipe Ingredients UI
 - Verify the recipe entered exists
 - Verify the ingredient entered exists for the given recipe
 - Ensure that all necessary public and private tables are updated

C. Evaluating Code

When evaluating our code, we focused on 3 main aspects:

Readability

As this was a team project with intent to be shown to others, readability was necessary for development and future showcasing. Our code needed to be legible to each other and for functions and classes to be easily understood from names and parameters alone. This also significantly decreased the time spent meeting to explain our code to each other, allowing us to instead focus on asking clarifying questions rather than an entire rundown of the new implementation.

Functionality

We needed to ensure the database and system were running as expected, such as by printing out the current pandas information to ensure the behavior matches our expectations. This expanded into ensuring the system not only had expected and replicable results, but also that those results were achievable in a timely manner. Pandas helped significantly with this portion, as it made data management easier and allowed us to quickly add, remove, and change any section we need.

This was also ensured through the previously listed test cases to prevent user error: this kept results in an expected range and prevented edge cases with the addition of duplicates or invalid variable types being inputted into the system.

Modularity

We developed this system with an object oriented approach in mind, so a necessary evaluation point was how modular the system was and if we desired the expand or change aspect of the application, could it be scaled up appropriately? We made sure our pandas system was relying on major classes, so any changes done to those modules would be reflected in the system as a whole.

D. Evaluating User Interface

To evaluate the UI of our system, we focused on a heuristic evaluation with 10 main points

1. Visibility of system status
 - a. When the user implements changes to their Personal CookBook, we ensure they are informed about the success or failure of those actions, as seen in UI 12 below where the removal of a recipe ingredient is returned to the user in a pop-up and the failure to do so because the ingredient is not present is likewise given to the user in UI 11
2. Match between system and the real world
 - a. Because we took a minimalist approach to our app design, we attempted to avoid too many icons but did implement one that is an easy match to the real world: the application icon. The chef icon makes it clear to all users the intent behind the app so when searching they know what our icon represents.
3. User control and freedom
 - a. This was especially important when confirming the navigation of the app and the individual actions the user can take for recipe creation, removal, and changes. We ensured the user didn't have to go through list after list to reach their desired section of the application, with all main functions easily located on the main menu and then sub actions through the following sub-menu. For user control, we also added freedom of unit types, the ability to remove a recipe from both the public and private list at once, and an option to easily select what the user wishes to change about a private recipe.
4. Consistency and standards
 - a. We kept to a consistent and minimalist color scheme, with buttons using cool colors, such as blue and green and keeping the submit button in the bottom left.
5. Error prevention
 - a. The entire list in "Summary of Test Cases per UI", is dedicated to preventing user errors with the system.
6. Recognition rather than recall
 - a. We do not expect users to recall everything about their recipes, offering basic information when they search for a recipe or look through their current list, as seen in UI 7.

7. Flexibility and efficiency of use
 - a. Similarly to user control and freedom, the user has many easily accessible actions available to them right from the menu and have minimal steps to go from any one part of the app to another.
8. Aesthetic and minimalist design
 - a. The application was developed with a neutral background and cool color scheme alongside a minimalist design to help prevent overloading the user with information. Each section of the application has notable white space and the color scheme helps guide the user to submission of actions.
9. Help users recognize, diagnose, and recover from errors
 - a. As mentioned above in the visibility of the system status, pop-ups will appear whenever a user's action fails. This can be from searching for something invalid or inputting a non-numeric value into units.
10. Help and documentation
 - a. We aimed to avoid this section of heuristic evaluation and ensure the application is as self-efficient and explanatory as possible to even new users, though the github is both readily available and all user inputs are carefully labeled to minimize confusion.

Structure and Modules

We employed the use of pandas using classes as our base to create an easily accessible and reliable database. These classes include: 1) ItemList which was used for our pantry and grocery items; 2) Recipes which were employed for public recipes and meal plans; 3) PrivateRecipes which was only used for our private recipe dataset; 4) and finally AllIng which was used for allIngPrivate, allIngPublic, and allIngMealPlan.

ItemList Class and Implementation

```
class ItemList:
    def __init__(self, items=[], quantity=[], units=[], unitType=[]):
        self.ItemName = items
        self.ItemQty = quantity
        self.Units = units
        self.UnitType = unitType

pantry = ItemList()
# pantry = {"ItemName": [], "ItemQty": [], "Units": [], "UnitType": []}
# pantry = pd.DataFrame(pantry)
pantry = pd.DataFrame(pantry.__dict__)
# groceries = {"ItemName": [], "ItemQty": [], "Units": [], "UnitType": []}
groceries = ItemList()
groceries = pd.DataFrame(groceries.__dict__)
```

Code snippet 9: The class above depicts how the pantry and groceries tables were created in our app.

Recipes Class and Implementation

```
class Recipes:
    def __init__(self, names=[], inst=[], serving=[], desc=[]):
        self.RecipeName = names
        self.Instructions = inst
        self.Servings = serving
        self.Description = desc
publicRecipes = Recipes()
publicRecipes = pd.DataFrame(publicRecipes.__dict__)
# mealPlan = {"RecipeName": [], "Instructions": [], "Servings": [], "Description": []}
mealPlan = Recipes()
mealPlan = pd.DataFrame(mealPlan.__dict__)
```

Code snippet 10: The class above depicts how the public recipe and meal plan tables were created in our app.

PrivateRecipes Class and Implementation

```
class PrivateRecipes:
    def __init__(self, recipes=[], instructions=[], servings=[], desc=[], isPublic=[]):
        self.RecipeName = recipes
        self.Instructions = instructions
        self.Servings = servings
        self.Description = desc
        self.isPublic = isPublic
privateRecipes = PrivateRecipes()
privateRecipes = pd.DataFrame(privateRecipes.__dict__)
```

Code snippet 11: The class above depicts how the private recipe table was created in our app.

AllIng Class and Implementation

```
class AllIng:
    def __init__(self, recipes=[], items=[], quantity=[], units=[], unitType=[]):
        self.RecipeName = recipes
        self.ItemName = items
        self.ItemQty = quantity
        self.Units = units
        self.UnitType = unitType

allIngPrivate = AllIng()
allIngPrivate = pd.DataFrame(allIngPrivate.__dict__)
# allIngPublic = {"RecipeName": [], "ItemName": [], "ItemQty": [], "Un
allIngPublic = AllIng()
allIngPublic = pd.DataFrame(allIngPublic.__dict__)
# allIngMealPlan = {"RecipeName": [], "ItemName": [], "ItemQty": [], "
allIngMealPlan = AllIng()
allIngMealPlan = pd.DataFrame(allIngMealPlan.__dict__)
```

***Code snippet 12:** The class above depicts how the three ingredient tables were created in our app.*

System Test Case Example

The following test case shows how the Personal Cookbook app will respond to user inputs when they attempt to remove an ingredient from one of their personal recipes. The functions shown in code snippets 9 and 10 ensure that only valid recipe ingredients are removed and the necessary tables are updated. If the user enters an invalid recipe name or ingredient, they will be shown UIs 9 and 10 respectively. Upon successful execution, UI 11 will be displayed.

Remove Recipe Ingredient Function

```
def removeRecipeIng(recipeName, itemName):  
    # checking if the listed ingredient is in both the public and private libraries  
    if (  
        privateRecipes[  
            (privateRecipes["RecipeName"] == recipeName) & (privateRecipes["isPublic"] == True)]).empty == False:  
        # removing the ingredient and updating the public table indices  
        allIngPublic.drop(  
            allIngPublic[(allIngPublic["RecipeName"] == recipeName) & (allIngPublic["ItemName"] == itemName)].index,  
            inplace=True)  
        # removing the ingredient and updating the private table indices  
        allIngPrivate.drop(  
            allIngPrivate[(allIngPrivate["RecipeName"] == recipeName) & (allIngPrivate["ItemName"] == itemName)].index,  
            inplace=True)
```

***Code Snippet 13:** This function searches for a specific ingredient listed for a given recipe and removes it from the private and public ingredient library when necessary.*

Executing the Remove Recipe Ingredient Function Through the UI

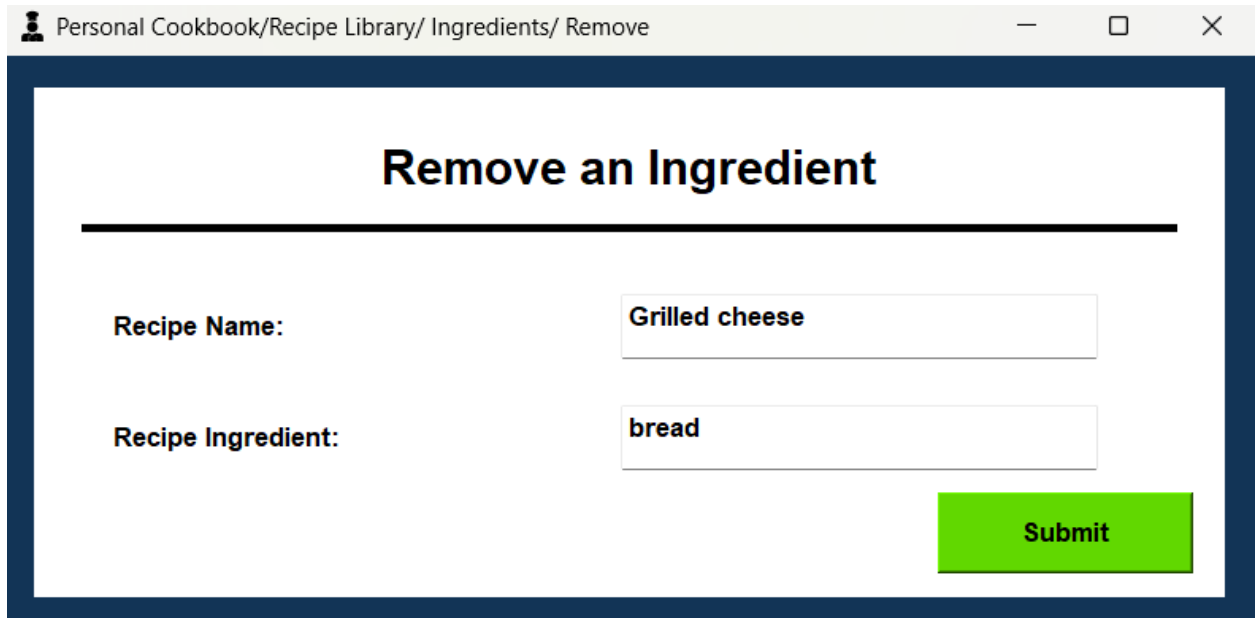
```

def submitRemoveRecipeIng(self):
    # creating notification message
    msg = QMessageBox()
    msg.setIcon(QMessageBox.Information)
    msg.setWindowIcon(QtGui.QIcon('chef.png'))
    msg.setWindowTitle("Remove Ingredient Info")
    recipeName = self.textEdit.toPlainText()
    recipeIngredient = self.textEdit2.toPlainText()
    # checking if recipe exists in the user's private recipe library
    y = rf.privateRecipes.loc[rf.privateRecipes["RecipeName"] == recipeName].all(1).any()
    if (y == True):
        # checking if the ingredient entered is listed for this recipe
        if (rf.allIngPublic.loc[(rf.allIngPublic["RecipeName"] == recipeName) &
                                (rf.allIngPublic["ItemName"] == recipeIngredient)].all(1).any() == True:
            rf.removeRecipeIng(recipeName, recipeIngredient)
            msg.setText("The ingredient \" + recipeIngredient + "\" has been removed from \" + recipeName + "\".")
        else:
            msg.setText("The ingredient you entered is not listed for this recipe.")
    else:
        msg.setText("The recipe you entered was not found in your library.")
    msg.exec_()

```

Code Snippet 14: The function above is called when the user attempts to remove a recipe ingredient through the 'Remove an Ingredient' UI (UI 9). To modify the necessary ingredient libraries, the function shown in code snippet 13 is called. To minimize user errors, this function checks 1) if the recipe name provided matches any recipe listed in the user's personal recipe library, 2) if the ingredient entered is listed as one of the ingredients for the provided recipe. If the recipe name is not valid, the user will see a pop-up message explaining the error (UI 10). Similarly, if the ingredient name is not valid, the user will see the pop-up message depicted in UI 11. When a recipe ingredient is successfully removed, the relevant tables will be updated and UI 12 will be shown.

Remove an Ingredient From a User's Private Recipe



Personal Cookbook/Recipe Library/ Ingredients/ Remove

Remove an Ingredient

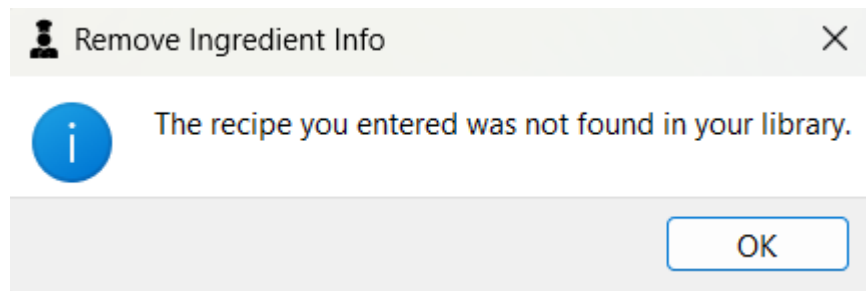
Recipe Name: Grilled cheese

Recipe Ingredient: bread

Submit

UI 9: This UI shows an example of the input parameters a user could enter to remove an ingredient from a given recipe. When successfully executed, the ingredient specified will be removed from all ingredient libraries containing this recipe name.

Removing a Recipe Ingredient: Info Message 1



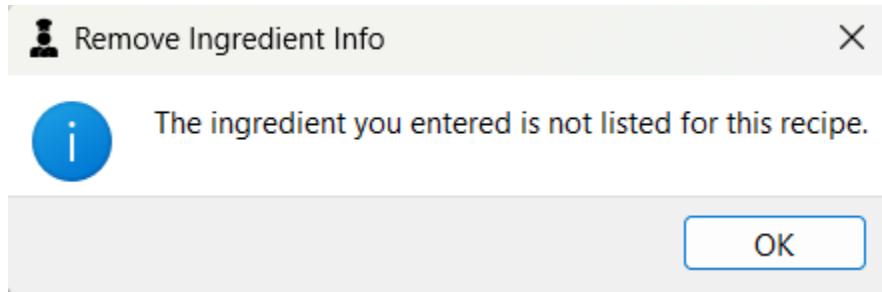
Remove Ingredient Info

The recipe you entered was not found in your library.

OK

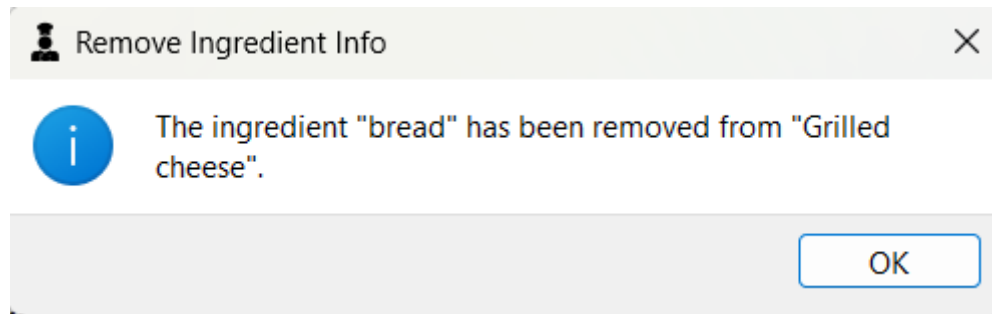
UI 10: This UI message is displayed to users when they enter a recipe that does not exist in their recipe library. To resolve this error users should select to remove ingredients only from recipes that already exist in their personal library.

Removing a Recipe Ingredient: Info Message 2



UI 11: This UI message is displayed to users when they select to remove an ingredient that does not exist for the recipe specified. To resolve this error, users should enter a valid ingredient.

Removing aFinal Code Recipe Ingredient: Info Message 3



UI 12: This UI message is displayed to users when they successfully remove an ingredient from a recipe. If the recipe they selected to modify is publicly available, this change will be reflected in both the public and private ingredient libraries. Otherwise, only the user's private recipe ingredient library will be affected.

Appendix:

[Final Code- Link to GitHub Repository](#)