cookpad

Better Cookpad
Cook Better

Table of Contents

Executive Summary

Cookpad community platform is a system that enables people to share recipe ideas and cooking tips. It's a global platform used by on average around 100 million people every month across the world. Over 4 million recipes have been created by people in almost 70 countries.

The four main features that Cookpad platform already had are "Explore", "Create", "Plan", "Search". For the Explore feature, the user could browse recipes shared from the Cookpad community to inspire originative cooking ideas. For the Create feature, the member could edit recipes with photos, ingredients, cooking methods, and tips, etc. Also, the member could publish recipes created if he/she is willing to do so. For the Plan feature, there are three parts, which are "Planning-to-Cook list", "Save-to-Cook list", "Cooked-Again list" respectively. Briefly, when the member creates a recipe or cook followed by recipes from other members, the recipe will be automatically added in Cooked-Again list. As the member find a favorite recipe and plan to cook in the future, the member could save the recipe to Save-to-Cook list. The last part, which is Planning-to-Cook list, allows the member to put recipes that are ready to cook and makes it convenient for the member to find recipes quickly. Another feature, Search, allows the user to search recipes based on the keywords inputted by the user. When the user inputs the keywords, the Cookpad interface page will list related recipes for the user.

After experiencing on our own for 2 weeks, we found that the existing system lacks, or can be improved, some useful features: filter and sorting and recommendation function. As we were searching recipes on the app, the bunch of results directly and instantly show on the screen, which is nice. However, sometimes the first few posts were less popular, out of dated, or we love it but the prepare time is too long for us. To prevent from these scenarios, implementing filter and sorting function may provide a great help. Therefore, we plan to add a content-based filter and sorter, which enable users to apply the constraints (popularity, posted time, and prepare time) on searching results.

Regarding to the recommendation function, different from the artificial keyword-based system, we design the one depends on TensorFlow API. Leverage on TensorFlow's functionality, the Word2Vec module allows us to achieve the goal. Word2Vec is a method to convert texts to vectors and thus

enable computers to apply algorithms on them. In our case, we utilized users' searching behaviors, recipes, and ingredients in their E-Fridge to calculate the correlations between them. By doing this, the system can automatically generate the recommended recipes at users' preferences.

We will deliver more details on our idea in the next section, and follow by the Context Diagram, Use Case Diagram, Use Case Descriptions, and Data Dictionary. Besides, we provide a few core conceptual interface designs in Interface Design section. This project also includes an entity relationship diagram (ERD) and some important scenarios that the system may function in real life in Database Design and Sequence Diagram respectively. Last but not least, pseudo codes for software designs are presented as well.

Problem Statement

Problems

1. The existing system only has keyword search optimization, which means it takes users plenty of time to find the specific cooking methods of desired recipes.
2. It's not easy for everyone always has ideas about daily meals, especially for people whose schedule is tight. Further, they usually waste time finding inspiration on searching posts and spend unnecessary money buying redundant ingredients.

Objectives

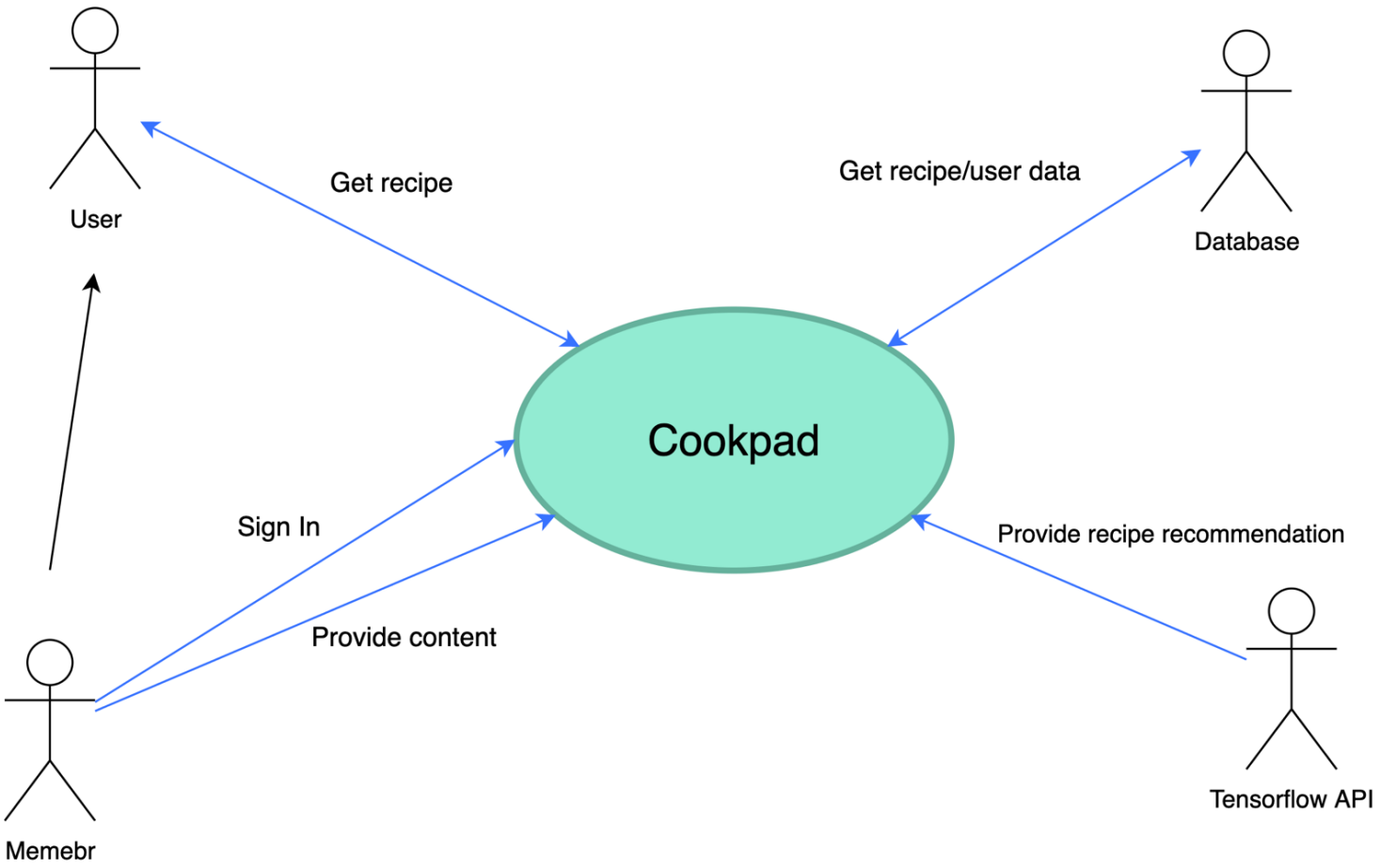
1. We aim to provide the more precise and more efficient results based on user's inputs. To achieve this goal, we add a filter and sorting features to the existing system that automatically collects and classifies each posted and uploaded recipe on the platform. User can thus apply filter to extract out their desired results and set the sorting criteria to make surfing experience easier.
2. Design an artificial intelligence (AI) recommendation model within the existing system to retrieve data from the users, analyze the data, and provide advice to the users when the user logs into the app. In other words, the model can give recommended recipes based on the user's

prior cook plans, recipes cooked repeatedly, recipes saved to cook, and searching behavior they performed. And all the above-suggested recipes are customized by either existing ingredients in the "E-Fridge" or the searching keywords.

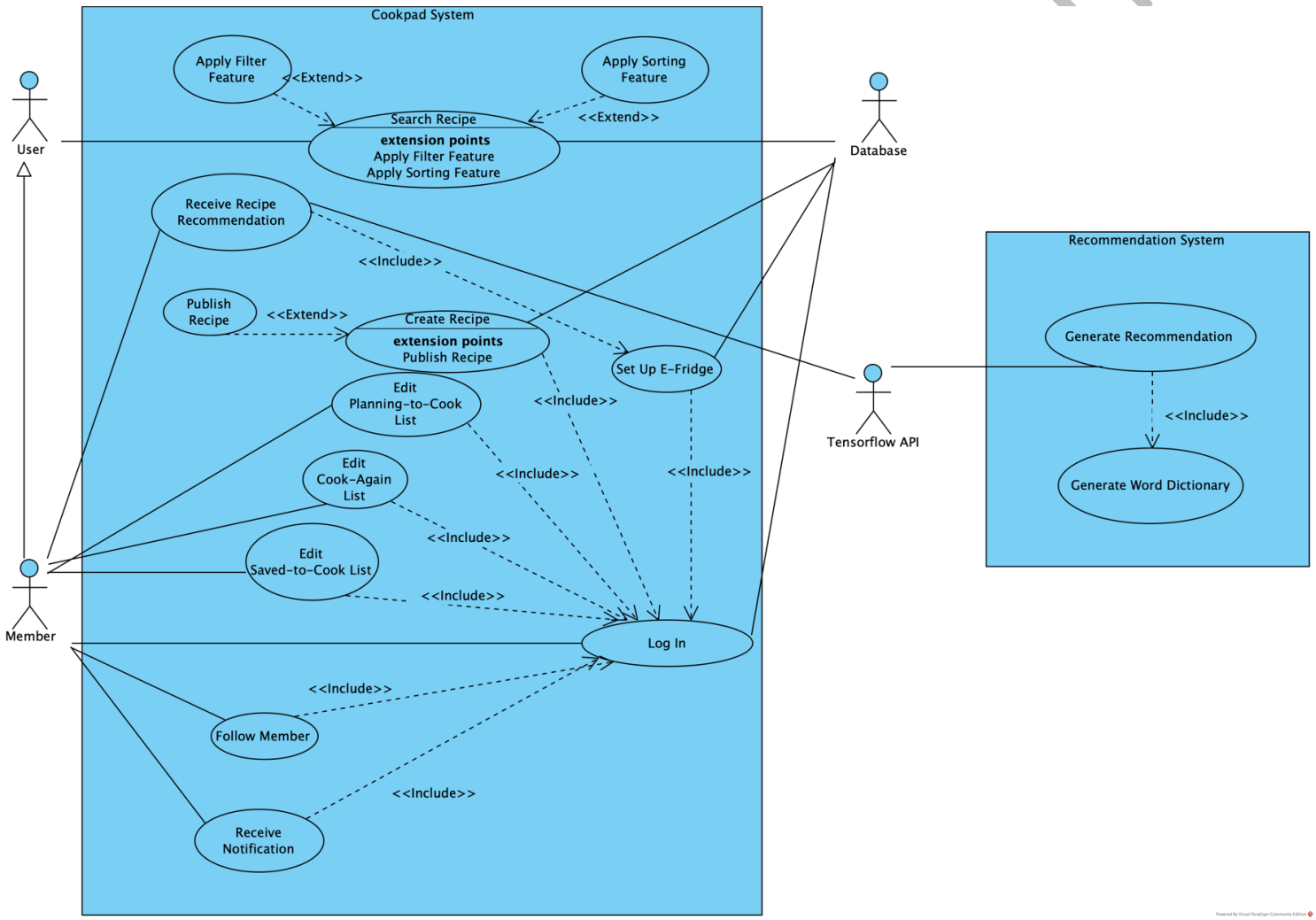
Scope

1. The project would primarily focus on improving the search recipes function by adding filter and sorting features, building the E-Fridge function and the Recommendation system.
2. The goal of the project is to provide brand new experience for members and motivate members to share more creative recipes with the Cookpad community.
3. The estimated cost of the new system is approximate $100,000.
4. The new system should be implemented within 6 months.
5. The development of new system shouldn't require additional manpower.

Context Diagram



Use case Diagram



Use Case Description

Use Case Description 1: Log In

Use Case Name:	Log In
Primary Actor:	Member
Stakeholders:	System, Member
Brief Description:	Member inputs email and password to log in to the System.
Trigger:	Member opens Cookpad App
Normal Flow of Events:	1. The System requests that the Member enter his/her email and password. 2. The Member inputs email and password and presses the Sign In button. 3. The System validates the Member's email and password and logs the Member into the system. 4. The System displays the main page of the App.
Exception Flow:	3a. The System displays the error message that the input email or password is invalid and goes back to step 2.

Use Case Description 2: Receive Recipe Recommendation

Use Case Name:	Receive Recipe Recommendation
Primary Actor:	Member
Stakeholders:	Member, Recommendation System
Brief Description:	Provide recommendation implementing the algorithms handled by TensorFlow API
Trigger:	Log in the system/Update E-fridge
Normal Flow of Events:	1. The Member logs in the System/Member updates the E-Fridge 2. The Recommendation System loads the data from system and runs the algorithms 3. The Recommendation System sends the results back to the System 4. The System displays the customized recommendations on main page
Exception Flow:	2a. If Member's database is empty, then skip step 3 and display the latest recipe posted by other users only

Use Case Description 3: Create Recipe

Use Case Name:	Create Recipe
Primary Actor:	Member
Stakeholders:	Member, Database
Brief Description:	Member creates his/her own recipe
Trigger:	Member clicks on the Create button
Normal Flow of Events:	1. The System displays New Recipe Request Screen 2. The Member enters Recipe Data 3. The Member clicks on the "Done" button 4. The System saves the recipe as a draft
Exception Flow:	

Use Case Description 4: Publish Recipe

Use Case Name:	Publish Recipe
Primary Actor:	Member
Stakeholders:	Member, Database, System
Brief Description:	Member publishes his/her own recipe
Trigger:	Member clicks on the Publish button
Normal Flow of Events:	1. The System checks the data validation 2. The System adds the recipe to database
Exception Flow:	1a. If any required item is empty, then display the error message and skip step 2

Use Case Description 5: Edit Planning-to-Cook List

Use Case Name:	Edit Planning-to-Cook List
Primary Actor:	Member
Stakeholders:	Member, Planning-to-Cook List, Cook-Again List
Brief Description:	Member adds the recipe to their Planning-to-Cook List
Trigger:	Member clicks on the “Add to plan”/ “Mark cooked” button
Normal Flow of Events:	1. The System modifies the Planning-to-Cook list 2. Move the recipe 2.1 If Add to plan: Add the recipe in Planning-to-Cook List 2.2 If Mark cooked: Move the recipe from Planning-to-Cook List to Cook-Again List
Exception Flow:	

Use Case Description 6: Set Up E-Fridge

Use Case Name:	Set Up E-Fridge
Primary Actor:	Member
Stakeholders:	Member, System
Brief Description:	Member manually input the ingredients into the system
Trigger:	Member submits a list of ingredients
Normal Flow of Events:	1. The Member clicks on the "E-Fridge" button. 2. The System opens the current Member's fridge. 3. The Member clicks on the "Add" button. 4. The System displays E-Fridge setup screen. 5. The System performs either subflow S-1, S-2, or S-3.
Subflow:	S-1: Add ingredient list to E-Fridge: a. The Member inputs the data of an ingredient list. b. The Member submits self-created ingredient list. c. The System updates Member's E-Fridge. S-2: Edit ingredient list in E-Fridge: a. The Member selected the existed ingredient list he/she wants to edit. b. The Member finishes editing and submits the modified ingredient list. c. The System updates Member's E-Fridge S-3: Cancel E-Fridge setup: a. The Member clicks on the "Cancel" button. b. The System displays confirmation message. c. The Member clicks on the "Discard change" button. d. The System closes E-Fridge setup screen without saving.
Exception Flow:	If the member enters quantity with zero, then display the message "Invalid quantity".

Use Case Description 7: Edit Saved-to-Cook List

Use Case Name:	Edit Saved-to-Cook List
Primary Actor:	Member
Stakeholders:	Member, System
Brief Description:	Member saves their favorite recipe
Trigger:	Member clicks on the "Save" button
Normal Flow of Events:	1. The System modifies the Member's Saved-to-Cook list 1.1. If the recipe is saved, the S-1 subflow Remove Recipe from Saved-to-Cook list is performed. 1.2. If the recipe is not saved yet, the S-2 subflow Add Recipe to Saved-to-Cook list is performed.
Subflow:	S-1: Remove Recipe from Saved-to-Cook list: a. The System removes the recipe from the Member's Saved-to-Cook list b. The System removes the shadow of the Save button (unmark it as unsaved). S-2: Add Recipe to Saved-to-Cook list: a. The System adds it to the Member's Saved-to-Cook list. b. The System adds a shadow to the Save button (mark it as saved).
Exception Flow:	

Use Case Description 8: Edit Cook-Again List

Use Case Name:	Edit Cook-Again List
Primary Actor:	Member
Stakeholders:	Member
Brief Description:	The Member marks a recipe in Planning-to-Cook List as cooked or plan a recipe to cook
Trigger:	The Member clicks on the "Mark cooked" button
Normal Flow of Events:	1. The System modifies the Cook-Again list 1.1. If the recipe is cooked before, the S-1 subflow is performed. 1.2. If the recipe is not cooked before, the S-2 subflow Move the recipe from Planning-to-Cook list to Cook-Again list is performed.
Subflow:	S-1: Remove the recipe from Planning-to-Cook list: a. The System removes the recipe from the Member's Planning-to-Cook list b. The System increases the recipe's cook times by 1 in Cook-Again list S-2: Move the recipe from Planning-to-Cook list to Cook-Again list: a. The System removes the recipe from the Member's Planning-to-Cook list b. The System adds the recipe in Cook-Again list
Exception Flow:	

Use Case Description 9: Search Recipe

Use Case Name:	Search Recipe
Primary Actor:	User
Stakeholders:	System, User
Brief Description:	The User inputs keywords and the System displays requested recipes for the User.
Trigger:	The User clicks on the search bar
Normal Flow of Events:	1. The User inputs desired keywords in the search bar. 2. The User clicks on the "Search" button. 3. The System accepts the input keywords. 4. The System runs a query to search recipes in the Database based on desired keywords. 5. The System displays requested recipes (search results) for the User.
Exception Flow:	

Use Case Description 10: Apply Filter Feature

Use Case Name:	Apply Filter Feature
Primary Actor:	User
Stakeholders:	System, User
Brief Description:	The User applies Filter Feature and the System operates this feature.
Trigger:	The User clicks the filter button
Normal Flow of Events:	1. The System displays filter options for the User. 2. The User selects desired filter options. 3. The User clicks the Complete button for the Filter Feature. 4. The System sets that only recipes which fit the filter options will be showed in search results.
Exception Flow:	

Use Case Description 11: Apply Sorting Feature

Use Case Name:	Apply Sorting Feature
Primary Actor:	User
Stakeholders:	System, User
Brief Description:	The User applies sorting feature and the System operates this feature.
Trigger:	The User clicks the sorting button
Normal Flow of Events:	1. The System displays sorting options for the User. 2. The User selects desired sorting options. 3. The User clicks the Complete button for the Sorting Feature. 4. The System displays search results based on the selected sorting criteria.
Exception Flow:	

Use Case Description 12: Follow Member

Use Case Name:	Follow Member
Primary Actor:	Member
Stakeholders:	System, Member
Brief Description:	Member follows another Member
Trigger:	The Member clicks the Follow button at a recipe page or another Member's profile page.
Normal Flow of Events:	1. The Member clicks the Follow button to follow another Member. 2. The system adds the followed Member to the Member's following list. 3. The system displays the Follow button as Following.
Exception Flow:	The Member can click the Following button again to unfollow another Member; and the system will remove him/her from the Member's following list.

Use Case Description 13: Receive Notification

Use Case Name:	Receive Notification
Primary Actor:	Member
Stakeholders:	System, Member
Brief Description:	The Member receives notifications which the System sends to his/her inbox.
Trigger:	The System sends a message to Members.
Normal Flow of Events:	1. The System sends a message to the Member's inbox. 2. The System displays the number of unread messages on the Inbox button. 3. The Member clicks the Inbox button to access the messages in his/her inbox.
Exception Flow:	

Use Case Description 14: Generate Dictionary

Use Case Name:	Generate Dictionary
Primary Actor:	System
Stakeholders:	System, Member
Brief Description:	Generate word dictionary from Member's data
Trigger:	Routine operator
Normal Flow of Events:	1. The System sends a query to fetch the data of all the searching records from members. 2. The System creates a dictionary for the whole system.
Exception Flow:	

Use Case Description 15: Generate Recommendation

Use Case Name:	Generate Recommendation
Primary Actor:	TensorFlow API
Stakeholders:	TensorFlow API, System, Member, Database
Brief Description:	Generate recommendations based on dictionary
Trigger:	Member logs in
Normal Flow of Events:	1. TensorFlow API accesses dictionary. 2. Recommendation accesses dictionary, Member's search history, Member's E-Fridge, and Member's recipes. 3. Recommendation system implements Word2Vec technique to generate the related keywords for Member. 4. Recommendation system returns keywords to Cookpad system. 5. The System runs the search query based on the keywords from step 3. 6. The System displays the recommendations on Member's main page.
Exception Flow:	2a. If the executive time exceeds a certain duration, the System displays the message "RuntimeError".

Data Dictionary

Use Case Name: Log In

Email = Data Element

Password = Data Element

Use Case Name: Set Up E-Fridge

Ingredient list = 1{Ingredient name + Quantity + Date}

Ingredient name = Data Element

Quantity = Data Element

Date = Data Element

Use Case Name: Create Recipe

Recipe data = Recipe ID + (Photo)+ Title + (Description) + (Serves) + (Cook time) + 1{Ingredients} + Method

Photo = Data Element

Title = Data Element

Description = Data Element

Serves = Data Element

Cook time = Data Element

Ingredients = Data Element

Method = 1{Step}

Step = Instruction + (Photo)

Instruction = Data Element

Use Case Name: Publish Recipe

Recipe data = Recipe ID + (Photo)+ Title + (Description) + (Serves) + (Cook time) + 1{Ingredients} + Method

Photo = Data Element

Title = Data Element

Description = Data Element

Serves = Data Element

Cook time = Data Element

Ingredients = Data Element

Method = 1{Step}

Step = Instruction + (Photo)

Instruction = Data Element

Use Case Name: Edit Saved-to-Cook List

Saved-to-Cook List = (Photo) + Title

Recipe data = Recipe ID + (Photo)+ Title + (Description) + (Serves) + (Cook time) + 1{Ingredients} + Method

Photo = Data Element

Title = Data Element

Description = Data Element

Serves = Data Element

Cook time = Data Element

Ingredients = Data Element

Method = 1{Step}

Step = Instruction + (Photo)

Instruction = Data Element

Use Case Name: Edit Cook-Again List

Cook-Again List = (Photo) + Title + Cook times

Recipe data = Recipe ID + (Photo)+ Title + (Description) + (Serves) + (Cook time) + 1{Ingredients} + Method

Photo = Data Element

Title = Data Element

Description = Data Element

Serves = Data Element

Cook time = Data Element

Ingredients = Data Element

Method = 1{Step}

Step = Instruction + (Photo)

Instruction = Data Element

Use Case Name: Edit Planning-to-Cook List

Planning-to-Cook List = (Photo) + Title

Recipe data = Recipe ID + (Photo)+ Title + (Description) + (Serves) + (Cook time) + 1{Ingredients} + Method

Photo = Data Element

Title = Data Element

Description = Data Element

Serves = Data Element

Cook time = Data Element

Ingredients = Data Element

Method = 1{Step}

Step = Instruction + (Photo)

Instruction = Data Element

Use Case Name: Search Recipe

Keywords = 1{Word}10

Word = Data Element

Use Case Name: Apply Filter Feature

Filter options = (Prepare time) + (Popularity) + (Post time)

Prepare time = [1-15 minutes|15-30 minutes|30-45 minutes| 45-60 minutes|60+ minutes]

Popularity = [< 5 reaction| 5-10 reaction| 10-15 reaction| 15-20 reaction| 20+ reaction]

Post time = [Today| One week| One month| Three month| This year]

Use Case Name: Apply Sorting Feature

Sorting options = (Prepare time) + (Popularity) + (Post time)

Prepare time = [Longest to shortest| Shortest to longest]

Popularity = [Most to least| Least to most]

Post time = [Latest to oldest| Oldest to latest]

Use Case Name: Follow Member

Member's following list = 0{Member}

Member = Name + Email

Name = First Name + Last Name

Email = Data Element

Use Case Name: Receive Notification

Member's inbox = 0{System message}

System message = Data Element

Use Case Name: Generate Dictionary

Dictionary = 0{Word + Word Index}

Word = Data Element

Word Index = Data Element

Use Case Name: Generate Recommendation

Recommendation = 0{Recommended recipe}

Recommended recipe = Recipe data

Recipe data = Recipe ID + (Photo)+ Title + (Description) + (Serves) + (Cook time) + 1{Ingredients} + Method

Photo = Data Element

Title = Data Element

Description = Data Element

Serves = Data Element

Cook time = Data Element

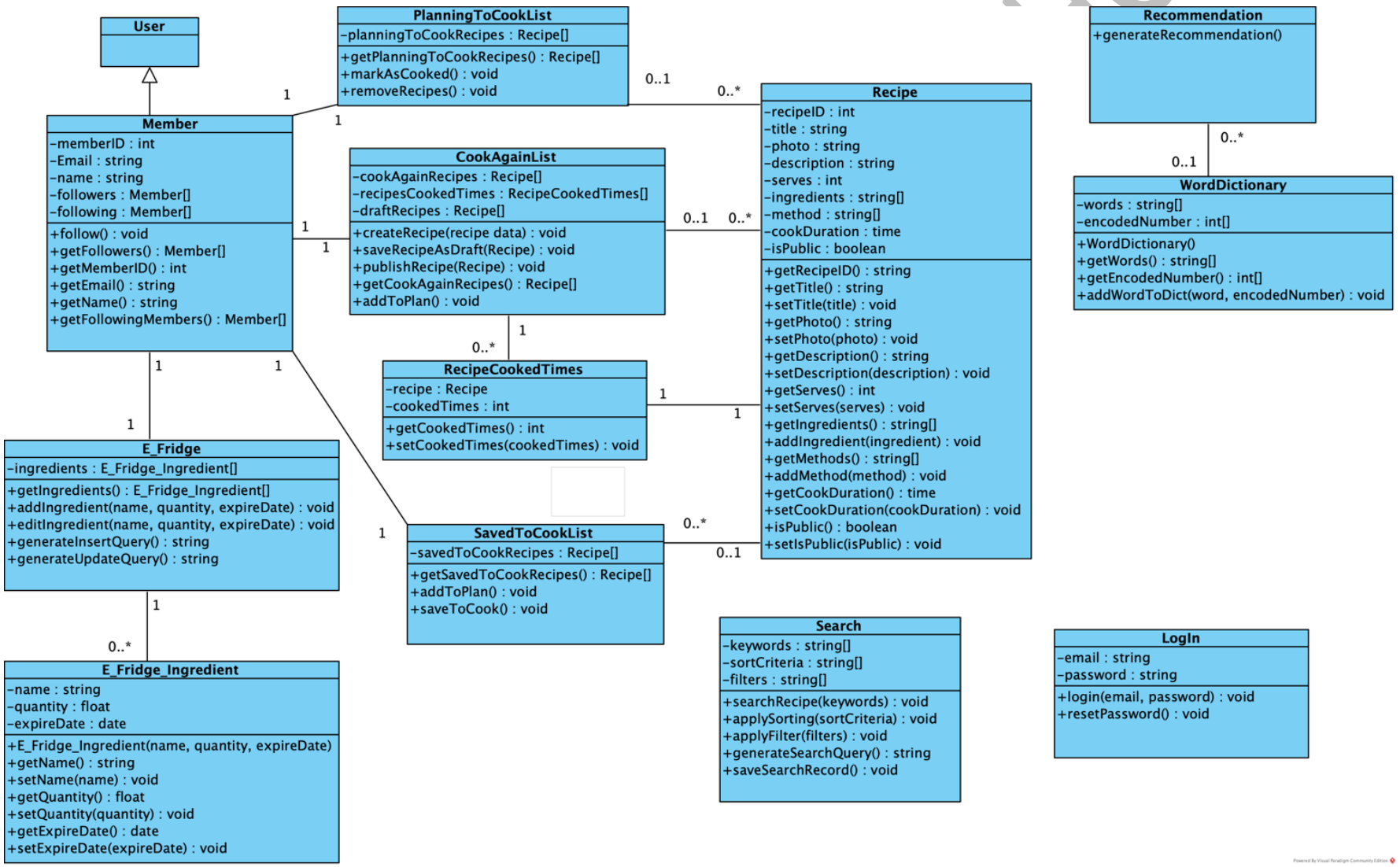
Ingredients = Data Element

Method = 1{Step}

Step = Instruction + (Photo)

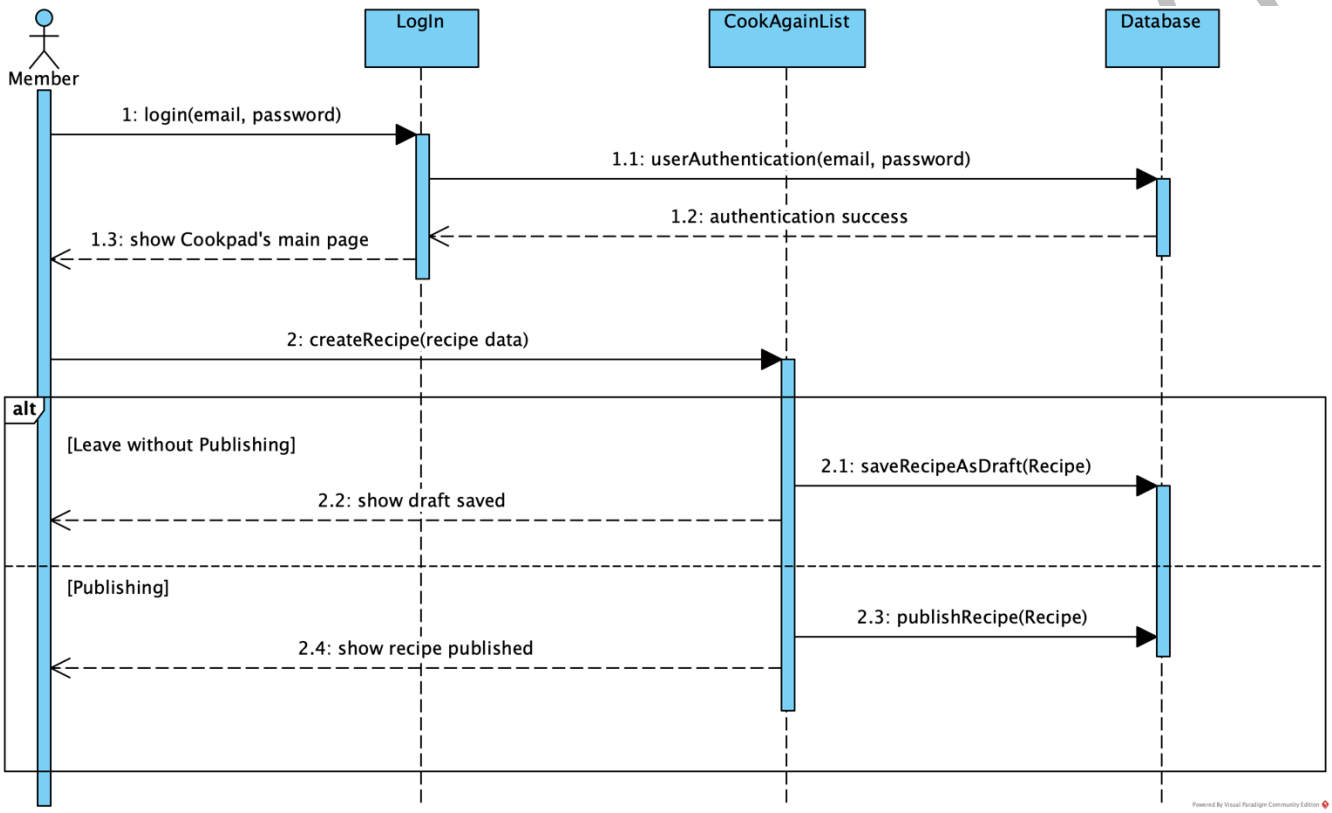
Instruction = Data Element

Class Diagram

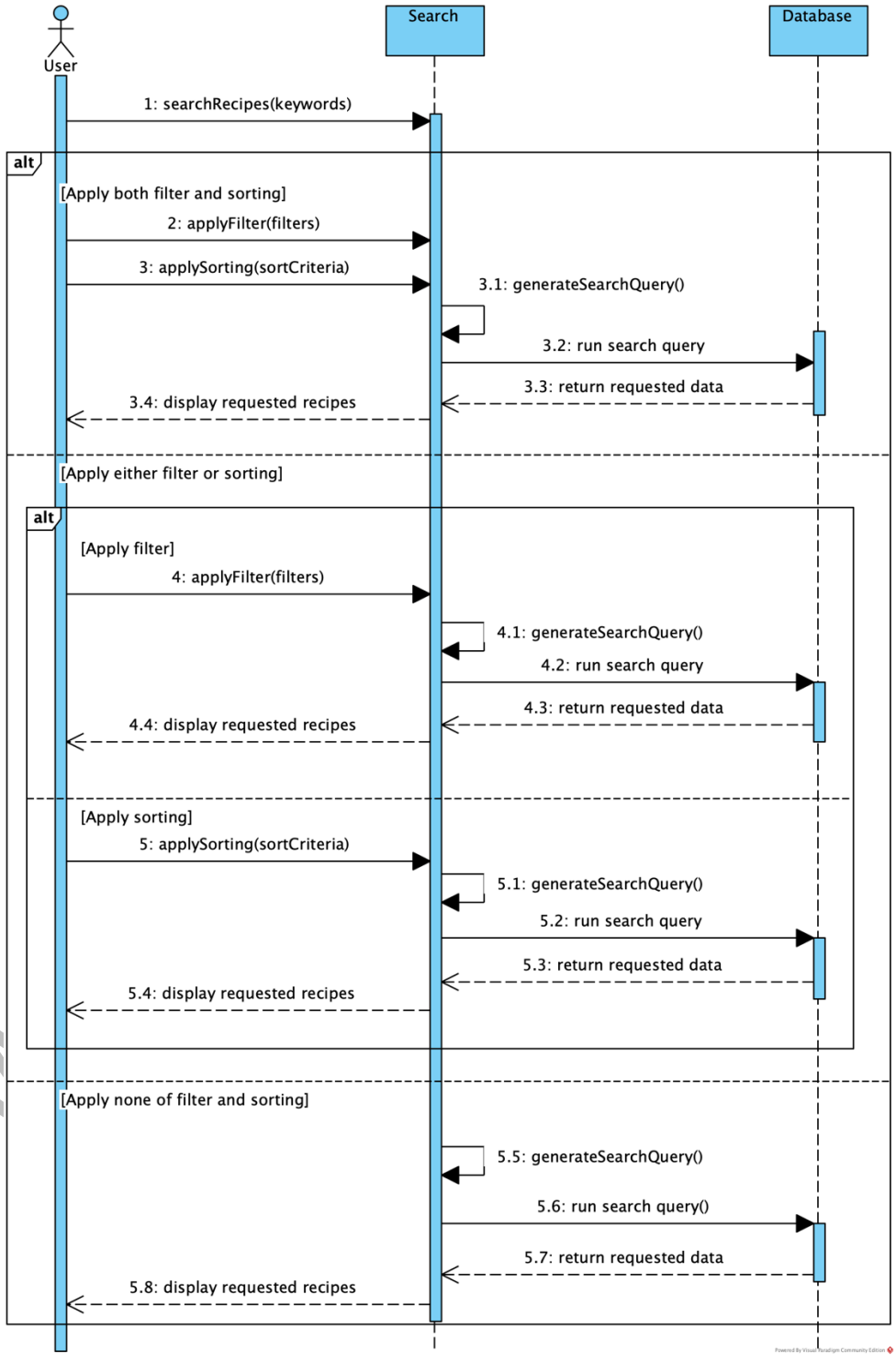


Sequence Diagram

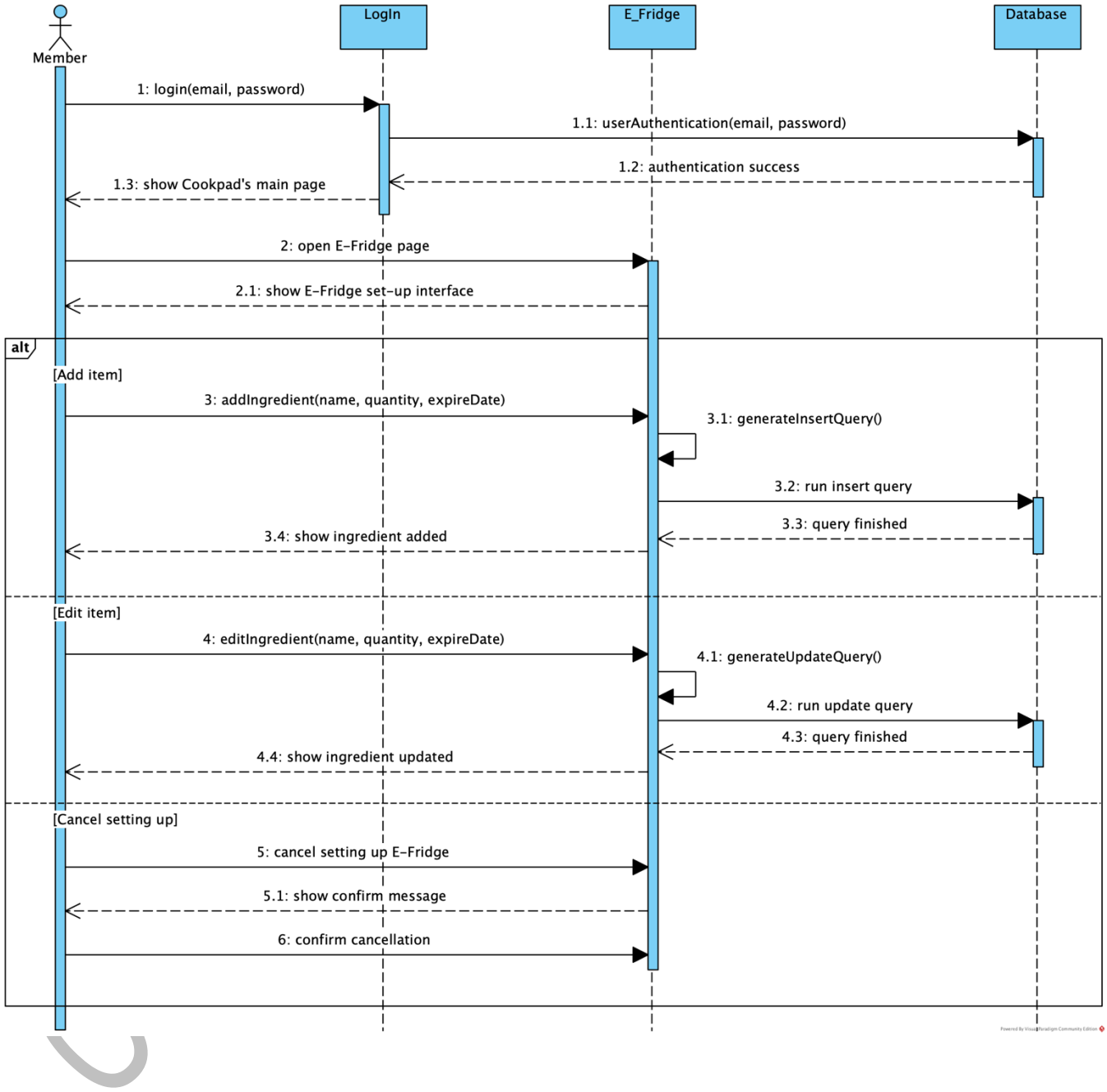
Create to Publish Recipe



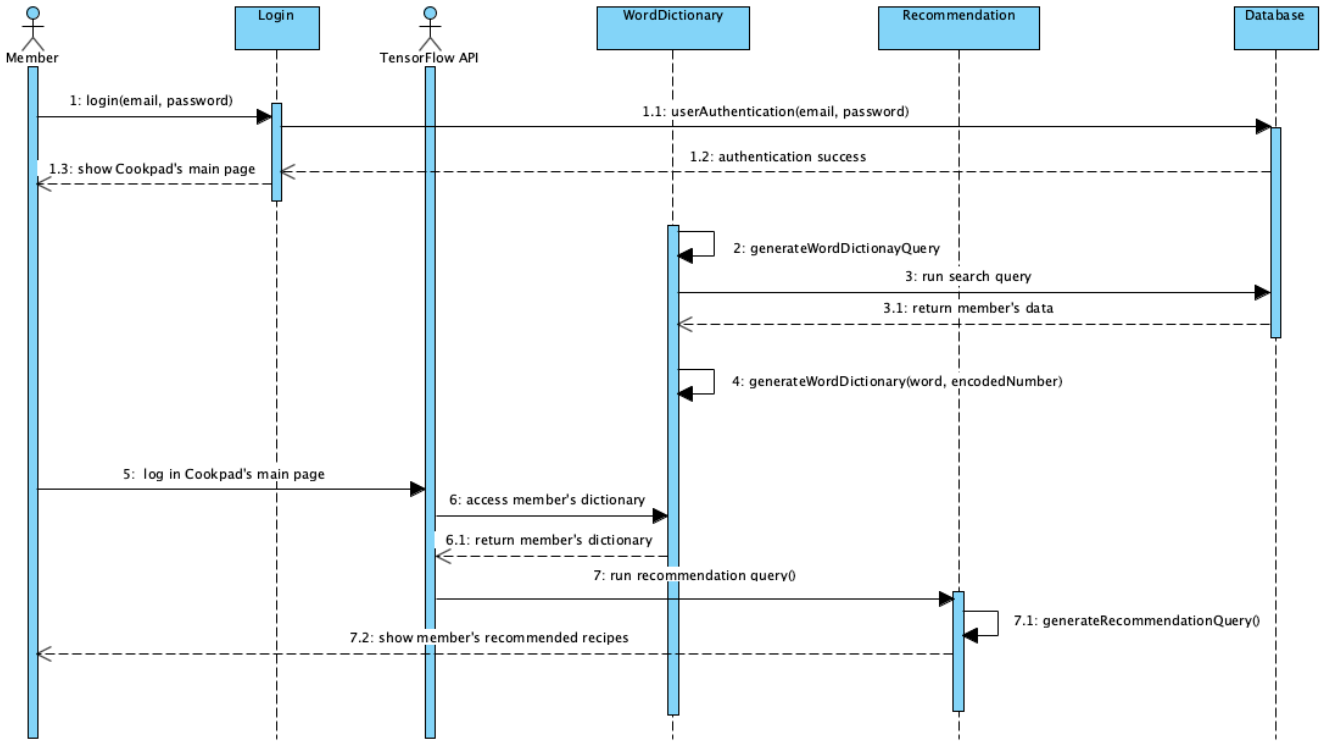
Search Recipes



Set Up E-Fridge



Generate Recommendation



Functional Specification

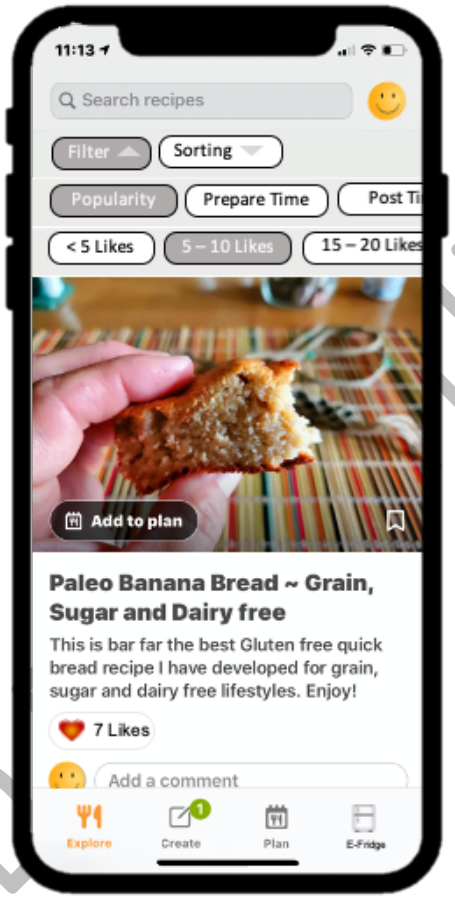
- Cookpad App will allow users to apply filter and sorting features when searching recipes on the platform.
    - Users will be able to use filter and sorting features separately and together.
    - Users can also choose to search without those features.
    - The filter and sorting features include options such as popularity (numbers of likes), prepare time, post time of recipes.

- Cookpad App will have the E-Fridge feature which lets members keep track of their own ingredients and provides data to generate recommendations.
    - The E-Fridge should allow members to input the ingredients they currently have along with ingredients' quantity and expiration date.
    - Data of Ingredients in E-Fridge will be collected periodically for building a word dictionary for generating recommendations.
    - The members can remove ingredients from E-Fridge by cooking recipes which use them or manually delete them.

- Cookpad App will be integrated with Tensorflow packages to provide recipe recommendations for individual members based on their own E-Fridge ingredients and search history.
    - Recipe recommendations will be generated through the Natural Language Processing algorithm in Tensorflow packages given members' data.
    - Recipe recommendations will be shown on members' main page after logging in to Cookpad App.

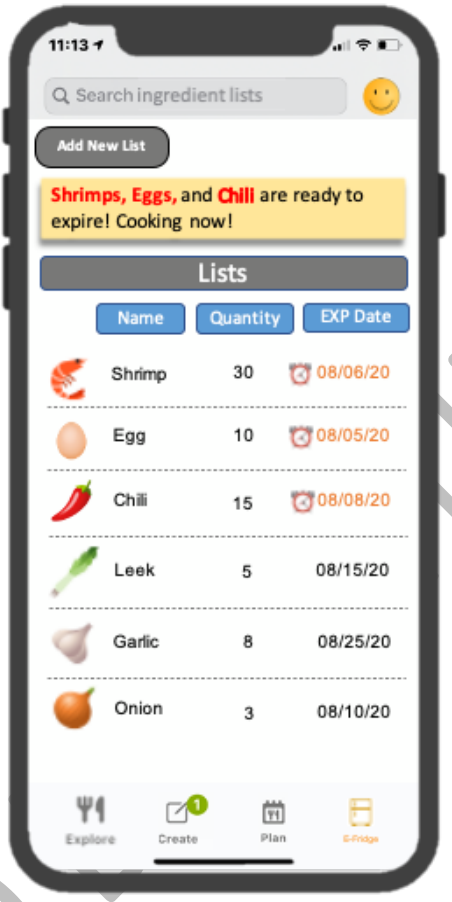
Interface Design

Sorting and Filter Feature



After tapping on the search recipes bar and inputting keywords, the user can select preferred “filter options”, “sorting options”, and both to search specific recipes. The system has three main options for users to choose, which are “Popularity”, “Prepare Time”, and “Post Time”. And each of the three main options has specific options such as how much likes for “Popularity”, how much preparation time for “Prepare Time”, and how long the recipes post for “Post Time”. Take the picture as an example, the user search banana bread by filtering “popularity” based on recipes received 5 to 10 likes. What’s more interesting is that the user can easily select those options by swiping the screen.

E-Fridge Feature

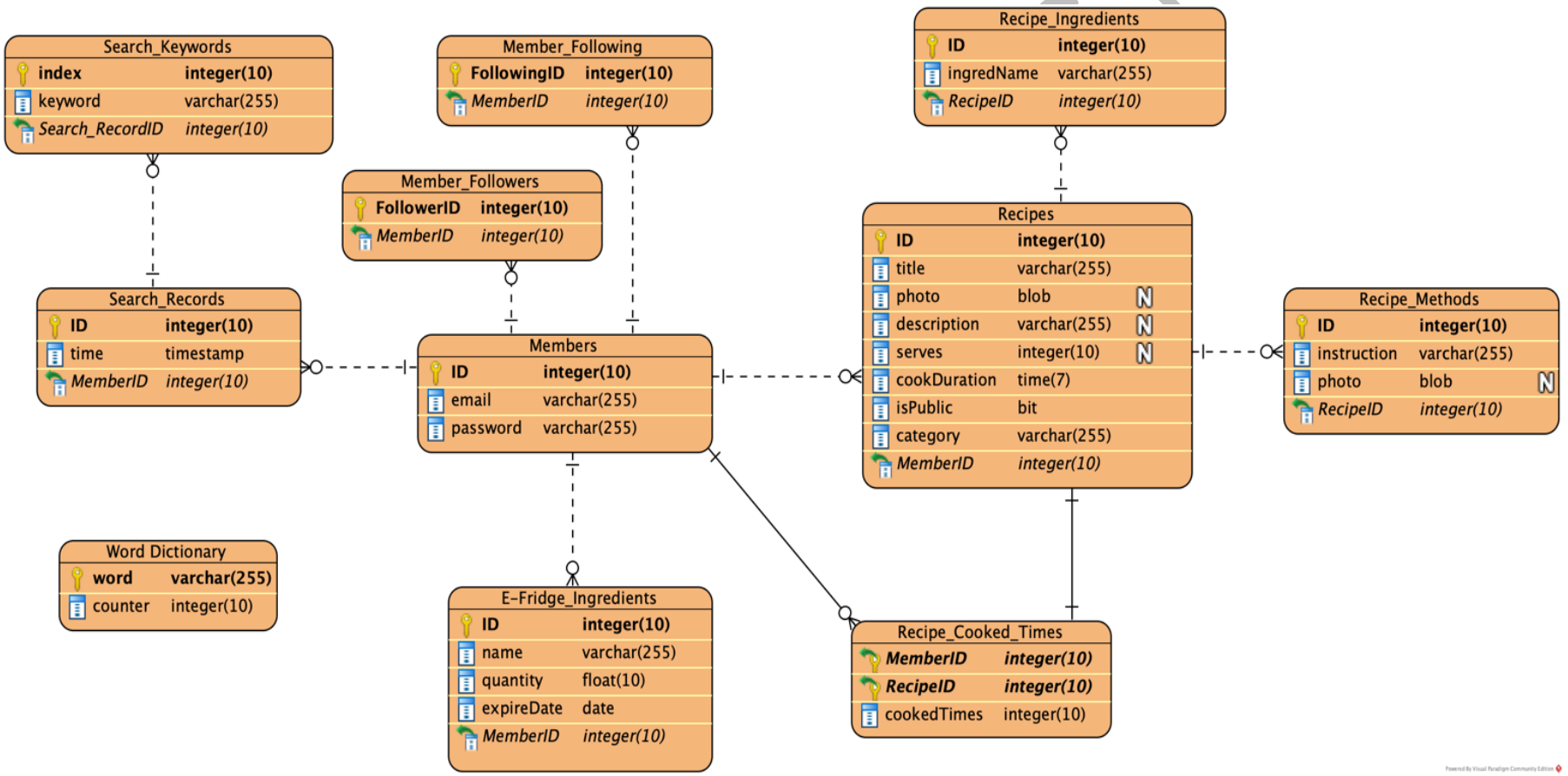


The member can set up E-Fridge function at the bottom of the Cookpad main page. The member can set up the E-Fridge function on the bottom of the Cookpad main page. After adding ingredient lists, the member can review what ingredients and their quantities stored in the fridge and which ingredients are going to expire. Also, the E-Fridge has a short summary for the member every time when the member opens the E-Fridge page. For instance, the photo above not only shows that the ingredients name, quantity, and expiration date, but also three ingredients are ready to expire within this week.

Recommendation Feature

As member logs in the app, he/she will immediately get the today's recommendation with the "Yummy" and "Dislike" emojis right below the picture of the recipe. By clicking on the picture, the card will flip around, and show the details. Membership users can either click on the emoji or swipe the cards to show their preferences. Through this process, users are actually training the recommendation system, making the app "smarter" and bring more relative recipes up from the deck.

Database Design



Software Design

create_recipe()

```
function create_recipe(title, photo, description, serves,
                       ingredients[], method[], cook_duration,
                       to_publish):

    construct a Recipe object (recipe) with input parameters

    generate INSERT SQL command for saving recipe

    if to_publish is TRUE:
        save recipe to cookAgainRecipes[]
    else:
        save recipe as a draft to draftRecipes[]

    run INSERT query on database
```

search_recipes()

```
function search_recipes(keywords[], isSortingApplied,
                        isFilterApplied):

    initialize sortCriteria[] as an empty list
    initialize filters[] as an empty list

    if isSortingApplied is TRUE:
        get selected sorting criteria
        save sorting criteria to sortCriteria[]

    if isFilterApplied is TRUE:
        get selected filters options
        save filters options to filters[]

    generate SQL query to search recipes based on keywords[],
                sortCriteria[], and filters[]
    run search query on database
    obtain query result
    display query result

    save search record to database
```

setup_E_Fridge()

```
function setup_E_Fridge(action):

    get member's E-Fridge settings and data
    show member's E-Fridge page

    while action is not "cancel":

        if action is "add":
            show input interface
            get member's input data
            generate INSERT SQL query
            run query on database
            show "ingredient added" message

        if action is "edit":
            get selected E-Fridge ingredient
            show edit interface
            get the updated ingredient data
            generate UPDATE SQL query
            run query on database
            show "ingredient updated message"

        set action as "add", "edit", or "cancel" based on member's
                choice

    close E-Fridge page
```

generate_word_dictionary()

```
function generate_word_dictionary():
                fetch keyword from Search_Keywords
                initialize dictionary to an empty dictionary
                initialize counter to zero

                for word in keyword:
                    if word is not in dictionary:
                         add word to dictionary as a key
                         set the value of word to counter
                         add one to counter
                    else:
                         pass

                return dictionary
```

Generate_recommendation()

```
function generate_recommendation(memberid, dictionary):
                fetch titles from Recipes
                set neighboring_size to two
                initialize data to an empty list

                for title in titles[]:
                    for word in title:
                         while neighbor within nighboring_size:
                              generate a word_pair with its neighbor
                              append word_pair to data

                convert data to integer by looking up the dictionary
                convert data to one-hot vectors

                call tensorflow api
                input data to word2vec module
                set vectors to the result from word2vec

                if MemberID is memberid:
                    fetch titles[] from Recipes
                    fetch ingredName from E-Fridge_Ingredients
                    fetch keyword from Search_Records

                for word in titles, ingredName, and keyword:
                    for vector in vectors:
                         caculate distance between word and vector

                set top_ten to the top ten closest words
                return top_ten
```

References

1. Cookpad app from AppStore
2. Cookpad Wikipedia, https://en.wikipedia.org/wiki/Cookpad
3. Term of Service by Cookpad, https://cookpad.com/us/terms
4. Word embeddings on TensorFlow tutorials, https://www.tensorflow.org/tutorials/text/word_embeddings

Minutes of Project Meetings

1.

Datetime: 6/15 22:00 – 23:30

Attention: All

Topic: Decide the data and topic

Upcoming: Context diagram and use case diagram

Tentative Meeting Date: 6/22

2.

Datetime: 7/22 20:30 – 22:00

Attention: All

Topic: Context diagram and use case diagram

Upcoming: Use case description

Tentative Meeting Date: 6/30

3.

Datetime: 6/30 15:00 – 16:00

Attention: All

Topic: Use case diagram modification

Upcoming: Use case description

Tentative Meeting Date: 7/6

4.

Datetime: 7/6 15:00 – 16:00

Attention: All

Topic: Finalize use case diagram and discuss on data dictionary

Upcoming: Class diagram

Tentative Meeting Date: 7/13

5.

Datetime: 7/13 15:00 – 16:00

Attention: All

Topic: Go over data dictionary and discuss on class diagram

Upcoming: Review the modifications

Tentative Meeting Date: 7/18

6.

Datetime: 7/18 15:00 – 16:00

Attention: All

Topic: Discuss on the class diagram

Upcoming: Discuss on sequence diagram

Tentative Meeting Date: 7/25 13:00

7.

Datetime: 7/25 13:00 – 14:00

Attention: All

Topic: Discuss on the class diagram and sequence diagram

Upcoming: Discuss on the database design

Tentative Meeting Date: 7/30 15:00

8.

Datetime: 7/30 15:00 – 16:00

Attention: All

Topic: Discuss on the database design and interface design

Upcoming: Finalize the documentation

Tentative Meeting Date: 8/1 15:00

9.

Datetime: 8/2 16:00 – 17:00

Attention: All

Topic: Discuss on the 1st version report

Upcoming: Modify and finalize the repot

Tentative Meeting Date: 8/3 15:00

10.

Datetime: 8/3 16:00 – 17:00

Attention: All

Topic: Discuss on the final version report

Upcoming: Modify and finalize the repot

Tentative Meeting Date: None