Software Engineering 14:332:452

Group #9

Full Report #1: ChefPal

Submission Date: 02/28/2021

Team Members:

Shahir Ghani
Malena Bashar
Dymytriy Zyunkin
Daniel Samojlik
Amanda Phan

Michael Fong
Malak Khalifa
Aswathy Aji
Azim Khan
Nirav Patel

Table of Contents

Individual Breakdowns

	Shahir	Malena	Nirav	Aswathy	Dymytriy	Michael	Azim	Amanda	Malak	Daniel
Customer Statements	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Decomposition into sub-problems	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Glossary of Terms	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Business Goals	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Enumerated Functional Requirements	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Enumerated Non-functional Requirements	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
User Interface Requirements	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Stakeholders	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Actors and Goals	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Use Cases	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
System Sequence Diagrams	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Preliminary Design	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
User Effort Estimation	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Identifying Subsystems	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Architecture Styles	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Mapping Subsystems to Hardware	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Connectors and Network Protocols	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Global Control Flow	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Hardware Requirements	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Project Size Estimation	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Plan of Work	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
References	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%

1) Customer Problem Statement

a. Prospective Customer Perspectives

College Students

Living at college is hard. Not only am I in a different environment from the one I've grown up in, but at the same time I have to manage my life by myself. I have to deal with the constant stress of classes, exams, and making sure all my assignments are completed and submitted on time. This means that I have to stay on top of all my work, while somehow managing my time and money efficiently. Of course, saving money means that I can't get a meal plan, simply because it costs way too much money per meal. According to the USDA, the average person spends about $163 to $367 per month. Meanwhile, with the meal plan, I'm spending $500 a month! It would save far more money by cooking meals for myself. That also means I have to stop eating out and ordering takeout all the time, but I don't want to get stuck simply making boxed mac & cheese in my microwave. Unfortunately, grocery stores in my area are limited, so I rarely have the ingredients needed to cook quality recipes at home.

It would be so much more convenient for me if there was something out there that could help me find recipes that I could make at home that only uses the limited amount of ingredients that I have. It would definitely be a more healthier option than ordering pizza every night, and it would also be a more financially safer option.

To solve my problems, I can use ChefPal. ChefPal would provide me with recipes based on the ingredients that I select in the app that I have on me right now. It would also provide me with recipes that require the same ingredients I have, but only require a few additional ingredients to make. Since the app automatically prioritizes showing me recipes with the least amount of extra ingredients to purchase, it's easy to find recipes that are financially feasible for me. Plus, I can save these recipes so that I can make them for a later time, which not only saves me money, but also time later on as well! Now, I have more time to focus on classes because ChefPal does all the hard work of matching me to healthy, affordable food options.

Amateur Home Cooks

I just got married but I am not a good cook. I've never had to make food because when I lived with my parents, they did all the cooking. Plus, when I lived alone, I would just purchase takeout because whenever I did try to cook, I would go out of my way to buy too many new grocery items needed for a recipe and then end up cooking it wrong, ultimately wasting my money. Then I would have all these leftover ingredients that I didn't know how to put to use. My husband works long hours and I want to be able to make food for the both of us, but I don't know how to be resourceful with the ingredients I purchase. An additional challenge is that my husband is vegan, so it is even harder for me to figure out what food to make for both of us,

while using the same ingredients at home. As a new couple, we want to be able to not only save money when providing meals for ourselves and future kids, but also ensure our family is eating healthy and that we fulfill each person's dietary needs.

ChefPal would be the most useful solution to my problems. Especially by using the feature in the app that filters by dietary needs, I can browse recipes that are vegan, which works perfectly for my husband. Then, I can change the dietary needs so I can find recipes that include meat, which I personally enjoy eating. Plus, this fixes my problem of using ingredients wisely, so I don't end up wasting ingredients I've already bought. The app will find recipes that use ingredients already in our home, saving money for my family. Even if I don't have every necessary ingredient, ChefPal will provide recipes that require the least amount of additional ingredients to make a fulfilling dish. The amazing fact that I don't need to buy new ingredients can help me when practicing how to cook well with the food in my fridge that I already have and not waste extra money on grocery shopping. ChefPal will be the best companion to have that will help me provide healthy and delicious meals for my family.

Soup Kitchen Managers

I am the manager of a local soup kitchen in my town and it is my responsibility to make sure I can provide food to the low income citizens in my community. We get ingredients from local grocery stores that are at the lowest price and also ingredients that we receive from donations. Sale items vary week to week based on nearby grocery stores so it is always a challenge of figuring out what to make for our less fortunate consumers. We also get our stock in bulk so when our volunteers use recipes, they need to find ones that will have serving sizes in large quantities for our large low income population. I have been a manager for many years here and have formed a bond with our usuals who need food. I believe that everyone has the right to eat tasty, healthy food, regardless of their economic status. I want to uplift our community by providing them with a variety of food options to ease their existing money related hardships.

ChefPal would be very useful for me as a manager to keep track of what groceries we already have in the soup kitchen because of how the app saves all the ingredients. The fact that this app can also show a recipe's measurement information and serving sizes will really help us tailor our cooking based on the amount of people we are trying to serve. Plus, we can create more meals in bulk for our community. The feature in ChefPal that provides information on the nearest grocery stores would be a great help in finding any missing ingredients quickly, since we can filter grocery stores from closest to furthest. As a soup kitchen and non-profit organization, it is really important we save money, so the purpose of using only the ingredients we have on us will help us financially. Since this app shows us a variety of recipes using the same ingredients we currently have, we will be able to meet our goal of providing people in need with a variety of delicious foods. I can't wait to use ChefPal because it will make a significant positive impact in my lower income community.

Last-Minute Party Planners

I love throwing impromptu parties for my friends and family but figuring out what to make can be tough. I do these so often that I don't want to end up repeating recipes and boring my invitees. Another problem I also have is when we decide to throw surprise parties after work, I don't have time to go out grocery shopping after work in time for me to be able to gather the ingredients that I need for the parties, so I have to figure out what to make pronto. It's never assured what could be in my fridge and sometimes I don't know what I could possibly make with the random ingredients I do have. When I am stumped on what to make, I end up having to buy takeout. This poses a lot of problems in their own way. Not only is it expensive, but I find it embarrassing because I am passionate about cooking and don't want to serve my party attendees food they could just go out and buy themselves.

How could ChefPal help me? ChefPal would be very helpful for me when I have to throw a party coming home right from work because it would provide recipes that only utilize ingredients that I already have. The variety of recipes ChefPal can offer with the same ingredients can also help me serve a variety of dishes to keep my parties lively. Internet recipes are so scattered and disorganized but this app has optional filters to even show recipes for the type of dish I am looking for (ex: snack, meal, dessert etc.). ChefPal would be a lifesaver and I can't wait to flex to my friends and family about how resourceful and tasty my cooking can be.

Fast-Paced Lifestyle Workers

Due to the job that I have, I lead a very busy lifestyle. From the second I wake up, my day is a complete blur. Every moment I spend not doing work is a moment wasted, so I don't have any time to lose. I barely have any free time, and by the time I come home from work I am completely exhausted. Most of the time, I just end up buying a hot dog or a burger from a fast food restaurant to sustain myself throughout the day, but in the long term, not only am I spending more money on buying food than I want to, it is also very unhealthy.

Putting the time into searching through countless recipe databases online is an enormous waste that ChefPal eliminates by giving me clear and concise instructions for recipes I can cook and what are the nearest grocery stores near me that sell all of the missing ingredients. In addition, since everything is online these days, this app would be much more useful for me to choose recipes I am interested in during my daily train ride to work with the "Save Recipe" feature it has, as well as being free on the app store unlike if I had to go buy a recipe book. With all the responsibilities I have related to work, this saves me time that could have been wasted

trying to look for a feasible recipe once I am back from work. ChefPal takes the guessing and searching out of planning my dinners and gives me more time to enjoy the cooking process.

b. Decomposition into Sub-problems

- **Problem 1**: Meal planning is a challenging issue for college students both financially and time-wise
    - University-provided meal plans are often low-value (high price per meal).
    - Purchasing new ingredients in bulk is expensive.
    - Students don't have enough time to regularly buy new ingredients and plan for different recipes
- **Problem 2:** Logistics of grocery shopping planning while balancing other responsibilities.
    - Time management can be very hard, especially for people with a busy work schedule, such as a stock broker, or ER doctors/nurses. Their work doesn't give them enough time to plan out their meals and physically purchase the necessary groceries for the few recipes they do know how to make.
- **Problem 3:** It is a tedious process to find recipes with one's matching ingredients.
    - Recipe books are often wordy and can cater to a particular cuisine or have specific ingredients required, which makes them not an optimal choice for everyday cooking, which may call for a variety of things.
    - Internet recipe resources are scattered, unorganized, and filled with pop-ups and advertisements.
    - The cost of buying physical copies of recipe books is high.
    - Trying to find recipes that include only the ingredients that you currently have, if not the majority of ingredients you currently have, is very hard to find manually.

c. Glossary of Terms

- **Amazon Aurora**- Amazon Aurora is a MySQL and PostgreSQL-compatible relational database built for the cloud, that combines the performance and availability of traditional enterprise databases with the simplicity and cost-effectiveness of open source databases.
- **Amazon Cognito** - lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Apple, Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0 and OpenID Connect
- **Amazon DynamoDB** - key-value and document database that delivers single-digit millisecond performance at any scale
- **AWS Amplify** - set of tools and services that can be used together or on their own, to help front-end web and mobile developers build scalable full stack applications, powered by AWS
- **AWS AppSync**- combines data from databases, APIs, and other backend systems offered by Amazon into a single GraphQL endpoint
- **Bugify Issue Tracker -** a simple PHP issue tracker, designed to offer powerful bug tracking capabilities in an easy to use system.
- **Cuisine** - type or style of cooking and food, particularly relevant to a particular region or country.
- **Dietary restrictions -** constraints on a users food intake due to allergies/diabetes, religious reasons, or preference.
- **Figma** - a graphic editing tool to help design the user interface of our mobile application. Is compatible for Android and IOS.
- **Flutter** - Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop from a single codebase
- **Google Places API** - service that returns information about places using HTTP requests. Places are defined within this API as establishments, geographic locations, or prominent points of interest.
- **Ingredients** - any food, spice, or substance that is used and combined in order to form a certain dish.
- **Recipe** - a set of instructions on how to create a certain dish; usually includes a list of ingredients required for the dish.
- **Spoonacular API** - an API that allows you to access a vast collection of recipes and their attributes, such as their ingredients and type of cuisine to allow recipe searching and filtering specifications defined by the developer implementing the API.

2) System Requirements: Goals, Requirements, and Analysis

a. Business Goals

Goal #1: Allow users to select specific dietary restrictions, preferences, and their current ingredients, overall making the recipe search unique to the user's needs.

Sub-goals:

1. Allowing the user to find recipes based on the ingredients the user currently has.
2. Allowing the user to save their dietary and allergen restrictions for long term use. Also offering filters like cuisine, type of meal, and preparation time for each individual search.
3. Saving the user's dietary restrictions and preferred location radius so they don't have to keep reentering the same information on every search. These settings can be changed in the user's profile.

Goal #2: Being able to offer a variety of recipes based off of Goal #1 (the user's restrictions and ingredients they have).

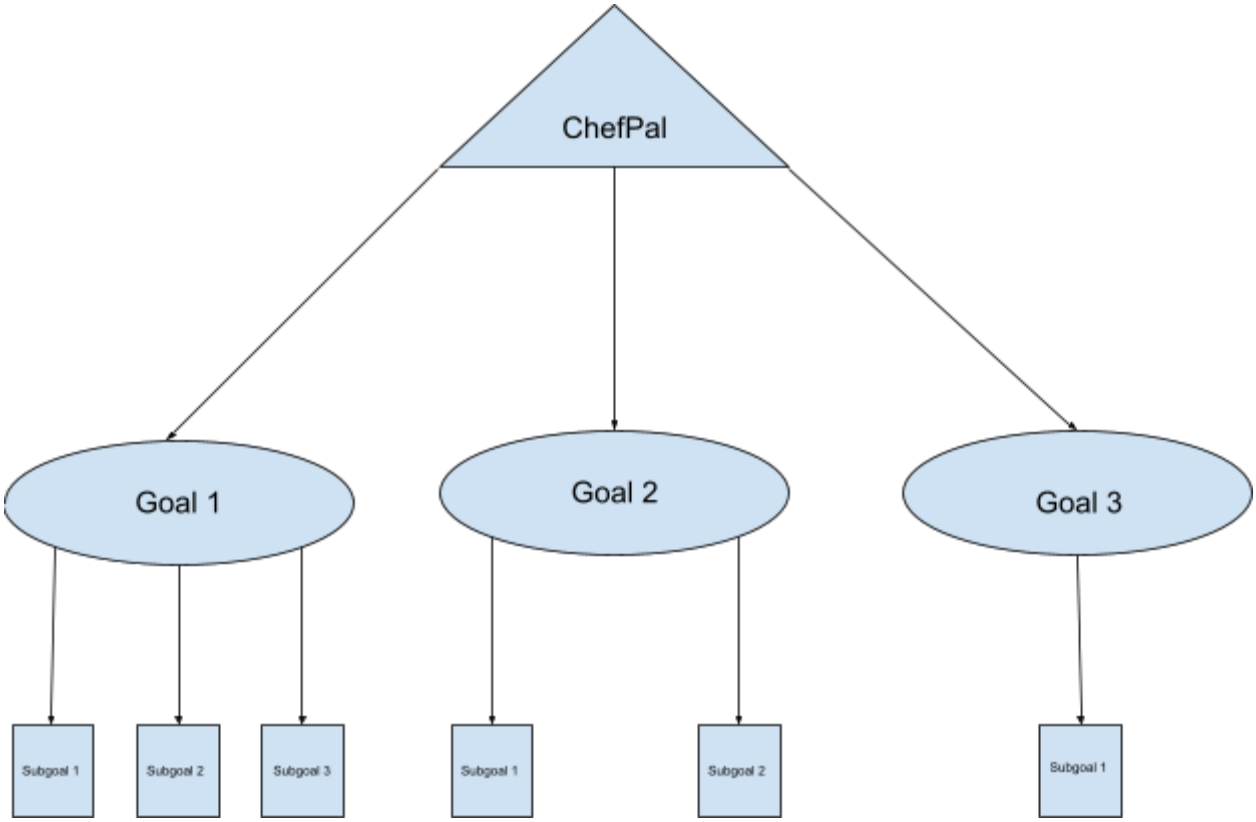
Sub-goals:

1. Allowing a user to save their favorite recipes to a "Likes" tab which is essentially a library of all the user's recipes that they have saved for future reference.
2. Notifying the user if there are missing ingredients that he or she needs to create a certain recipe.

Goal #3: Based on the user's missing ingredients for a recipe, offer grocery stores which can supply the missing ingredients.

Sub-goal:

1. Providing an address of the stores that have the missing ingredients available. The stores listed will be dependent on the user's current location.



b. Enumerated Functional Requirements

Identifier:	Priority:	Requirement:
REQ-1	5	As a user, I should be able to login using my unique username and password.
REQ-2	4	I should then be able to check off my dietary restrictions (vegetarian, vegan, keto, etc.) to save towards my profile.
REQ-3	3	As a user, I should be able to set a distance preference for grocery stores.
REQ-4	5	As a user, I should also be able to input my current available ingredients.
REQ-5	5	I should be able to see a list of recipes, prioritizing the ones that require the least amount of missing ingredients. IE: starting with recipes that have zero extra missing ingredients, so they use all the ingredients I have in my possession.

REQ-6	4	I should be able to view recipes that are within my dietary restrictions that I have previously specified in the profile setup.
REQ-7	3	As a user, I should now be able to filter recipes based on cuisine.
REQ-8	3	I should next have the ability to filter recipes based on type of meal (breakfast, dinner, snack, etc.).
REQ-9	3	I should be able to filter recipes based on preparation time.
REQ-10	3	I should be able to filter out recipes based on calorie and macronutrient count.
REQ-11	4	Now that I've seen a list of options, I should be able to save any recipe into a different tab organized into customized lists.
REQ-12	5	I get to see a display of all the recipe's required ingredients, preparation time, serving size, calorie count, and list of steps to make the meal upon clicking on a recipe.
REQ-13	5	When I click on a missing ingredient in a recipe I'm viewing, I should be redirected to a generated list of grocery stores that sells the missing ingredient
REQ-14	3	I should be able to filter grocery stores based on proximity.
REQ-15	5	I should have access to my personal profile where I can change login information, dietary preferences, and my distance preference for grocery stores.
REQ-16	2	If there are any bugs or issues, I should be able to report them through a "user report" feature.
REQ-17	3	I should be able to search all the recipes in the database based on a keyword I enter.

c. Enumerated Nonfunctional Requirements

Identifier:	Priority:	Requirement:
REQ-18	4	The application should be compatible with IOS and Android OS.
REQ-19	4	The application should be able to handle 500 daily users and can be scaled as needed.
REQ-20	5	The application should keep user information private by securely storing passwords and other sensitive information.
REQ-21	4	As a system, all stored and collected data should be secure.
REQ-22	3	The application layout should be aesthetically appealing and minimalistic with clutter free UI.
REQ-23	4	The application should operate at 60Hz or120 Hz for supported devices.
REQ-24	3	The application should be easy and simple to download and navigate for nearly any person who has access to a smartphone and internet.
REQ-25	3	As a system, user requests and issues should be checked daily and addressed as soon as possible.
REQ-26	3	As a system, unit testing should be done after any changes to the system and before any version updates releases.

d. User Interface Requirements

Identifier:	Priority:	Requirement:	Graphic:
REQ-27	5	The sign up page for a new user.	
REQ-28	5	The login screen for a returning user.	

REQ-29	4	If the user is new, then there is a list of dietary preferences to check off from.	
REQ-30	3	If the user is new, then there is a preferred distance range for grocery stores to select.	

REQ-31	5	A list of available ingredients, which are categorized by dairy, vegetables, meat, etc., to check off from in the "Basket" tab.	
REQ-32	5	A list of recipe titles you can scroll through in the "Search" tab. There are filters included at the top, such as cuisines, calorie and macronutrient count, preparation time, and type of meal (breakfast, snack, lunch, dinner, dessert, etc). A given recipe that a user clicks on will have a heart next to it as to save the recipe.	

REQ-33	5	When a user clicks on a recipe to view, the recipe includes its required ingredients, preparation time, serving size, calorie count, and numerated steps to make the meal.	
REQ-34	4	If there is an ingredient in a given recipe that a user doesn't own, it will be noticeably clickable and will redirect to grocery store locations that offer that ingredient.	

REQ-35	3	Generated list of grocery stores with filters at the top, such as price and proximity, appear when the missing ingredient is clicked.	
REQ-36	2	"Profile" tab is where a user can change login information, dietary preferences, and distance preferences for grocery stores.	

REQ-37	2	“My Likes” tab where the liked recipes are saved in an “All Likes” list. The user’s customized lists also appear here with an option to create more customized lists.	 

3) Use Cases

Use Cases are granular requirements with high specificity that are created by UX designers in tandem with client feedback to gauge what the end user wants from the software and accommodate their specific needs. The overarching goal of use cases is to create a set of specific goals that the end client wants to achieve and communicate them concisely and effectively to the designer. Identified below are the following two groups who contribute to the creation of use cases - stakeholders and actors. Stakeholders refer to any person or legal entity who has a vested interest in the success of the application. Actors are any people who interact with our application, both from the supply and demand side.

a. Stakeholders

Stakeholders are people, either singular or an organization, that have an interest in our system. A good way to organize these stakeholders is to categorize them into primary and secondary stakeholders. For our system, our primary stakeholders are college students, home cooks, fast-paced lifestyle people, and soup kitchen managers. They are all people who have a demand for quickly prepared home-cooked meals, and all face various external constraints. The college students and soup kitchen managers are largely limited by the products available to them as well as the cost of production. Home cooks and people with a fast-paced lifestyle, however, are more so limited by the time they have to go grocery shopping. Our secondary stakeholders are people who want to advertise their own recipes from their websites, companies that want to advertise their product, and advertising agencies who want to promote produce and grocery stores. Listing their produce and recipes on our application will greatly increase the consumer exposure to their goods. Internal stakeholders will include the project team and funders.

b. Actors and Goals

The actors involved in this application and their respective goals are broken down into two groups - initiating users and participating users. Initiating users are actively interacting with the application and take advantage of the entire spectrum of the application's functionality. Participating users are all the entities that allow the application to work and provide the back-end functionality.

i. Initiating Actors

Actors	Actor's Goals	Use Case
User	Sign up for the app and create a profile. Add information and dietary restrictions to their profile.	Registration

User	Login to app and view profile.	Login
User	Edit ingredients available. Through this tab, a user can select or deselect what ingredients he or she has.	Ingredient selection
User	While searching for recipes, users can filter to their likings and what they want to cook (sweet, savory, etc.).	Filtering ingredient generated recipes
User	Search for recipes on the entire database via a user inputted keyword	Recipe search
User	A user can save any recipe for future use.	Save recipes
User	A user can see a list of all saved recipes, as well as the customized lists that certain saved recipes can go into.	View saved recipes
User	A button next to a missing ingredient in a given recipe will allow the user to locate where they can purchase the recipe's required ingredient. It will redirect the user a list of grocery stores in the preferred radius that will sell the missing ingredient.	Grocery store locator
User	User can report any errors or problems they have faced throughout their process of selecting and finding recipes	Bug and error reporting
User	Users have access to this feature to change any profile information, location and dietary restriction info.	View/ Change Profile

ii. Participating Actors

Actors	Roles
Amazon Cognito	Will handle all user registration and authentication
Spoonacular API	Will take in all the users ingredients & filters and process their search.
Google Places API	Will be used to search for nearby grocery stores upon request by the user.
Bugify Issue Tracker	Will handle user bug reports.

c. Use Cases

i. Causal Description

1. **UC-1: Registration**

   Allow users to create a login with a unique username and password that is stored in the database.

2. **UC-2: Primary Preference Selection**

   The user will be prompted to enter in their primary preferences, such as any dietary preferences or restrictions, as well as enter their location and a radius that they will be willing to go grocery shopping in.

3. **UC-3: Login**

   The user can enter in their username and password, allowing the user to be logged into their own account with their own personalized app.

4. **UC-4: Ingredient Selection**

   The user will be able to select the ingredients they have by checking off boxes next to ingredients that are sorted into lists by food groups.

5. **UC-5: Generated Recipes**

   Based on the ingredient selection, the user is presented with a generated list of recipes. These recipes are based on the dietary preferences in the user's stored profile settings and the ingredients they have selected. The recipes that require the least number of extra missing ingredients appear first.

6. **UC-6: Filtering System for Generated Recipes**

   After seeing the generated recipes, users will have the option to further narrow down these recipes based on additional filters. These filters would include cuisine, meal preparation time, type of meal, and calories.

7. **UC-7: Recipe Search**

   The user will be able to search the entire recipe database, regardless if the user has the necessary ingredients to make that recipe. Once being able to access all of the recipes and search for any recipe they desire, they can still filter out their results by cuisine, meal prep, etc. However, these recipes will not be personalized to the user's ingredients, rather it is meant more for the user to explore the available recipes and see the selection at hand.

8. **UC-8: Save Recipes**

   After searching for recipes, the user will be able to add recipes that he or she likes to a custom list, which the user can create. The user will have a page of the app dedicated to recipe lists, which he or she has saved and will allow for more lists to be created.

9. **UC-9: View/Change Profile**

   The user will be able to change their login information, edit their dietary preferences and food intolerances. The user will also be able to alter their address in order to change the list of available grocery stores.

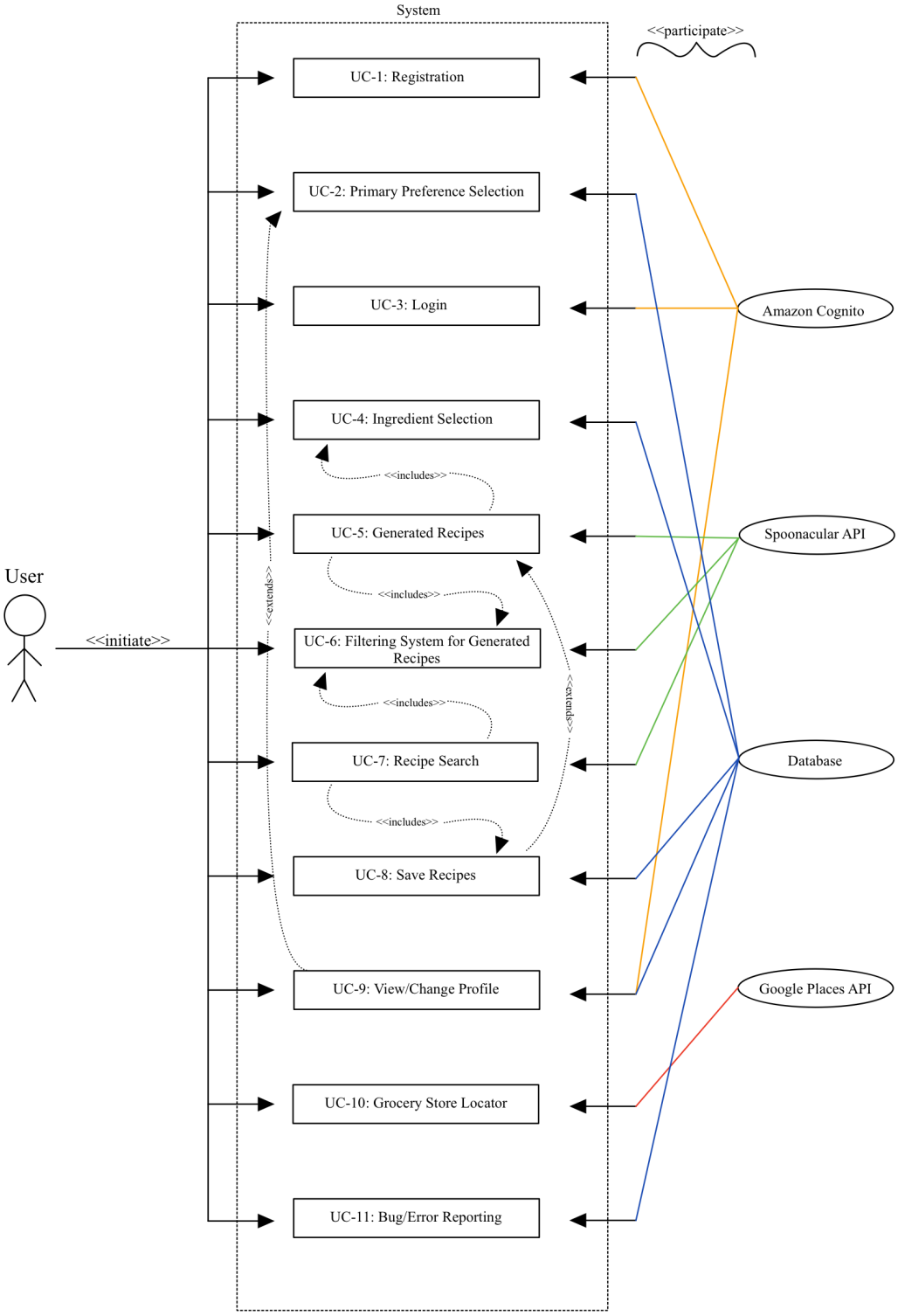
10. **UC-10: Grocery Store Locator**

    The user will be able to find stores nearby which should have the ingredients that the user is missing. This will allow the user to buy the necessary missing ingredients to make the recipe.

11. **UC-11: Bug/Error Reporting**

    The user will be able to report any bug or error that occurs. This way, the developers can be aware of any problems that the user is facing and help approach them via a hotfix.

ii. Use Case Diagram



iii. Traceability Matrix

The Traceability Matrix allows the reader to examine the requirements described in the Use Cases above. The matrix points to which use cases fulfill each requirement, and scales them to account for the priority of each use case.

Req't	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11
REQ-1	5	✔		✔								
REQ-2	4		✔							✔		
REQ-3	3	✔									✔	
REQ-4	5				✔							
REQ-5	5					✔						
REQ-6	4					✔		✔				
REQ-7	3						✔					
REQ-8	3						✔					
REQ-9	3						✔					
REQ-10	3						✔					
REQ-11	4								✔			
REQ-12	5							✔				
REQ-13	5										✔	
REQ-14	3										✔	

REQ-15	5									✔		
REQ-16	2											✔
REQ-17	3							✔				
REQ-18	4											
REQ-19	4											
REQ-20	5	✔		✔						✔		
REQ-21	4									✔		
REQ-22	3											
REQ-23	4											
REQ-24	3											
REQ-25	3											✔
REQ-26	3											
REQ-27	5	✔										
REQ-28	5			✔								
REQ-29	4		✔									
REQ-30	3	✔									✔	
REQ-31	5				✔							

REQ-32	5					✔	✔					
REQ-33	5							✔				
REQ-34	4										✔	
REQ-35	3										✔	
REQ-36	2									✔		
REQ-37	2								✔			
Total Weight		21	8	15	10	14	17	17	6	20	21	5

iv. Fully-Dressed Descriptions

UC-1: Registration
Related Requirement: REQ-1, REQ-3, REQ-20, REQ-27, REQ-30
Initiating Actor: User
Actor's Goal: To create a new account with a username and password
Participating Actors: Amazon Cognito
Pre-Conditions: This is the first time the user is using and operating the app
Post-Conditions: The user can now select their dietary preferences and their username and password will be saved to the database
Minimal Guarantees: The registration fails if the password doesn't meet the requirements, or if the user already has an account, or if another user has the same username.
Success Guarantees: The new user is stored into a database.
Flow of event for Success Scenario: ● → **User** selects the "Username" text box and enters in their preferred username. ● ← **Application** verifies that the username is unique. ● → **User** selects the "Password" text box and enters in their preferred password. ● ← **Application** navigates the user to personalized their home screen.
Flow of Events for Alternate Scenario: ● → **User** selects the "Username" text box and enters in their preferred username. ● ← **Application** finds the same username has already been used for another account. ● → **User** selects the "Password" text box and enters in their preferred password. ● ← **Application** alerts the user that the login credentials have already been taken and prompts them to pick a different username and password.

UC-2: Primary Preference Selection
Related Requirement: REQ-2, REQ-29
Initiating Actor: User
Actor's Goal: To add dietary information,restrictions, and distance preference to their profile
Participating Actors: Amazon Cognito
Pre-Conditions: The user has been registered in the system and log in into the system using his/her authentication credentials.
Post-Conditions: The user can now get recipe suggestion based on the preferences and grocery locations.
Minimal Guarantees: The registration fails if the password doesn't meet the requirements, or if the user already has an account, or if another user has the same username.
Success Guarantees: The user preferences are stored in the database and the user's profile
Flow of event for Success Scenario: ● → **User** checks the different boxes that align with their preferences ● → **User** inputs their distance preference ● ← **Application** stores the preferences into the user's information ● → **User** confirms all the selections ● ← **Application** personalizes the user's suggested recipe based on the selections
Flow of Events for Alternate Scenario: ● → **User** has no preference and skips the selections ● ← **Application** alerts the user that a minimum of one box must be checked

UC-3: Log in
Related Requirement: REQ-1, REQ-20, REQ-28
Initiating Actor: User
Actor's Goal: To regain access into their ChefPal account
Participating Actors: Amazon Cognito
Pre-Conditions: The user has been registered in the system and log in into the system using his/her authentication credentials.
Post-Conditions: The user can gain access back to their account and begin searching for recipes based off of ingredients available to them
Minimal Guarantees: The log-in fails if the password does not match the username, or if the entered username is not in the database.
Success Guarantees: The user is logged into the system and is able to interact with the app.
Flow of event for Success Scenario: ● → **User** selects "Username" text box and enters in their personal username. ● → **User** selects the "Password" text box and enters in their personal password associated with that username. ● ← **Application** validates user authentication credentials against stored hash and prompts the user to their home screen.
Flow of Events for Alternate Scenario (Login Error): ● → **User** selects "Username" text box and enters in their personal username. ● → **User** selects the "Password" text box and enters in their personal password associated with that username. ● ← **Application** alerts the user that their login credentials do not match records and prompts them to re-enter login and password.

UC-4: Ingredient Selection
Related Requirement: REQ-4, REQ-31
Initiating Actor: User
Participating Actors: Database
Actor's Goal: Having the user enter in the ingredients that they currently have in their possession
Pre-Conditions: The user has been registered in the system and log in into the system using his/her authentication credentials.
Post-Conditions: Entering any preference regarding their meal such as meal type or cuisine of meal that they would like
Minimal Guarantees: If no ingredients are selected, then no recipes will be generated.
Success Guarantees: User will be able to view available recipes based off of any of the selected ingredients
Flow of event for Success Scenario: ● → **User** selects available ingredients from the provided menu. ● ← **Application** will show that ingredient has been selected ● → **User** selects the "Search" button
Flow of Events for Alternate Scenario: ● → **User** selects ingredient ● ← **Application** does not successfully show the ingredient has been selected ● → **User** issues a bug report

UC-5: Generated Recipes
Related Requirement: REQ-5, REQ-6, REQ-32
Initiating Actor: User
Actor's Goal: To select and filter out recipes based on their preset dietary restrictions and current ingredients, as well as other filters
Pre-Conditions: User inputted their own preferences and dietary needs in order to be able to generate a list of recipes based on the selected ingredients
Post-Conditions: Recipes will be pulled pertaining to user's selected preferences and current ingredients on hand
Minimal Guarantees: N/A (If there are no matching recipes based on the user's input, the application will suggest some other recipes based on previous history)
Success Guarantees: Users will be able to view a list of generated recipes based on their dietary preferences and selected ingredients.
Flow of event for Success Scenario: ● → **User** searches with selected ingredients ● ← **Application** will return a list of recipes that contain the most ingredients in common with the selected ingredients, where recipes lower on the list contain less amounts of common ingredients.
Flow of Events for Alternate Scenario: ● → **User** searches with selected ingredients ● ← **Application** does not successfully display results ● → **User** issues a bug report

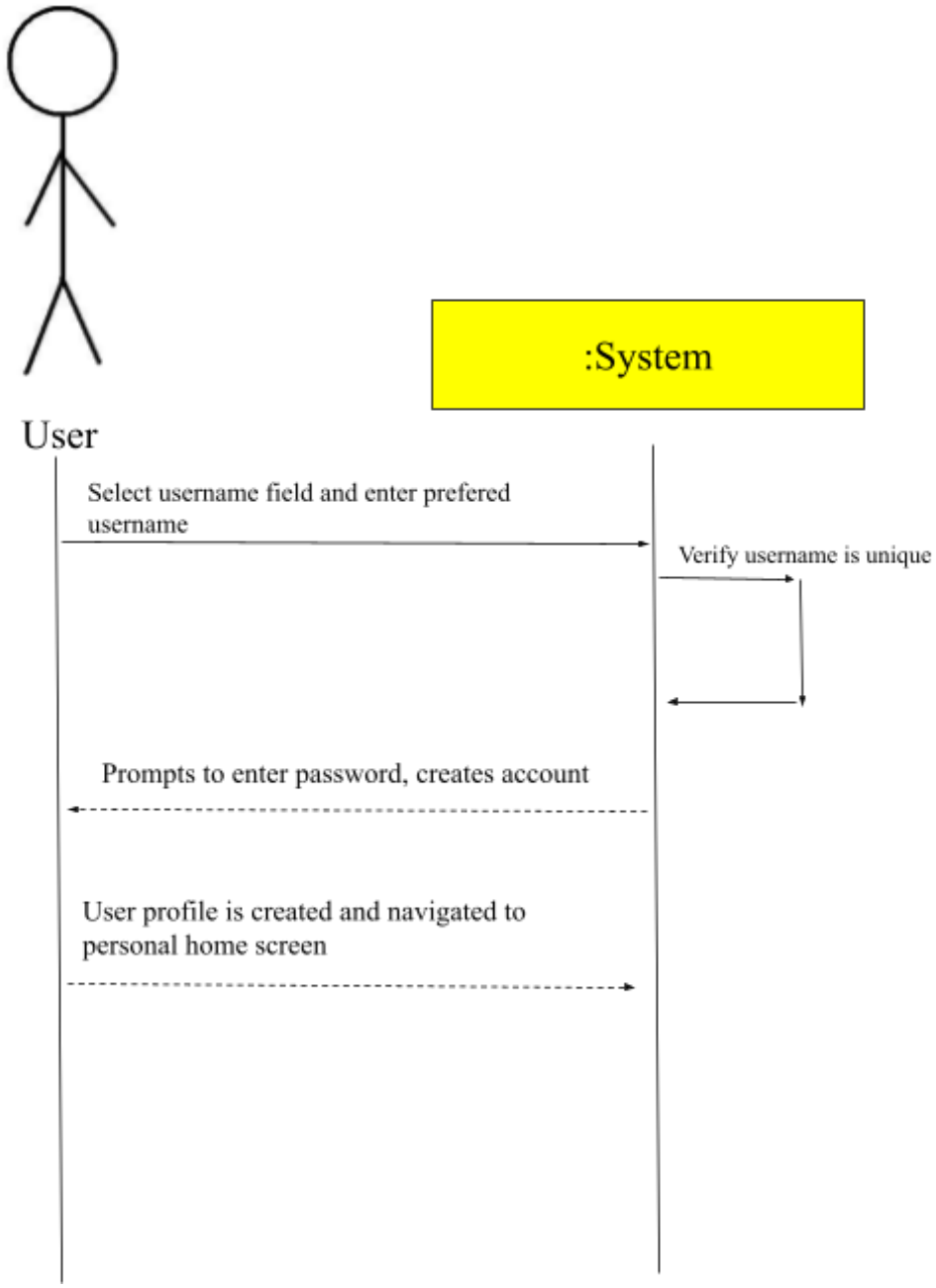
UC-6: Filtering Generated Recipes
Related Requirement: REQ-7, REQ-8, REQ-9, REQ-10, REQ-32
Initiating Actor: User
Actor's Goal: User will be able to filter through more categories, such as meal type, cuisine, prep time, calorie count, etc., to help generate more specific recipes.
Pre-Conditions: The application have generated recipes that correspond the user's ingredients
Post-Conditions: User gets to narrow down the results given by choosing what type of meal they want, which cuisine they prefer, whether they have a calorie count specifications, etc…
Minimal Guarantees: User will get generated recipes that do not include extra filtration.
Success Guarantees: User will get more filterized recipes based off of their selections.
Flow of event for Success Scenario: ● ← **Application** generated recipes based on users dietary restrictions and ingredients ● → **User** inputs preference filters such as type of meal, type of cuisine, prep time, calories ● ← **Application** will regenerate recipes based on the new filters
Flow of Events for Alternate Scenario: 1. Display Error a. → **User** filters with generated recipes b. ← **Application** does not successfully display results c. → **User** issues a bug report

UC-7: Recipe Search
Related Requirement: REQ-6, REQ-12, REQ-17, REQ-33
Initiating Actor: User
Actor's Goal: Allowing the user to look up any recipe based off of their search preferences Ex: User searches Shrimp Fried Rice
Pre-Conditions: User selects their dietary restriction and inputs available ingredients and preferences.
Post-Conditions: User will be able to see all available recipes for searched recipe
Minimal Guarantees: No recipes will output
Success Guarantees: User will be able to view recipes base on their search
Flow of event for Success Scenario: ● → **User** enters in a recipe to be searched with given keywords into a search bar ● ← **Application** successfully displays results ● → **User** scrolls through the result
Flow of Events for Alternate Scenario: ● → **User** searches with given keywords ● ← **Application** does not have any recipes related to that keyword search, or no matches occur ● ← **Application** does not not output any recipes

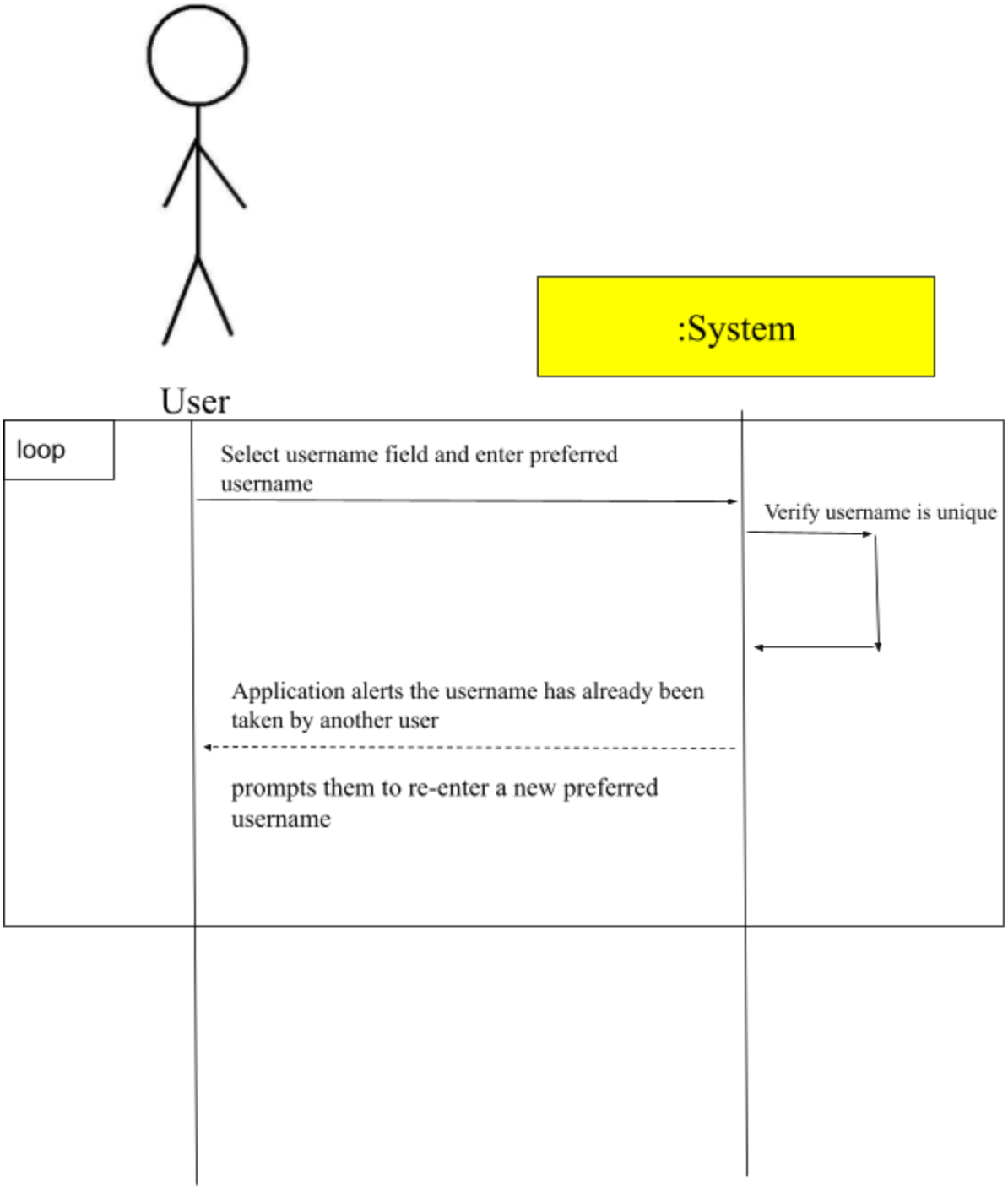
UC-10: Grocery Store Locator
Related Requirement: REQ-3, REQ-13, REQ-14, REQ-30, REQ-34, REQ-35
Initiating Actor: User
Actor's Goal: Locating a store that has the missing ingredients available to the user
Pre-Conditions: User has identified all of the ingredients that they currently have and given the fact that they have minimal ingredients, they need to go to the grocery store. User has also indicated their current location and the distance they are willing to travel for grocery shopping.
Post-Conditions: The application gives the user the name and address of the grocery store nearest to their location that has the necessary ingredients for the selected recipe.
Minimal Guarantees: No grocery store is available in the requested area.
Success Guarantees: The app will display nearby grocery stores with items that the user will need for their chosen recipe.
Flow of event for Success Scenario: ● → **User** selects a recipe ● → **User** presses find missing ingredients button if there are any missing ingredients ● ← **Application** successfully displays grocery store list. ● → **User** views the results
Flow → **User** selects a recipe ● → **User** presses find missing ingredients button if there are any missing ingredients ● ← **Application** prompts user to enable location services ● → **User** accepts the app's use of location services. ● ← **Application** successfully displays grocery store list. ● → **User** views the results

d. System Sequence Diagrams

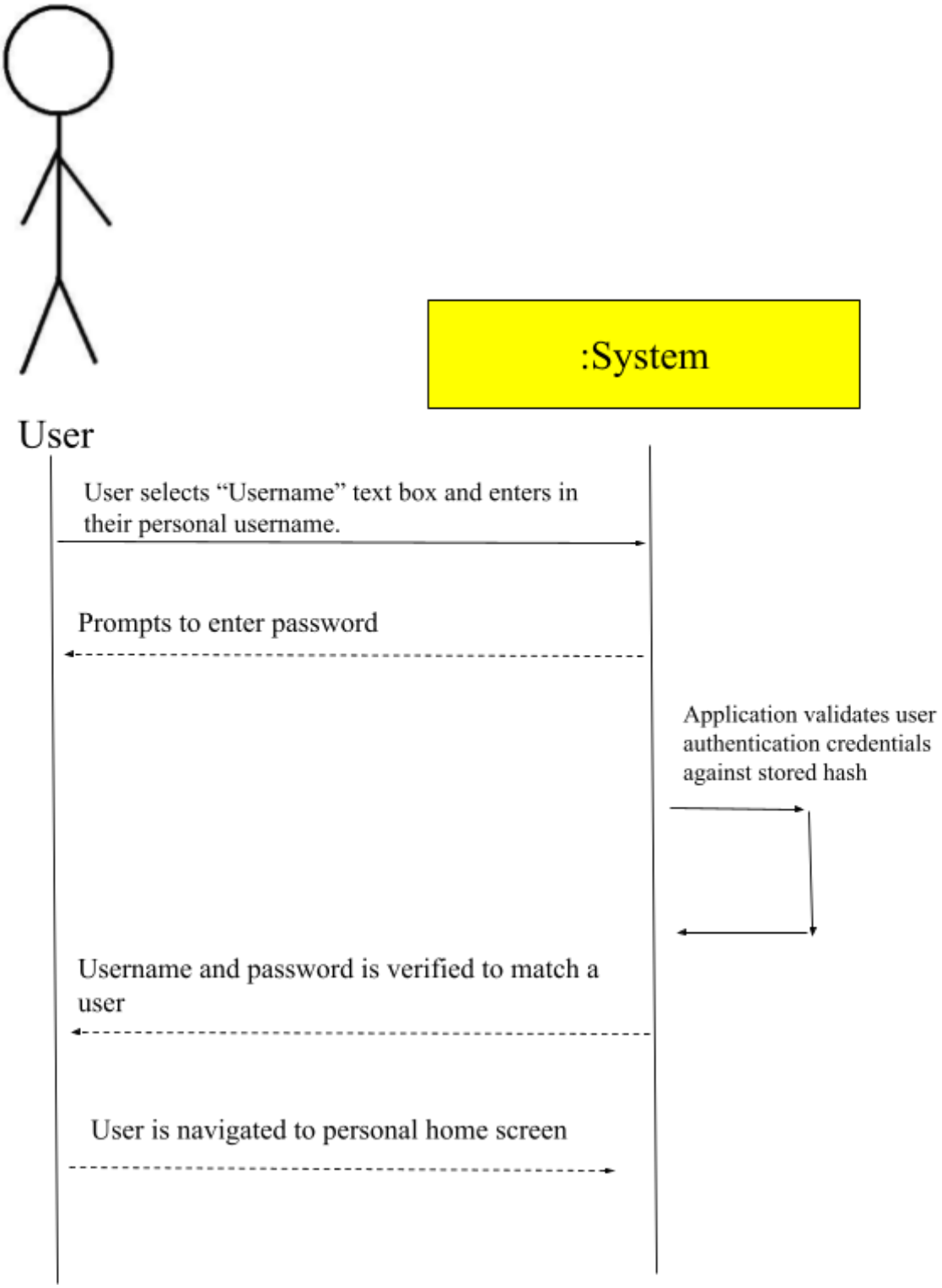
UC - 1 Sign- Up Main Success Scenario



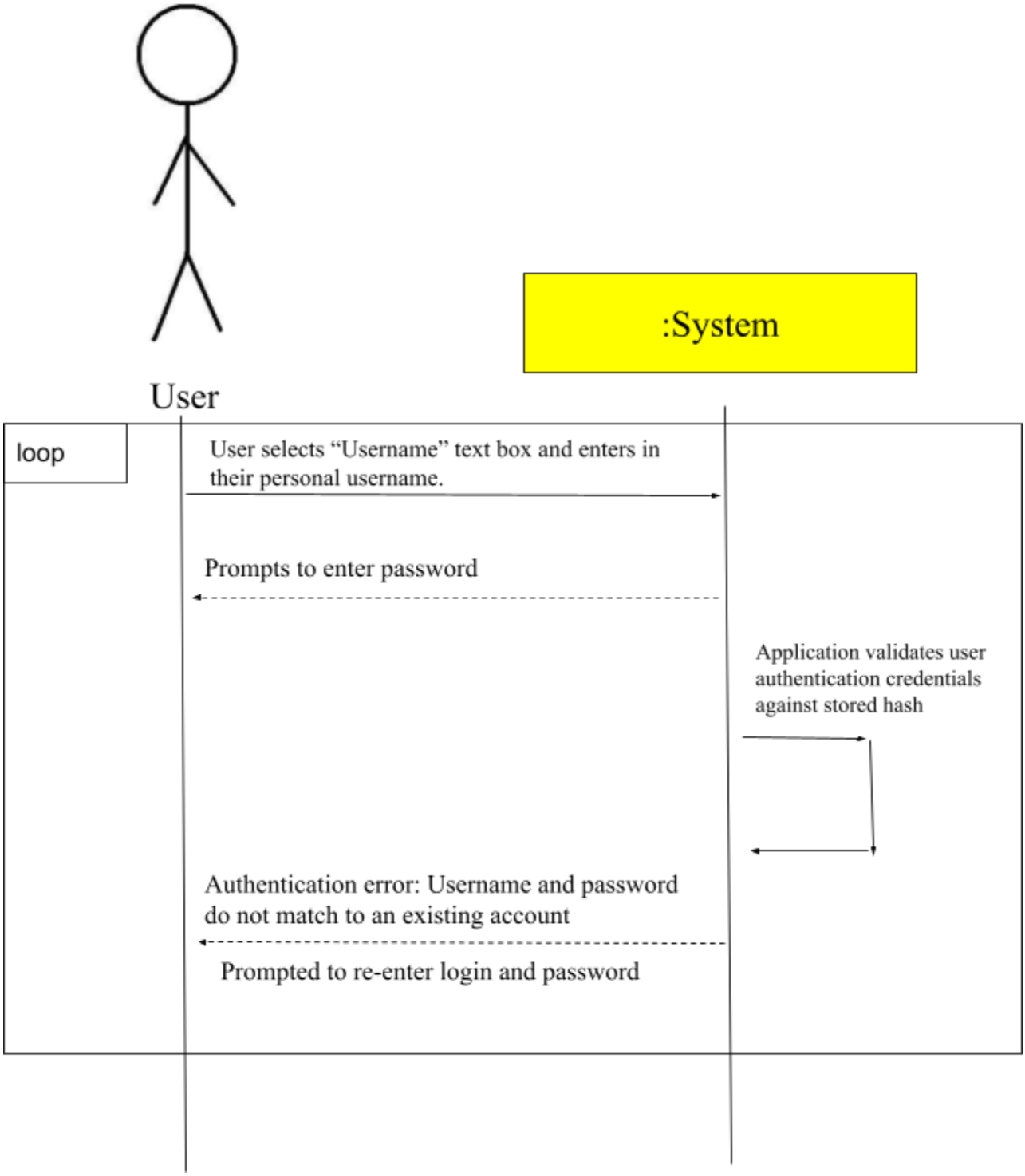
UC - 1 Sign-Up Alternate Success Scenario



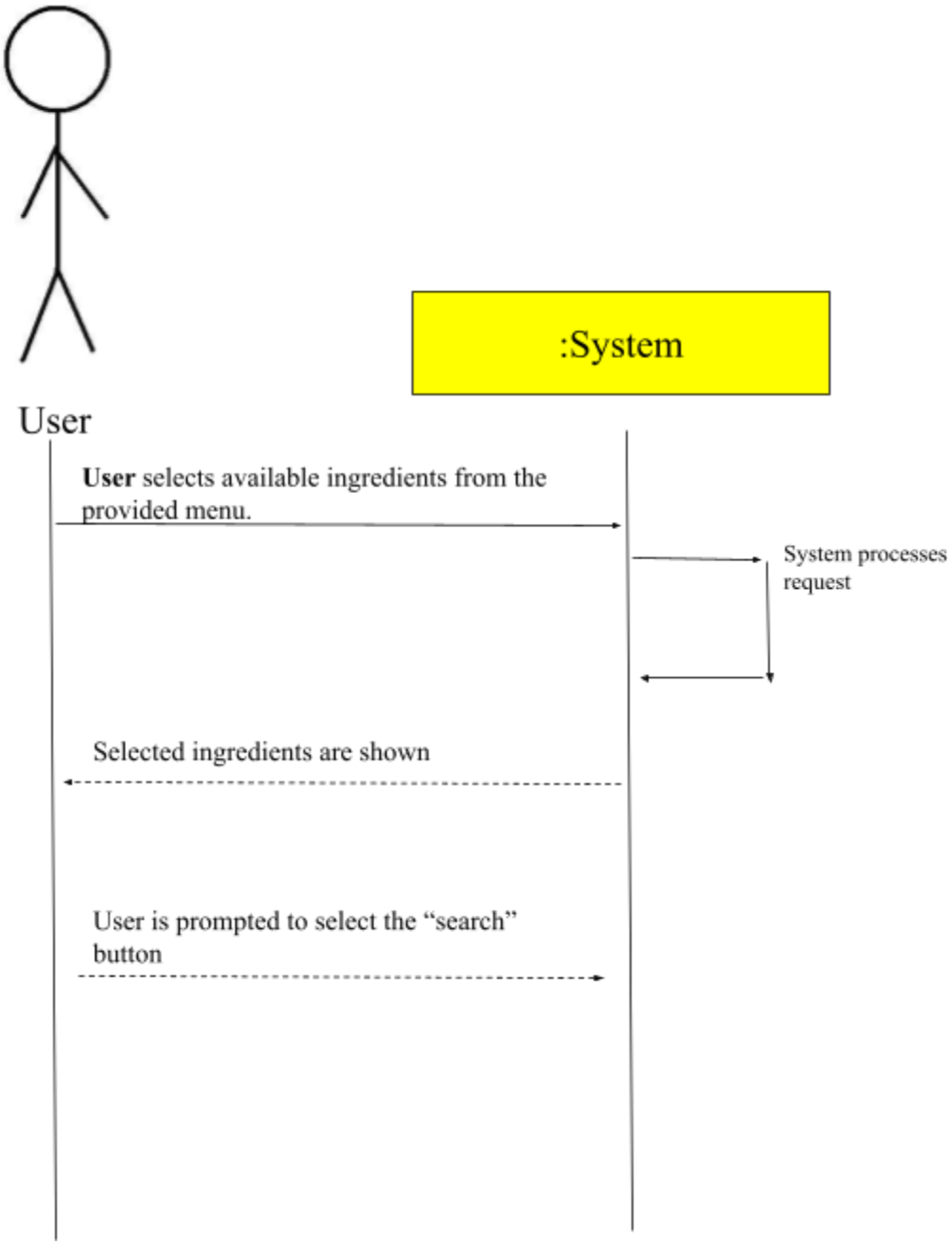
UC - 3 Login Main Success Scenario



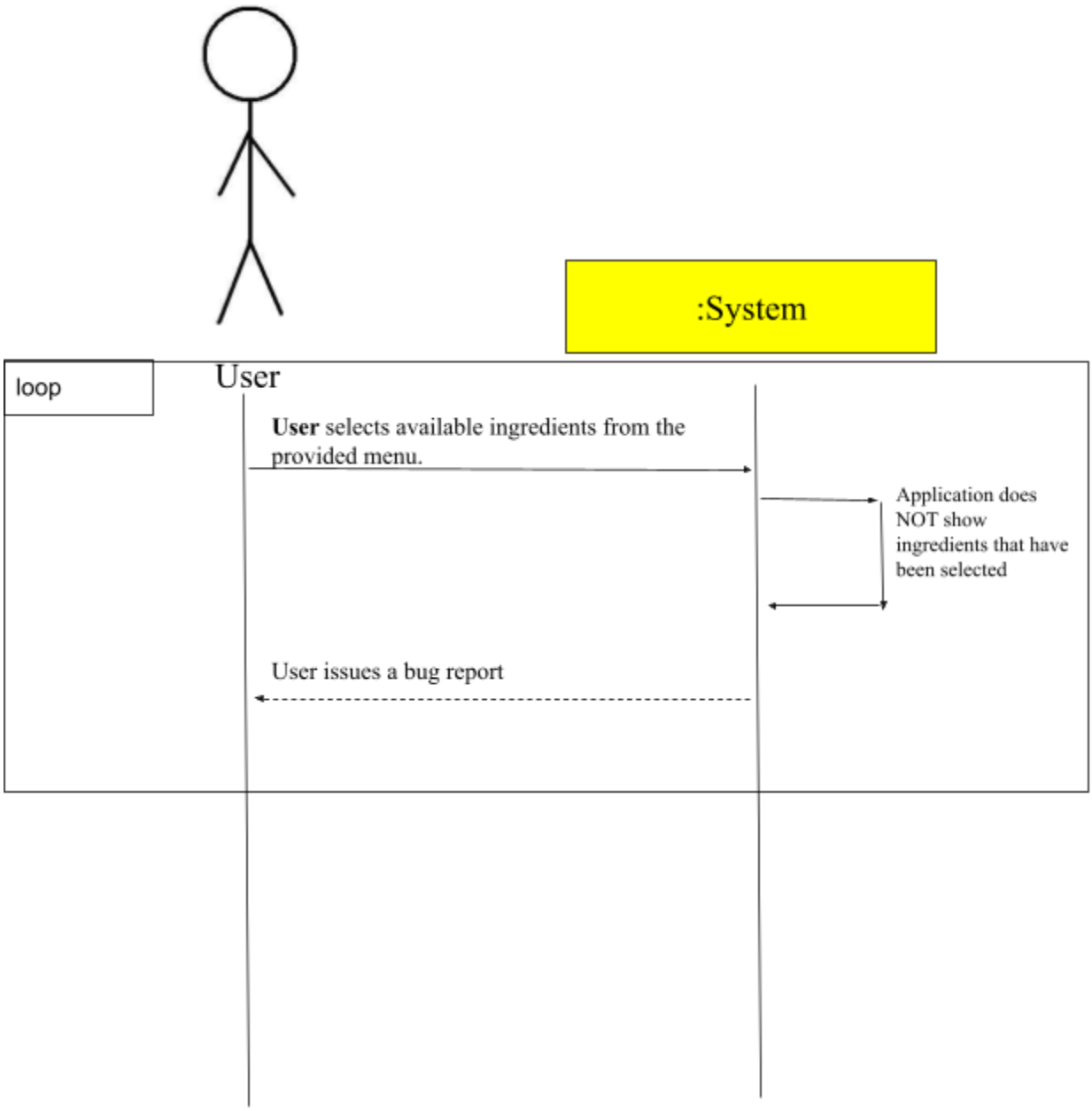
UC - 3 Login Alternate Success Scenario



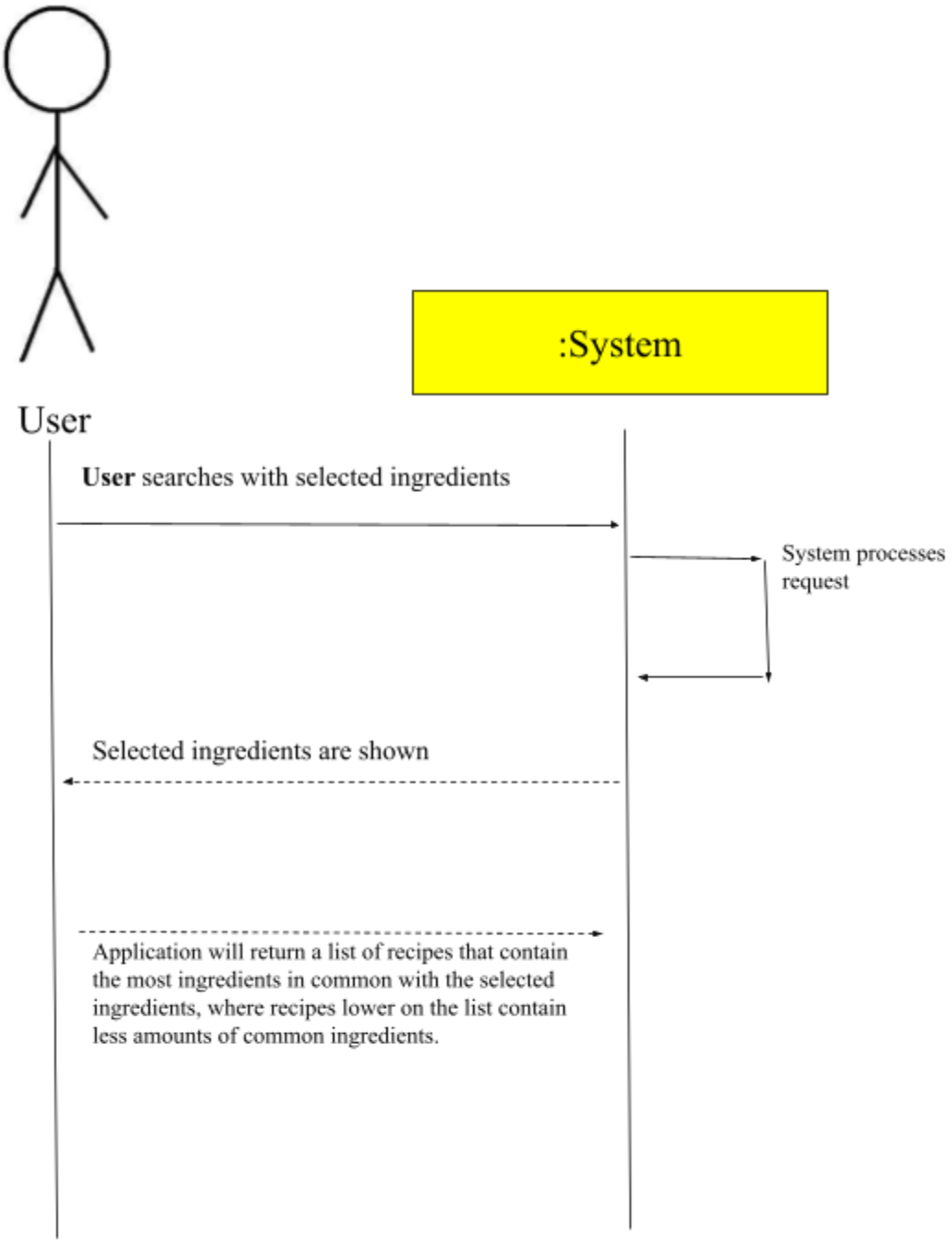
UC - 4 Ingredient Selection Main Success Scenario



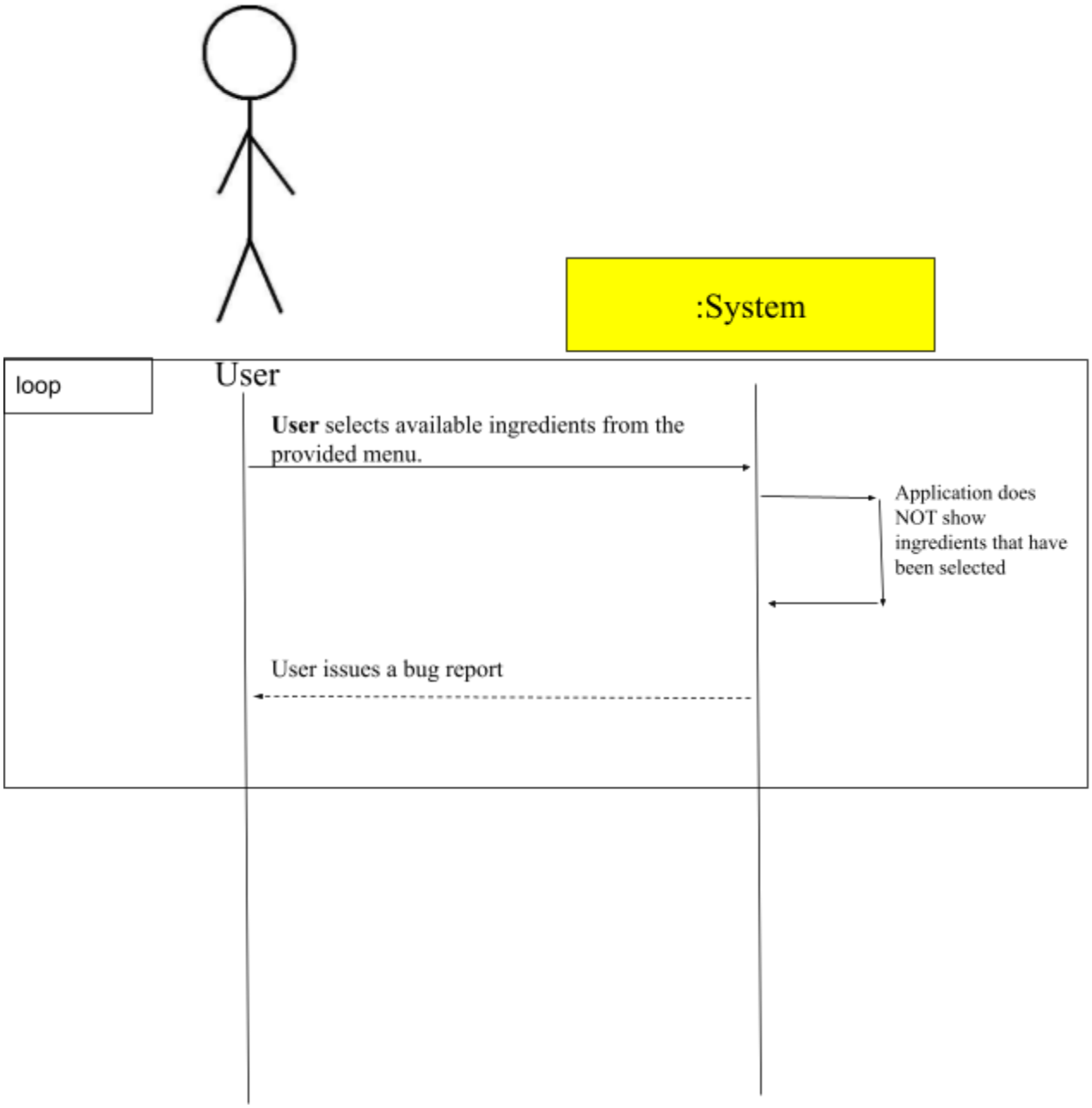
UC - 4 Ingredient Selection Alternate Success Scenario


:System
loop
User
User selects available ingredients from the provided menu.
Application does NOT show ingredients that have been selected
User issues a bug report

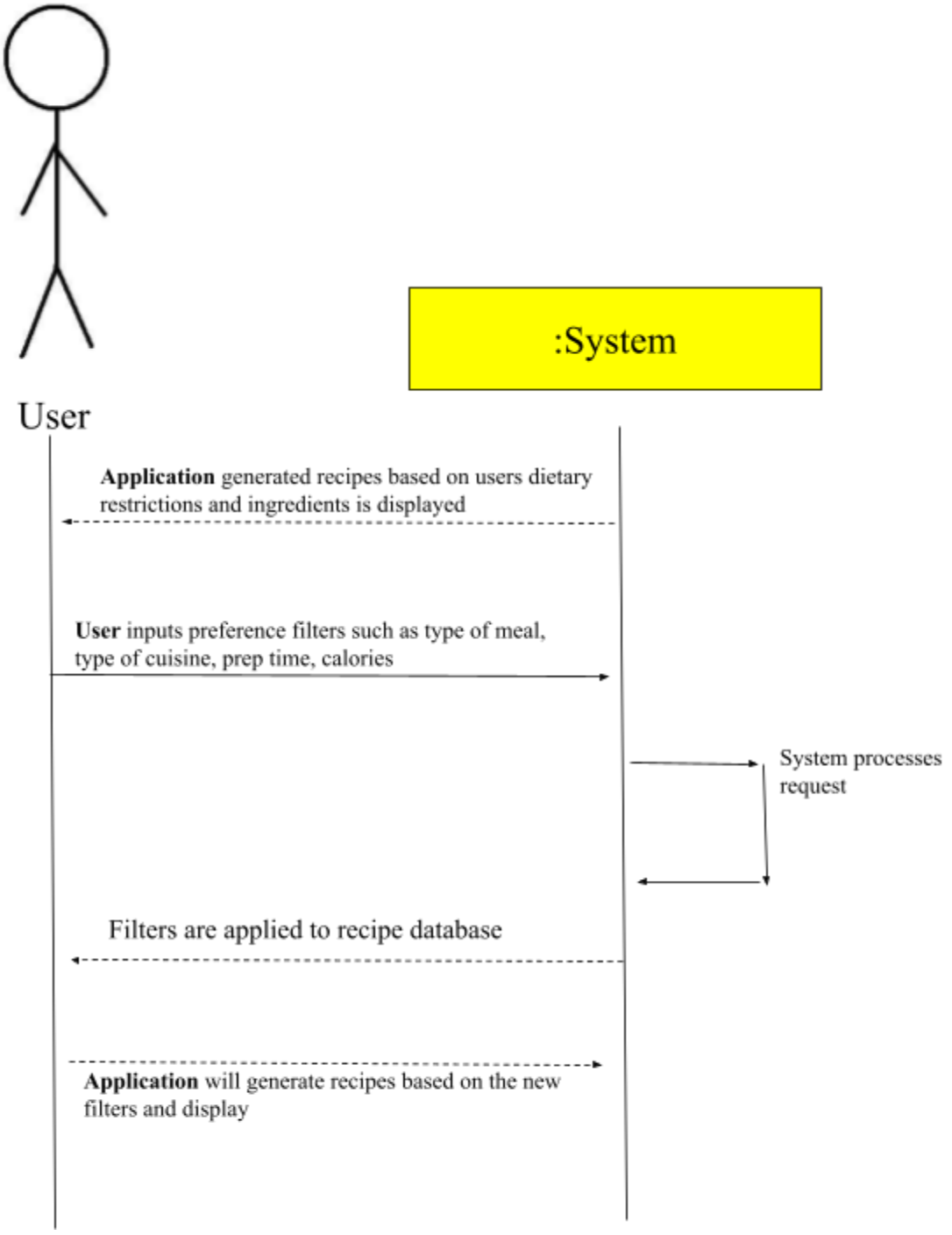
UC-5 Generated recipes Success Scenario



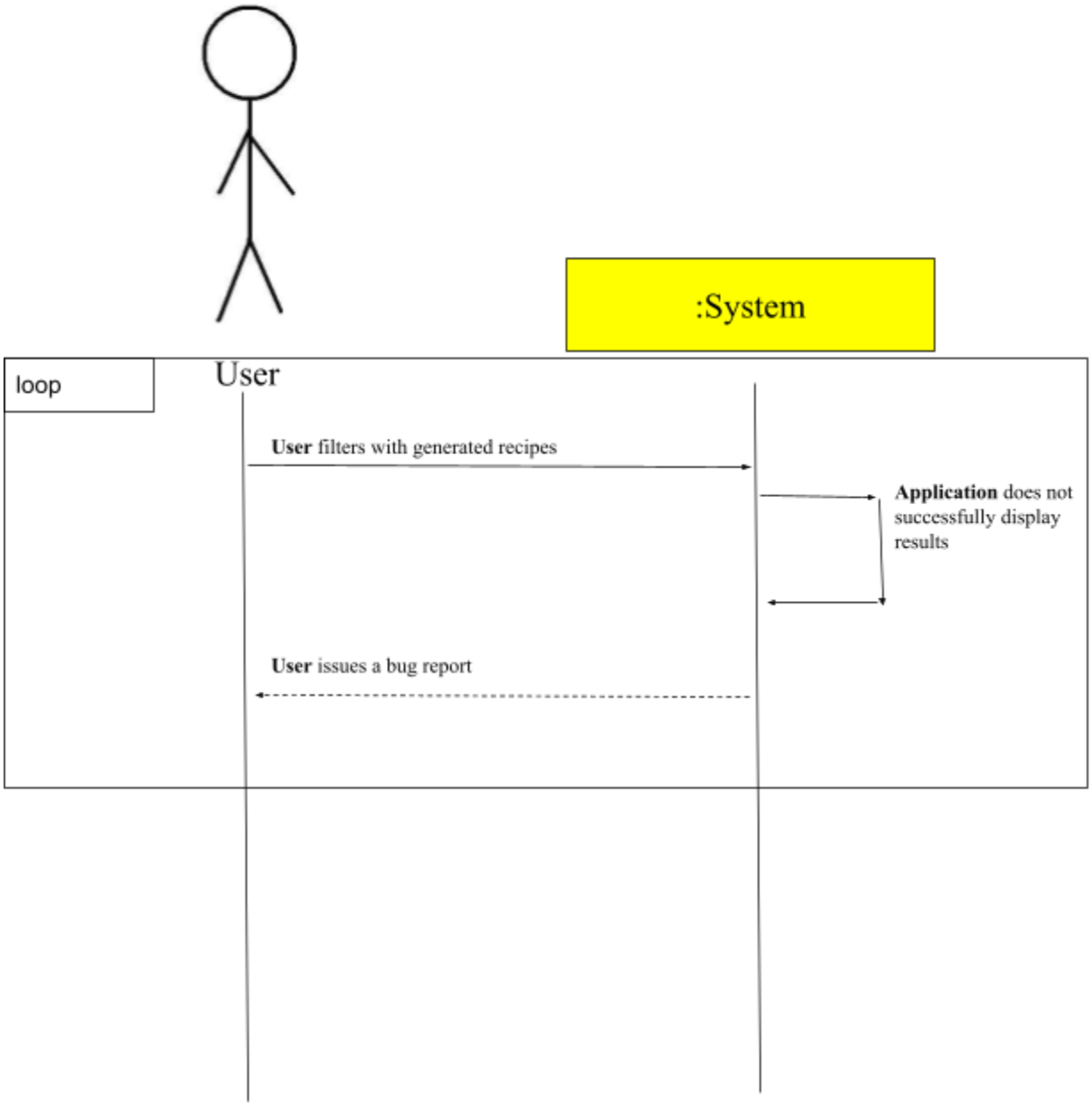
UC- 5 Generated recipes Alternate Success Scenarios



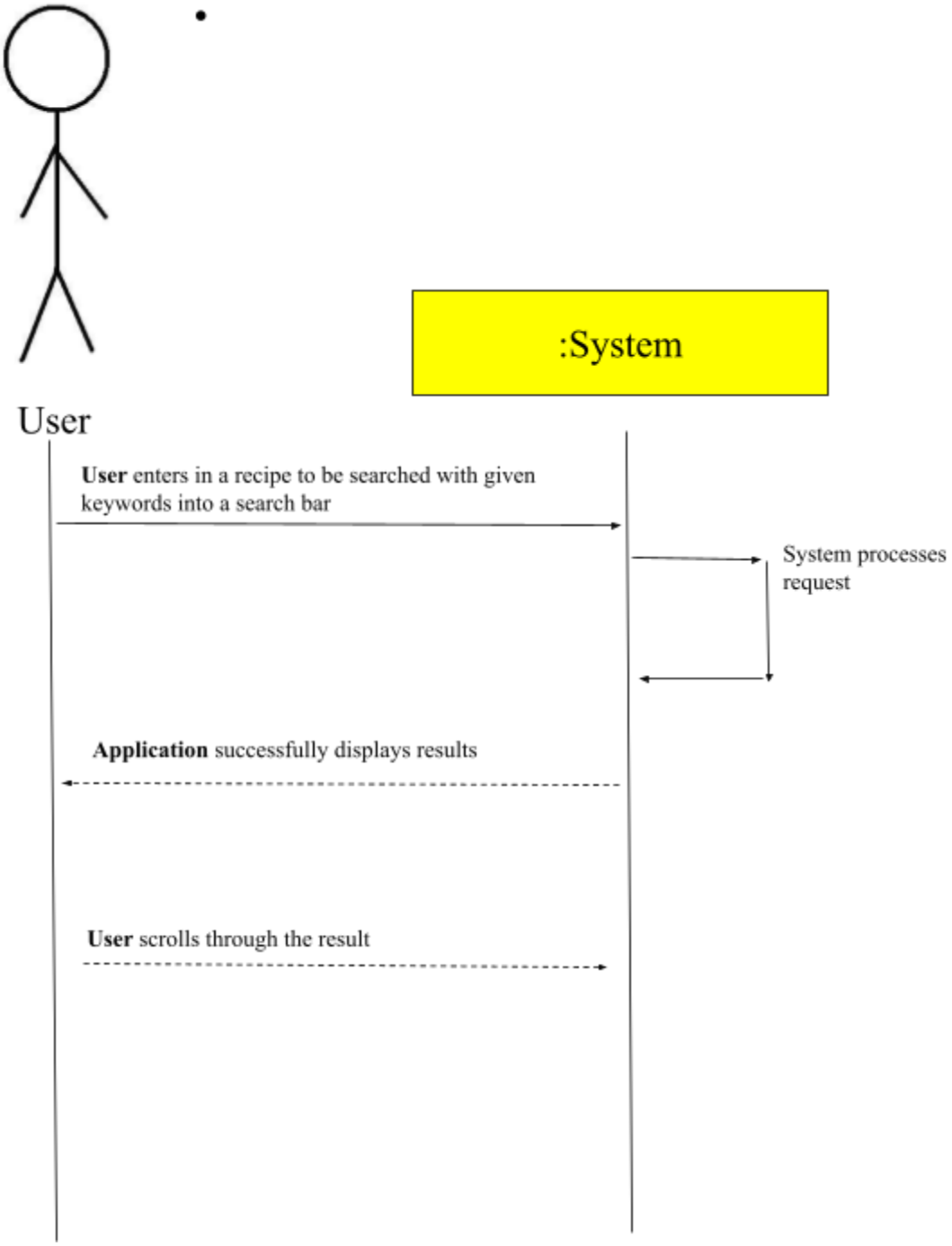
UC-6 Filtering Generated Recipes Success Scenario



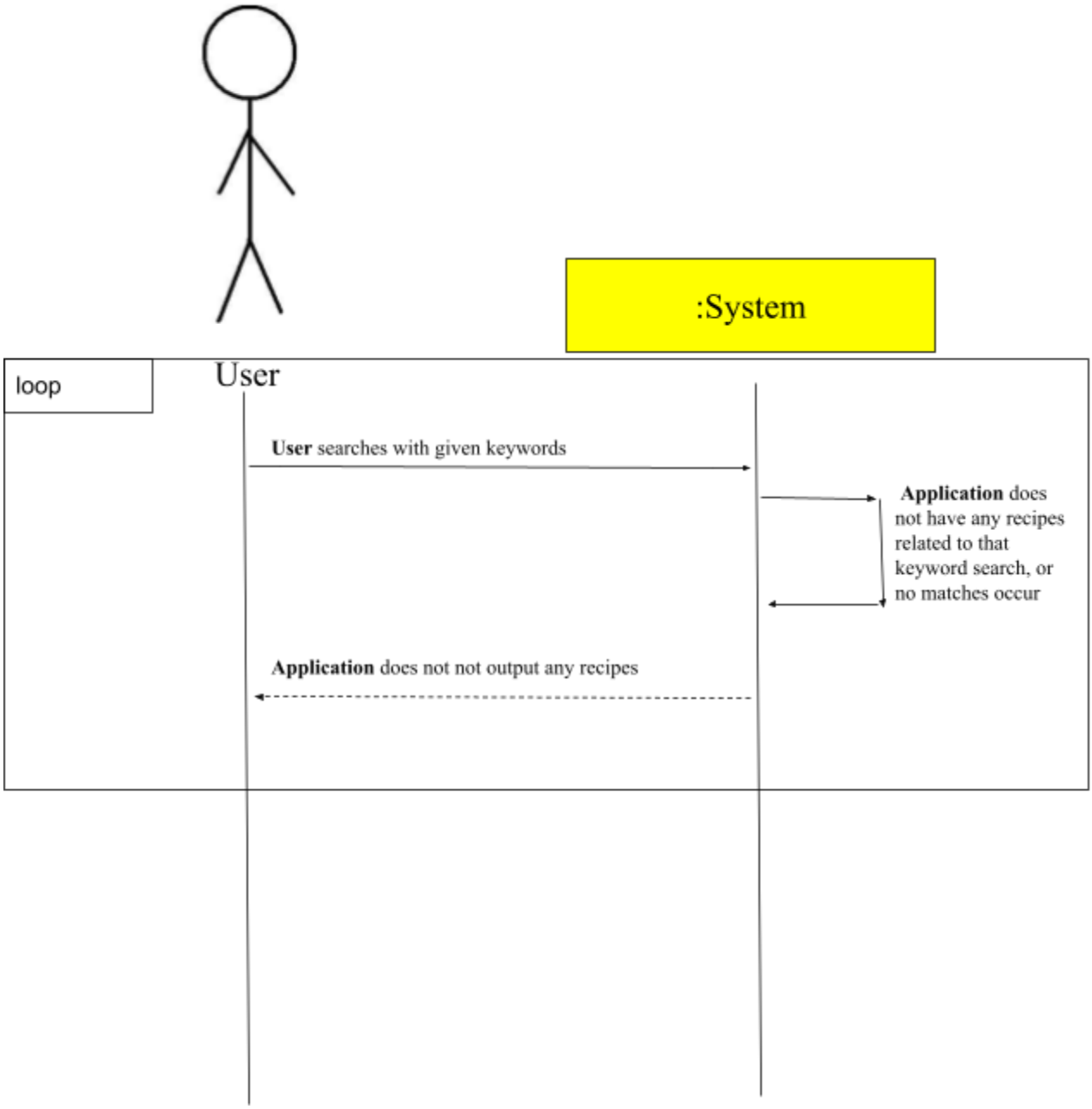
UC-6 Filtering Generated Recipes Alternate Success Scenario



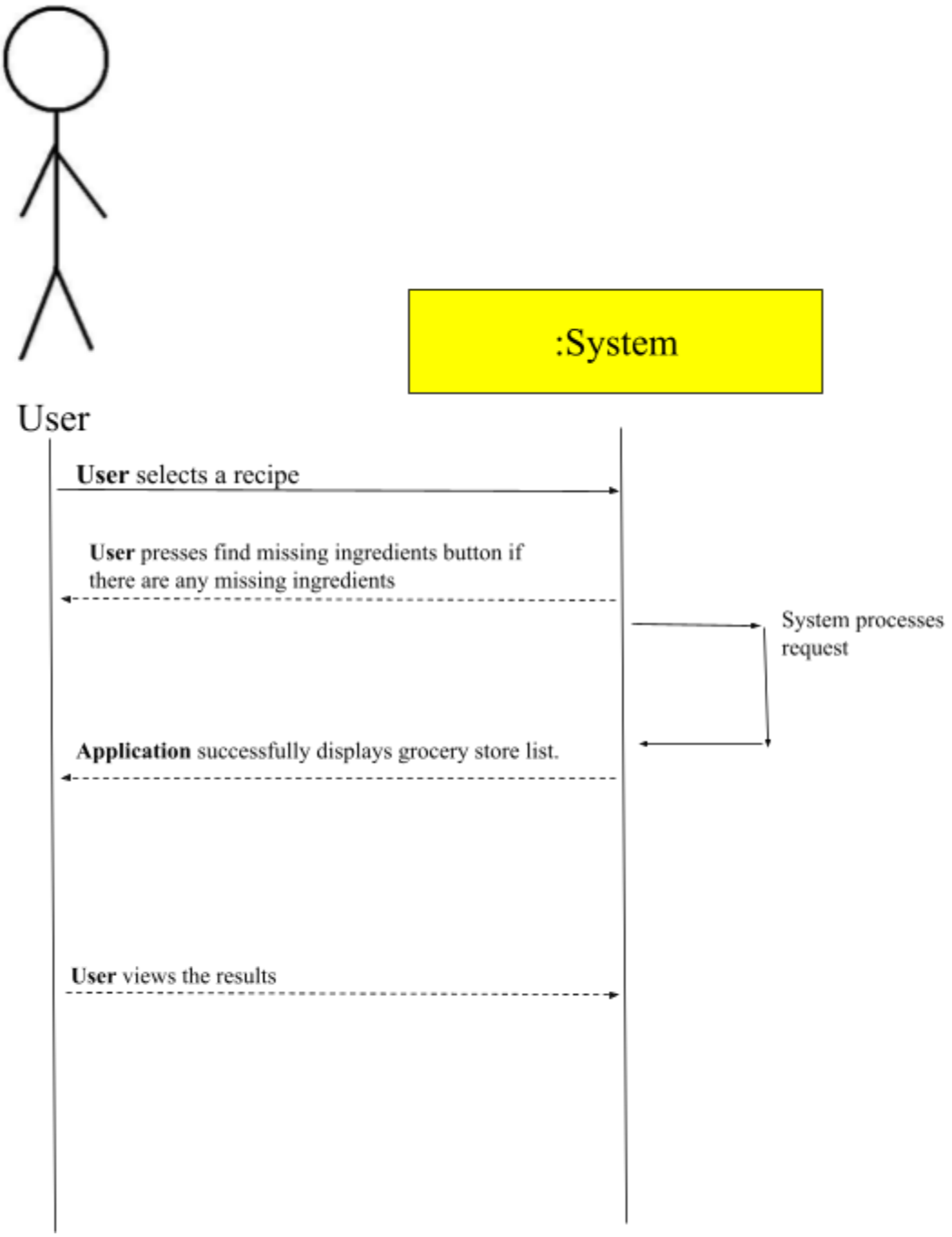
UC-7 Recipe Search Success Scenario



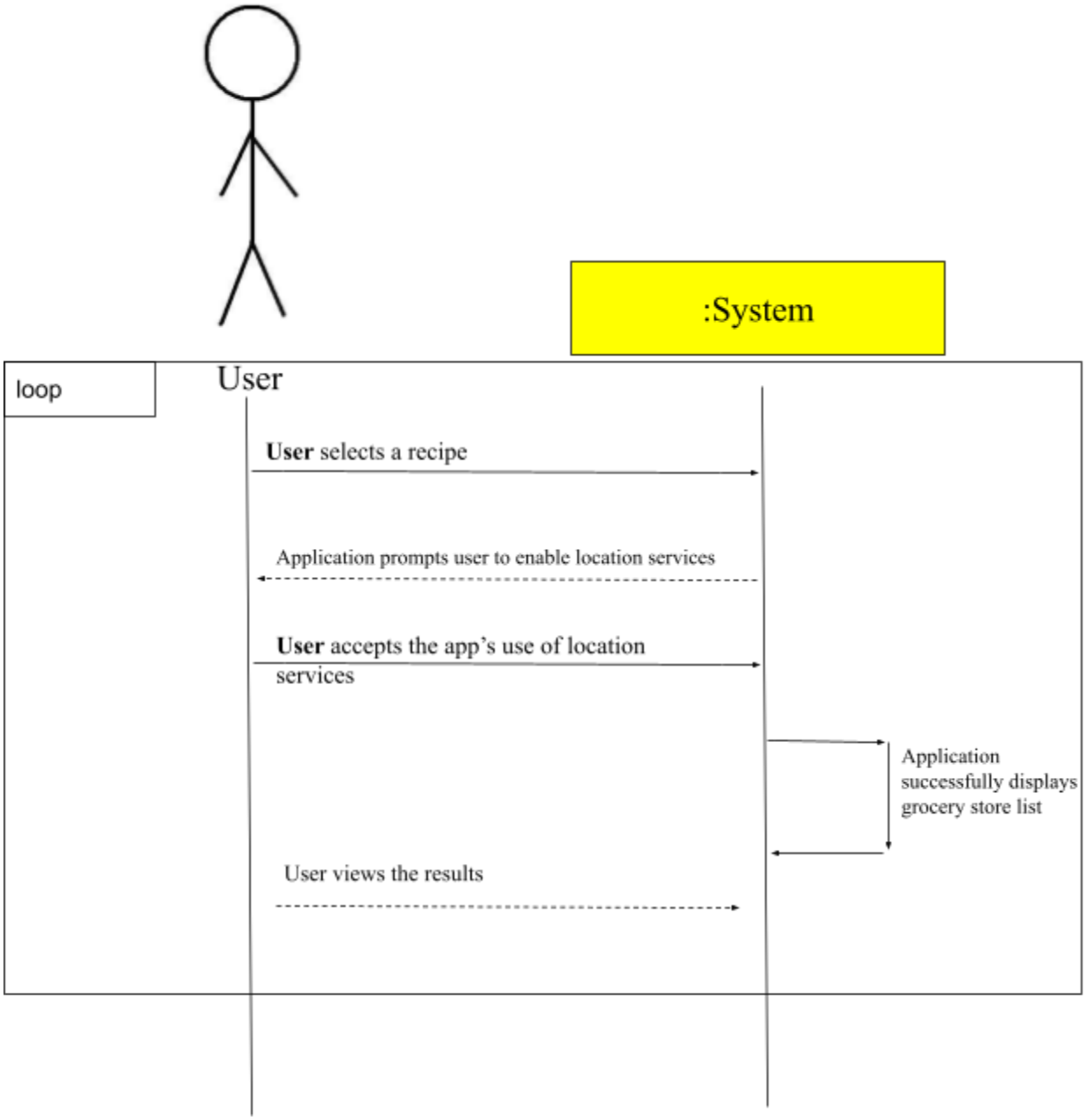
UC-7 Recipe Search Alternate Success Scenario



UC-10 Grocery Store Locator Success Scenario


:System
User
User selects a recipe
User presses find missing ingredients button if there are any missing ingredients
System processes request
Application successfully displays grocery store list.
User views the results

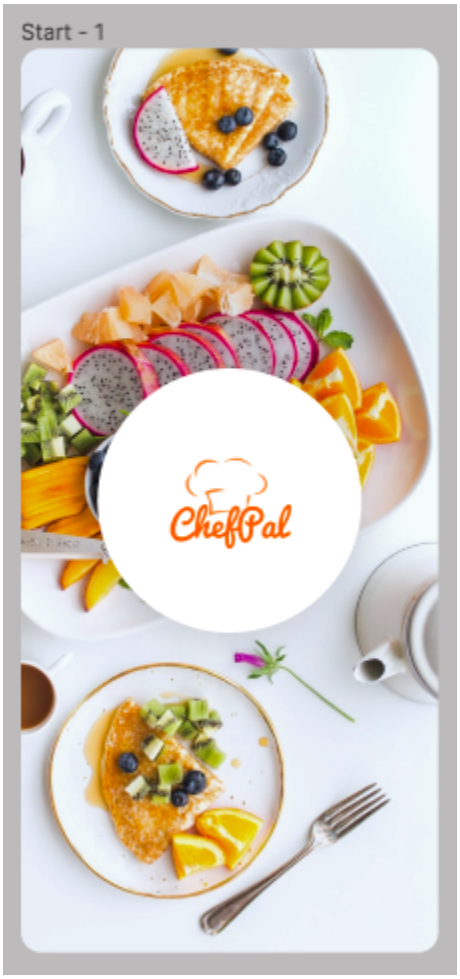
UC-10 Grocery Store Locator Alternate Success Scenario

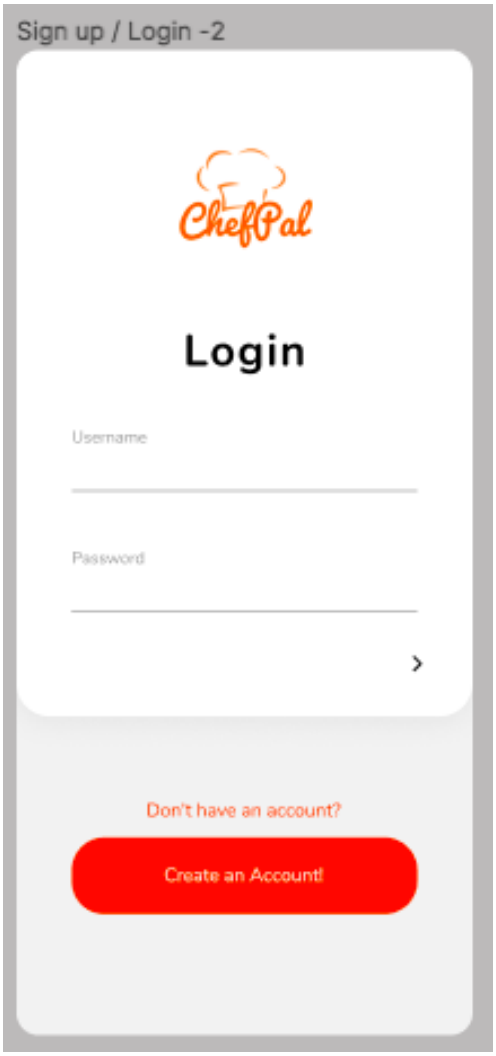


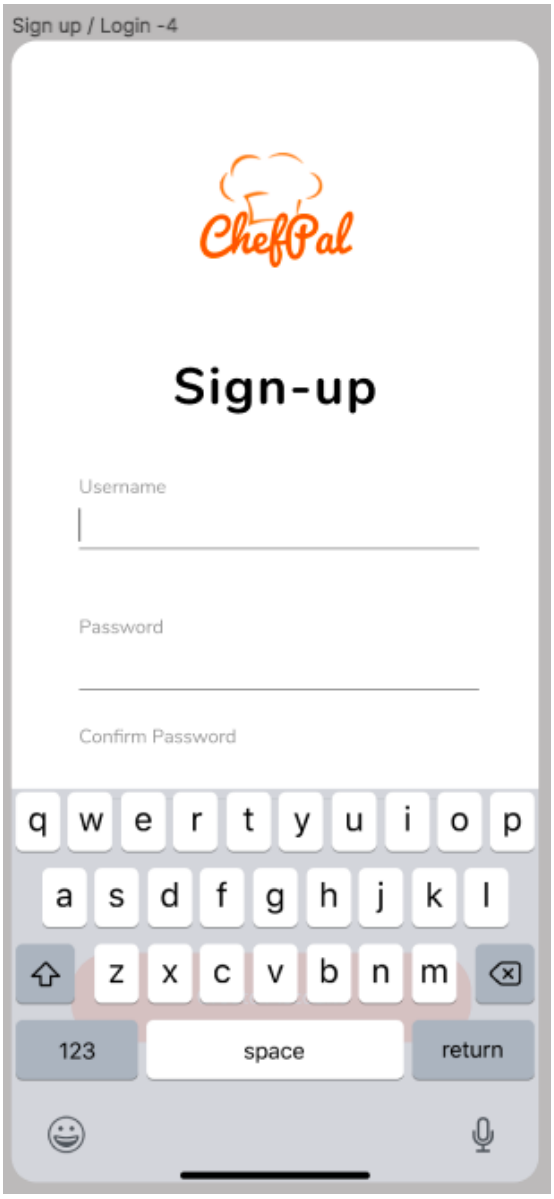
4) User Interface Specification

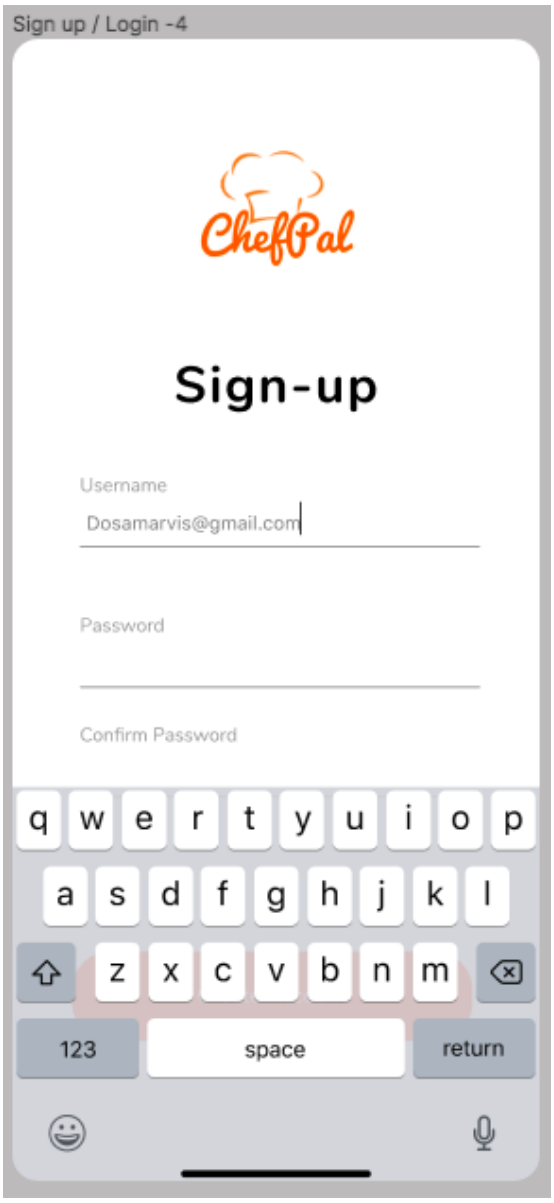
a. Preliminary Design

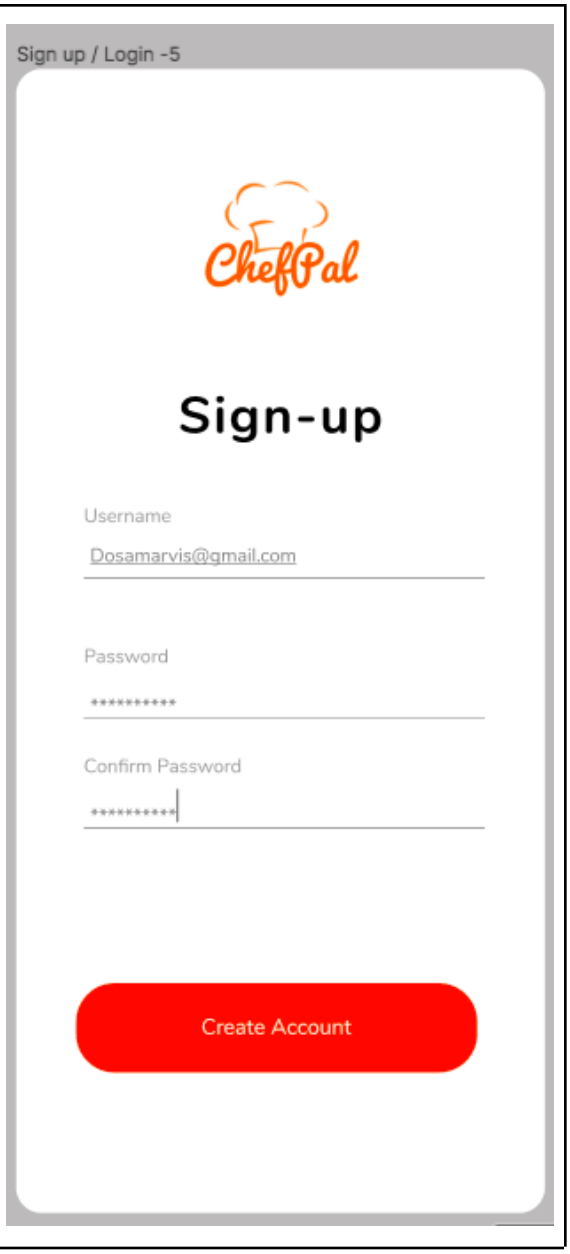
UC-1: Registration

1) User opens the application and sees welcome screen while application loads	

2) User clicks “Create an Account!” and is redirected to the sign-up screen.	

3) User clicks on each textbox to input data.	

4) User inputs their email and password associated with the account.	

5) User clicks “Create Account” once all required fields are entered.	

UC-2: Primary Preference Selection

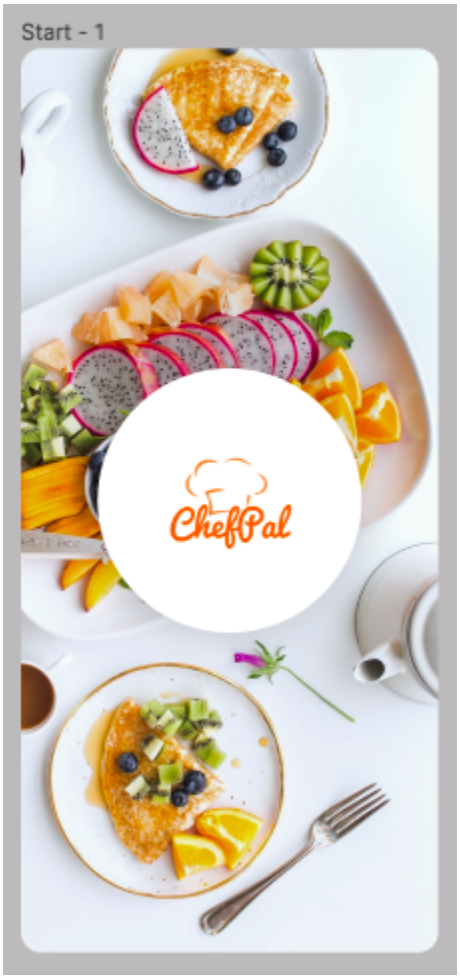
1) After creating an account, a checklist of types of dietary restrictions is displayed. The user can check off as many or as little as he/she desires.	

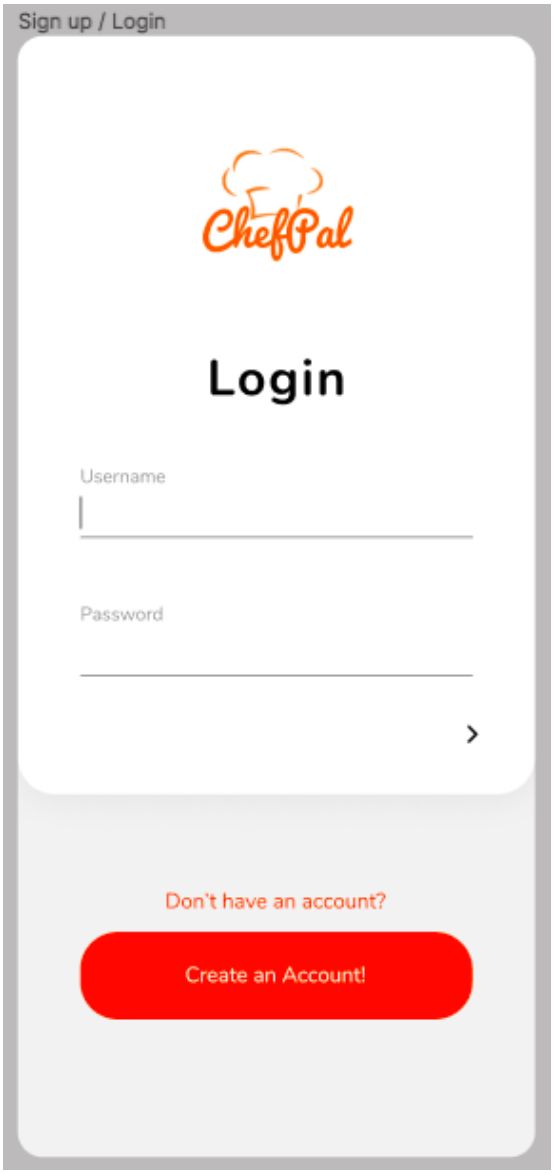
2) User clicks “Submit”	

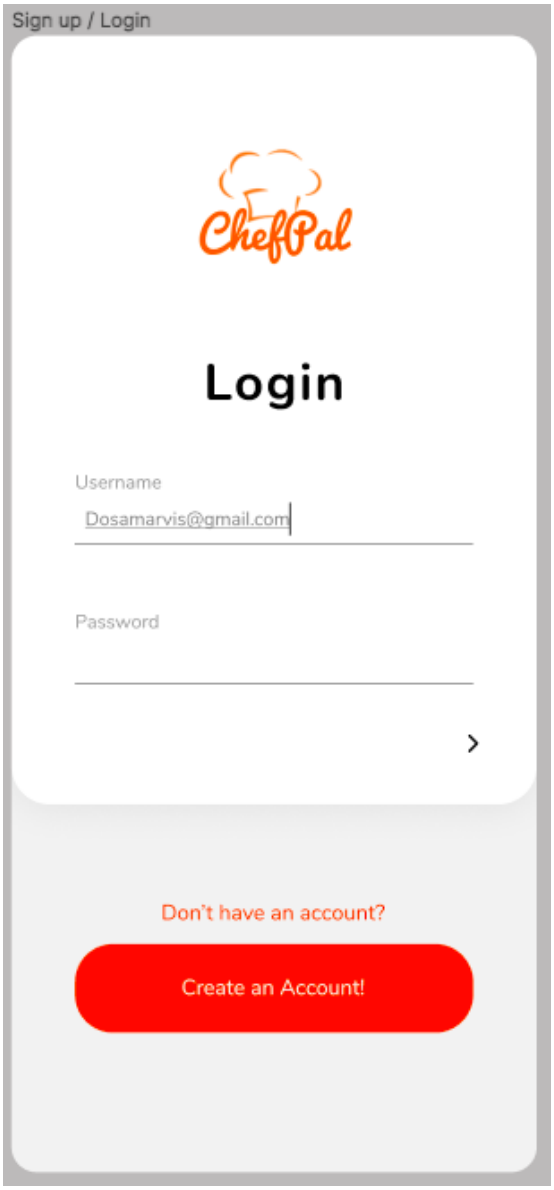
3) User is redirected to a distance radi to choose from. This will be the preferred range of distance between the user and the presented grocery stores for missing ingredients they may have.	

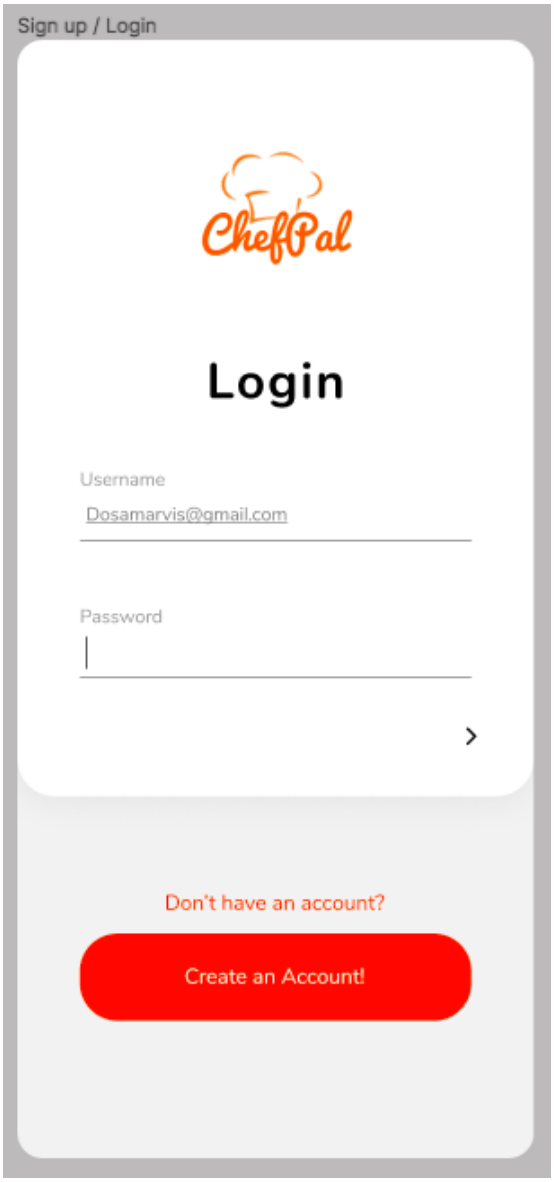
4) User clicks “Submit” on their preferred radi	

UC-3: Login

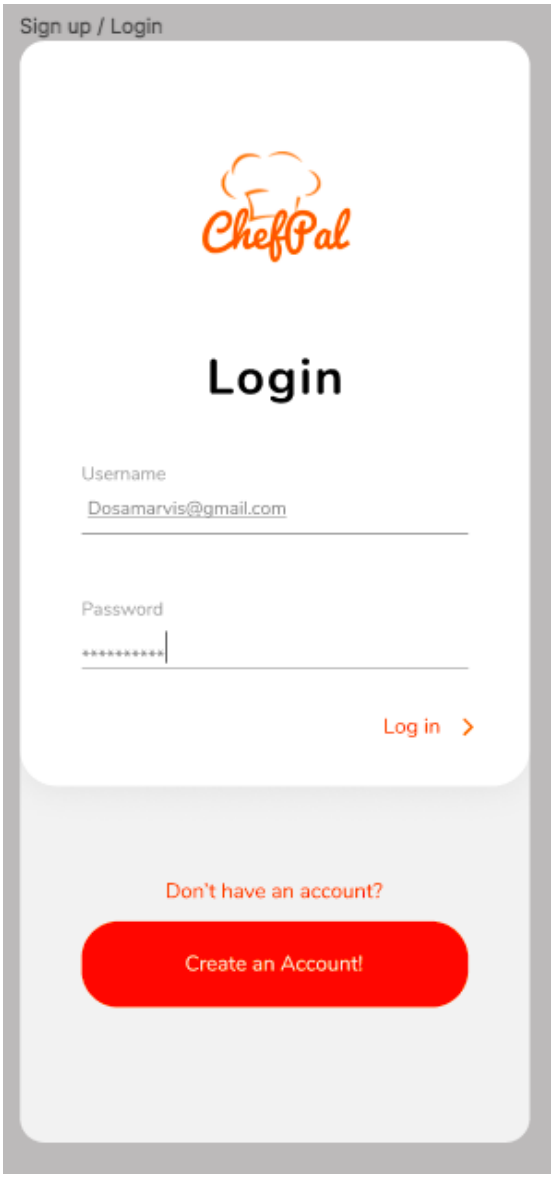
1) User clicks to open the application	

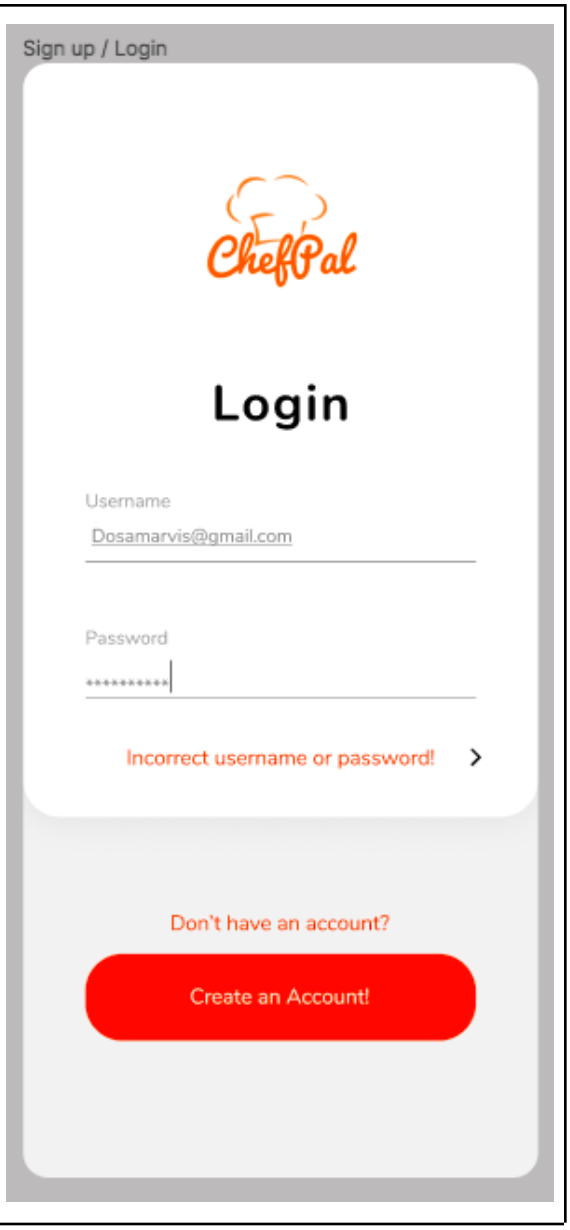
2) User clicks username textbox	

3) User enters username	

4) User clicks password textbox	

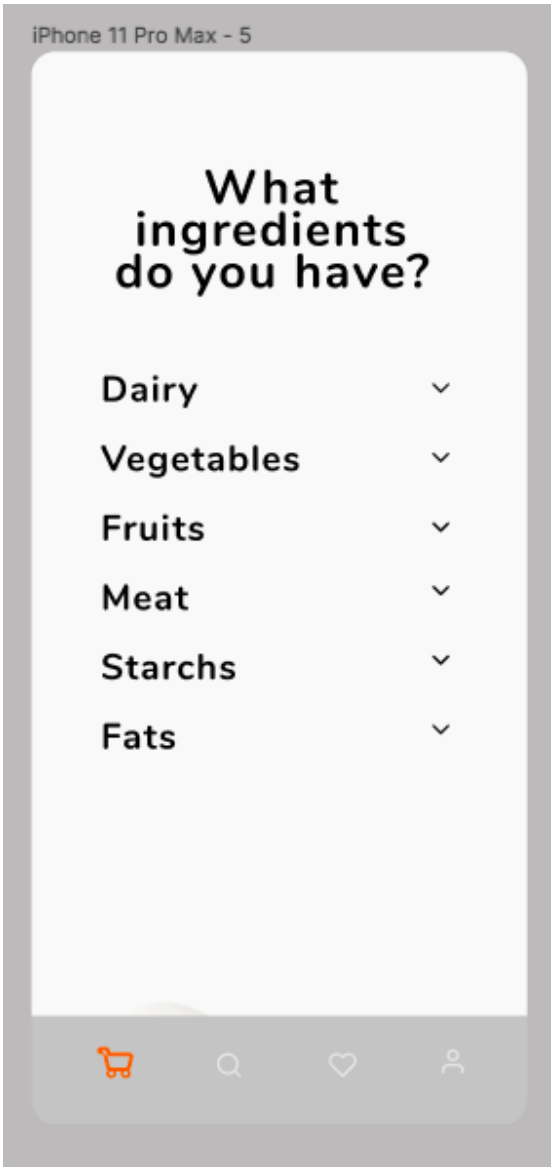
5) User enters password	

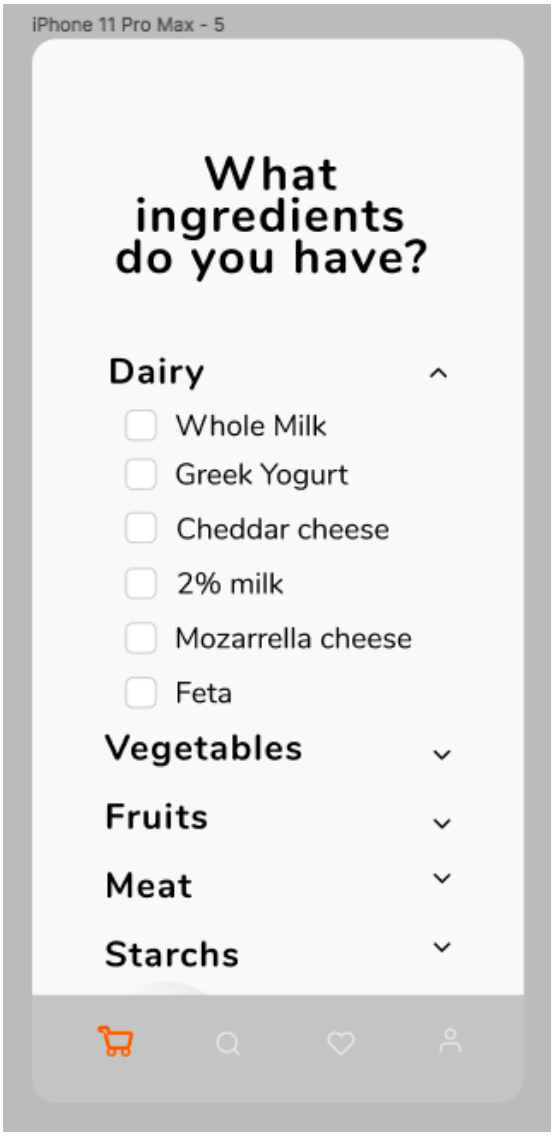
6) User clicks “Log In”	

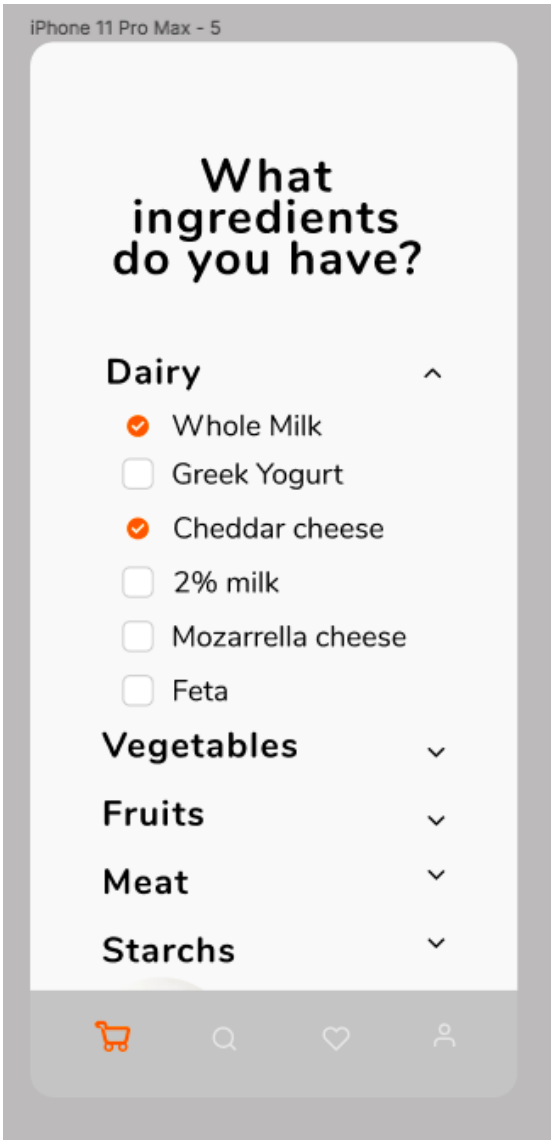
7) In the case the username or password is incorrect, the user gets the following screen	

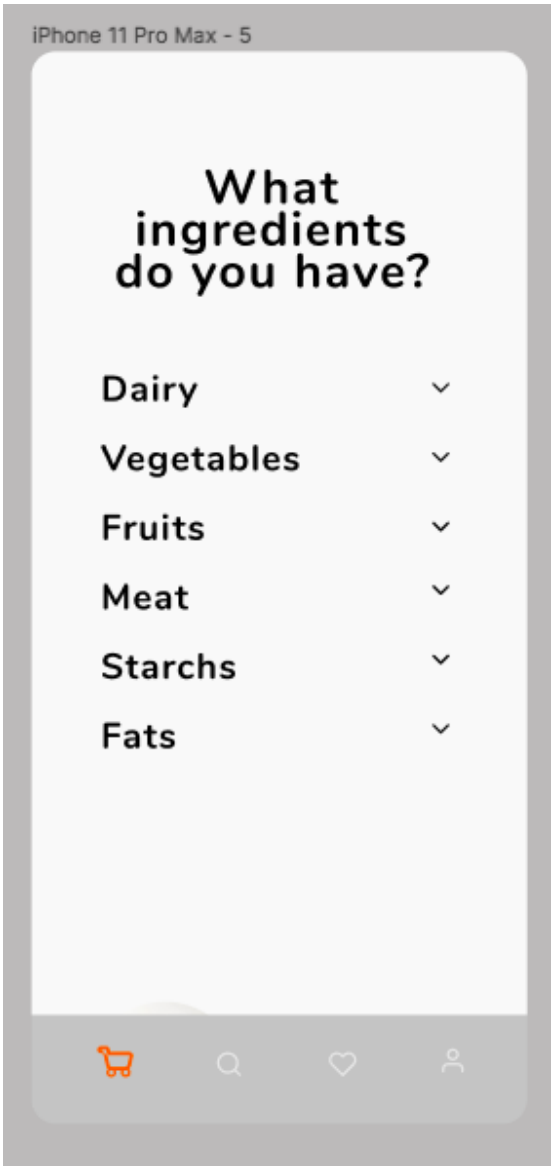
UC-4: Ingredient Selection:

Step-by-Step of how user enters information:

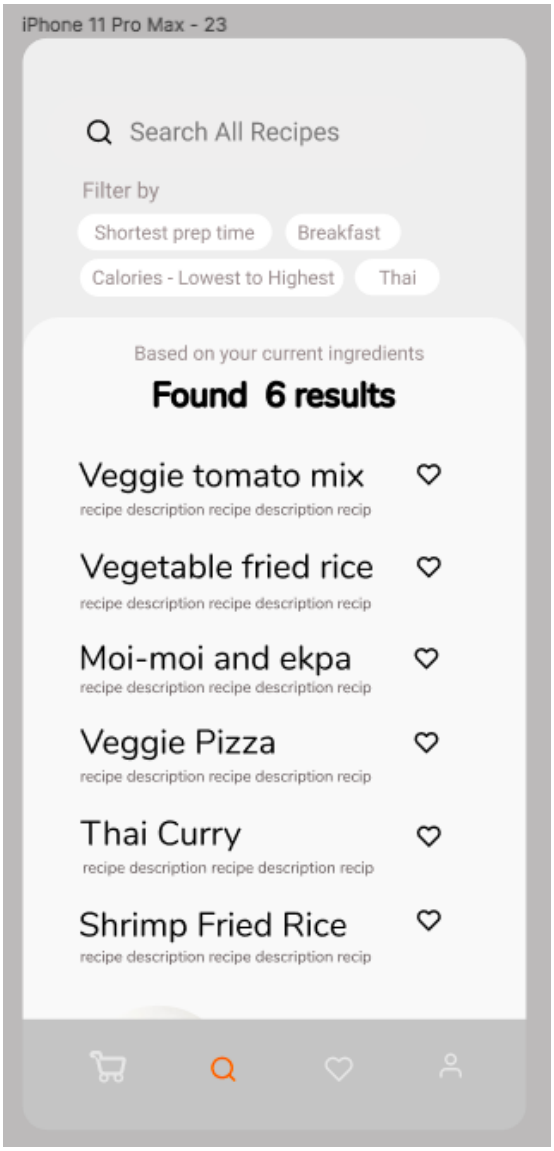
1) The user clicks on the shopping basket icon on the bottom tab	

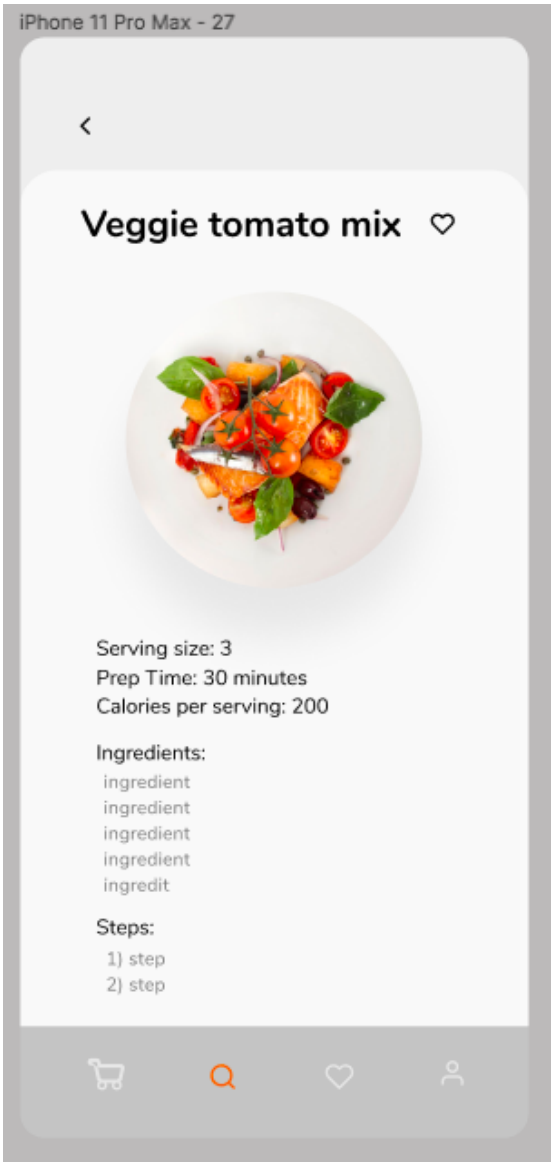
2) The user is presented with the titles of food categories that, when clicked, shows a drop down check list of ingredients	

3) The user can click the buttons next to the ingredient to check it off	

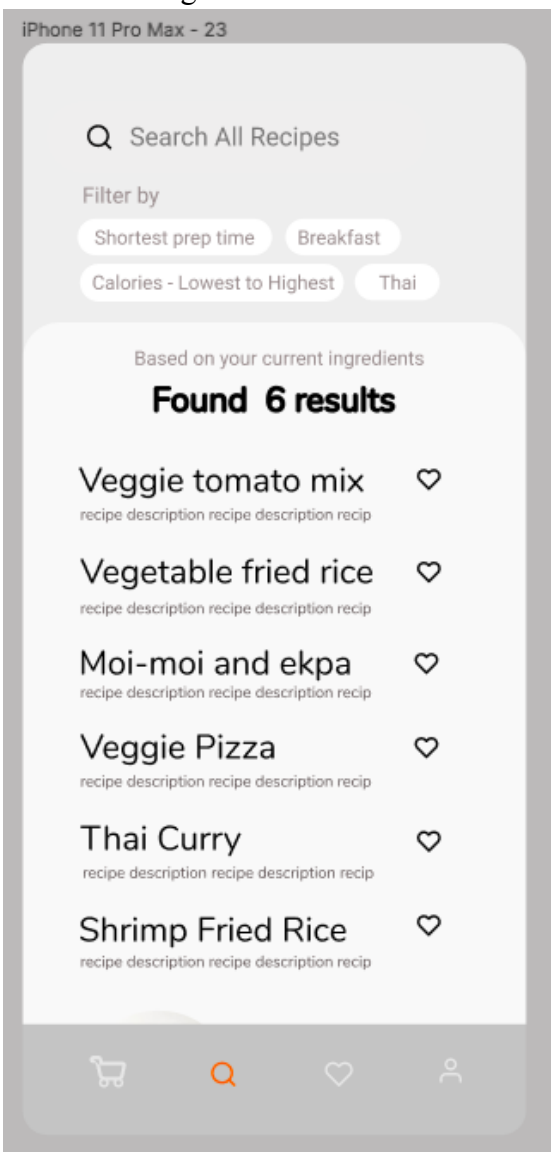
4) The user can click on the title of the drop down check list to minimize it again back to the original display	

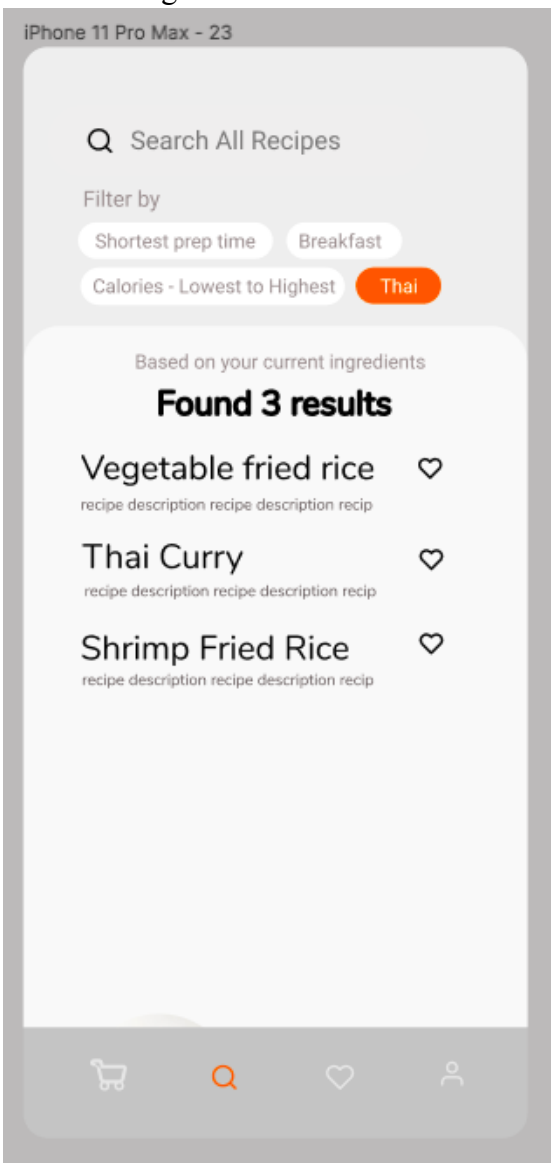
UC-5: Generated Recipes

1) User clicks “Search” tab at the bottom and is presented with the generated list of recipes based on their previously entered dietary preferences and available ingredients.	

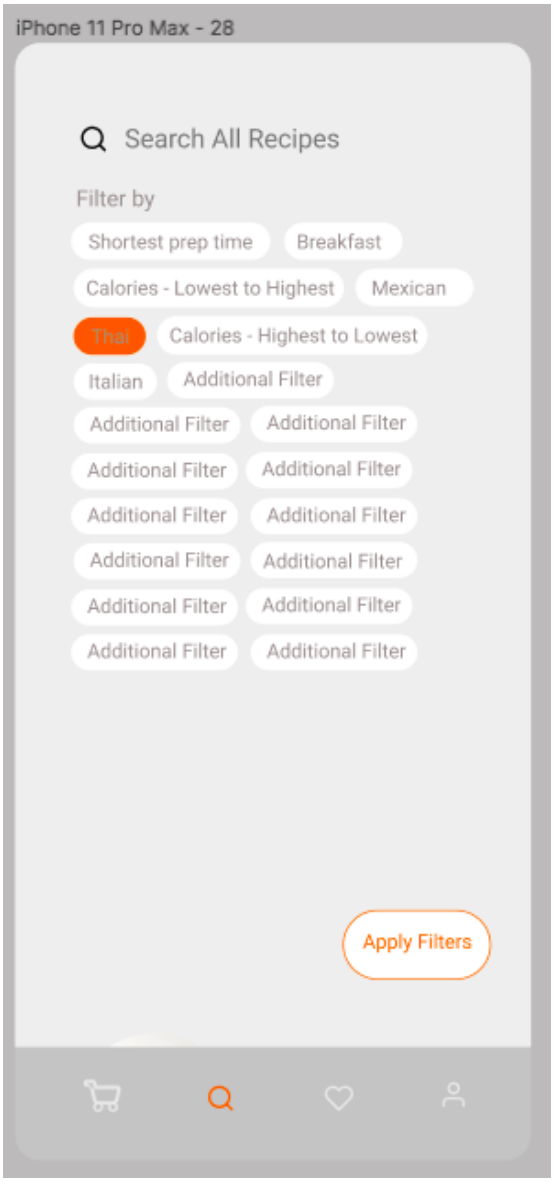
2) User can view a recipe by clicking on a title from the list. Users can click the back arrow to go back to the generated list.	

UC-6: Filtering System for Generated Recipes

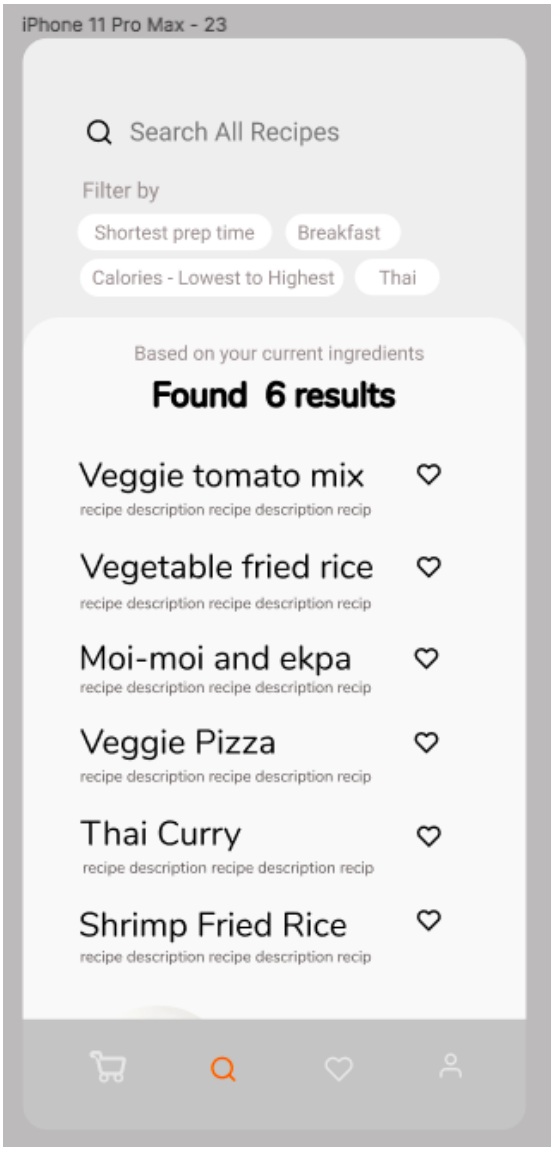
1) The user has the option to click on one or more of the filters at the top to further narrow down this list accordingly. Upon clicking a filter, the user will see the updated list:	Before clicking filter: 

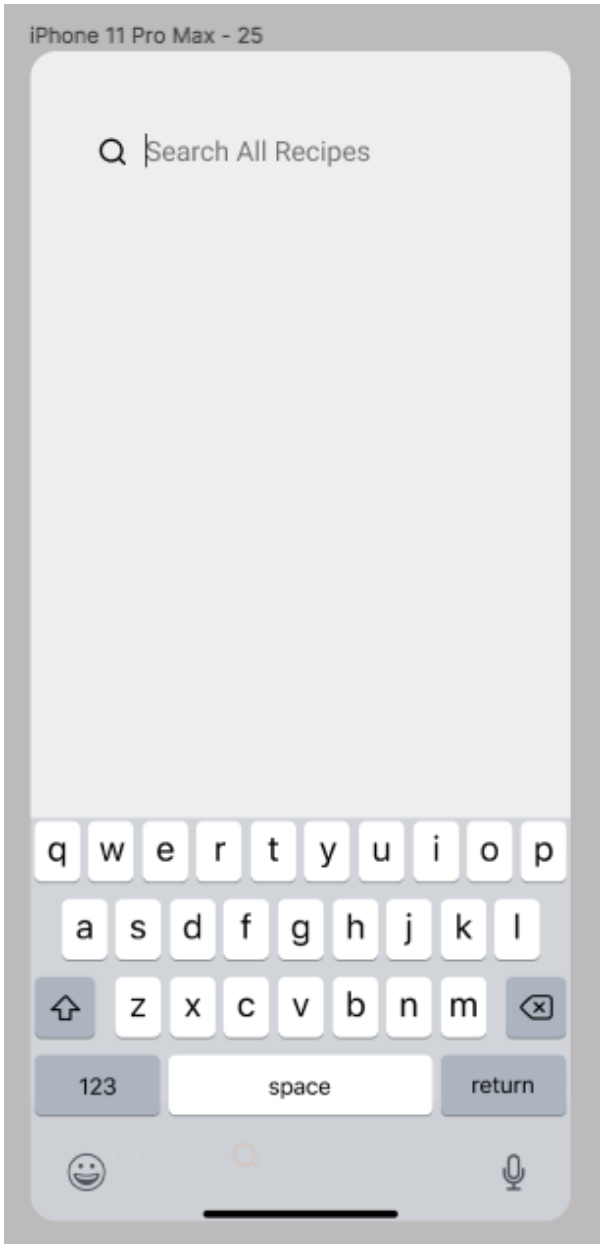
	After clicking filter: 

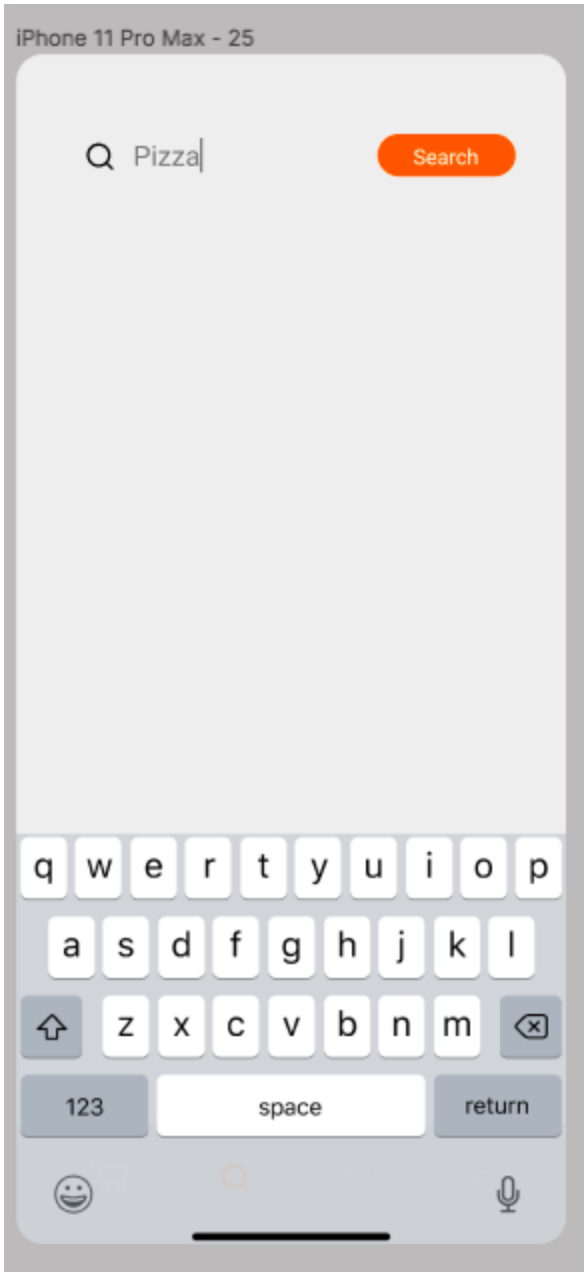
2) Users can see a full list of filters by clicking “Filter by” and can choose the filters there. They can then apply these by clicking “apply filters”

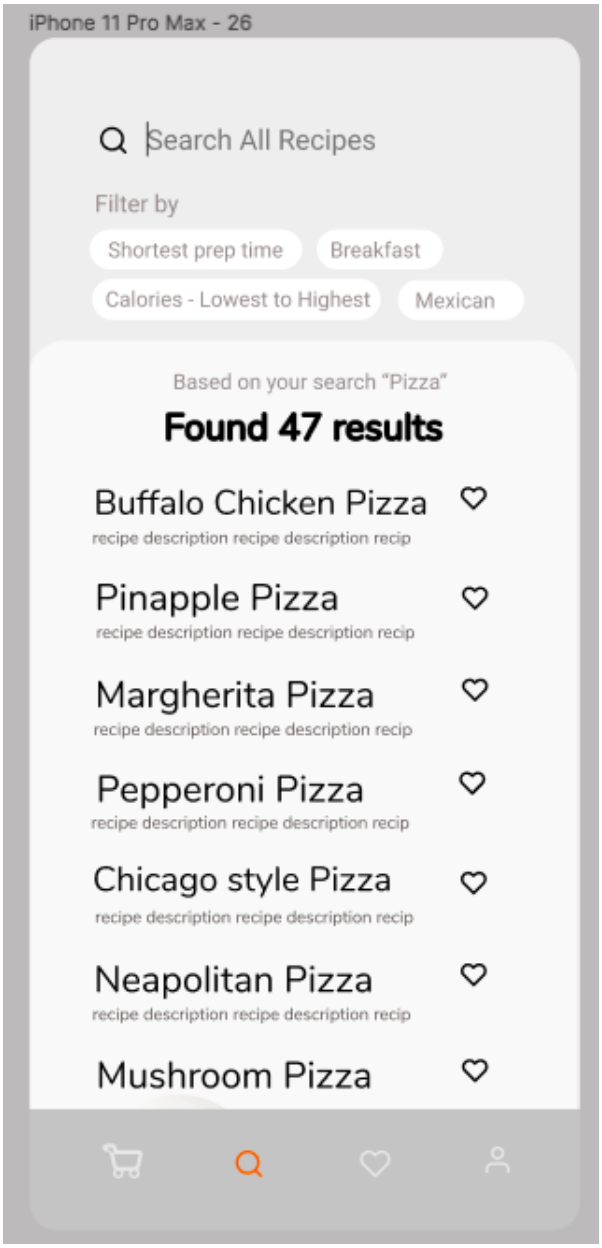


UC-7: Keyword Recipe Search

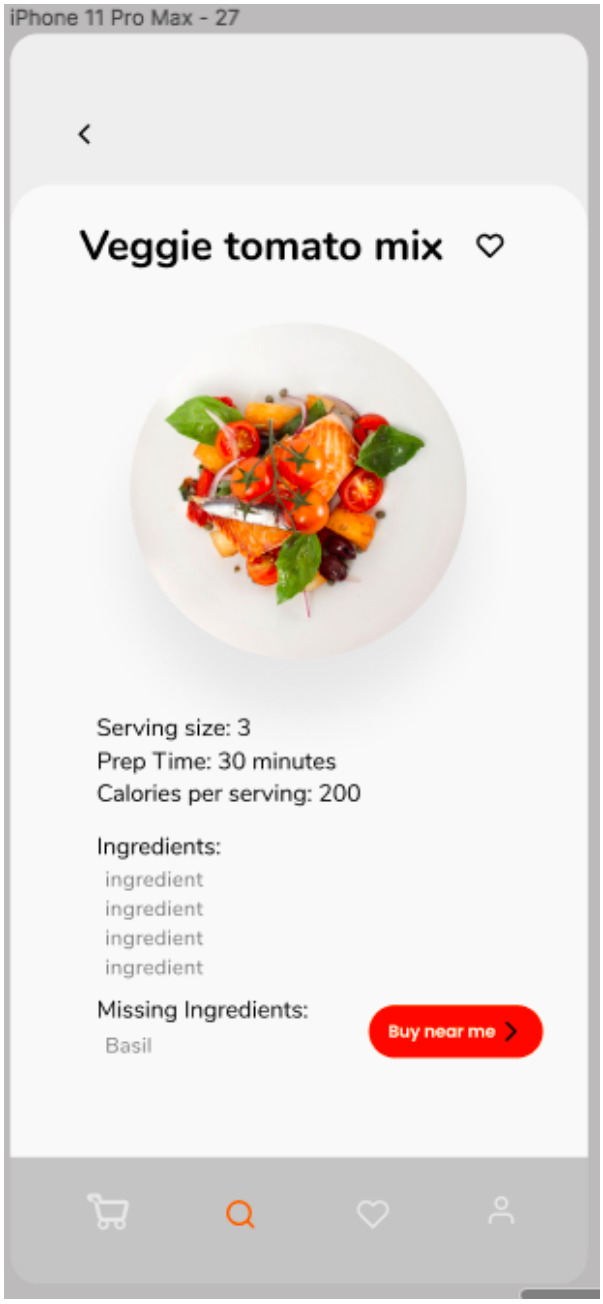
1) User clicks on search bar	

	 iPhone 11 Pro Max - 25 Search All Recipes q w e r t y u i o p a s d f g h j k l z x c v b n m 123 space return

2) user inputs keywords for recipe search and clicks the search button	

3) User gets a generated list of recipes related to keyword, regardless if they have the ingredients or not. They can filter these recipes in the same way as the previous use case. They can also view these recipes in further detail as seen in UC-5.	

UC-10: Grocery Store Locator

1) User should have a recipe open to view that requires an ingredient that the user does not have. User clicks on “Buy Near Me”	

2) User is redirected to a list of stores that sell the missing ingredient	

b. User Effort Estimation

Note: N = user-interface navigation; D = clerical data entry

UC-1: Registration

- User clicks to open the application (N) → user clicks "Create New Account" (N) → user clicks for each textbox to input data for username and password (D) → user inputs data into each textbox (D) → user clicks "Create Account" once all required fields are entered (D)
- Total clicks = 1 + 1 + 1 * (# of textboxes) + (keystrokes for data entry for each textbox) + 1
- Maximum of 50% of clicks are for user-interface navigation; at least 50% of the clicks (keystrokes) will be for clerical data entry

UC-2: Primary Preference Selection

- User clicks which dietary preferences to apply (the user is directed to this screen after the user successfully creates a new account) (D) → user clicks "Submit" (D)
- User clicks slider to apply distance preference (the user is directed to this screen after the user successfully selects a dietary preference) (D) → user clicks "Submit" (D)
- Total clicks = (# of dietary preferences selected) + 1 + 1 for distance slider + 1
- 0% of clicks are for user-interface navigation; 100% of clicks are for clerical data entry

UC-3: Login

- User clicks to open the application (N) → user clicks username textbox (D) → user enters username (D) → user clicks password textbox (D) → user enters password (D) → user clicks "Log In" (D)
- Total clicks = 1 + 1 + (keystrokes for username) + 1 + (keystrokes for password) + 1
- Maximum of 5% of clicks are for user-interface navigation; at least 95% of clicks are for clerical data entry

UC-4: Ingredient Selection

- User clicks ingredients that are available (D) via:
    - Clicking drop down menu (N) → Selecting box next to ingredients in that menu to select the ingredient (D)
- Total clicks:
    - (# of drop down menus selected) + (# of ingredients within a certain drop down)
- Maximum of 50% of clicks are for user-interface navigation; at least 50% of clicks are for clerical data entry

UC-5: Generated Recipes

- User clicks on the “Search” icon at the bottom tab (N) after all ingredients are selected, to which the user is directed to a screen with the generated recipes. User can click on certain recipes to read them in detail
- Total clicks = 1, depends on if user clicks recipes to read in more detail
- 100% of clicks are for user-interface navigation; 0% of clicks are for clerical data entry

UC-6: Filtering System for Generated Recipes

- User clicks on “Filter by” (N) and is navigated to a menu with filter options → user clicks on a filter to apply (N), to which a menu of specific filters is shown → user clicks on a filter to apply (D) → user clicks “Apply Filters” (D)
- Total clicks = 1 + 1 * (# of filters user wants to apply) + 1 * (# of specific filters that user applies) + 1
- About 50% of clicks are for user-interface navigation; about 50% of clicks are for clerical data entry

UC-7: Recipe Search

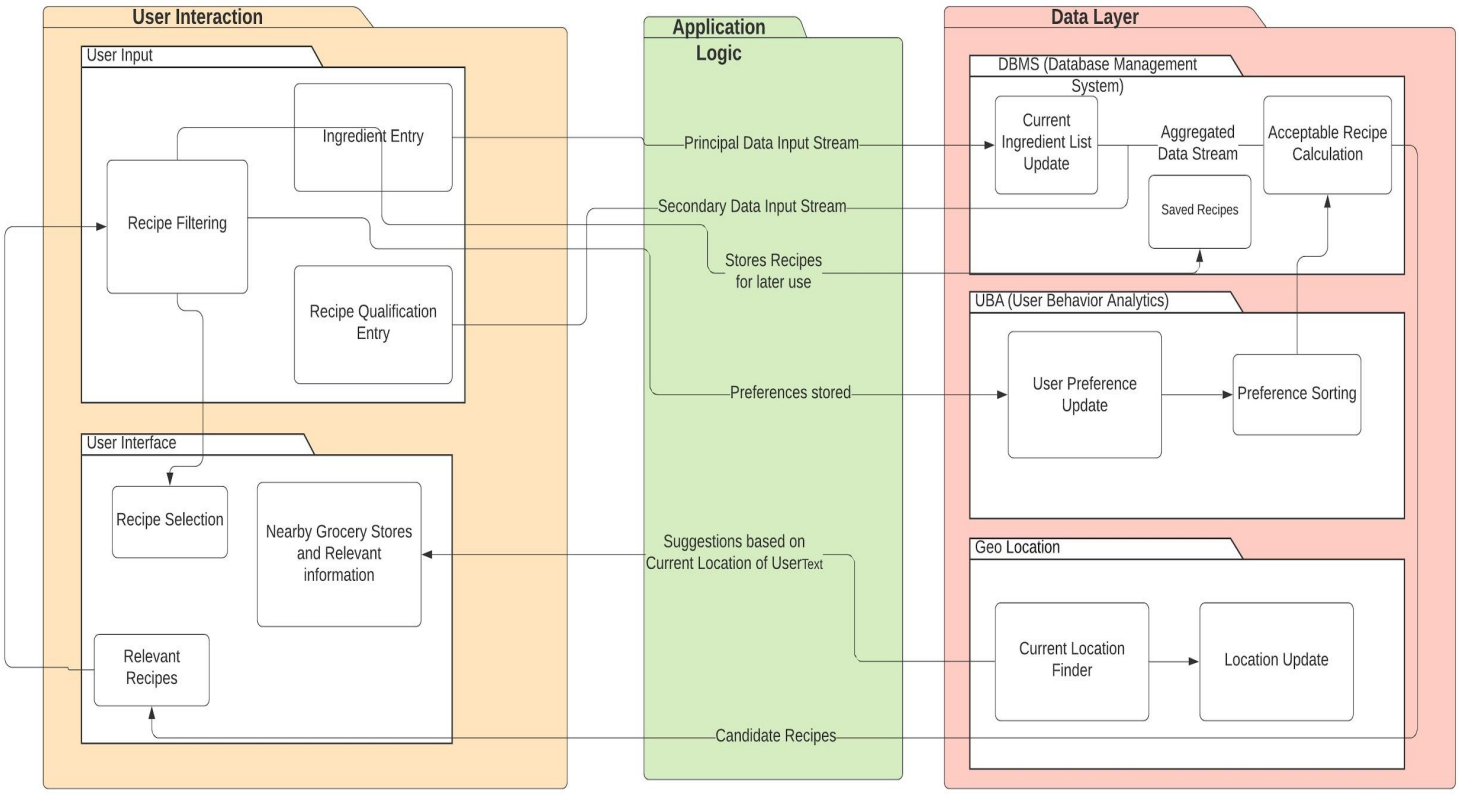
- User clicks on “Search Recipe” tab (N) → user clicks on search bar (D) → user inputs keywords for recipe search (D) → user selects “Search” button (D)
- Total clicks = 1 + 1 + (keystrokes for keywords) + 1
- About 10% of clicks are for user-interface navigation; about 90% of clicks are for clerical data entry

UC-10: Grocery Store Locator

- User picks the recipe (N) → clicks “Buy Near Me” next to the desired missing ingredient (N) → user allows for location use (D) → user clicks on store (D)
- Total clicks = 1 + 1 + 1 + 1
- About 50% of clicks are for user-interface navigation; about 50% of clicks are for clerical data entry.

5) System Architecture

a. Identifying Subsystems



Pictured above is the UML diagram to aid in subsystem identification

The UML diagram provided in this section demonstrates the procedural behavior of users and associated backend procedure systems. The diagram is divided into two principal modules: the User Interaction and the Data Layer modules. The purpose of this division is to separate all user-generated activity from the procedurally generated machine activity of the application. Each module is further subdivided into various subsystems. Below is a breakdown of the principal objectives of each module and subsystem:

- User Interaction Module - Everything remotely related to user actions. Subdivided into the following categories:
    - User Input - The necessary and sufficient information that the user needs to input for the application to render a successful decision.

        - Ingredient Entry - Fulfilling UC-4, the Ingredient Entry module allows the user to enter the ingredients that are available to them. This information is then passed as the Principal Data Input Stream to the subsequent decision making processes.
        - Recipe Qualification Entry - Partially fulfilling UC-2, the Recipe Qualification module allows the user to select all attributes that "qualify" a recipe to be suitable for him or her. This includes dietary restrictions, cuisine preferences, and macronutrient/caloric requirements. This information is passed as the Secondary Data Input Stream to the decision making protocols. It is secondary since the user can choose to omit entering any information in this section, thus it is not critical.
        - Recipe Filtering - In the penultimate step of the user experience, the user is given the option to choose from the recipes provided by the application. The process of selecting said recipes by the application is not conducted in this module.
    - User Interface - This module hosts everything that the user sees on his or her display but does not necessarily interact algorithmically.
        - Recipe Selection - A display of the selected recipes that the user wished to cook. This is the final step of the user experience.
        - Relevant Recipes - A collection of all the recipes that have been provided by the application for the user to select from. This module is chronologically between the Recipe Selection and Recipe Filtering.
        - Nearby grocery stores and Relevant Information - A list of grocery stores in the area provided to the user by the application. This is where the capabilities of the application come to an end, and the user may choose to go to the suggested grocery store and purchase the necessary items.
- Data Layer Module - This module hosts all of the back-end that the user does not directly interact with. It hosts tools for managing the various databases it interacts with, as well as the engine that is responsible for suggesting recipes to the user based on past history.
    - DBMS (Database Management System) - This serves as the engine for handling all API calls, as well as accessing the local database of recipes that the Selection Algorithm has chosen for the user.
        - Current Ingredient List Update - Maintains a running list of ingredients available at the users disposal. The running list may account for non-perishables that the user has previously input, or may be updated altogether.
        - Acceptable Recipe Calculation - After taking into account the ingredients that the user has on hand, as well as past preferences from UBA, the console cross-references the database to determine which recipes the user is able to cook.

        - Saved Recipes - A running log of the recipes that the user has favorited in the past. The Saved Recipes log is saved locally on the user's machine, and is factored into the Acceptable Recipe Calculation.
    - UBA (User Behavior Analytics) - This module contains all the decision-making process behind the suggestion generation for preferred recipes.
        - User Preference Update - Using preferences that users put towards the process of filtering recipes (whether that be cuisine, type of meal, saved recipes etc.), these particular preferences will be stored to the user's profile for future use.
        - Preference Sorting - Using information stored from User Preference Update, those user specific preferences will be used towards helping filter out relevant recipes that they may search for in the future.
    - Geolocation - This module is responsible for finding the current location of the user while using ChefPal.
        - Current Location Finder - Based on where the user is currently located while using the application, nearby grocery stores will be recommended to the user if they need to get additional ingredients
        - Location Update - Last used location is stored in case the next time the user needs nearby groceries, they are accessing from the same place (e.g. their home)

b. Architecture Styles

Our application will be a combination of the Layered Architectural style, and the Client-Server architectural style. In the Layered Architecture we will have 3 layers: user interface, application logic and database. In the user interface layer, the user will directly communicate with the application, from a front-end perspective. The application logic layer is incredibly important, because this is the layer that directly interacts with the database layer and is in charge of what the application will do next depending on the specific user input. The application logic handles the API requests that will connect between the user and the data stored in the AWS server. The database layer is where all the user information is stored, as well as the ingredients, recipes, and all the information about those recipes is stored as well. This style is ideal for our application because it clearly separates the different responsibilities of each layer, so that layer only has to focus on one part. This does not mean, however, that each part is independent of the other layers. It simply means that each layer has its own priorities to handle, and working together with the other layers can develop a fully functional application.

Our application also uses a Client-Server style architecture. In this case, the part of the architecture that is always waiting for a command is our database. The recipes and information are always ready to be obtained, and are simply waiting to be accessed. The client, or the user, will be the one performing the actions in order to tell the server that it is looking for some

particular information. In our application, the API is the connector from the user to the server. This style works really well with our application because all of the information is stored in a database, while multiple clients all obtain information from that one server.

c. Mapping Subsystems to Hardware

In this application, the user will be able to access and interact with the software using a cell phone or tablet. This device will handle all the user inputs that will require information to be retrieved from the data layer. Any input that will require information to be retrieved, or even navigation of the application will be done through the touch screen on the mobile device or tablet.

There will also be a backend server, which will contain the databases for ingredients, recipes, as well as saved recipes. This will also hold any additional backend code that we need for the app to operate. The server will communicate with the client through different API requests, depending on how the user interacts with the application. Multiple users will be able access the database on various different devices, although it will only be one user per device.

There will also be a connector, which will send the API requests to get the information from the data layer to the user. The connector facilitates API requests, so that the user never needs to directly interact with the database, and allows for processing of contextual information.

d. Connectors and Network Protocols

Our application will run on a user's mobile devices and will communicate with our own database which will store the user's preferences, saved recipes, and other information. This database will be stored on AWS using either AmazonAurora or AmazonDynamoDB. This data will need to be accessed using an API provided by AWS. AWS offers both REST & GraphQL APIs to communicate with data you have stored and other backend services you may have stored on the server such as AWS Lambda. In order to perform recipe searches and get information about nearby grocery stores we would be using existing REST APIs to get the necessary information from the existing data pools. To get the recipe data we will use the Spoonacular REST API and for the grocery stores we will use the Google Places API. These APIs would follow the same control flow as the API we would be using for our own database, except we do not have to implement the connection from the API to the database. The user would perform an action on the app such as a recipe search. The app would then make HTTP requests to the API. The API would then fetch the data from the database and pass it back to the API. Then the API will finally pass the given information back to the user.

e. Global Control Flow

Our system is event-driven. In this sense, there is no “linear” set of instructions that the user has to go through each time they use the application. The user may be able to input ingredients, or immediately search the general database for recipes. The system will wait for an instruction to come from the user, and depending on the user’s input, the system will respond accordingly. In our case, our participating actors, or the Spoonacular and Google Places APIs, both use HTTP requests with the response data as either JSON or XML. This will allow the system to wait until any instructions are received from the user, and then send out the relevant information based on which instruction to perform. Our system is an event-response system because the app responds to events initiated by the user. For example, the user will initiate an event and AWS AppSync will process that event request and serve the user the data or service they require. This will be used to serve the user the data or service they require.

Since our application is event driven there is no concern for realtime.The application can be used at any time of day where there is a stable internet connection to connect with the server. There is no real-time constraint, and the system can be executed at any time.

f. Hardware Requirements

Our application should be compatible with any iOS device running iOS 11 or above and with any android device running android 5.0 or above. Due to these OS requirements, the device should have a color display with size restriction of a 4 inch display (iphone 5s) or above for smartphones; for tablets, there should be a 7.9 inch display or above. Our application also restricts the user to operate only in portrait mode. The size of our application is still under review so as default we would require the user to have a minimum of 500 mb of free storage space on their device. We would also require a device with GPS service for our grocery store locator function. We will update the hardware requirements as the design of the software progresses.

6) Project Size Estimation Based on Use Case Points

UCP = (UUCW + UAW) x TCF x ECF = (100 + 8) x 0.815 x 1 = **88.02**

- UUCW
  - UUCW = 4 x Simple + 5 x Average + 2 x Complex = 20 + 50 + 30 = 100

Group	Use Case	Description	Category	Weight
1	UC- 1: Registration	Allow users to create a login with a unique username and password that is stored in the database.	Simple	5
1	UC- 2: Primary Preference Selection	The user will be prompted to enter in their primary preferences, such as any dietary preferences or restrictions, as well as enter their location and a radius that they will be willing to go grocery shopping in.	Average	10
1	UC- 3: Login	The user can enter in their username and password, allowing the user to be logged into their own account with their own personalized app.	Simple	5
2	UC- 4: Ingredient Selection	The user will be able to select the ingredients they have by checking off boxes next to ingredients that are sorted into lists by food groups.	Simple	5
2	UC-5: Generated Recipes	Based on the ingredient selection, the user is presented with a generated list of recipes. These recipes are based on the dietary preferences in the user's stored profile settings and the ingredients they have selected. The recipes that require the least number of extra missing ingredients appear first.	Average	10
2	UC-6: Filtering System for Generated Recipes	After seeing the generated recipes, users will have the option to further narrow down these recipes based on additional filters. These filters would include cuisine, meal preparation time, type of meal, and calories.	Average	10
3	UC-7: Recipe Search	The user will be able to search the entire recipe database, regardless if the user has the	Complex	15

		necessary ingredients to make that recipe. Once being able to access all of the recipes and search for any recipe they desire, they can still filter out their results by cuisine, meal prep, etc. However, these recipes will not be personalized to the user's ingredients, rather it is meant more for the user to explore the available recipes and see the selection at hand.		
3	UC-8: Save Recipes	After searching for recipes, the user will be able to add recipes that he or she likes to a custom list, which the user can create. The user will have a page of the app dedicated to recipe lists, which he or she has saved and will allow for more lists to be created.	Average	10
4	UC-9: View/Change Profile	The user will be able to change their login information, edit their dietary preferences and food intolerances. The user will also be able to alter their address in order to change the list of available grocery stores.	Simple	5
4	UC-10: Grocery Store Locator	The user will be able to find stores nearby which should have the ingredients that the user is missing. This will allow the user to buy the necessary missing ingredients to make the recipe.	Complex	15
4	UC-11: Bug/Error Reporting	The user will be able to report any bug or error that occurs. This way, the developers can be aware of any problems that the user is facing and help approach them via a hotfix.	Average	10

- UAW
  - UAW= 1 x Simple + 2 x Average + 1 x Complex = 1 + 4 + 3 = 8

Actors	Roles	Complexity	Weight
Amazon Cognito	Will handle all user registration and authentication.	Simple	1
Spoonacular API	Will take in all the users ingredients and filters, and it will process their search.	Average	2
Google Places API	Will be used to search for nearby grocery stores upon request by the user.	Average	2

Bugify Issue Tracker	Will handle user bug reports.	Complex	3

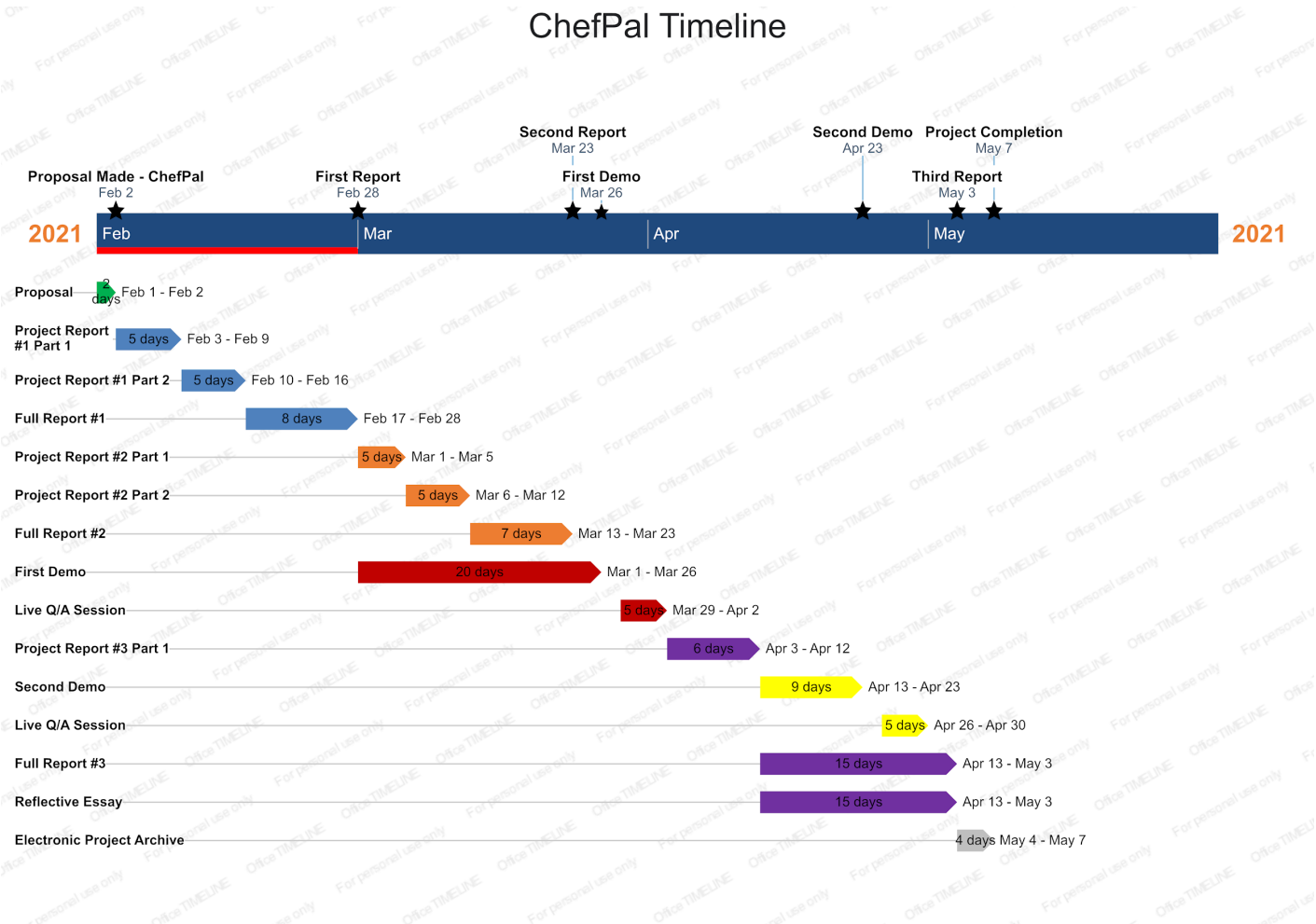
- TCF
  - TCF= 0.6 + (TF/100)
  - TCF= 0.6 + (21.5/100) = 0.815

Group	Identifier	Description	Weight	Perceived Complexity	Calculated Factor
1	REQ-18	The application should be compatible with IOS and Android OS.	1	3	3
3	REQ-19	The application should be able to handle 500 daily users and can be scaled as needed.	1	3	3
2	REQ-20	The application should keep user information private by securely storing passwords and other sensitive information.	2	1	2
2	REQ-21	As a system, all stored and collected data should be secure.	2	3	6
1	REQ-22	The application layout should be aesthetically appealing and minimalistic with clutter free UI.	1	1	1
1,2,3,4	REQ-23	The application should operate at 60 Hz or 120 Hz for supported devices.	2	1	2
4	REQ-24	The application should be easy and simple to download and navigate for nearly any person who has access to a smartphone and internet.	2	1	3
3	REQ-25	As a system, user requests and issues should be checked daily and addressed as soon as possible.	0.5	2	1
1	REQ-26	As a system, unit testing should be	0.5	1	0.5

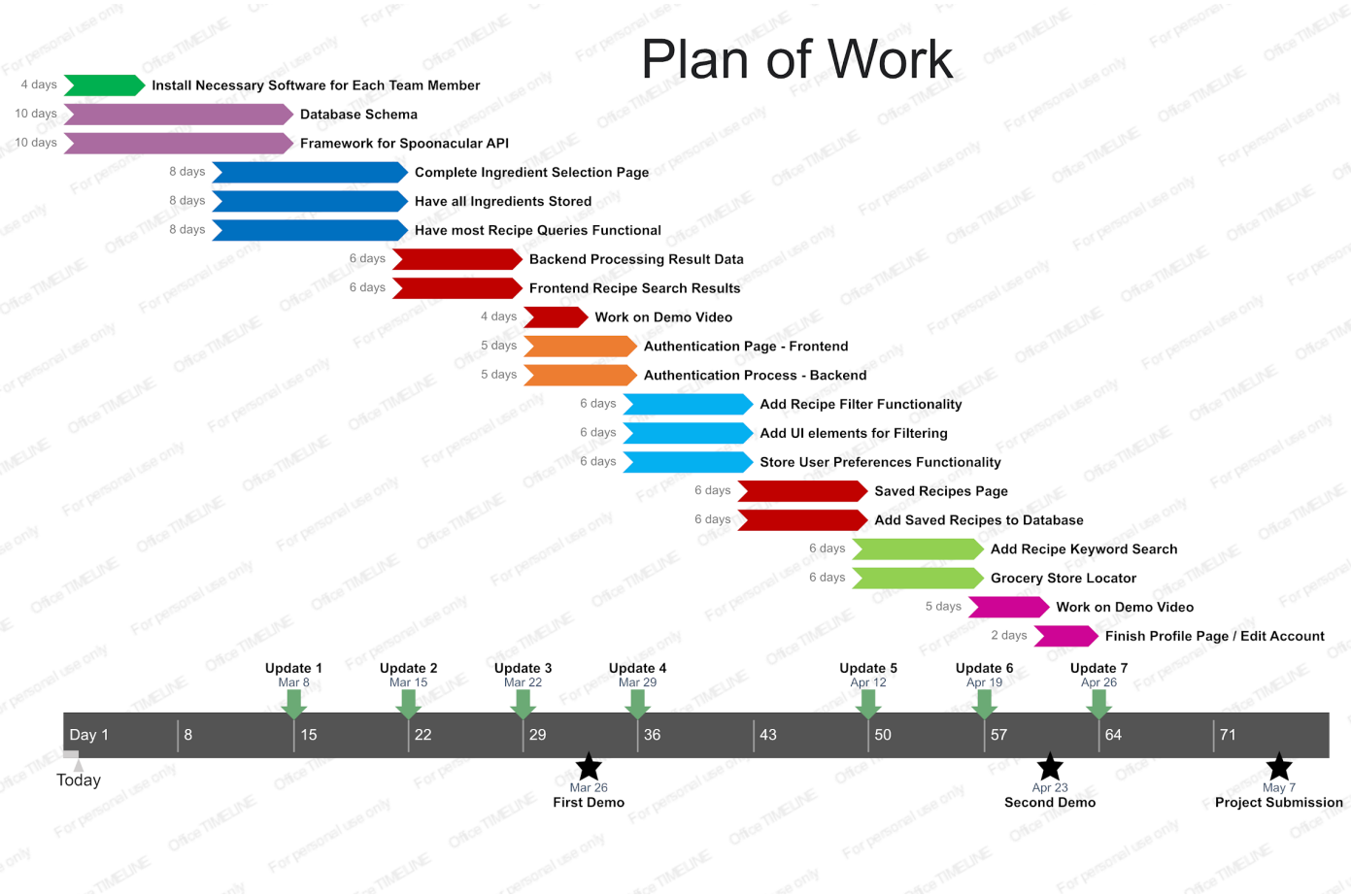
		done after any changes to the system and before any version updates releases.			
				Technical Factor Total :	21.5

7) Plan of Work

Deadlines for ChefPal Project:



Deadlines for Project Creation and Management:



Product Ownership Description:

So far, each group member has contributed equally to the totality of the first report, as shown in the individual breakdown at the beginning of the report. Each group member has discussed, collaborated, and executed each task asked for the project. In the future, we will continue to develop and elaborate on the production of our mobile application. We will be focusing on creating the frontend, backend, and data management of the mobile app in order to prepare for our first demo. We have divided the project into four major parts so far and have established who will be responsible for what part.

Group 1 - Malena and Azim:

- Allowing users to check off the ingredients they have available to them.
- Allow users to check off what specifications/ restrictions they want, whether it is any dietary restrictions, cuisine or types of meals.

Group 2 - Nirav, Shahir, and Dymytriy:

- Offering a variety of recipes based on user preferences and inputted ingredients
    - Recipes shown are based on the filter selected by the user
    - Overarching goal is to create an analytics engine capable of basing recipe suggestions off past choices.

Group 3 - Michael, Malak, and Daniel:

- Storing both a user's favorite recipes in a recipe book and their previously checked off ingredients and preferences for future use.

Group 4 - Aswathy, Amanda, and Azim:

- Offering a variety of grocery stores based on user preferences, such as convenience, based on the missing ingredients in the recipe

Currently, our group is starting to prepare for establishing the design of the mobile app and creating the second report. As we get ready to design the application, group members are also starting to download and learn specific coding languages that we will be using in order for the creation of ChefPal. We will start off by learning how to operate Flutter and figure out how to share code with Github. If any difficulty arises throughout the project, we will gladly work together and help each other through any obstacles we are facing.

Project Management:

In order to organize our group's thoughts and ideas, our group is using an instant messaging platform called Discord. With Discord, we are able to communicate with one another through messaging and voice chat. In addition, our groups had established specific times where we all would get together and discuss the next stage in our project and make sure we are all on the same page and everything is going smoothly. The meetings are every Monday and Thursday at around 12 PM, which last for about an hour. We have one group member working asynchronously from a different time zone, and one with scheduling conflicts. Also, we got together on the weekend and Tuesday nights to work and finalize our report. We used breakout rooms in Discord to work in smaller groups for specific sections of the report. In order for everyone to contribute towards writing and formatting the report, we used Google Docs in order to share and collaborate on the completion of the report as a group.

For the coding portion of the project, we will be using JavaScript - backend - and Dart - for frontend - in addition to whichever modules work best with mobile app integration. Furthermore, we decided to use Flutter for app development, which uses Dart.

For the UI design we had used Figma to illustrate the application and the steps a typical user would go through to access/ find recipes. Mostly used for the preliminary design, Figma

was able to allow us to specify what we need and want our mobile application to look like once all is said and done. With this user interface design, we are able to offer a preview of what our users should be expecting to see and be using from this app.

For the database and storage of our code, we will be using AWS to allow us to store the user's data and other data required for this project. With the use of AWS, we will most likely be using SQL web services. In addition, we will be using Github to share code with one another, which allows us to collaborate on certain coding aspects of the project.

8) References

Amazon Web Services, Inc. 2021. *AWS Amplify Pricing | Front-End Web & Mobile | Amazon Web Services*. [online] Available at: <https://aws.amazon.com/amplify/pricing/> [Accessed 24 February 2021].

Amazon Web Services, Inc. 2021. *Build a Flutter Application on AWS*. [online] Available at: <https://aws.amazon.com/getting-started/hands-on/build-flutter-app-amplify/> [Accessed 24 February 2021].

Amplify Framework Documentation. 2021. *Amplify Framework Documentation*. [online] Available at: <https://docs.amplify.aws/lib/datastore/sync/q/platform/flutter> [Accessed 24 February 2021].

Eceweb1.rutgers.edu. 2021. [online] Available at: <http://eceweb1.rutgers.edu/~marsic/books/SE/projects/Restaurant/2019-g13-report3.pdf> [Accessed 16 February 2021].

Ers.usda.gov. 2021. *USDA ERS - Food Prices and Spending*. [online] Available at: <https://www.ers.usda.gov/data-products/ag-and-food-statistics-charting-the-essentials/food-prices-and-spending/#:~:text=In%202019%2C%20households%20in%20the,representing%208.0%20percent%20of%20income> [Accessed 16 February 2021].

Flutter.dev. 2021. *Firebase*. [online] Available at: <https://flutter.dev/docs/development/data-and-backend/firebase> [Accessed 24 February 2021].

Flutter.dev. 2021. *Write your first Flutter app, part 1*. [online] Available at: <https://flutter.dev/docs/get-started/codelab> [Accessed 24 February 2021].

Google Developers. 2021. *Overview | Places API | Google Developers*. [online] Available at: <https://developers.google.com/places/web-service/overview> [Accessed 16 February 2021].

Marsic, I., 2021. *Software Engineering Project Report - Requirements*. [online] Ece.rutgers.edu. Available at: <https://www.ece.rutgers.edu/~marsic/Teaching/SE/report1.html> [Accessed 16 February 2021].

Medium. 2021. *AWS Amplify for Flutter*. [online] Available at: <https://medium.com/flutter-community/aws-amplify-for-flutter-ed7890f75493> [Accessed 24 February 2021].

Medium. 2021. *Size matters: Reducing Flutter App size best practices*. [online] Available at: <https://suryadevsingh24032000.medium.com/size-matters-reducing-flutter-app-size-best-practices-ca992207782> [Accessed 24 February 2021].

Spoonacular.com. 2021. *spoonacular recipe and food API*. [online] Available at: <https://spoonacular.com/food-api> [Accessed 16 February 2021].