

Software Engineering: 14:332:452:01
Group 1
Home Kitchen Automation



<https://github.com/max-legrand/chefcart>

Project Report 1
23 February 2021

Team Members:

- Indrasish Moitra
- Milos Seskar
- Shreyas Heragu
- Kevin Lin
- Maxwell Legrand
- Jonathan Wong
- Allen Chang
- Elysia Heah
- Mark Stanik
- Brandon Luong

Individual Contributions

All team members contributed equally

Table of Contents

Cover Page	1
Individual Contributions	2
Table of Contents	3
Work Assignment	5
Customer Problem Statement	9
Problem Statement	9
Problem Decomposition	13
Glossary Terms	15
Goals, Requirements, and Analysis	16
Business Goals	16
Enumerated Functional Requirements	16
Enumerated Nonfunctional Requirements	18
User Interface Requirements	19
Use Cases	23
Stakeholders	23
Actors and Goals	23
Casual Description	24
Use Case Diagram	27
Traceability Matrix	28
Fully Dressed Description	29
System Sequence Diagrams	35
User Interface Specification	39
Preliminary Design	39
System Architecture	50
UML Package Diagram	50
Architectural Styles	51

Mapping Subsystems to Hardware	51
Connectors and Network Protocols	52
Global Control Flow	52
Hardware Requirements	53
Project Size Estimation	54
Unadjusted Use Case Weight	54
Technical Complexity Factor	56
Use Case Points	56
Plan of Work	57
Gantt Chart	58
References	58

Work Assignment

We chose to divide our team into four sub teams based on the project functions. Group members were free to choose their own sub team and encouraged to join a team that could utilize their individual competencies.

Sub Team 1

Primary Task: Implement the “Digital Pantry” function. Focused on database management and creating ingredient structures to hold relevant metrics like names, quantities, weights, expiry dates, etc.

Members & Individual Competencies:

- Mark Stanik
 - Python
 - Google Cloud Platform
 - Java
 - HTML/CSS
- Dear all,
-
- Please send us the contribution of each team member for the current work ASAP.
Thanks.
-
- Your team is divided into 3-4 subgroups. Each group implemented different subprojects (modules) and each member is assigned different tasks. Also, your demo presentation showed that some subprojects are completed. Some are in the implementing phase and some will be done in the next step. Please identify the contribution of each team member and send it to us.
-
- Best regards,
- Bin
-
- Indrasish Moitra:
 - Google Cloud Platform (BigQuery, IoT, Cloud Function, Google APIs)
 - Python
 - C++
 - Microcontrollers and other hardware devices (piezoelectrics, load cells)

Sub Team 2

Primary Task: Implement the “Recipe Generator” function. Create an appropriate data structure to contain recipe components (recipe, ingredients, dietary restrictions, allergens, etc.). Leverage the Spoonacular API and query the Digital Pantry database to search for recipes. Integrate with the Grocery List function to add missing items to lists.

Members & Individual Competencies:

- Indrasish Moitra:
 - Google Cloud Platform (BigQuery, IoT, Cloud Function, Google APIs)
 - Python
 - C++
 - Microcontrollers and hardware devices (piezoelectrics, load cells)
- Brandon Luong:
 - Python
 - C
 - Ocaml
 - API
- Milos Seskar:
 - Python
 - C
 - JavaScript
 - Java
 - Hardware (Arduino, RaspberryPi)
 - R

Sub Team 3

Primary Task: Implement the “Grocery List” function. Focus on appropriate data structures to manage addition and removal of ingredients to the shopping list. Integrate “ingredient structures” from Sub Team 1. Utilize grocery store APIs to simplify workflow.

Members & Individual Competencies:

- Shreyas Heragu:
 - Java
 - JSP
 - Python
 - MySQL
- Kevin Lin:
 - Java
 - C
 - Python
- Elysia Heah:
 - Python
 - Java
 - C
 - MySQL

Sub Team 4

Primary Task: Create the Web App Display to house the project features for the user. Focus on User Interface and User Experience goals like attractive design, intuitive interface, and informative architecture.

Members & Individual Competencies:

- Jonathan Wong:
 - Java
 - ReactJS
 - JavaScript
 - HTML/CSS
- Maxwell Legrand:
 - SQL
 - Web Development
 - JavaScript
 - Python
 - Go
- Allen Chang:
 - Java
 - JavaScript
 - ReactJS
 - React Native
 - HTML/CSS, Python

Customer Problem Statement

Problem Statement:

Cooking is a skill that many people struggle to learn. Cooking carries many benefits, as it allows the chef to both save money and have better control over their nutritional intake. In 2016, results from a survey demonstrated that 80% of millennials thought that cooking at home was a “good way to live”. However, despite all the positive attributes associated with cooking, only 45% of those ages 18-24 and 64% of those 25 to 34 consider themselves to be “somewhat good or “good” at cooking. If cooking is portrayed to be such an important skill, then why do so few people know how to do it?

Learning how to cook is deceptively complex and time consuming for the chef. Cooking requires ingredients, which involves being able to keep track of what the chef already owns, while also having to periodically buy groceries from the supermarket. To top it off, chef’s need to find recipes that both match what they want to eat and can be created from what they want in the fridge. Specific dietary needs have to be met, and finding particular recipes that match certain diets can be frustrating. Cooking takes a lot of effort, and those who don’t see the effort worth the benefits in cost and nutrition will not bother trying to learn how to cook in the first place.

With the rise of the COVID-19 pandemic, a record high amount of people are staying home due to isolation practices. This has driven up the need of home-cooking, as many restaurants are closed or limited in capacity. Cooking from home provides a safe way for people to get a good meal, without having to put themselves at risk by going to a restaurant.

Software can alleviate these issues by automating the tedious parts of cooking. Problems such as memorizing ingredients at home and ingredients that need to be bought can be solved using a digital list. Many people use their web browsers at many times throughout the day and therefore can check their ingredients digitally whenever they want. The API spoonacular contains price and nutrition information about ingredients, while also containing different recipes. Spoonacular can search for specific recipes based on an ingredient list and dietary restrictions. Using this API can allow a user to easily generate recipes depending on what they have in their fridge and has a ranking system to determine the difficulty of each recipe. This would eliminate the time needed to search online for a recipe that hits all the factors of: matching ingredients, dietary restrictions, and desired dish of the user.

This product would be beneficial to not only beginners, but advanced home cooks as well. Taking a look at multiple people's perspectives will help solidify a need for our product.

Non-home cooks/beginners:

Cooking at home can be extremely difficult and time consuming. Not only do I need to keep track of key ingredients I have, I have trouble finding recipes for the ingredients I do have. I could always go grocery shopping for missing ingredients, but I have a 9-5 job and just want to relax after work. It can be really time consuming to pick out a good recipe beforehand and have time to go grocery shopping while also keeping in mind the ingredients I already have. Because of all this, it becomes much easier to just get restaurant takeout every night.

I would consider cooking at home more if I had something to let me know what ingredients I have, while also picking appropriate recipes for the ingredients I do have or giving me a missing ingredient list alongside a recipe. This would save me a lot of hassle and worry about missing ingredients and picking good recipes. I would have more time in my busy day and will be able to save more money and eat healthier by cooking at home more.

How will ChefCart help me? This application will be able to help me keep track of ingredients I already have via a digital pantry, so I can always see what I have and what I need right from my computer. A great part of the application being a web app is that in the grocery store, I can also log into my account on my phone and pick up where I left off. The application will also help me pick recipes that are simple enough to make for my level and will taste good. If

desired, the ChefCart account can apply to my whole family and not just myself. Involving other people would make cooking more fun and enjoyable.

Picky Eaters:

I love to cook at home, but I can never find new recipes that I can actually eat. With being a vegan and having gluten sensitivity, my options are limited and I am often stuck making the same meals throughout the week. I would love to explore new cuisines and cultures, but it is difficult with all my food restrictions. I have to spend a lot of time researching different dishes and altering them myself to get a version that abides with all my restrictions. Furthermore, finding grocery stores that sell certain ingredient substitutions can be challenging. Going to a store only to find it does not have what I need is often demoralizing and makes me want to give up on finding new dishes.

I would definitely cook more diverse dishes if I could only find good recipes for all my food needs. It would be great if there were something to give me certain recipes given all my restrictions. If there were also some way to know what ingredients I need and where to find all the necessary ingredients, it would save me a lot of time and frustration and make cooking more appealing.

How will ChefCart help me? This application will generate a set of recipes for me to cook. I only need to input my restrictions and then I will be able to pick from a bunch of recipes that I know will be safe for me to cook and eat. Whenever I want to make a meal, I can select a recipe and it will tell me what ingredients I am missing and where to find them. The application will use Walmart and Wegmans apis as well as web scraping to find local stores that carry the proper items.

Advanced home cooks:

I often cook at home and have many ingredients already. Many times I would like to just cook a quick meal with the ingredients I already have; however, it can be a pain to scour recipe sites in search of recipes I can readily make. I would love to explore new dishes, but I often find I am missing a few key ingredients. If there were a way for me to find recipes without having to make a grocery run for only a few items, I would definitely be willing to cook more diverse dishes.

How will ChefCart help me? This application will keep track of the ingredients I already have. When I want to make a quick, unique meal, I can search for specific cuisines and ChefCart can generate recipes that only contain ingredients that I already have, saving me a trip to the supermarket. This will make exploring new dishes and recipes much easier. A great feature of ChefCart is how it will keep track of existing ingredients. By updating different weights or number of ingredients, the application will automatically deduct certain quantities when a recipe is used, alerting me if I am running low on certain ingredients and giving me warnings if I do not have enough for a recipe.

Problem Decomposition

Problem 1: Keep track of pantry items

- Ingredient Data Structure - Ingredients will be treated as a dictionary data type. The keys of the dictionaries will be relevant information to be recorded for each ingredient. This includes name, quantity, weight, volume, expiry dates, etc. For example, we can create a dict as follows:

```
bananas = {name: "bananas", quantity: "7", weight: "", volume: "", expiry:
"2/18/21"}
```

We will then design sections of the web app to appropriately show key value pairs for ingredient dictionaries so the user can see useful information about the ingredients.

- Adding/Changing ingredients - We will create inline buttons to add ingredients to the pantry. On clicking the button the user will be given text boxes for key value pairs. After filling the boxes the user can submit the new dictionary to the Digital Pantry database. We will also add inline buttons to discrete metrics like quantity or weight so that they can be changed without having to repopulate the other key value pairs. Finally, we will add a button for removing pantry ingredients. This feature will be paired with notifications that are tied to expiry dates and ingredient quantities. When quantities/weights reach zero or less, the user will be given a pop up that will direct them to the button for removing the target ingredients.

Problem 2: Manage the grocery list

- Grocery List Data Structure - The grocery list will be contained in an SQL database that can be queried on demand into temporary tables for the user. The tables will have fields for useful parameters like price, quantity, comments, or desired stores.
- Add and Remove Items from grocery list - Similar to the digital pantry, we will have inline buttons for adding, removing, or editing items to/on the grocery list. We will use the user input from these buttons to update the grocery list tables, and then periodically refresh the database with these tables.
- Integrate with Digital Pantry - Use the ingredient dictionary structure to update ingredients that have been purchased using the grocery list. Use new quantities and weights from the list to edit values in the current pantry database.

- Find a Store - Once the grocery list is generated the user must be able to locate these ingredients at local stores. This will be achieved by leveraging grocery stores APIs like the Wegman's API to search for ingredients at local stores. When the ingredients are found the store names and prices can be added to the ingredient dictionaries. We can also use a database table to temporarily store the results of API calls and then query them using commands to sort by price, distance, availability, etc.

Problem 3: Recipe Generator

- Integrate Spoonacular API - Create "go-between" functions that ask the user for necessary Spoonacular inputs, especially ingredients, and package them to deliver to the API. We must also explore different Spoonacular functions to add user parameters like dietary restrictions. Finally, we need to find API limitations and assess the user's need for those features, so that we can choose to develop them as needed.
- Identify missing ingredients - Query the Digital Pantry database with the ingredients for a selected recipe. Do a join on these tables to find the ingredients missing from the Digital Pantry but found in the recipe. Offer to add these ingredients to the grocery list.
- Sort recipes by important metrics - This will be achieved using SQL commands. Most commonly we can use "order by" on price, complexity, prep time, and other fields that will be available for the generated recipes. We will also use "where" to choose binary parameters like whether or not a recipe is vegetarian.

Glossary of Terms

Home Cook - An individual that creates food in their own kitchen with their own ingredients rather than ordering from a restaurant

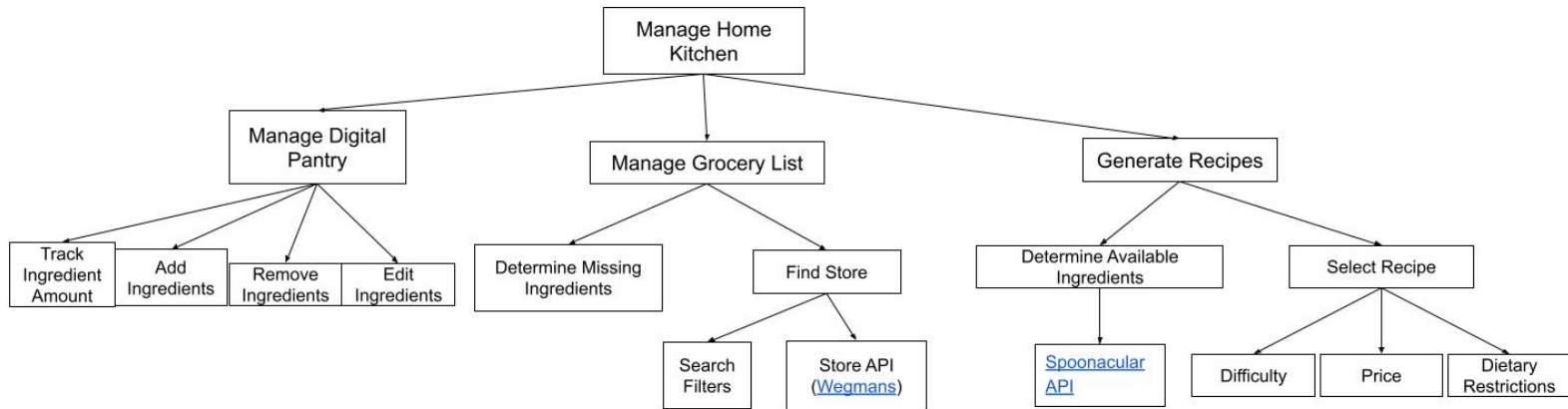
Pantry - A small room or closet where food and ingredients are kept.

Recipe - A set of instructions to create and prepare a certain dish. Usually will also list the ingredients that you need.

Spoonacular - A website that allows you to plan your meals in advance and save different recipes from around the web so you can have it all in one place

Goals, Requirements, and Analysis

Business Goals



The Business goals are divided into three main parts. The Digital Pantry, Grocery List, and Generate Recipes are the main parts. The digital pantry needs to be able to track and update ingredients. The grocery list needs to be able to determine missing ingredients and be able to find a store with filters. To generate recipes, we need to be able to determine available ingredients and be able to select a recipe based on difficulty, price and dietary restrictions.

Enumerated Functional Requirements

*Higher numbers are used to represent higher priority.

*User Authentication and Account Editing are assumed.

Web App General Usage

Label	Priority Weight	Description
REQ-1	3	ChefCart will allow users to switch between the Digital Pantry, Recipe Generator, Home Page, and Grocery List features as tabs on a Web App.
REQ-2	2	ChefCart will allow users to set their account dietary restrictions in their Home Page.

Digital Pantry

Label	Priority Weight	Description
REQ-3	5	ChefCart will allow users to add and remove ingredients to a digital pantry as well as edit their amounts.
REQ-4	5	ChefCart will allow users to save their digital pantry for later usage.
REQ-5	4	ChefCart will allow users to set their required threshold for each ingredient in their digital pantry on a weekly basis and show a warning if below threshold.
REQ-6	4	ChefCart will allow users to view the ingredients and their amounts by mass or volume in their digital pantry.

Grocery List

Label	Priority Weight	Description
REQ-7	3	ChefCart will allow users to determine missing ingredients in their digital pantry and create a grocery list from it for viewing.
REQ-8	4	ChefCart will allow users to find a store to purchase their grocery list items with filter/sort for price and distance.
REQ-9	3	ChefCart will allow users to add and delete additional ingredients to their grocery list if desired.

Recipe Generator

Label	Priority Weight	Description
REQ-10	3	ChefCart will allow users to input ingredients, instead of using the digital pantry.

REQ-11	5	ChefCart will allow users to generate recipes that they can cook with as much of their available ingredients and view them via Web App.
REQ-12	4	ChefCart will allow users to sort and filter the generated recipes by important metrics and dietary preferences (ex. price, difficulty).
REQ-13	4	ChefCart will allow users to select a generated recipe to view required ingredients, instructions, price, missing ingredients and difficulty.

Additional Functional Requirements

Label	Priority Weight	Description
REQ-14	4	ChefCart will allow users to search ingredients that they can add.
REQ-15	3	ChefCart will allow users to add missing recipe ingredients to their grocery list.

Enumerated Nonfunctional Requirements

Label	Priority Weight	Description
REQ-16	4	ChefCart should securely maintain user data. Data will not corrupt, will not be accessible by anyone other than the account user and will save automatically with 99% accuracy.
REQ-17	4	ChefCart should be able to run on the iOS 10 and above, Android 8.0 Oreo and above, Windows 7 and above, and MacOS Catalina (10.15) and above as well as the major Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari browsers.
REQ-18	2	ChefCart should be able to startup and shut down quickly within 3 seconds.
REQ-19	3	ChefCart should have smooth transitions from page to page that last less than 1 second.
REQ-20	3	ChefCart should present relevant data within 3 seconds and would allow a person with average eyesight to read it from 1 meter.
REQ-21	3	ChefCart should display properly when resizing the application window.

User Interface Requirements

*User authentication User Interface Requirements are assumed.

Web App First Time Usage

Label	Priority Weight	Description
UIREQ-1	3	A user that launches ChefCart will be greeted by a login page where they can enter their account username and password or register a new account with the Sign Up button.
UIREQ-2	3	Upon clicking the login button in the login page with the correct credentials, the user will be taken to the Digital Pantry.
UIREQ-3	5	All pages beyond the login page will have tabs to navigate between the “Account” page, Digital Pantry, Grocery List, and Recipe Generator pages.
UIREQ-4	5	Clicking on any of the Account, Digital Pantry, Grocery List, and Recipe Generator tabs will take the user to the corresponding page.
UIREQ-5	4	On pages not listed as a tab or the login page, there will be a button option to navigate back to the previous page.
UIREQ-6	3	In the Account page, the user can edit their account details and save them.
UIREQ-7	4	Clicking the Logout button lets the user logout of ChefCart and returns them to the login page.

Digital Pantry

Label	Priority Weight	Description
UIREQ-8	5	In the Digital Pantry, clicking the amount in an ingredient lets the user edit the amount for that ingredient
UIREQ-9	5	In the Digital Pantry, clicking Add will let the user add a new ingredient with its amount
UIREQ-10	5	In the Digital Pantry, clicking the delete icon will let the user delete the

		corresponding ingredient
UIREQ-11	5	In the Digital Pantry, allow users to view the current ingredients in the pantry in a tabular fashion.

Grocery List

Label	Priority Weight	Description
UIREQ-12	4	In the Grocery List, missing ingredients will already display and the user can click Add to add more ingredients to their grocery list and delete them as in the digital pantry.
UIREQ-13	5	In Grocery List, clicking on Find Store will display stores for users to choose from to purchase the list in the Choose Store Page.
UIREQ-14	3	In the Choose Store Page, there will be an option to sort stores by price and distance.
UIREQ-15	4	In the Choose Store Page, a user can click on a store to show its location, desired ingredients and their prices in the Store Page.

Recipe Generator

Label	Priority Weight	Description
UIREQ-16	5	In the Recipe Generator, a user can add ingredients they want or click Use Pantry to use any of their Digital Pantry ingredients and click Discover to generate recipes. The user will then be sent to the Recipes page.
UIREQ-17	3	In the Recipes page, the list of recipes displayed can be filtered by difficulty, price and percentage of ingredients available.
UIREQ-18	4	In the Recipes page, a user can click on one of the recipes to display its instructions, ingredients, missing ingredients and estimated price range in the Meal Page.

Additional Requirements

Label	Priority Weight	Description
UIREQ-19	4	In each page that manipulates ingredients, allow for the user to search for valid ingredients that they can add.
UIREQ-20	4	In the Digital Pantry, allow for users to see a warning when their ingredients are below a certain threshold.
UIREQ-21	3	In the Meal Page, they can click “Shop” to add missing recipe ingredients to their grocery list, sending them to the Grocery List page.

User Interface Graphics: See part 2 section “User Interface Specifications” for UI mockups and detailed navigation breakdowns.

Acceptance Test Cases:

1. REQ-1
 - a. ATC1.01 Ensure a user can switch between the 4 tabs: Digital Pantry, Recipe Generator, Home Page, and Grocery List.
2. REQ-2
 - a. ATC2.01 A user can change their account dietary restrictions, appropriately filtering out recipes shown.
3. REQ-3
 - a. ATC3.01 Add and remove ingredients by quantity, weight, volume as well as edit expiry date and verify by visual update.
 - b. ATC3.02 Disable removal if there are no ingredients to remove.
 - c. ATC3.03 Fail if inputted quantity, weight or volume is not a positive number.
4. REQ-4
 - a. ATC4.01 Ensure a working network connection to edit ingredients, save them, navigate out of the Digital Pantry and back to verify the persistence of data.
5. REQ-5
 - a. ATC5.01 Set an existing ingredient threshold above their current amount and verify that a warning is shown.
6. REQ-6
 - a. ATC6.01 Ensure a user can view the ingredient amount by quantity,

- weight, volume, and expiry date
- b. ATC6.02 Notifies the user if there are no ingredients to view.
7. REQ-7
- a. ATC7.01 Manually determine missing ingredients in the Digital Pantry and verify that the generated Grocery List restocks those ingredients above the set threshold when generated.
8. REQ-8
- a. ATC8.01 A user can generate stores to purchase their ingredients.
 - b. ATC8.02 Ensure the user can sort and filter the ingredients.
 - c. ATC8.03 Notify the user if no stores are generated.
 - d. ATC8.04 Fail if no ingredients are in the grocery list.
9. REQ-9
- a. ATC9.01 Add and delete ingredients in the grocery list and verify visual update in the grocery list table of ingredients.
 - b. ATC9.02 Fail if ingredient quantity, weight, or volume is not a positive integer.
10. REQ-10
- a. ATC10.01 Input manual ingredients that are not part of the Digital Pantry and generate recipes to ensure those ingredients were used.
11. REQ-11
- a. ATC11.01 A user can generate recipes using their digital pantry.
 - b. ATC11.02 Ensure the user can view the recipes
12. REQ-12
- a. ATC12.01 Sort and filter recipes by price and difficulty and verify order.
13. REQ-13
- a. ATC13.01 Select a recipe and verify the display of ingredients, instructions, price, missing ingredients and difficulty.
14. REQ-14
- a. ATC14.01 Ensure that ingredients added are provided by a drop down populated using the Spoonacular API.
15. REQ-15
- a. ATC15.01 Add missing recipe ingredients to their grocery list

Use Cases

Stakeholders:

The stakeholders listed below are going to be interested in ChefCart:

1. Aspiring chefs such as home cooks, parents, and college students will have direct usage of ChefCart and have interest in using it to manage their pantry, grocery shopping, and meal planning.
2. Grocery Stores have an interest because ChefCart's grocery shopping functions will list them as potential options for home chefs to purchase from, bringing more business to them.
3. Healthcare and insurance companies will have an indirect interest in ChefCart due to the project's positive effect on user health. Users will be able to decrease consumption of fast food and unhealthy takeout by encouraging home cooked meals
4. Environmentalists have a stake in ChefCart because the project will reduce food waste and excess consumption

Actors and Goals:

Initiating Actors:

Actor	Role	Goal
Home Chef (User)	The user of the application. They can add or remove ingredients from their digital pantry, search for new recipes based off of the ingredients in their pantry, and create grocery lists for ingredients they need and/or are missing	The goal for the home chef is to have a stress-free cooking experience when managing the home kitchen.

Participating Actors:

Actor	Role
Database/System	The database system records the user's digital pantry ingredients, dietary restrictions, and account information. The system queries the grocery store APIs to provide the client side with information on the inventory and pricing of ingredients to help the user shop and view different stores. The system also queries the Spoonacular API to provide information and data about ingredients and recipes.

Casual Description:

General Usage

UC-1: Signup - Allows individuals to sign up for an account through the webapp

Derivations: UIREQ-1

UC-3: Login - Allows users to login in with their account credentials

Derivations: UIREQ-2, UIREQ-3

UC-10: Account Editing - Allows users to view account setting and edit them. The setting includes, dietary restriction, username, password, and location.

Derivations: REQ-2, UIREQ-6

UC-11: Logout - Allows users to logout of any page and be sent back to login screen

Derivations: UIREQ-7

UC-18: Search Ingredients - Allows the user to search for an ingredient when trying to add it to either the grocery list or digital pantry.

Derivations: REQ-14, UIREQ-19

Digital Pantry

UC-2: Track Pantry - Allows users to view and manipulate ingredients from their digital pantry.

Derivations: REQ-1, REQ-3, REQ-4, REQ-5, REQ-6, REQ-14, UIREQ-2, UIREQ-3, UIREQ-4, UIREQ-8, UIREQ-9, UIREQ-10, UIREQ-11

UC-12: View Pantry Ingredients - Allows the user to view ingredients currently present in the digital pantry.

Derivations: REQ-4, REQ-6

UC-13: Add Pantry Ingredients - Allows the user to add ingredients or add existing ingredient amounts to the digital pantry.

Derivations: REQ-3, REQ-4, REQ-14, UIREQ-8, UIREQ-9, UIREQ-19

UC-14: Remove Pantry Ingredients - Allows the user to remove ingredients or remove existing amounts from the digital pantry.

Derivations: REQ-3, REQ-4, UIREQ-8, UIREQ-10

UC-19: Ingredient Threshold Warning - Allow the user to be warned when the ingredient amount is below a threshold

Derivations: REQ-5, UIREQ-20

Grocery List

UC-4: Create Grocery List - Allows users to generate a grocery list based on missing ingredients in the digital pantry and ingredients they would like to purchase.

Derivations: REQ-1, REQ-7, REQ-9, REQ-14 UIREQ-4, UIREQ-12, UIREQ-3

UC-5: View Grocery Stores - Allows users to view grocery stores that have the ingredients for them to purchase and complete their grocery list.

Derivations: REQ-8, UIREQ-5, UIREQ-13, UIREQ-15

UC-6: Sort Grocery Stores- Allows users to sort the list of stores based off of any set of user-specified preferences such as distance and price

Derivations: REQ-8, UIREQ-5, UIREQ-14

UC-15: View List Ingredients - Allows the user to view ingredients currently present in the grocery list.

Derivations: UIREQ-12

UC-16: Add List Ingredients - Allows the user to add ingredients or add existing ingredient amounts to the grocery list.

Derivations: REQ-9, REQ-14, UIREQ-12, UIREQ-19

UC-17: Remove List Ingredients - Allows the user to remove ingredients from the grocery list or remove existing ingredient amounts.

Derivations: REQ-9, UIREQ-12

Recipe Generator

UC-7: Discover Recipes - Allows users to view a list of generated recipes based off of ingredients available in their pantry as well as any manually-inputted items

Derivations: REQ-1, REQ-10, REQ-11, REQ-12, REQ-14, UIREQ-4, UIREQ-16, UIREQ-19, UIREQ-3

UC-8: Sort/Filter Recipes - Allows users to sort and filter the list of generated recipes based off of any set of user-specified dietary preferences such as price, prep time, difficulty, percentage of ingredients available

Derivations: REQ-12, UIREQ-5, UIREQ-17

UC-9: Viewing Recipe - Allows users to view the recipe instructions and other relevant information such as required ingredients, price, difficulty, prep time, etc.

Derivations: REQ-13, UIREQ-5, UIREQ-18

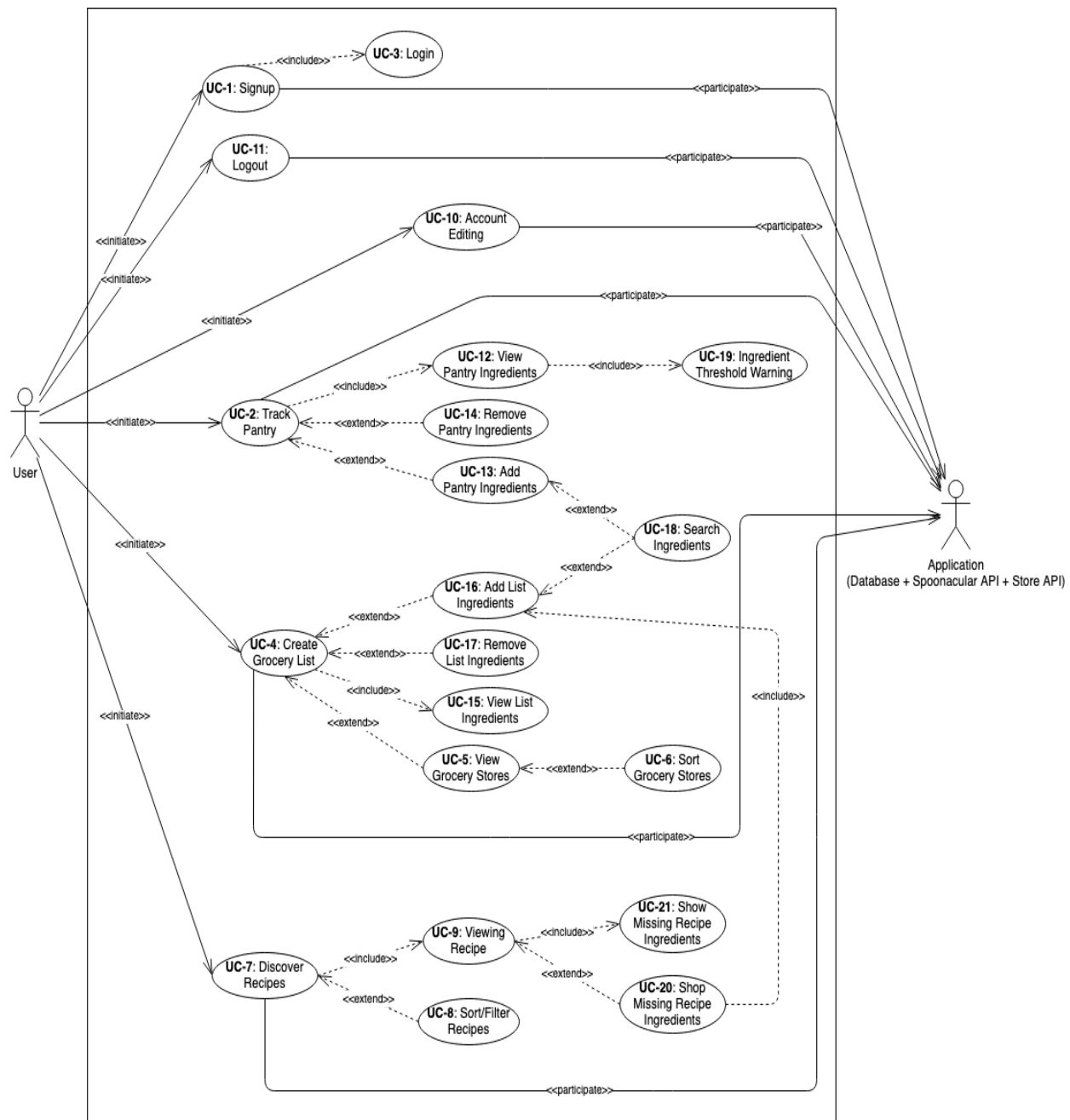
UC-20: Shop Missing Recipe Ingredients - Allow the user to add missing recipe ingredients to their grocery list after viewing a recipe.

Derivations: REQ-15, UIREQ-21

UC-21: Show Missing Recipe Ingredients - Allow the user to be shown the missing ingredients that they do not have for a recipe

Derivations: UIREQ-18

Use Case Diagram



The use case diagram has an initiating actor (**user**) and a participating actor (**application**). The base cases are shown to be directly initiated by the **user**. Subroutines are related to their parent routines through an “**extends**” relationship if the subroutine MAY occur during a parent routine, and through an “**includes**” relationship if the subroutine ALWAYS occurs during a parent routine.

Note: We didn’t connect **Login** to the **user** because we were advised not to connect any subroutine to an initiating actor (**Login** is an **included** subroutine of **Signup**).

Traceability Matrix

Requirements mapped to each use case (Use cases on the columns)

Req't	PW	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
REQ-1	3		X		X			X														
REQ-2	2			X							X											
REQ-3	5		X	X										X	X							
REQ-4	5		X										X	X	X							
REQ-5	4		X																	X		
REQ-6	4		X										X									
REQ-7	3				X																	
REQ-8	4					X	X															
REQ-9	3				X												X	X				
REQ-10	3							X														
REQ-11	5							X														
REQ-12	4							X	X													
REQ-13	4									X												
REQ-14	4		X		X			X						X			X		X			
REQ-15	4																				X	
UIREQ-1	3	X																				
UIREQ-2	3		X																			
UIREQ-3	5		X		X			X														
UIREQ-4	5		X		X			X														
UIREQ-5	4					X	X		X	X												
UIREQ-6	3										X											
UIREQ-7	4											X										
UIREQ-8	5		X											X	X							
UIREQ-9	5		X											X								
UIREQ-10	5		X												X							

UIREQ-11	5		X																		
UIREQ-12	4				X										X	X	X				
UIREQ-13	5					X															
UIREQ-14	3						X														
UIREQ-15	4					X															
UIREQ-16	5							X													
UIREQ-17	3								X												
UIREQ-18	4									X											X
UIREQ-19	4							X						X			X		X		
UIREQ-20	4																		X		
UIREQ-21	3																			X	
MAX PW		3	5	5	5	5	4	5	4	4	3	4	5	5	5	4	4	4	4	4	4
TOTAL PW		3	53	7	27	17	11	38	11	12	5	4	9	28	24	4	15	7	8	8	4

Fully Dressed Description

→ This symbol will be used to denote when the user performs an action.

← This symbol will be used to denote when the system performs an action.

UC-2: Track Pantry	
<p>Related Requirements:</p> <ul style="list-style-type: none"> ● REQ-1 ● REQ-3 ● REQ-4 ● REQ-5 ● REQ-6 ● REQ-14 ● UIREQ-2 ● UIREQ-3 ● UIREQ-4 ● UIREQ-8 ● UIREQ-9 ● UIREQ-10 	

<ul style="list-style-type: none"> ● UIREQ-11
<p>Initiating Actor:</p> <ul style="list-style-type: none"> ● Home Chef
<p>Actor's Goal:</p> <ul style="list-style-type: none"> ● To track ingredients in the user's physical pantry.
<p>Participating Actors:</p> <ul style="list-style-type: none"> ● Database / System
<p>Preconditions:</p> <ul style="list-style-type: none"> ● The user has logged in and has navigated to the Digital Pantry tab.
<p>Minimal Guarantees:</p> <ul style="list-style-type: none"> ● The application will provide the user the ability to manually update their food stock in the digital pantry. Each field (ingredient: string, quantity: int, weight: float, volume: float, expiry: date, threshold to be quantity, weight or volume) will be implemented such that there is no possibility for input error (i.e. for number fields, you use a number picker, for dates you use a date picker, and for ingredient names, you can search/select from a list).
<p>Success Guarantees:</p> <ul style="list-style-type: none"> ● The application will update the corresponding ingredients with their quantity, volume or weight, and expiration date.
<p>Plan of Action for Main Success Scenario:</p> <p>Adding a new ingredient</p> <ol style="list-style-type: none"> 1. → The user enters a new ingredient or updates an existing ingredient by inputting the fields for ingredient quantity, volume, weight, threshold and expiry date in a popup. 2. → If the user cancels the addition of the ingredient, nothing will be changed and the user will be returned to the digital pantry precondition state, ending the use case. 3. → The user confirms the addition of this ingredient. 4. ← The system verifies this ingredient with the Spoonacular API. 5. ← If the ingredient is invalid, the use case ends and the ingredient is not added. 6. ← The ingredient is verified and valid, so the system updates the database with the new ingredient attributes accordingly. 7. → The new ingredient fields are displayed back to the user in the Digital Pantry.
<p>Plan of Action for Alternate Scenarios or Contingencies:</p> <p>Editing/Updating ingredients</p> <ol style="list-style-type: none"> 1. → The user selects an ingredient field for editing. 2. → If the user unselects the field, editing will cancel and the unedited Digital Pantry information will be redisplayed, returning them to the precondition state and the use case ends. 3. → The user inputs their new ingredient value for either of quantity, volume, weight, threshold and expiration date. 4. → The user confirms the edit.

5. ← The system updates the ingredient in the database and redisplay the updated table.

Deleting Ingredients

1. → The user clicks on the trash icon to delete the corresponding ingredient.
2. ← The system asks the user for confirmation of the delete with the popup.
3. → If the user cancels the delete, the user is returned to the precondition state and the use case ends.
4. → The user confirms the delete.
5. ← The system will delete the corresponding ingredient in the database and redisplay the updated table to the user.

UC-4: Create Grocery List
<p>Related Requirements:</p> <ul style="list-style-type: none"> ● REQ-1 ● REQ-7 ● REQ-9 ● REQ-14 ● UIREQ-4 ● UIREQ-12 ● UIREQ-3
<p>Initiating Actor:</p> <ul style="list-style-type: none"> ● Home Chef
<p>Actor's Goal:</p> <ul style="list-style-type: none"> ● Generate a grocery list based on missing ingredients in the digital pantry and ingredients they would like to purchase.
<p>Participating Actors:</p> <ul style="list-style-type: none"> ● Database / System
<p>Preconditions:</p> <ul style="list-style-type: none"> ● The user has launched the ChefCart app. ● The user has successfully logged into ChefCart. ● After logging in, the user has navigated to the Grocery List Tab.
<p>Minimal Guarantees:</p> <ul style="list-style-type: none"> ● Same as success guarantees: no failure involved.
<p>Success Guarantees:</p> <ul style="list-style-type: none"> ● The grocery list will reflect the user's desired changes.
<p>Plan of Action for Main Success Scenario:</p> <ol style="list-style-type: none"> 1. ← The system generates a grocery list based on the ingredients under a user set amount

threshold in the digital pantry.

2. ← The grocery list is displayed to the user, allowing for further addition, deletion and editing of ingredients.

Plan of Action for Alternate Scenarios or Contingencies:

a. Deleting Ingredients

1. → The user presses the trash can icon to delete the relevant ingredient.
2. ← The system asks for confirmation of the delete action with a popup.
3. → If the user cancels the delete, the ingredient will not be deleted and the Grocery List page will be redisplayed and the use case ends.
4. → The user confirms the delete.
5. ← The grocery list refreshes itself with updated changes and is redisplayed.

b. Editing Ingredients

1. → The user clicks on the field of an existing ingredient.
2. ← The system provides the user with a picker or text field input for the user to input the new value.
3. → If the user cancels the edit by unselecting the field input, the grocery list is redisplayed and the use case ends.
4. → The user inputs their new value.
5. ← The system updates the value and redisplay the updated information to the user.

c. Adding Ingredients

1. → The user presses the *Add Ingredient* button to open up an input window
2. → If the user cancels the add, the grocery list is redisplayed and the use case ends.
3. ← The application opens up an ingredient searcher through the Spoonacular API.
4. → The user searches for the desired ingredient.
5. → The user finds the desired ingredient.
6. → The user adds the relevant quantity of the ingredient to the grocery list.
7. ← The ingredient searcher closes, displaying an updated grocery list.

UC-5: View Grocery Stores

Related Requirements:

- REQ-8
- UIREQ-5
- UIREQ-13
- UIREQ-15

Initiating Actor:

- Home Chef

Actor's Goal:

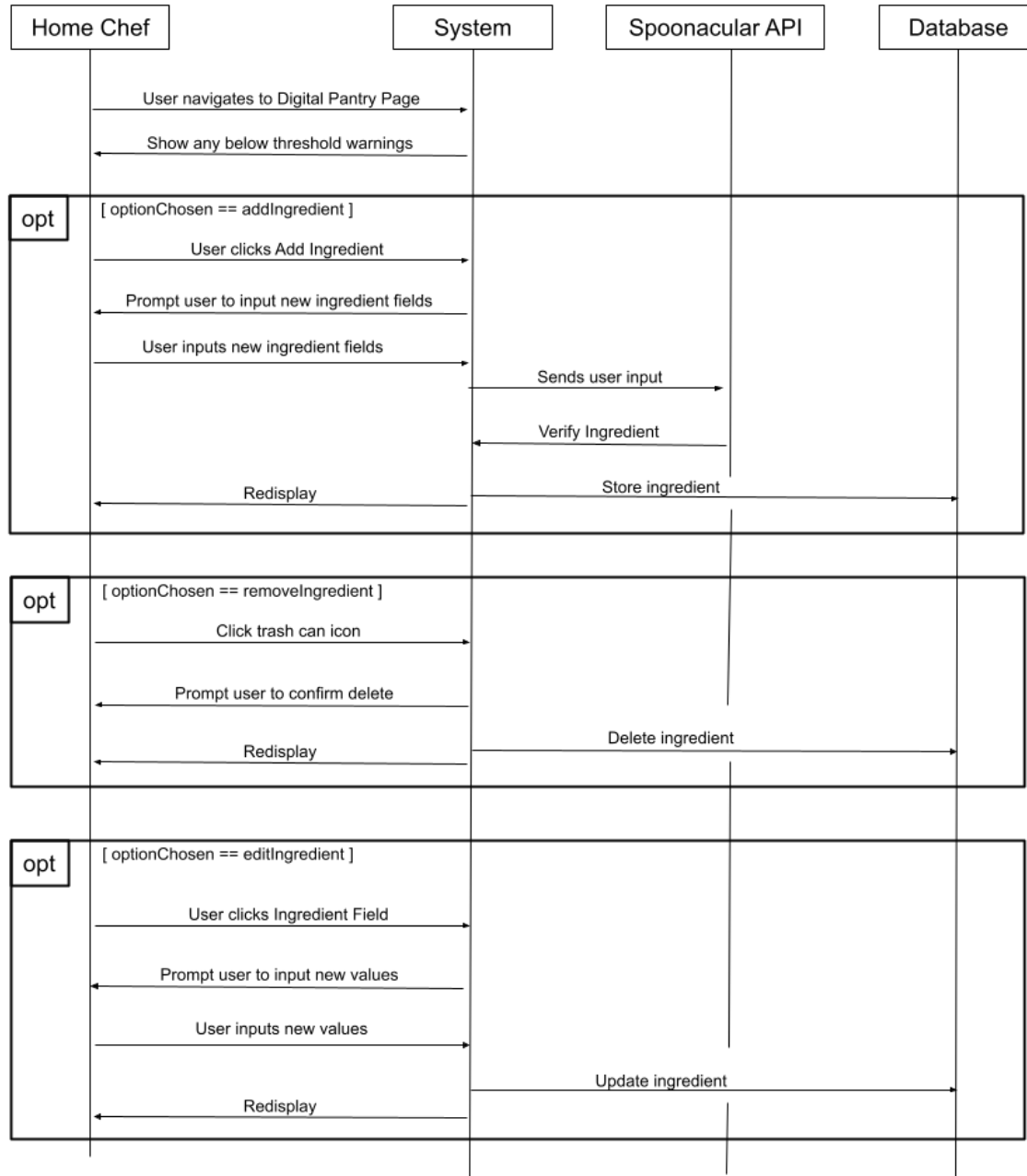
<ul style="list-style-type: none"> To find a suitable store to purchase ingredients from the grocery list.
Participating Actors: <ul style="list-style-type: none"> Database / System
Preconditions: <ul style="list-style-type: none"> The user has launched and logged into ChefCart The user was on the grocery list page, generated a grocery list and clicked on the “Choose Store” button.
Minimal Guarantees: <ul style="list-style-type: none"> There is no failure involved, it will be the same as success guarantees.
Success Guarantees: <ul style="list-style-type: none"> The displayed stores, if any, will allow users to click on them to view their information in the Store Page.
Plan of Action for Main Success Scenario: <ol style="list-style-type: none"> ← The system searches for stores using the stores APIs available. ← The system displays the grocery stores to the user sorted by price. → The user chooses to sort them by price and/or distance. → The user clicks on a store. ← The system displays the store’s information (address, distance) along with the ingredients and their prices.
Plan of Action for Alternate Scenarios or Contingencies: <ol style="list-style-type: none"> N/A

UC-7: Discover Recipes
Related Requirements: <ul style="list-style-type: none"> REQ-1 REQ-10 REQ-11 REQ-12 REQ-14 UIREQ-4 UIREQ-16 UIREQ-19 UIREQ-3
Initiating Actor:

<ul style="list-style-type: none"> ● Home Chef
<p>Actor's Goal:</p> <ul style="list-style-type: none"> ● To find suitable recipes to cook at home.
<p>Participating Actors:</p> <ul style="list-style-type: none"> ● Database / System
<p>Preconditions:</p> <ul style="list-style-type: none"> ● The user has launched and logged into ChefCart ● The user has navigated to the Recipe Generator page.
<p>Minimal Guarantees:</p> <ul style="list-style-type: none"> ● No failure involved, this will be the same as success guarantees. ● The system will only allow correct user input when inputting ingredients. An ingredient searcher will be used to properly help the user select an ingredient.
<p>Success Guarantees:</p> <ul style="list-style-type: none"> ● The system will display any found recipes to the user.
<p>Plan of Action for Main Success Scenario:</p> <ol style="list-style-type: none"> 1. → The user chooses to use Digital Pantry items by checking the "Use Digital Pantry Ingredients" box. 2. → The user clicks on the "Discover Recipes" button. 3. ← The application will call the Spoonacular API to generate recipes based on the ingredients input and the user's account dietary restrictions, displaying them. 4. ← The system sends the user to the Recipes Page and displays the generated recipes to the user.
<p>Plan of Action for Alternate Scenarios or Contingencies:</p> <p>Adding manual ingredients</p> <ol style="list-style-type: none"> 1. → The user may or may not use Digital Pantry items by checking the "Use Digital Pantry Ingredients" box. 2. → The user clicks on the "Add Ingredient" button. 3. ← The system displays a popup with fields to add an ingredient to search for recipes. 4. → If the user cancels the addition of an ingredient, the user is sent back to the Recipe Generator page and the use case ends. 5. → The user confirms the addition of the ingredient. 6. ← The system adds the ingredient and displays the added ingredient for recipe searching. 7. → The user clicks on the "Discover Recipes" button. 8. ← The application will call the Spoonacular API to generate recipes based on the ingredients input and the user's account dietary restrictions, displaying them.

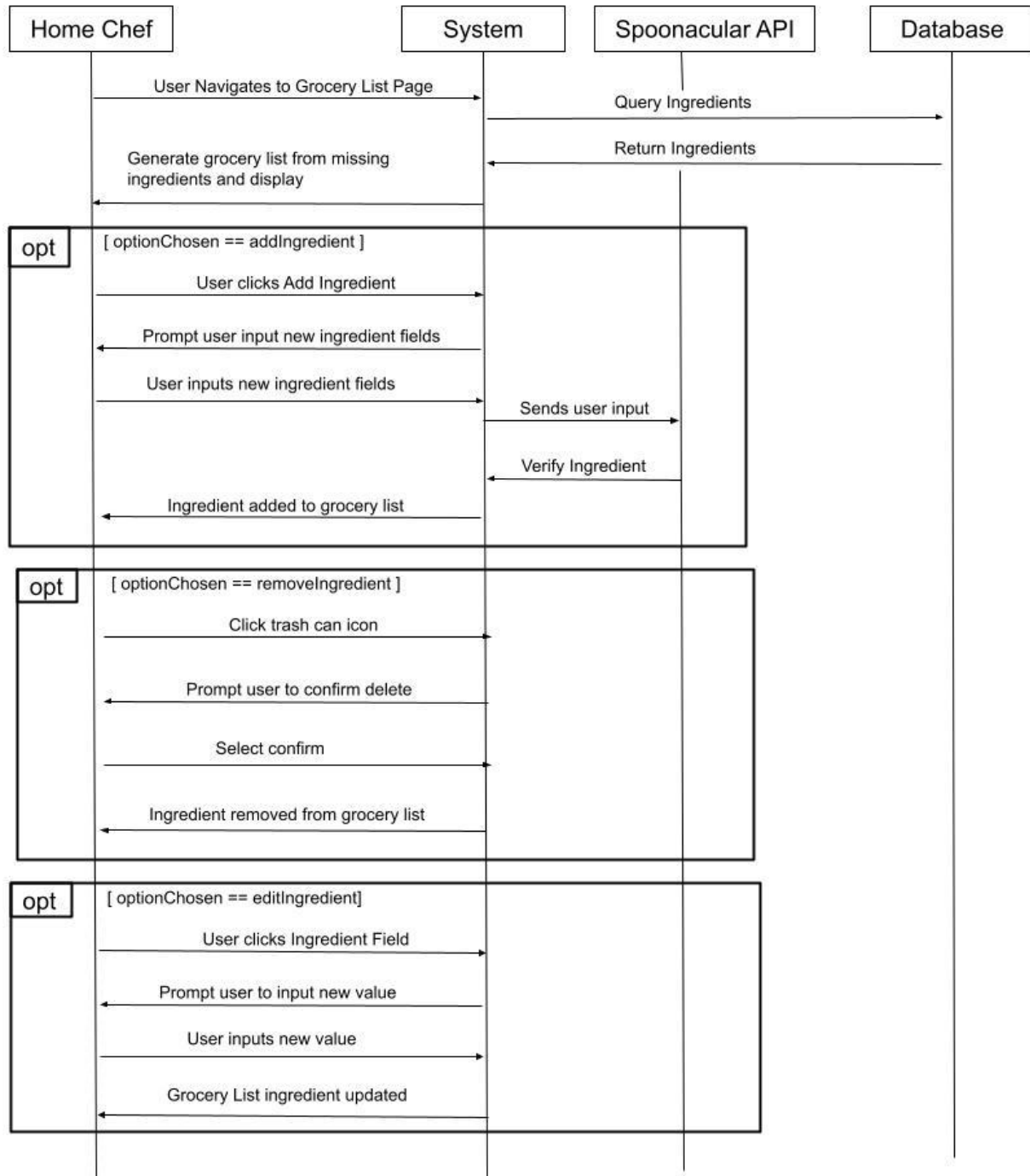
System Sequence Diagrams

UC-2: Track Pantry



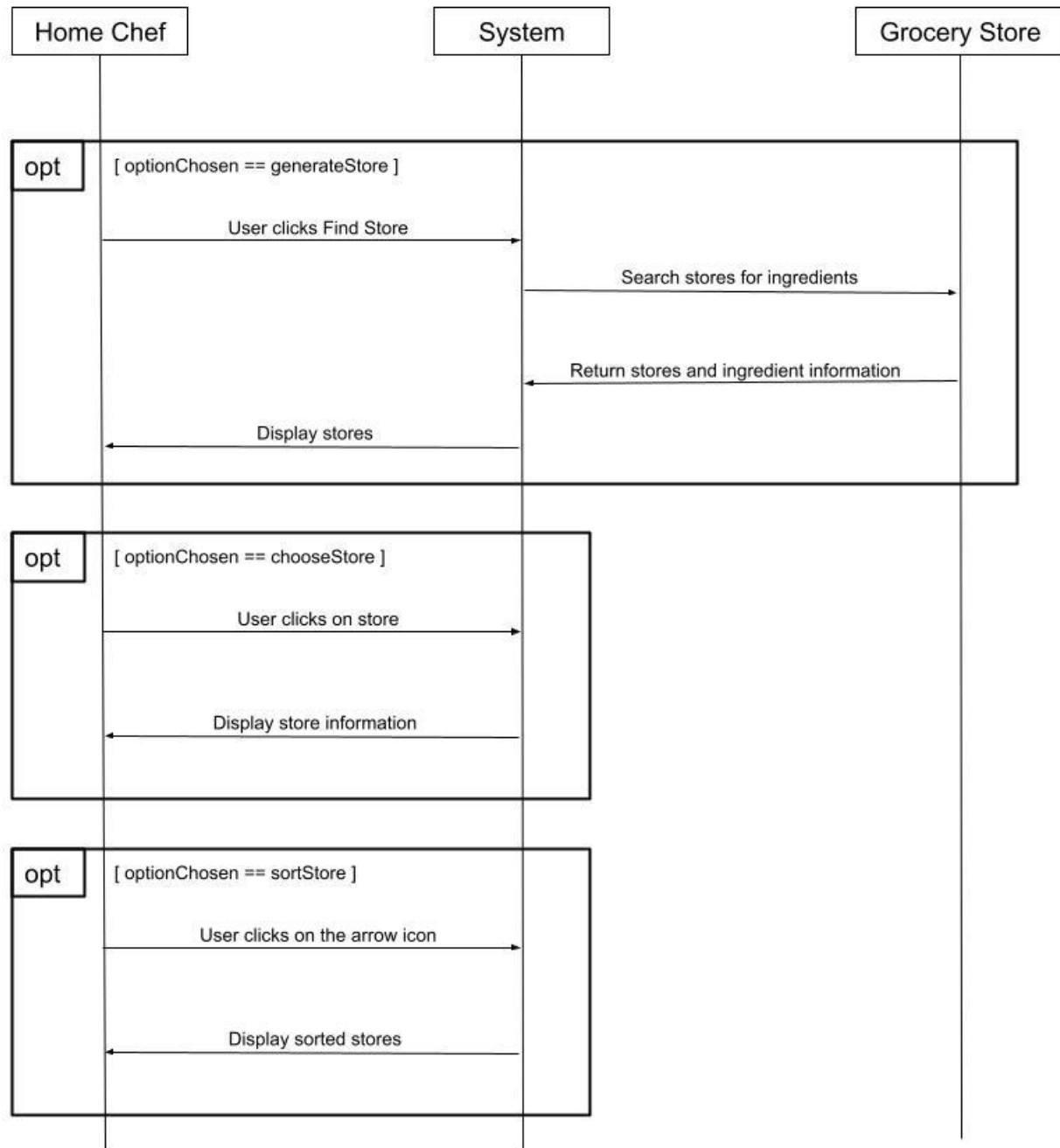
Note that the user may cancel the addition, editing, and deleting of ingredients. To track the pantry, the user first navigates to the Digital Pantry Page which will immediately warn the user if there are ingredients that are low. Then, the user can choose from 3 options to add, remove or edit ingredients. By inputting the desired updated ingredient information, the database saves it and redisplayes the corresponding updates to the user.

UC-4: Create Grocery List



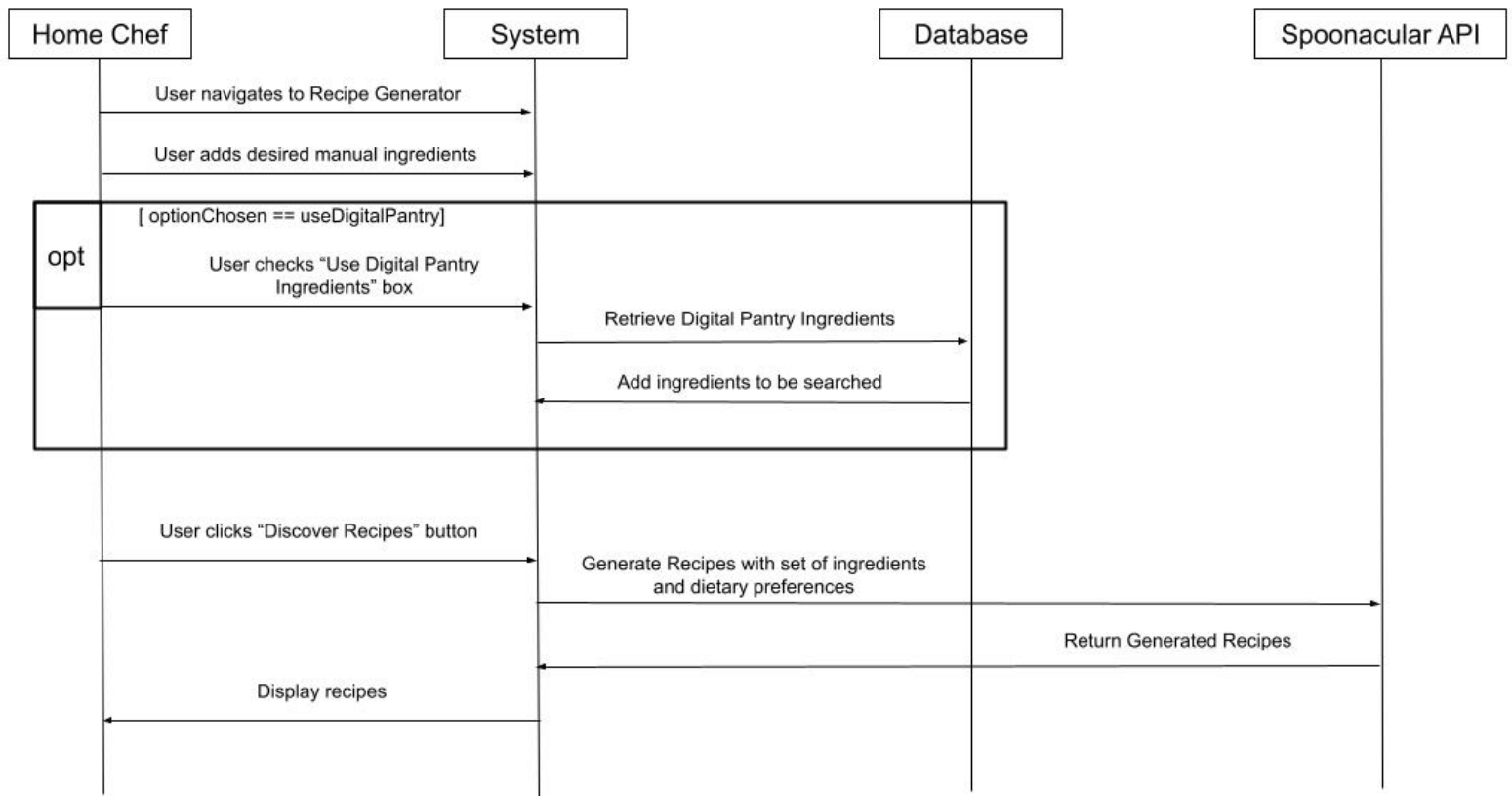
Note that the user may cancel the addition, editing, and deleting of ingredients. To create a grocery list, the user first navigates to the Grocery List Page. The system will then query the database for the missing ingredients that the user normally has. As in the digital pantry, the user may add, remove or edit ingredients in the generated grocery list.

UC-5: View Grocery Stores



To find a store, the user must have generated a grocery list and clicked on the Find Store button. We comb grocery store APIs to search for stores with the right ingredients and return them to display to the user. Then, the user will have the option to sort the stores listed and click on one to see its information such as total price and store location.

UC-7: Discover Recipes



To discover recipes, the user first must navigate to the Recipe Generator page. The user may choose to add manual ingredients or opt to use all digital pantry ingredients in their search for a recipe. Once they click Discover Recipes, Spoonacular will help generate the recipes with the set of ingredients and display them to the user.

User Interface Specifications

Preliminary Design

UI/UX (Account Handling) subproject:

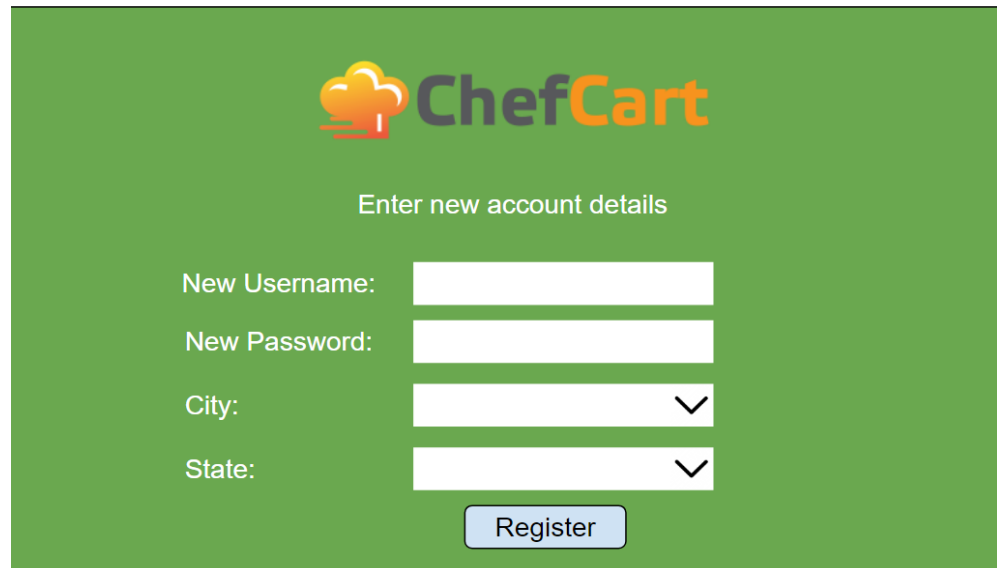
Login Page

A login page with a green background. At the top center is the ChefCart logo, which consists of an orange icon resembling a chef's hat or a stylized 'C' followed by the text 'ChefCart' in a bold, sans-serif font. Below the logo are two white input fields. The first field is labeled 'Username:' and the second is labeled 'Password:'. Below these fields are two light blue buttons with rounded corners. The top button is labeled 'Login' and the bottom button is labeled 'Sign Up'.

Description:

- If the user does not have a ChefCart account, then the user will click on the *Sign Up* button to create an account. After clicking the button, it will lead the user to the *Sign Up Page*.
- If the user does have a ChefCart account, then the user will input their username and password in the respective fields. After inputting the username and password, the user will click the *Login* button to sign in. It will lead the user to the *Digital Pantry Page*.

Sign-Up Page



The image shows a sign-up page for ChefCart. The background is a solid green color. At the top center is the ChefCart logo, which consists of a stylized orange and yellow chef's hat icon followed by the text "ChefCart" in a bold, sans-serif font. Below the logo, the text "Enter new account details" is centered. There are four input fields arranged vertically: "New Username:" with a white text box, "New Password:" with a white text box, "City:" with a white text box and a downward arrow, and "State:" with a white text box and a downward arrow. At the bottom center is a blue button with the word "Register" in white text.

Description:

- When creating a new account, the user is prompted to enter a username and password, while also inputting their current city and state of residence. This information will be stored in the database for when the user logs in a separate time. Upon clicking the *Register* button, the user will be led to the *Login Page* where they can properly login (see Login Page).

Edit Account Page

Description:

- When in the *Account Page*, the user can edit their accounts. The *Account Page* consists of...
 - New Username - The user can change their username by inputting a new username into the field.
 - New Password - The user can change their password by inputting a new password into the field.
 - City - The user can click on the drop down icon and a list of cities will appear. The user can change their city by clicking on one of the cities in the list.
 - State - The user can click on the drop down icon and a list of states will appear. The user can change their state by clicking on one of the states in the list.
 - Dietary Restriction - The user can click on the drop down icon and a list of dietary restrictions will appear. The user can change their dietary restriction by clicking on one of the dietary restrictions in the list. If the user wants to remove a dietary restriction, the user can click on the “x” button next to the dietary restriction.
- The user can save all the account changes by clicking on the *Save* button.

Digital Pantry subproject:

Digital Pantry

[Username]

Logout

Digital Pantry







Grocery List

Recipe Generator

Account

Digital Pantry

Add Ingredient

	Ingredient	Quantity	Weight (g)	Volume (ml)	Expiry	
	Tomatoes	1	N/A	N/A	2/9/21	
	Bread	N/A	1000	N/A	2/10/21	
	Oil	N/A	N/A	800	8/5/21	

Description:

- When in the *Digital Pantry*, the user will be able to see what is in their digital pantry along with key information: ingredient name, quantity, weight in grams, volume in milliliters, and expiration date.
- The user has two main actions in the digital pantry:
 - The *Add Ingredient* button opens a pop-up window that allows users to manually add ingredients and information. Once the ingredient is added, the pop-up will close and the *Digital Pantry* will refresh, showing the new ingredient(s) included.
 - The *Trash Can* button allows the user to manually delete an ingredient from their *Digital Pantry*. Upon clicking the *Trash Can* button, the ingredient will be deleted and the *Digital Pantry* will refresh.

Recipe Generator subproject:

Recipe Generator

The screenshot shows a web application interface for the 'Recipe Generator' subproject. At the top, there is a green header bar containing a '[Username]' placeholder and a 'Logout' button. Below the header is a navigation bar with four buttons: 'Digital Pantry', 'Grocery List', 'Recipe Generator' (which is highlighted), and 'Account'. The main content area is titled 'Recipe Generator' and contains a checkbox labeled 'Use Digital Pantry Ingredients' which is checked. To the right of this checkbox is an 'Add Ingredient' button. Below these elements, the text 'No Custom Ingredients Entered' is displayed. At the bottom of the main content area is a 'Discover Recipes' button.

Description:

- The *Recipe Generator* page shows the user what ingredients to include when discovering recipes.
- The user can check or uncheck the *Use Digital Pantry Ingredients* signifying if they want recipes that use the ingredients stored in the *Digital Pantry*. Interacting with the checkbox keeps the user in the *Recipe Generator*.
- The user also has an option to add specific ingredients that are not in the *Digital Pantry*. Clicking the *Add Ingredient* button opens a pop-up window that allows the user to input ingredient(s) and specific information such as quantity, weight in grams, volume in milliliters, and/or expiration date. The button does not add to the *Digital Pantry*, it just adds it to a list of ingredients for a recipe. When the pop-up closes, the *Recipe Generator* page will refresh and show the custom ingredient(s) added.
- The user can press the *Discover Recipes* button which will lead the user to the *Recipes Page* (shown below).

Recipes Page

[Username]			Logout
Digital Pantry	Grocery List	Recipe Generator	Account
Back	Recipe Results		Filters
Recipe	Difficulty: 1 - 10 ^	% Ingredient Available ^	
Garlic Bread	3	80%	
Bruschetta	4	100%	
Tomato Soup	7	70%	

Description:

- The user can view the list of recipes available to them. The list contains the recipe name, difficulty, and percentage of ingredients available.
- By clicking on the arrow icon next to the “Difficulty” or “% Ingredient Available” headers, the user can sort the corresponding column in ascending or descending order.
- By clicking on the *Filter* button, it opens a pop-up window that allows users to manually add any set of user-specified preferences.
- By clicking on the *Back* button, it allows the user to return back to the *Recipe Generator* when they click it.

Meal Page

[Username]
Logout


Digital Pantry
Grocery List
Recipe Generator
Account

Back
Garlic Bread

Instructions:

Step 1: Preheat oven to 350°F.
Step 2: Prepare the garlic bread: cut the loaf in half, horizontally.

Missing Ingredients

Ingredient	Quantity	Weight (g)
 Bread	N/A	100

Shop

Description:

- In the *Meal Page*, the user can view the instructions for the recipe and the missing ingredient(s).
- The user can click on the *Back* button to return back to the *Recipe Page*.
- The user can scroll up or down to look at the instructions by clicking on the scroll bar and drag it up or down.
- The user can click on the *Shop* button to add the missing ingredients to their grocery list. This will then navigate them to the Grocery List page.

Grocery List subproject:

Grocery List

The screenshot shows a web application interface for a 'Grocery List'. At the top, there is a green header bar with a '[Username]' placeholder and a 'Logout' button. Below the header is a navigation bar with four buttons: 'Digital Pantry', 'Grocery List' (which is highlighted), 'Recipe Generator', and 'Account'. The main content area is titled 'Ingredients You Need' and features an 'Add Ingredient' button. Below this is a table with the following columns: 'Ingredient', 'Quantity', 'Weight (g)', 'Volume (ml)', and 'Expiry'. The table contains two entries: 'Tomatoes' with a quantity of 2, weight of N/A, volume of N/A, and expiry of 2/18/21; and 'Oil' with a quantity of N/A, weight of N/A, volume of 1000, and expiry of 1/4/22. To the left of the table are images of a tomato and an oil bottle. To the right of each row is a trash can icon. At the bottom of the main content area is a 'Find Store' button.

Ingredient	Quantity	Weight (g)	Volume (ml)	Expiry
Tomatoes	2	N/A	N/A	2/18/21
Oil	N/A	N/A	1000	1/4/22

Description:

- When in the *Grocery List*, the user will be able to see what is in their grocery list along with key information: ingredient name, quantity, weight in grams, volume in milliliters, and expiration date.
- The user has three main actions in the *Grocery List*:
 - By clicking on the *Add Ingredient* button, it opens a pop-up window that allows users to manually add ingredients and information. Once the ingredient is added, the pop-up will close and the grocery list will refresh, showing the new ingredient(s) included.
 - The *Trash Can* button allows the user to manually delete an ingredient from their grocery list. Upon clicking the *Trash Can* button, the ingredient will be deleted and the grocery list will refresh.
 - By clicking on the *Find Store* button, it will lead to the *Choose Store Page*. (Shown below)

Choose Store Page

[Username]

Logout

Digital Pantry

Grocery List

Recipe Generator

Account

Back




Choose Store

Name	Distance (mi) ^	Total Price (\$) ^
Wegmans	4	27.32
Walmart	10	39.18
Whole Foods	12	41.47

Description:

- The user can view the list of stores available to them. The list contains the store name, distance, and total price.
- By clicking on the arrow icon next to the “Distance” or “Total Price” headers, the user can sort the corresponding column in ascending or descending order.
- By clicking on the *Back* button, it allows the user to return back to the *Grocery List* when they click it.

Store Page (*Back* → Choose Store Page)

[Username]		Logout
Digital Pantry	Grocery List	Recipe Generator Account
Back	Wegmans	
  	Ingredient	Price (\$)
	Tomatoes	9.23
	Bread	10.36
	Oil	7.73
Total Price:		\$27.32

Description:

- In the *Store Page*, the user can view the different ingredients, the price for each ingredient and the total price.
- The user can click on the *Back* button to return back to the *Choose Store Page*.

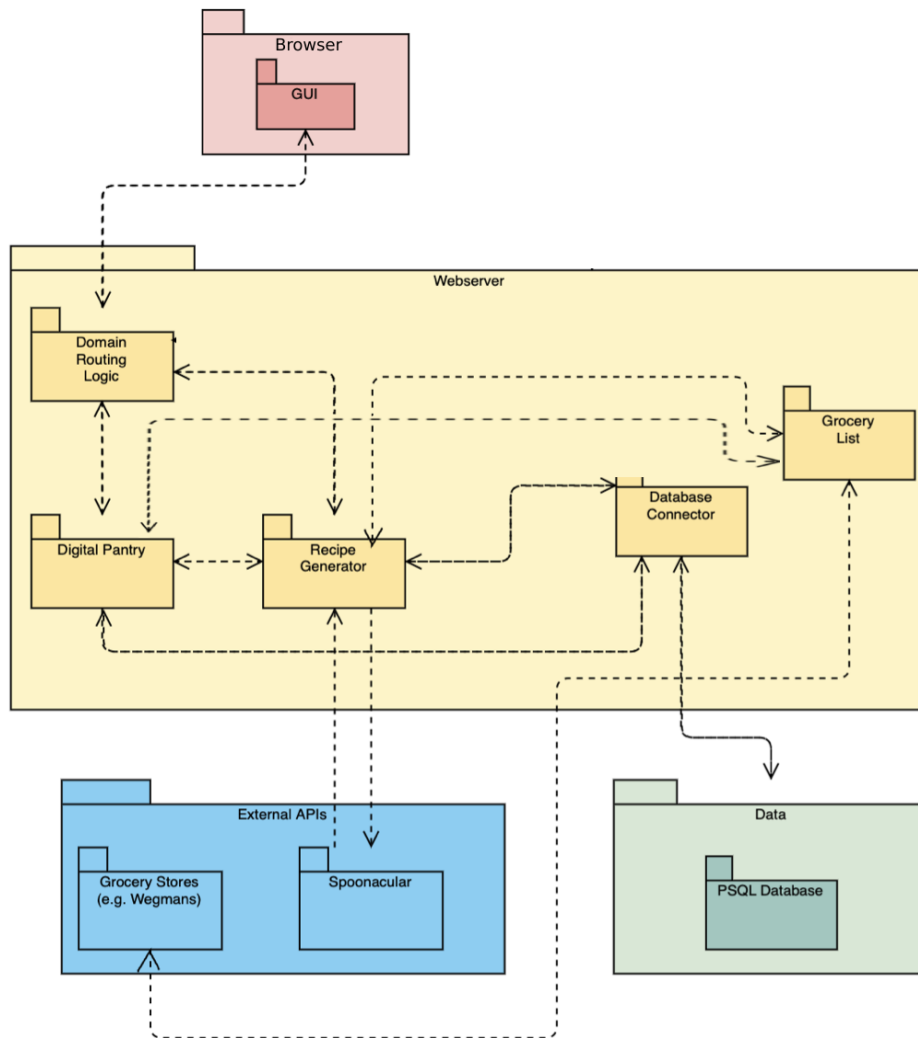
User Effort Estimation

See Project Size Estimation in Part 3

System Architecture

UML Package Diagram

ChefCart utilizes different APIs to develop different features. Using the domain routing logic, the results of each of the Digital Pantry, Recipe Generator, and the Grocery List features will be displayed to the user in the Client GUI. Feature pairs such as the Digital Pantry and the Recipe Generator, the Grocery List and the Digital Pantry and the Recipe Generator need to communicate with each other. A grocery list needs to know what is in the pantry and so does the recipes generator. In addition, the recipe generator may refer the user to add recipe ingredients to their grocery list. Each feature is connected to the API they need to call and the database to retrieve ingredients used in the Digital Pantry. Our web server, database and client GUI are separated so that each feature can be easily tested modularly.



Architectural Styles

In our design, we utilize two main architectural styles:

1. Event-driven
2. Model-view-controller.

The event-driven architecture is commonly used in modern applications. It is a software architecture and model for application design. It consists of three parts: event producer, event router, and event consumer. In the event producer, the user inputs and updates information through our web application. Then, the data will be sent to the event router. The event router will filter and push the event to the appropriate consumer. In the event consumer, our application will generate (recipes, stores), update, remove, or add content (ingredients, user's personal information) depending on the user input.

The model-view-controller is commonly used for developing the user interface. It is an application design model composed of three interconnected parts. The three parts are model, view, and controller. In the view section, the users are able to view information such as ingredients, recipes, and personal information depending on what page they are on. After the user inputs, update or add/remove ingredients or other information, the controller will update the model and view accordingly. The model contains the data, mainly ingredients, used by the application.

Mapping Subsystems to Hardware

In our application, there are three subsystems.

1. Website Hosting
2. Database Content
3. Client Access

All three subsystems need to run on multiple computers. The first subsystem, Website Hosting, will be using an external server like Amazon Web Services (AWS) and Heroku. AWS and Heroku are the two most commonly used cloud services that allow us to deploy, monitor, and scale web apps. The second subsystem, Database Content, is hosted on an external database like PSQl and will be hosted on a server to be accessible by ChefCart. The third subsystem, Client Access, will allow the clients to access their personal computer or mobile device.

Connectors and Network Protocols

Our team will be implementing a REST API to server HTTP requests. This is common in the modern industry when design web applications with clients and servers. HTTP will help us separate the client from the server and will allow for the client-side to send requests every now and then from the server so that it can return the appropriate information for the client-side to use. This makes sense because the user will be manipulating the digital pantry only ever so often, whose update information is passed through as an HTTP request to propagate the change in our database. In addition, we will use protobuf messages or Protocol Buffers in order to send messages to pass information between the client and the server. Since our intended language is going to be Go, this method is handled well in Go and is well documented for it. Protobuf messages allow flexible creation and receiving of messages as data can be directed into any of several data structures of our choosing.

Global Control Flow

Execution Orderness:

ChefCart is largely event-driven rather than procedure-driven. Some aspects of ChefCart will be linear, but each user can generate their actions in different orders. In particular, all users must login with their account before accessing the system features. Once logged in, users can then choose any of the features whether it be the digital pantry, recipe generator, or the grocery list pages. Users have access to each of these features in a nonlinear fashion. In addition, when manipulating the digital pantry, the user may choose to add, edit or delete ingredients in any order of their choosing. In each of the recipe generator and the grocery list pages, the user must go through these features in a linear fashion. When using the grocery list feature, they first must generate their grocery list before looking at potential stores and they must click a store before they can view its detailed information. As for the recipe generator, a user must input ingredients before they can start searching for recipes that mostly use them and they must click on a recipe before being able to view the recipe instructions.

Time Dependency:

ChefCart does not have timers in its system. However, some of the aspects of ChefCart change based on external APIs. For example, when calling the grocery store API to retrieve available ingredients that a user might buy, the price of the ingredients and the stock is dependent on the state of the store at that time. Other than that, there are no timers.

Hardware Requirements

ChefCart utilizes a database to store data and uses Wi-Fi to have the database communicate with the end-user client. Users should be able to access ChefCart by running it on iOS 10 and above, Android 8.0 Oreo and above, Windows 7 and above, and MacOS Catalina (10.15) and above as well as the major Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari browsers. These will cover most devices on the market today. Our application requires a color display with a minimum resolution of 1330 x 700 pixels and will work up to 1920 x 1080 pixels. In order to communicate with our servers, users need a 1 Mbps download and upload speed. Users will not need storage on their device to access ChefCart as it will be a web application. As long as the user has Wifi, they can access the web app. The requirements of the database storage space will expand with the number of users and the server requirements will not need to scale since we are not opting for microservice architecture.

Project Size Estimation

Unadjusted Use Case Weight

Use Case Classification	Class Description	Weight
Simple	1 participating actor 1 to 3 steps	5
Average	2 participating actors 4 to 6 steps	10
Complex	3 participating actors 7 or more steps	15

Use Case	Description	Weight
UC-1: Signup	Populate text fields and dropdowns, query and edit database, click button, redirect user. 3 steps, 1 actor	Average 10
UC-2: Track Pantry	Click button, populate fields, click button, query and edit database. 5 steps, 1 actor.	Average 10
UC-3: Login	Populate text fields and click button, query database. 3 steps, 1 actor	Simple 5
UC-4: Create Grocery List	Click button, query two databases on a join, create temporary table with list of ingredients. 6 Steps, 1 actor	Average 10
UC-5: View Grocery Stores	Click button, input radius, add filters. 3 steps, 2 actors	Simple 5
UC-6: Sort Grocery Stores	Sort existing database table. 1 step, 1 actor	Simple 5
UC-7: Discover Recipes	Choose filters, query database tables, return tables to user for display. 4 steps, 1 actor	Average 10
UC-8: Sort/Filter Recipes	Choose filter params, choose sorting params, query table 2-3 times for all filters and sorting params. Create new temporary table for return to user. 8 steps, 1 actor	Complex 15

UC-9: Viewing Recipe	Simple UI, display organized page to user. 1 step, 1 actor	Simple 5
UC-10: Account Editing	Navigate to tab, edit fields and dropdowns, query and update database. 3 steps, 1 actor	Simple 5
UC-11: Logout	Click button. 1 step, no actors	Simple 5
UC-12: View Pantry Ingredients	Navigate to tab. 1 step, no actors	Simple 5
UC-13: Add Pantry Ingredients	Click button, edit fields and dropdowns, query and update database. 3 steps, 2 actors	Average 10
UC-14: Remove Pantry Ingredients	Click button, confirm alert. 2 steps, 1 actor	Simple 5
UC-15: View List Ingredients	Navigate to tab. 1 step, no actors	Simple 5
UC-16: Add List Ingredients	Click button, edit fields and dropdowns, query and update database. 3 steps, 2 actors	Average 10
UC-17: Remove List Ingredients	Click button, confirm alert. 2 steps, 1 actor	Simple 5
UC-18: Search Ingredients	Fill text field, click button, query database and return table. 2 steps, 2 actors	Average 10
UC-19: Ingredient Threshold Warning	Query database, return alert, confirm alert. 3 steps, 1 actor	Simple 5
UC-20: Shop Missing Recipe Ingredients	Click button, redirect to tab, run query on three database tables, return temporary table to user, confirm additions for each row to grocery list table. 8 steps, 2 actors	Complex 15
UC-21: Show Missing Recipe Ingredients	Click button, redirect to tab, run query on three database tables, return temporary table to user. 7 steps, 2 actors	Complex 15

$$UUCW(\text{ChefCart}) = 11 * \text{Simple} + 7 * \text{Average} + 3 * \text{Complex} = 11 * 5 + 7 * 10 + 3 * 15 = 170$$

Technical Complexity Factor

Technical Factor	Characteristics	Weight	Complexity (Relative, 1-5)	Calculated Factor (W*C)
T1	Ease of use for Home Chef (user)	2	4	$2 * 4 = 8$
T2	Concurrent use is not required (1 actor only)	0.5	0	$0.5 * 0 = 0$
T3	Reasonably efficient use of hardware resources	1	3	$1 * 3 = 3$
T4	No access for third party users	0.5	0	$0.5 * 0 = 0$
T5	Generality/portability for all users	1	4	$1 * 4 = 4$
T6	Reasonably fast performance, on the order of seconds/operation	1	3	$1 * 3 = 3$
T7	Users do not require training to use system	2	5	$2 * 5 = 10$
T8	Minimal developer maintenance	1	2	$1 * 2 = 2$

Technical Factor Total = 30

$TCF = 0.6 + (\text{Technical Factor Total} / 100) = 0.6 + (0.3) = 0.9$

Environmental Complexity Factors

Assume ECF = 1

Use Case Points

$UCP = UUCW * TCF * ECF$

Unadjusted Use Case Weight = 170

Technical Complexity Factor = 0.9

Environmental Complexity Factor = 1

$UCP = 170 * 0.9 * 1 = 153$

ChefCart has a final Use Case Points (UCP) or 153 points.

Plan Of Work:

Upon completion of this initial report, we plan to begin working as our 4 separate teams. In the short term, our initial goals are to start development of the Digital Pantry while other groups begin to explore the API and information available to be integrated into the Recipe Generator, Grocery List, and Webapp. Due to the modular nature of these functions, they can work independently, however for the final implementation they will all be linked together, and as such, development for each individual piece is staggered after the initial development of the Digital Pantry portion. Each team will divide work amongst the individual members, and all work in each sub-team will be distributed evenly to ensure every member makes meaningful contributions.

At the present, each team's individual work has been focused around planning out relevant features, requirements, and relevant tech stack in order to ensure that these modular pieces will be able to interact with one another. Currently we are in the process of starting development and working with each subteam to set up a collaborative development environment for a streamlined workflow. As we progress further into the course, development for each modular piece will wrap up at which point team members will perform continuous testing and transition to the role of assisting integration into the remaining features.

In order to collaborate in an efficient way, we have created an overarching Github repository in order to keep the project up-to-date, and have established numerous text/voice channels for collaboration and feedback. In addition to sub-team communication, the entire group meets twice a week to discuss updates and progress reports in order to ensure we adhere to the Gantt chart provided below.

