Software Requirements Specification

for

<EasyFood>

Version 2.0 approved

Prepared by <Team HTTPS>

<06th November 2022>

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Anusha Agarwal	10/9/22	Incremental Development	1.0
Anusha Agarwal	06/11/22	Final	2.0

1. Introduction

1.1 Purpose

EasyFood is an open source Android mobile application that provides multiple services to local customers such as being able to search and save their favorite recipes and restaurants. EasyFood aims to serve the user by taking into account all combinations of users' dietary requirements, preferred calorie intake, and cuisine of choice. Users can also add reviews to a restaurant after visiting. EasyFood allows users to see the route to their chosen restaurant from their current location, by using Geolocation API and Google Map. It acts as an intermediary between restaurants in Singapore and consumers.

Project Mission Statement –
The app that brings you closer to your favorite restaurant by either suggesting a restaurant that will be a perfect fit for your mood or a recipe that you can cook on your own. Foodies from all over Singapore can use this app to plan for whatever cuisine they want to have, whenever they want to have it.

This document serves to define the software and hardware requirements of the application – EasyFood. It includes relevant information such as product purpose, scope, target audience, design considerations during planning stages to help understand the functionality of the app.

1.2 Document Conventions

1.2.1 Priorities for Higher-Level Requirements

Priorities for higher-level requirements are assumed to be inherited by detailed requirements, where applicable.

1.2.2 Typographical Conventions

UML	Unified Modeling Language
API	Application Programming Interface
Millennials	A person born in the 1980s or 1990s
Baby Boomer	A person born between 1946 and 1964

SDK	Software Development Kit

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, both front end and back end to understand the functionalities of the app. Since EasyFood is an open source application, developers are welcome to review, suggest and add new functionalities to the app. This document clearly states the scope, functional and non-functional requirements, as well as UML diagrams to understand the structure of EasyFood application.

Project testers shall use this documentation for building test strategies. This documentation provides users with methodically-organized testing processes such as through white box and black box testing. Back-end engineers may use this documentation to understand the entity classes such as database, and other API's used in this application. The API's are encapsulated and well-documented for users to implement. The application also uses the principle of dependency to support the same.

1.4 Users and Stakeholders

1.4.1 Target Audience

EasyFood aims to target the Millennials and Baby Boomer generation of Singapore who are not as tech savvy as Gen-Z and get overwhelmed by too complicated apps which offer too many functionalities. These users can appreciate the user friendliness and simplicity of EasyFood in contrast to its competitors.

Any new user can figure out the functionalities of EasyFood within 15 seconds.

EasyFood uses focus strategy from Porter's competitive strategies by providing limited features focused towards the needs and wants of targeted market segments.

1.4.2 Stakeholders

EasyFood collects data from external APIs such as Spoonacular for recipes, and Kaggle dataset for restaurant information. The app also uses other interfaces such as Geolocation API, Google map and Firebase. Since users can add reviews on a restaurant they visited, local Singapore restaurants might also be affected by the application.

1.5 Product Scope

EasyFood is a food app and as the name suggests makes it easier for users to find food of their choice in a fun and easy manner. EasyFood mainly focuses on two aspects –

(i) Suggests recipes based on criteria input by user such as calorie intake, ingredients available. It allows users to save the recipe and refer to it at a later time as well.

(ii) Suggests restaurant based on criteria input by user such as choice of cuisine, dietary requirement, and distance from current location. EasyFood also allows users to add reviews to a particular restaurant and also read reviews written by others before visiting. It also shows the route to the restaurant using Google map and shows the estimated time taken to reach there considering the present traffic conditions.

EasyFood aims to unite people with their favorite food in a seamless and noise free manner.

1.6 Data Dictionary

Term	Definition
User	Any Person who wishes to use the restaurant app service
Gmail acc/ username	A unique identifier assigned to users during registration. It is used during the login process to authenticate users.
Main Menu	The menu which appear after sign up which allows users to choose between recipes or restaurants
Recipe Search	A feature that allows users to search for desired recipes using keywords and criteria input by user
Restaurant Search	A feature that allows users to search for their desired restaurants using criteria input by user
Filter	A feature that allows users to sort restaurants or recipes based on certain criteria
Star	Star icon which when clicked saves or removes recipe/restaurant from saved list.
Saved Restaurants	A feature that allows users to add/retrieve restaurants to a ‘Saved’ list to allow for easier access later.
Saved Recipes	A feature that allows users to add/ retrieve recipes to a ‘Saved’ list to allow for easier access later
Restaurant Review	Optional comments on the restaurants written by users after visiting the restaurant. These are available for other customers to see as well
Calorie Count	Refers to the amount of calories provided by a recipe
Distance	Refers to the distance between users’ current location and chosen restaurant
Dietary Preference	Refers to the type of food the user wants to eat in the restaurant such as vegan/Dessert/ BBQ etc.

1.7 References

[1] *Spoonacular API | Spoonacular API* [Online] Available: https://spoonacular.com/food-api Accessed [13th of October 2022]
[2] "Android Developers," Android Developers. [Online]. Available: https://developer.android.com/. [Accessed: 15-Oct-2022].
[3] Google Maps Platform | Google Developers. [Online]. Available: https://developers.google.com/maps/documentation/. [Accessed: 15-Oct-2022].
[4] 16000 Grab Restaurants in Singapore. [Online]. Available: https://www.kaggle.com/datasets/polartech/16000-grab-restaurants-in-singapore [Accessed: 16-Oct-2022]
[5] Firebase Authenticator. [Online]. Available: https://firebase.google.com/products/auth?gclid=CjwKCAjw8JKbBhBYEiwAs3sxN6T9N39CGc3Tz212qFCbHMOfRwPcAWWUrrp4_SWnkUCn5mCnjYa92xoCEAkQAvD_BwE&gclsrc=aw.ds [Accessed: 12-Oct-2022]
[6] Firebase Cloud Firestore. [Online]. Available: https://firebase.google.com/products/firestore?gclid=CjwKCAjw8JKbBhBYEiwAs3sxN0WGoKG1DgYPWg8aGbXS0TrCHZbt7gUY-mrBcvHsqjH3mqHbO4k5lRoC5-sQAvD_BwE&gclsrc=aw.ds [Accessed: 12-Oct-2022]
[7] Geolocation API | Google Developers. [Online]. Available: https://developers.google.com/maps/documentation/geolocation/overview [Accessed: 15-Oct-2022]

2. Overall Description

2.1 Product Perspective

EasyFood is a self contained, open source product developed and deployed on Android platform. Its main features include suggesting suitable restaurants and recipes to users as per their requirement. It acts as a personal food diary where users' can save their favorite recipes. The usability of this app connects users and local restaurants. The app takes into account weird food combinations and dietary requirements and makes informed suggestions.

2.2 Product Functions

- 1. Sign in using Gmail
    - 1.1Validation of account
- 2. Retrieve recipes
    - 2.1 Save favorite recipe
    - 2.2 Retrieve list of saved recipes
    - 2.3 Remove recipe from saved list
    - 2.4 Retrieve recipe information (using Spoonacular API)
- 3. Retrieve restaurants
    - 3.1 Save favorite restaurant
    - 3.2 Retrieve list of saved restaurants
    - 3.3 Remove restaurant from saved list
    - 3.4 Retrieve restaurant's information (Using Kaggle dataset)
    - 3.5 Add review for restaurants
    - 3.6 Retrieve location from user's device (Using Geolocation API)
    - 3.7 Show direction to restaurant (Using Google Map)

2.3 User Classes and Characteristics

Our user class majorly comprises Singaporeans/ People in Singapore looking to find food that matches their requirements. EasyFood is very easy to use and only requires novice level knowledge of mobile apps.

2.4 Operating Environment

EasyFood is built on Android Studio using Flutter. It uses external interfaces such as Firebase, Firestore, Spoonacular API, Kaggle Dataset, Geolocation API and Google Maps.

It operates on mobile devices with Android operating system installed. Since its Minimum API Level is 21, the application will work on Android SDK packages with Version 5.0 and above. This shows the app is well adapted. The app can co exist with other apps without creating any issues.

2.5 Design and Implementation Constraints

- The app apk is about 24 MB, and requires the user to have at least 25 MB memory free for installation.
- The app is currently in English.
- The app currently takes into account only restaurants in Singapore.
- Database used: Firestore.

2.6 User Documentation

The application targets the general public. The app is user friendly with a seamless flow. It is designed with an interactive User Interface that is intuitive and allows any new user to figure out the workings of the app within seconds. Hence, users do not need documentation to understand the app.

2.7 Assumptions and Dependencies

Some of the assumptions taken into account are

- The user must have a stable internet connection
- The user has an android device with sufficient memory
- The user allows the app to access their device's live GPS location
- The accuracy of information of the app depends on accuracy of the data streamed in using Geolocation API and Google Maps.
- User has Android operating system installed with version >5.0
- The user must be planning to use the app in Singapore

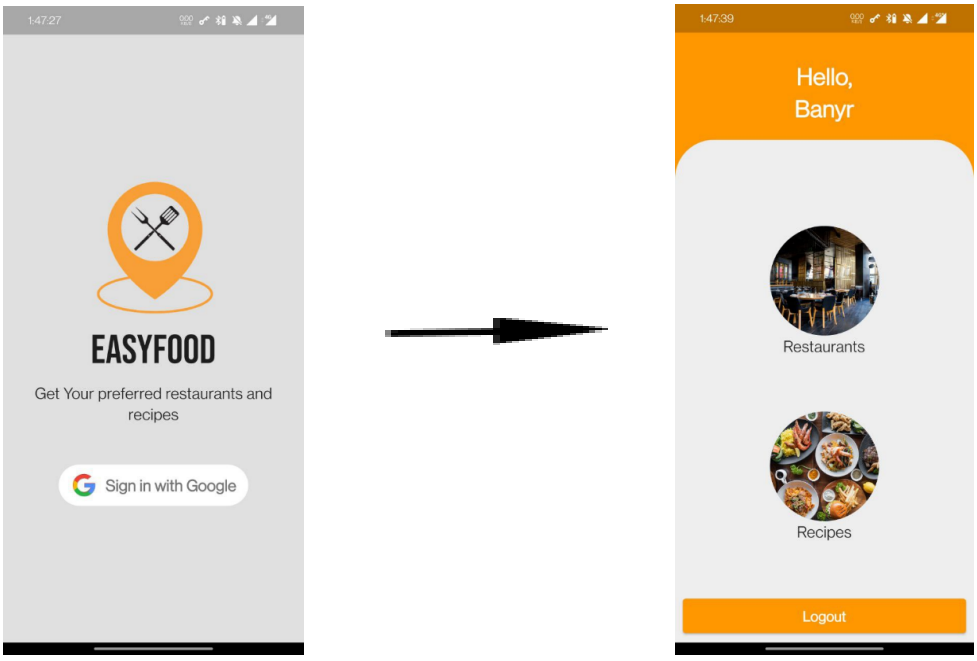
2.8 Future Extensions:

The app uses dependency inversion principle such that developers can change external interfaces as required without having to make changes in the core code. The classes have loose coupling and high cohesion to support the same.

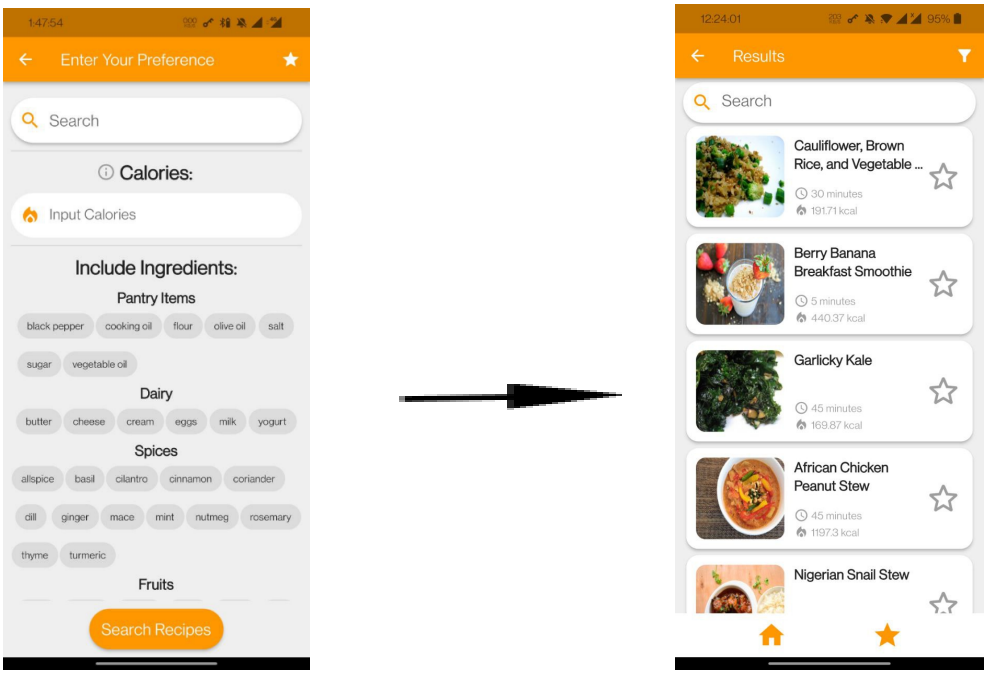
3. External Interface Requirements

3.1 User Interfaces

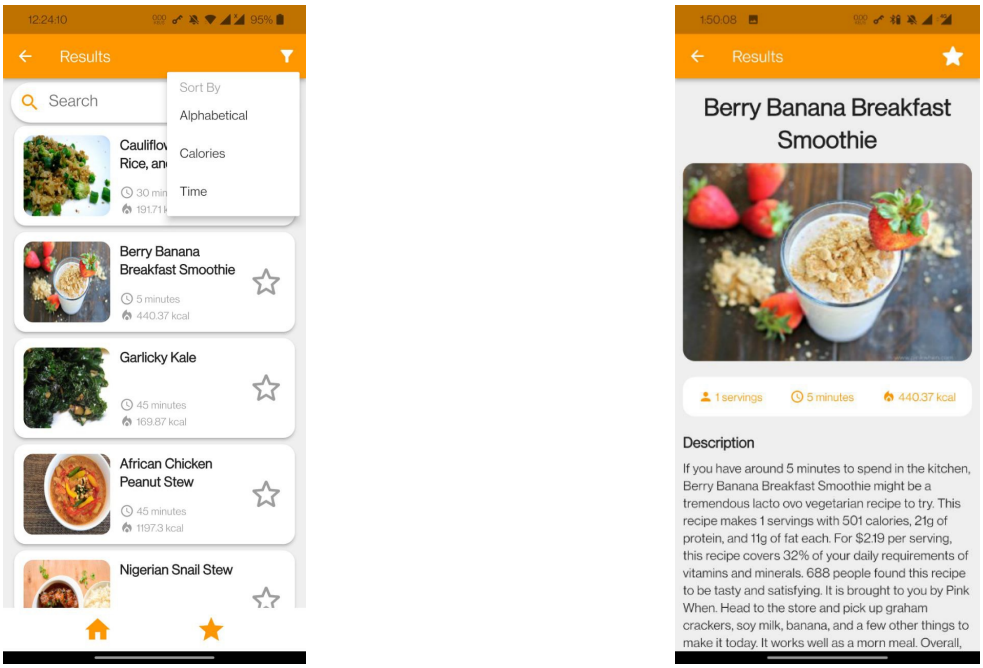
3.1.1 This is the welcome page when a user first opens the app. It asks the user to sign in with their gmail account. It then redirects the user to the main menu with a personalized hello message. It allows users to choose between restaurants and recipes.



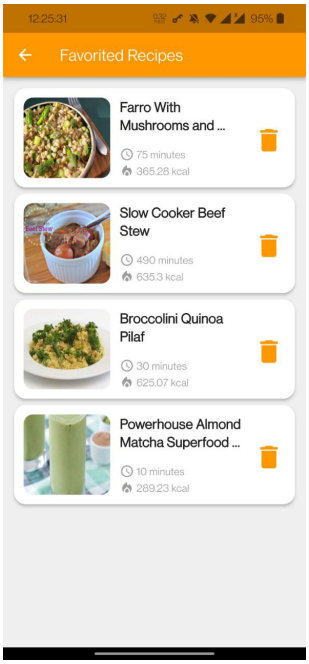
3.1.2 When the user clicks on the recipe icon, it redirects the user to a page where they can fill their required criteria such as Calories, Available ingredients. The app then shows a list of recipes matching the criteria.



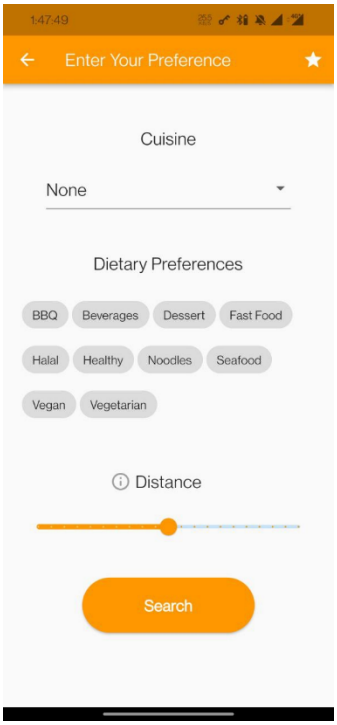
3.1.3 Users can sort the list by alphabetical order, time duration taken or preferred calorie intake. The app then sorts the list. The user can choose a recipe and click on it to see the description and procedure to make the dish.

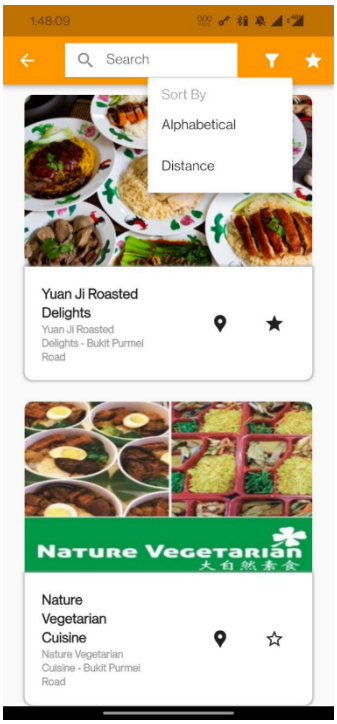


3.1.4 Users can click on the star beside the recipe to add it to their favorites. The use can also open this list to reference at a later time :

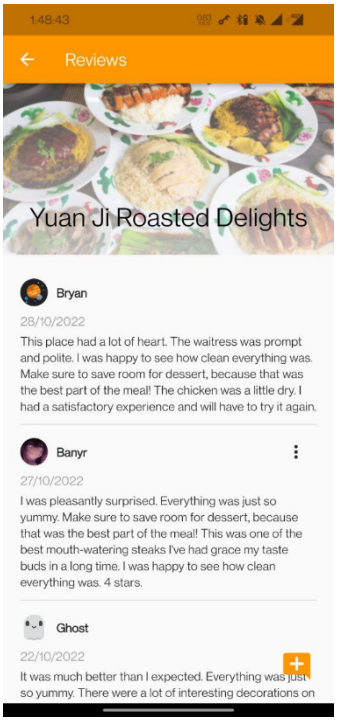


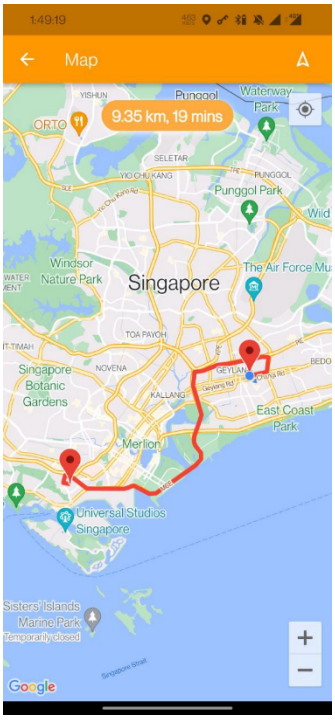
3.1.5 If a user clicks on a restaurant, the page below is shown where they can input criteria such as cuisine, Dietary Preference, and Distance radius from current location. The app then displays a list of restaurants matching the criteria. It also allows users to sort the list using alphabetical order or distance.



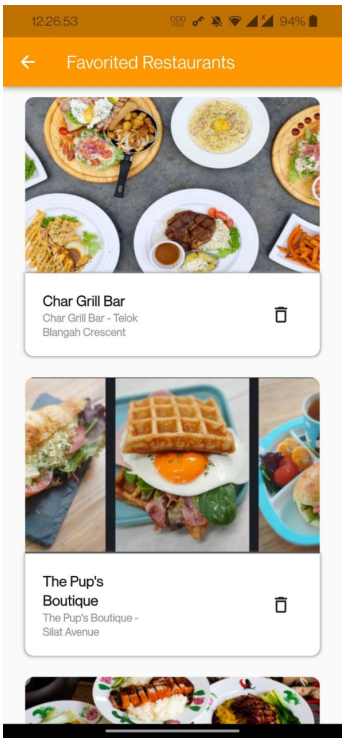


3.1.6 Users can add reviews to a restaurant/ read other reviews by clicking on the restaurant. They can also click on the location icon to see the route to the chosen restaurant.





3.1.7 EasyFood allows users to add restaurants to their favorite list:



3.2 Hardware Interfaces

EasyFood is currently responsive and can be installed and used on Android devices. It is mainly based on touch screens where direct manipulations are supported. Users can touch the screen, tap on buttons to manipulate on screen objects or methods. The response to user input is conducted after when users finish typing on the virtual keyboard. Internal hardware such as GPSmudules and WiFi modules are also used.
If the user has Google Map installed, the app opens the route to the restaurant using the app otherwise it opens it on a web interface.

3.3 Software Interfaces

The app is built on Android Studio with the help of Flutter and Dart Language. EasyFood is built with structured layout objects and UI controls that allows users to operate on GUI of application.
Database used - Firestore
External API's used -

Interface Used	Purpose
Google Map API	Displays Google Map within the application and shows the route to chosen restaurant from current location extracted through Geolocation API
Geolocation API	Extracts the current location of users' device after getting permission from user
Firebase	Allows app to authenticate user login via gmail account
Spoonacular API	App uses this API to extract information about recipes
Kaggle Dataset	App uses this dataset to retrieve information about Restaurants in Singapore
Firestore	Firestore is a cloud data storage platform to store, retrieve and update user information. Acts as database (used in entity class) to store information about user, their favorites, recipes and restaurants.

3.4 Communications Interfaces

EasyFood is developed to work on an android mobile software, and all the aforementioned features are available through the app. Class diagram, Dialog Map and Sequence Diagram depict clearly how objects and classes interact among each other.

4. System Features

4.1 Sign up

4.1.1 The user must be able to sign in using google account.
4.1.2 The external interface must validate the user's google account
4.1.3 The system will redirect users to the application main menu.

4.2 Main Menu

4.2.1 The system must display the options to choose between "Recipes to cook" and "Nearby Restaurant" to the user.
4.2.2 The system must redirect the user to the page according to the option chosen

4.3 Recipes Search

4.3.1 The user must be able to input criteria to retrieve recipes
    4.3.1.1 Criteria include:
        4.3.1.1.1 Available ingredients
        3.3.1.1.2 Calorie count
        4.3.1.1.3 Preferred Nutrients
4.3.2 The user must be able to sort the list using:
    4.3.2.1 Alphabetical order
    4.3.2.2 Calorie count
    4.3.3.3 Time duration

4.4 Save Recipes

4.4.1 The user must be able to save the recipes they like for future references
4.4.2 The user must be able to retrieve the list of saved recipes
4.4.3 The user must be able to remove their saved recipes

4.5 Restaurant Search

4.5.1 The system must be able to display information of the restaurant according to criteria inputted by the user –
    4.5.1.1 Criteria include:
        4.5.1.1.1 Cuisine

4.5.1.1.2 Dietary Preferences
4.5.1.1.3 Distance from current location
4.5.2 The user must be able to sort the displayed restaurants using :
4.5.2.1 Alphabetical order
4.5.2.2 Calorie count

4.6 Save Restaurants

4.6.1 The user must be able to save the restaurants they like
4.6.2 The user must be able to retrieve list of saved restaurants
4.6.3 The user must be able to remove their saved restaurant

4.7 Add review to Restaurants

4.5.1 The user must be able to add their review for a restaurant.
4.5.2 The review would be available for others to see.

4.8 Google Maps

4.8.1 The system must be able to show the direction of the restaurant chosen using google maps
4.8.2 The user must be able to view the traffic condition in the direction to the chosen restaurant on the map

4.9 Interface with other system

4.9.1 The system must be able to retrieve location from the user's device via Geolocation API.
4.9.2 The system must be able to retrieve the list of restaurant’s information from Kaggle Dataset
4.9.3 The system must be able to display the list of urls of recipes using Spoonacular API.
4.9.4 The system must be able to show directions to the restaurant with traffic conditions using Google Maps API.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 System shall provide a recommended list of recipes according to the user's search preferences within 10 seconds.
5.1.2 System shall provide the nearest restaurants to the user according to the user's preferences within 10 seconds.
5.1.3 System shall list out the routes to the restaurant of the user's choice after the user clicks on the restaurant from the fastest route to the longest route within 10 seconds.
5.1.4 The application interface should not have anything that could hinder the user input.
5.1.5 Whenever the application is interrupted by any third-party applications, the application must be able to save and return to the same page before it was interrupted.
5.1.6 The system shall not crash without saving user data and current progress
5.1.7 The system must be able to update its data with minimal down time.

5.2 Safety Requirements

5.2.1 The application shall be extended in future.
5.2.2 HTTPS team shall maintain the code
5.2.3 The application might be extended to IOS and other desktop version when it gets popular

5.3 Security Requirements

5.3.1 Any personal data that the user enters into the system e.g. Email, Phone Number etc. it will be enclosed in an encrypted database and only authorized personnel will be able to access it.
5.3.2 All the app data should be secured and be encrypted with minimum needs so that it is protected from the outside environment and from internal attacks.
5.3.3 The application only uses location for certain features. This location is not share with any other unauthorized applications.
5.3.4 The system shall hide the password field during login to prevent shoulder surfing.
5.3.5 The system shall not leak user information.
5.3.6 Restart shall be acceptable when failure happens

5.4 Software Quality Attributes

5.4.1 Conveniency - Users are able to log in with their Google account and are able to continue where they left off.
5.4.2 Usability - 80% of new users are able to search for the recipe of their choice within 2 minutes.
5.4.3 Consistency - The design of the app is consistent throughout the whole app such that every section has the same or similar background and the layout of the texts and images are to be the same for every section.
5.4.4 Scalability - When the data that the user stores increases, the system should be able to handle them without delay by optimizing how the storage is done and accessed.
5.4.5 Extensibility- System has to be able to expand and increase its features in the future should the designer decide to add more.
5.4.6 Supportability - The database must be replaceable with any commercial product supporting standard SQL queries.
5.4.7 Robustness - The user is able to go back to the previous page if the user were to accidentally perform unwanted actions.
5.4.8 The application services are free of charge.

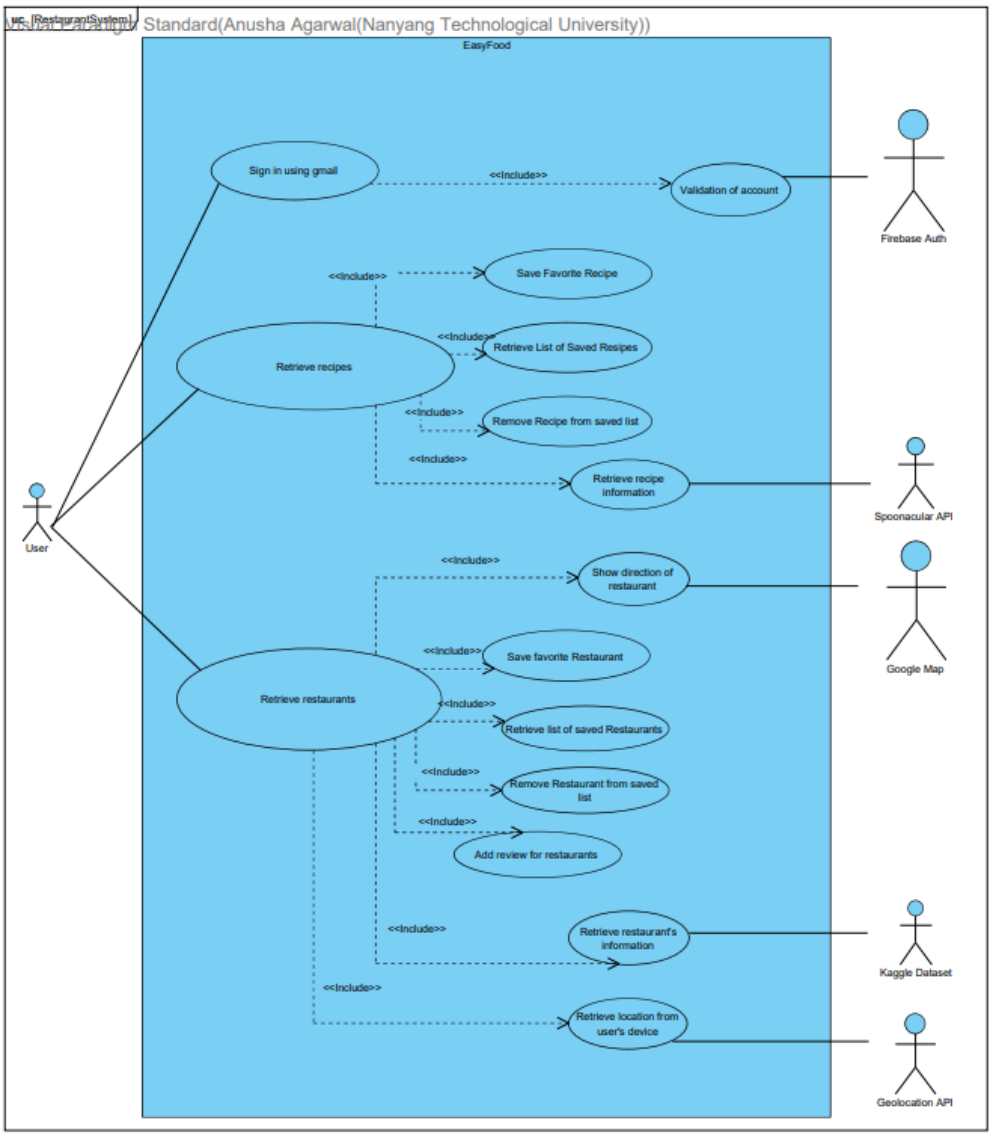
6. Use Cases

6.1 Use Case List -

- UC 1.0 - Sign in using Gmail account
    - UC 1.1 - Validation of account
- UC 2.0 - Retrieve Recipes
    - UC 2.1 - Save favorite recipes
    - UC 2.2 - Retrieve list of saved recipes
    - UC 2.3 - Remove Recipe from saved list
    - UC 2.4 - Retrieve recipe information (from Spoonacular API)
- UC 3.0 - Retrieve Restaurants
    - UC 3.1 - Save favorite restaurant
    - UC 3.2 - Retrieve list of saved restaurants
    - UC 3.3 - Remove Restaurants from saved list
    - UC 3.4 - Retrieve restaurants information (from Kaggle dataset)
    - UC 3.5 - Add review for restaurants
    - UC 3.6 - Retrieve location from user's device (using Geolocation API)
    - UC 3.7 - Show direction to restaurant (Google Map)

6.2 Use Case Diagram

Graphical representation of functional requirements to communicate various aspects of the structure, functionality, and dynamic behavior of EasyFood app.
It identifies the actors involved and describes how the app works.



6.3 Use Case Description

Use Case ID:	UC 1.0		
Use Case Name:	Sign in using gmail account		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022

Actor:	User

Description:	It allows the user to sign in and login the app using a gmail account and allows the app to store the user's choices in recipes or restaurants even when they logout.
Preconditions:	1. The user has downloaded the application 2. The user has an Android device with latest OS 3. The user must have a gmail account 4. The user has a good internet connection 5. The user plans to use it in Singapore
Postconditions:	The user must be able to sign in and be redirected to the main menu page.
Priority:	High
Frequency of Use:	Every time the user wants to use the app and is not already logged in.
Flow of Events:	1. The system will prompt the user to sign in using gmail 2. When clicked, it will redirect the user to gmail login page. 3. The user signs in. 4. The system checks if it is correct using firebase authentication. 5. The user is then redirected back to the main page of the mobile app.
Alternative Flows:	UC 1.0 – AF 1 1. The user inputs invalid mal id/ invalid credentials 2. The app prompts an error message
Exceptions:	1. The user may have forgotten their password 2. Internet connection not available
Includes:	Validation of account (Use Case ID – 1.1)
Special Requirements:	User remembers their username (gmail account) and inputs correct username and password
Assumptions:	The pre-conditions are met.

Use Case ID:	UC 1.1		
Use Case Name:	Validation of Account		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022

Actor:	Firebase Auth
Description:	It validates if the person has a valid gmail account and has input the correct password to be able to login to the app.
Preconditions:	1. User has downloaded the app 2. The user must have a valid gmail account

	3. User is willing to share their location with the app 4. The user has an Android device with the latest OS installed 5. The user plans to use the app in Singapore
Postconditions:	The app redirects the user to main menu page where the app greets them with their name.
Priority:	High
Frequency of Use:	Every time the user wants to login to the app using their gmail account
Flow of Events:	1. User inputs username-password for their gmail account 2. The system checks if the gmail account is valid using firebase authentication 3. The app redirects the app to the main menu where user gets to choose between Recipes or Restaurants 4. The app greets the user with a personalized message
Alternative Flows:	UC 1.1 – AF 1 1. If invalid credentials are entered, the app displays an error message. 2. The app prompts the user to try again
Exceptions:	1. No network connection 2. The user does not have a gmail account 3. The user has forgotten their password
Includes:	UC 1.0, AF 1
Special Requirements:	N/A
Assumptions:	The pre-conditions are met

Use Case ID:	UC 2.0		
Use Case Name:	Retrieve recipes		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022

Actor:	User
Description:	The user inputs in some criteria to find the desired recipe
Preconditions:	1. The user must have installed the app 2. The user must be having a stable internet connection 3. The user is logged in the app 4. The use chose the recipe option from the main menu
Postconditions:	The user must be displayed a variety of recipes based on the criteria input by him
Priority:	Medium
Frequency of Use:	Every time the user chooses recipes from the main menu.
Flow of Events:	1. The user chooses the recipe option from main menu

	2. The system redirects user to a page to select filters to get desired recipe 3. The user inputs preferred – a. Calorie count b. Ingredients available 4. The app suggests a variety of recipes considering the filters input by user
Alternative Flows:	UC 2.0 – AF 1 1. The user chooses the recipe option from main menu 2. The system redirects user to a page to select filters to get desired recipe 3. The user inputs preferred – a. Calorie count (left blank) or b. Ingredients available (left blank) 4. The app suggests a variety of recipes considering the filters input by user 5. The user sorts the list using a. Alphabetical order b. Calorie count c. Time taken to make the dish 6. No recipe found then app returns 'No result found'
Exceptions:	1. There is no recipe which matches the criteria input by user then app returns no result found 2. User's internet connection is not stable
Includes:	1. Save favourite recipes (UC 2.1) 2. Retrieve list of saved recipes (UC 2.2) 3. Remove Recipe from saved list (UC 2.3) 4. Retrieve recipe information (UC 2.4)
Special Requirements:	N/A
Assumptions:	Pre-conditions are met

Use Case ID:	UC 2.1		
Use Case Name:	Save favorite recipes		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022

Actor:	User
Description:	The user can make a list of their favourite recipes and retrieve them quickly when required
Preconditions:	1. The user must have downloaded the app 2. The user must be logged in the app

	3. The user must have chosen recipes option from the main menu
Postconditions:	The recipe must be saved and must not disappear when app is closed and reopened again.
Priority:	Medium
Frequency of Use:	The user has an option to save the recipe when they like it for future reference. The frequency may depend on how often the user accesses the app
Flow of Events:	1. The user chooses the recipe option from main menu 2. The app redirects the user to a page to input criteria 3. The user inputs in criteria such as calorie count to retrieve desired recipe 4. The user must be able to click on the heart symbol to save their favourite recipe which they already tried and like or might want to come back to for future reference. 5. The recipe gets added to the database of the user so that it stays when user logs out and comes back again
Alternative Flows:	UC 2.1 – AF 1 1. The user chooses the recipe option from main menu 2. The app redirects the user to a page to input criteria 3. The user inputs in criteria such as calorie count to retrieve desired recipe 4. The user does not save any recipe
Exceptions:	1. Users' internet connection is not stable
Includes:	N/A
Special Requirements:	N/A
Assumptions:	Pre-conditions are met

Use Case ID:	UC 2.2		
Use Case Name:	Retrieve list of saved recipes		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022

Actor:	User
Description:	The person can make a list of their favourite recipes which gets stored in the database of the user and retrieve the list when they wish to
Preconditions:	1. The user must have downloaded the app 2. The user must be logged in the app 3. The user must have chosen recipes option from the main menu 4. The user has (optional) some recipes saved already
Postconditions:	The list of saved recipes must be displayed

Priority:	Medium
Frequency of Use:	The user has an option to retrieve the list of saved recipes. The frequency may depend on how often the user accesses the app
Flow of Events:	1. The user chooses the recipe option from main menu 2. The user clicks on the saved restaurants button in the top right corner which looks like a star 3. A list of their saved recipes is shown by the app
Alternative Flows:	UC 2.2 AF 1 1. The user chooses the recipe option from main menu 2. The app redirects the user to a page to input criteria 3. The user inputs in criteria such as calorie count to retrieve desired recipes 4. The user saves a recipe by clicking on the star next to a particular recipe 5. The user then clicks on the star on top right corner of the window to see the saved recipes in a list form 6. The app displays the list of saved recipes 7. If not recipe is found, the app displays 'No Result found'
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	Pre-conditions are met

Use Case ID:	UC 2.3		
Use Case Name:	Remove recipe from saved list		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022

Actor:	User
Description:	The person can remove a recipe from their list of saved recipe if they are no longer interested in it
Preconditions:	1. The user must have downloaded the app 2. The user must be logged in the app 3. The user must have chosen recipes option from the main menu 4. The user has some recipes saved already
Postconditions:	The recipe must be removed from the saved list
Priority:	Low

Frequency of Use:	The user has an option to delete a recipe from the list of saved recipes. The frequency may depend on how often the user accesses the app and is usually less than that
Flow of Events:	1. The user chooses the recipe option from main menu 2. The user opens the saved recipes 3. The system displays the saved list 4. The user unclicks on the darkened heart symbol to remove the recipe from the saved list 5. The app should then remove the recipe from the list. 6. The app should also delete the entity from the user database
Alternative Flows:	UC 2.3 AF 1 1. The user chooses the recipe option from main menu 2. The user opens the saved recipes 3. The system displays the saved list 4. The user does not unclick on any saved recipe 5. The app does not take any action 6. The user exits UC 2.2 AF 2 1. The user chooses the recipe option from main menu 2. The user opens the saved recipes 3. The system displays the saved list 4. There is no recipe in the saved list 5. User exits
Exceptions:	1. User's internet connection is not stable
Includes:	N/A
Special Requirements:	The user should have some recipes saved already (preferably)
Assumptions:	Pre-conditions are met

Use Case ID:	UC 2.4		
Use Case Name:	Retrieve Recipe information		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	12/09/2022	Date Last Updated:	29/10/2022

Actor:	Spoonacular API
Description:	The app uses the Spoonacular API to
Preconditions:	1. The user has downloaded the app
Postconditions:	The recipe information is added to the entity class for recipes so it is available for use
Priority:	High
Frequency of Use:	If the user has downloaded the app,
Flow of Events:	1. The user downloads the app

	2. User logs in 3. User goes to recipes option in the main menu 4. The user inputs criteria 5. The app retrieves information about recipes and sorts it according to the input queries
Alternative Flows:	N/A
Exceptions:	1. The API is faulty and data does not load successfully
Includes:	N/A
Special Requirements:	N/A
Assumptions:	Pre-conditions are met

Use Case ID:	UC 3		
Use Case Name:	Retrieve restaurants		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022
Actor:	User		
Description:	The user inputs in some criteria to find the desired restaurant		
Preconditions:	1. The user must have installed the app 2. The user must be having a stable internet connection 3. The user is logged in the app 4. The user chose the restaurant option from the main menu 5. The user plans on using the app in Singapore		
Postconditions:	The user must be displayed a variety of restaurant based on the criteria input by him		
Priority:	Medium		
Frequency of Use:	Every time the user chooses restaurant from the main menu.		
Flow of Events:	1. The user chooses the restaurant option from main menu 2. The system redirects user to a page to select filters to get desired restaurant 3. The user inputs preferred – a. Cuisine b. Dietary Preference c. Distance from current location 4. The app suggests a variety of restaurants considering the filters input by user		
Alternative Flows:	UC 3.0 – AF 1 1. The user chooses the restaurant option from main menu 2. The system redirects user to a page to select filters to get desired restaurant 3. The user inputs preferred – a. Cuisine (left blank) or b. Dietary Preference (left blank) c. Distance from current location (left blank)		

	a. The app suggests a variety of retaurants considering the filters input by user 4. The user sorts the list using a. Alphabetical order b. Distance 5. No restaurant found then app returns 'No result found'
Exceptions:	1. There is no restaurant which matches the criteria input by user then app returns no result found 2. User's internet connection is not stable
Includes:	1. Save favourite restaurants (UC 3.1) 2. Retrieve list of saved restaurants (UC 3.2) 3. Remove Recipe from saved list (UC 3.3) 4. Retrieve recipe information (UC 3.4) 5. Add review for Restaurant (UC 3.5) 6. Retrieve location from users' device (UC 3.6) 7. Show direction to the restaurant (UC 3.7)
Special Requirements:	N/A
Assumptions:	Pre-conditions are met

Use Case ID:	UC 3.1		
Use Case Name:	Save favourite restaurant		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022

Actor:	User
Description:	The user can make a list of their favourite restaurants by saving them and retrieve them quickly when required
Preconditions:	1. The user must have downloaded the app 2. The user must be logged in the app 3. The user must have chosen restaurant option from the main menu
Postconditions:	The restaurant must be saved and must not disappear when app is closed and reopened again.
Priority:	Medium
Frequency of Use:	The user has an option to save the restaurant when they like it for future reference. The frequency may depend on how often the user accesses the app
Flow of Events:	1. The user chooses the restaurant option from main menu 2. The app redirects the user to a page to input criteria 3. The user inputs in criteria such as cuisine, dietary requirements and distance to retrieve desired restaurant

	4. The user must be able to click on the heart symbol to save their favourite restaurant which they already tried and like or might want to come back to for future reference. 5. The restaurant gets added to the database of the user so that it stays when user logs out and comes back again
Alternative Flows:	UC 3.1 – AF 1 1. The user chooses the restaurant option from main menu 2. The app redirects the user to a page to input criteria 3. The user inputs in criteria such as cuisine, dietary requirements and distance to retrieve desired restaurant 4. The user does not save any restaurant
Exceptions:	2. Users' internet connection is not stable
Includes:	N/A
Special Requirements:	N/A
Assumptions:	Pre-conditions are met

Use Case ID:	UC 3.2		
Use Case Name:	Retrieve list of saved recipes		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022

Actor:	User
Description:	The person can make a list of their favourite restaurant which gets stored in the database of the user and retrieve the list when they wish to
Preconditions:	1. The user must have downloaded the app 2. The user must be logged in the app 3. The user must have chosen restaurant option from the main menu 4. The user has (optional) some restaurant saved already
Postconditions:	The list of saved restaurants must be displayed
Priority:	Medium
Frequency of Use:	The user has an option to retrieve the list of saved restaurants. The frequency may depend on how often the user accesses the app
Flow of Events:	1. The user chooses the restaurant option from main menu 2. The user clicks on the saved restaurants button in the top right corner which looks like a star 3. A list of their saved restaurant is shown by the app
Alternative Flows:	UC 3.2 AF 1 1. The user chooses the restaurant option from main menu 2. The app redirects the user to a page to input criteria 3. The user inputs in criteria such as cuisine, dietary requirements, and distance to retrieve desired restaurant

	4. The user saves a recipe by clicking on the star next to a particular restaurant 5. The user then clicks on the star on top right corner of the window to see the saved restaurant in a list form 6. The app displays the list of saved restaurant 7. If no restaurant is found, the app displays 'No Result found'
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	Pre-conditions are met

Use Case ID:	UC 3.3		
Use Case Name:	Remove restaurant from saved list		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	27/08/2022	Date Last Updated:	29/10/2022

Actor:	User
Description:	The person can remove a recipe from their list of saved recipe if they are no longer interested in it
Preconditions:	1. The user must have downloaded the app 2. The user must be logged in the app 3. The user must have chosen restaurant option from the main menu 4. The user has some restaurants saved already
Postconditions:	The restaurant must be removed from the saved list
Priority:	Low
Frequency of Use:	The user has an option to delete a restaurant from the list of saved restaurants. The frequency may depend on how often the user accesses the app and is usually less than that
Flow of Events:	1. The user chooses the restaurant option from main menu 2. The user opens the saved restaurant 3. The system displays the saved list 4. The user unclicks on the darkened heart symbol to remove the restaurant from the saved list 5. The app should then remove the restaurant from the list. 6. The app should also delete the entity from the user database
Alternative Flows:	UC 3.3 AF 1 1. The user chooses the restaurant option from main menu 2. The user opens the saved restaurant 3. The system displays the saved list

	4. The user does not unclick on any saved restaurant 5. The app does not take any action 6. The user exits UC 3.3 AF 2 1. The user chooses the restaurant option from main menu 2. The user opens the saved restaurant 3. The system displays the saved list 4. There is no restaurant in the saved list 5. User exits
Exceptions:	1. User's internet connection is not stable
Includes:	N/A
Special Requirements:	The user should have some restaurant saved already (preferably)
Assumptions:	Pre-conditions are met

Use Case ID:	UC 3.4		
Use Case Name:	Retrieve Restaurants information		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	12/09/2022	Date Last Updated:	29/10/2022

Actor:	Kaggle Dataset
Description:	The app uses the Kaggle Dataset to
Preconditions:	1. The user has downloaded the app
Postconditions:	The restaurant information is added to the entity class for restaurant so it is available for use
Priority:	High
Frequency of Use:	If the user has downloaded the app,
Flow of Events:	1. The user downloads the app 2. User logs in 3. User goes to restaurant option in the main menu 4. The user inputs criteria 5. The app retrieves information about restaurant and sorts it according to the input queries
Alternative Flows:	N/A
Exceptions:	1. The dataset is faulty and data does not load successfully
Includes:	N/A
Special Requirements:	N/A
Assumptions:	Pre-conditions are met

Use Case ID:	UC 3.5
Use Case Name:	Add review for Restaurants

Created By:	Anusha	Last Updated By:	Anusha
Date Created:	12/09/2022	Date Last Updated:	30/10/2022

Actor:	User
Description:	The person can add review for a restaurant they went to
Preconditions:	1. The user must have downloaded the app 2. The user must be logged in the app 3. The user must have chosen restaurant option from the main menu 4. The user plans on using this app in Singapore
Postconditions:	The review must be added on the restaurant database and is displayed to other users who check it out
Priority:	Low
Frequency of Use:	Every time the user follows the recommendation of the app and checks the restaurant out, the user has the option to input their hones reviews about the place
Flow of Events:	1. The user chooses the restaurant option from main menu 2. The app redirects the user to input their criteria to find the desired restaurant 3. The user inputs the desired filters such as cuisine, dietary requirements, and distance 4. The person visits the recommended restaurant or already knows about the restaurant 5. The person opens the restaurant tab and finds the particular restaurant. 6. The person then adds their review of the restaurant accordingly.
Alternative Flows:	N/A
Exceptions:	1. Users' internet connection is not stable 2. The user might input false review and the app cannot check that
Includes:	N/A
Special Requirements:	N/A
Assumptions:	Pre-conditions are satisfied

Use Case ID:	UC 3.6		
Use Case Name:	Retrieve location from user's device		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	12/09/2022	Date Last Updated:	29/10/2022

Actor:	Geolocation API, User

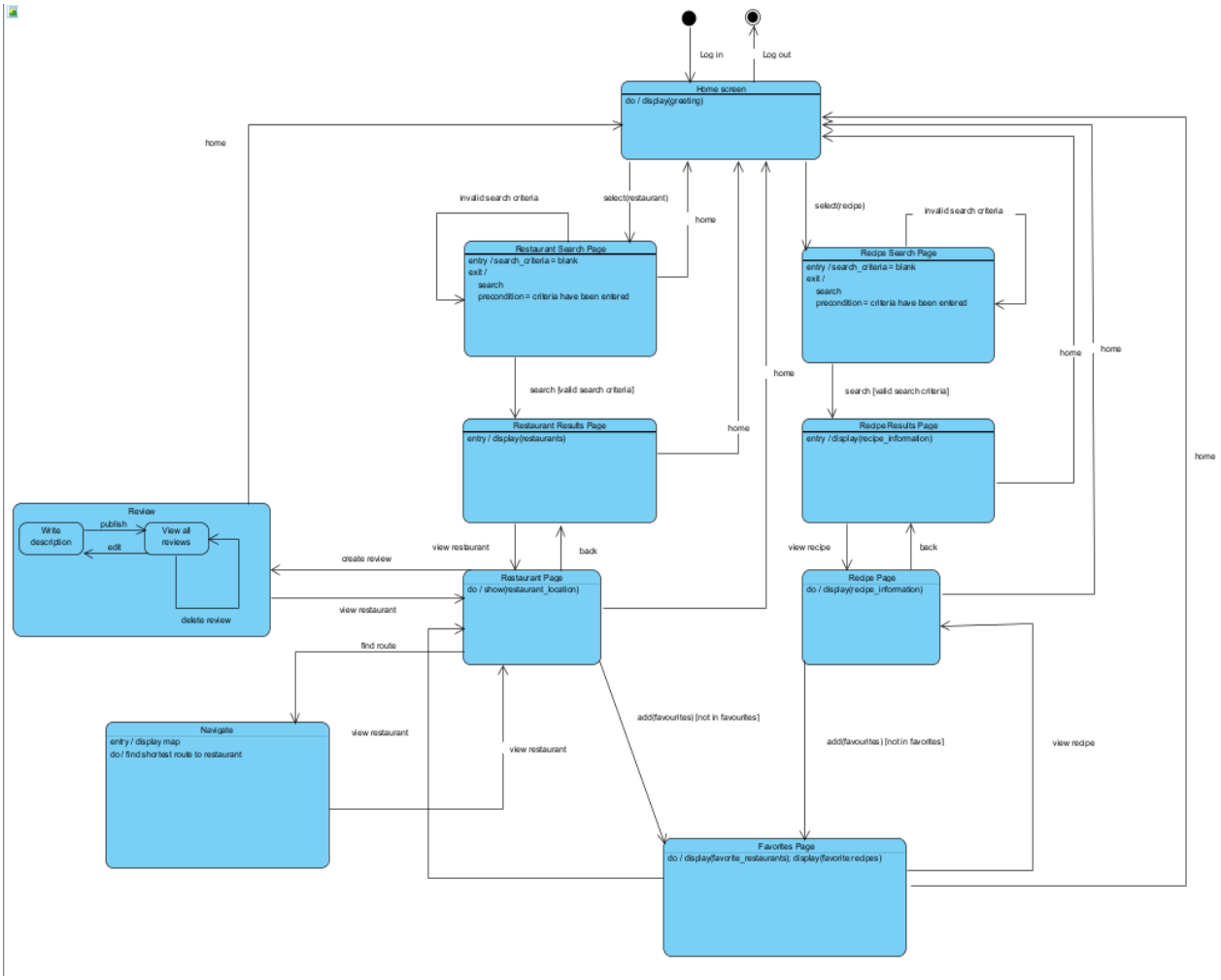
Description:	The app uses the API to find the current location of the user to recommend restaurants based on the how far they are from users' current location and also used to find the way using google map
Preconditions:	1. User has downloaded the app 2. User is in Singapore 3. User has allowed the app to use his current location
Postconditions:	The app has access to users' location
Priority:	Low
Frequency of Use:	When user uses the app to find the restaurant, and inputs distance from current location as a criterion
Flow of Events:	1. User downloads app 2. User allows app to access his location 3. The user chooses restaurant option from main menu 4. User inputs distance as a criteria 5. App uses users' geolocation to make recommendations 6. App displays a list of restaurants to choose from
Alternative Flows:	UC 3.6 AF 1 1. User downloads app 2. User allows app to access his location 3. The user chooses restaurant option from main menu 4. User inputs distance as a criteria 5. App uses users' geolocation to make recommendations 6. App displays a list of restaurants to choose from 7. User chooses a restaurant and opens map to see the way to reach there 8. App uses users' geolocation to show the user the path to reach the restaurant using google map
Exceptions:	1. User does not allow the app to access the users' current location 2. The user is not in Singapore
Includes:	N/A
Special Requirements:	User plans to use the app in Singapore
Assumptions:	Pre-conditions are satisfied

Use Case ID:	3.7		
Use Case Name:	Show direction to restaurant		
Created By:	Anusha	Last Updated By:	Anusha
Date Created:	12/09/2022	Date Last Updated:	29/10/2022

Actor:	Google Map, API
Description:	The app uses the google map to show the user the path to reach their chosen restaurant
Preconditions:	1. User has downloaded the app

	2. User is in Singapore 3. User has allowed the app to use his current location 4. User has Google map already downloaded on their device
Postconditions:	User can see the way to their chosen restaurant
Priority:	Low
Frequency of Use:	When user chooses a restaurant to visit
Flow of Events:	1. User downloads app 2. User allows app to access his location 3. The user chooses restaurant option from main menu 4. User inputs criteria to find the desired restaurants 5. App displays a list of restaurants to choose from 6. User chooses a restaurant and opens map to see the way to reach there 7. App uses users' geolocation to show the user the path to reach the restaurant using google map
Alternative Flows:	The user does not use this feature (optional)
Exceptions:	1. App is unable to redirect to google map 2. User does not have google map downloaded 3. Users' internet connection is not stable 4. User did not allow the app to use users' current location 5. User is not in Singapore
Includes:	N/A
Special Requirements:	N/A
Assumptions:	Pre-conditions are satisfied

6.4 Dialog Map



6.5 Testing

We have used Black box and White Box testing to test the various use cases of the app.

6.5.1 Blackbox Testing

Blackbox testing is used to test the functionality of the software without knowing the internal structure of the system. It helps with analysis and verification of client requirements and specifications.

6.5.1.1 Search Recipes

Query	Max Calories	Ingredients Selected	Expected Result	Actual Result
Pizza	1000	None	Recipes matching criteria returned	Recipes matching criteria returned
None	500	'fish', 'broccoli'	Recipes matching criteria returned	Recipes matching criteria returned
None	Any String Input	None	Recipes returned with max calories criteria ignored	Recipes returned with max calories criteria ignored
None	-1	None	No results found	No results found

6.5.1.2 Filter recipe results

Filter	Expected Result	Actual Result
Alphabetical	Results sorted in alphabetical order	Results sorted in alphabetical order
Calories	Results sorted in ascending calorie count	Results sorted in ascending calorie count
Time	Results sorted in ascending time	Results sorted in ascending time

6.5.1.3 Requesting permission for user's location to search restaurants

Permission Status	Location	Expected Result	Actual Result
Not Granted	Disabled	Search bar disabled	Search bar disabled
Not Granted	Enabled	Search bar disabled	Search bar disabled
Granted	Disabled	Search bar disabled	Search bar disabled
Granted	Enabled	Search bar enabled	Search bar enabled

6.5.1.4 Save favorite restaurant

User Action	Expected Result	Actual Result
Add favorite restaurant	Restaurant added to favorites list	Restaurant added to favorites list
Remove favorite restaurant	Restaurant removed from favorites list	Restaurant removed from favorites list

6.5.1.5 Restaurant review

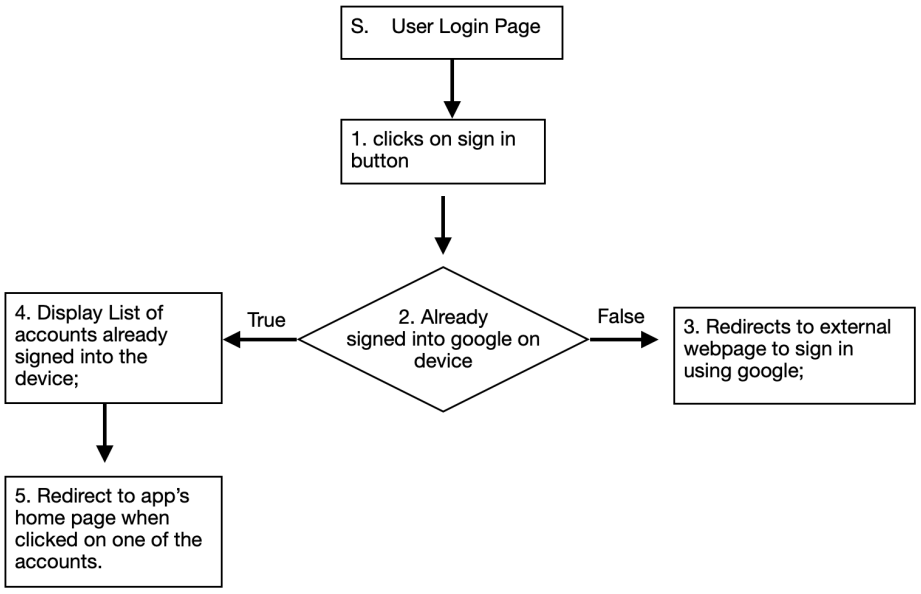
User Action	Review Status	Expected Result	Actual Result
Click Add/Edit review	Already reviewed	Edit review dialog shown	Edit review dialog shown
Click Add/Edit review	Not yet reviewed	Add review dialog shown	Add review dialog shown
Click delete review	Already reviewed	Review deleted	Review deleted

6.5.2 White Box Testing

White Box is used to verify that every path of the System under Test (SUT) has been identified and tested and functions properly. It helps do unit testing of every individual component of related units. We choose different inputs to exercise paths through the code and determine if actual output matches with expected output.

Use Case 1: User Login

1. Control Flow Diagram



2. Cyclomatic Complexity - 1 + 1 = 2

3. Basis Paths

Path 1: 1 — 2 — 4 — 5
Path 2: 1 — 2 — 3

4. Test Cases for Basic Paths

Path 1

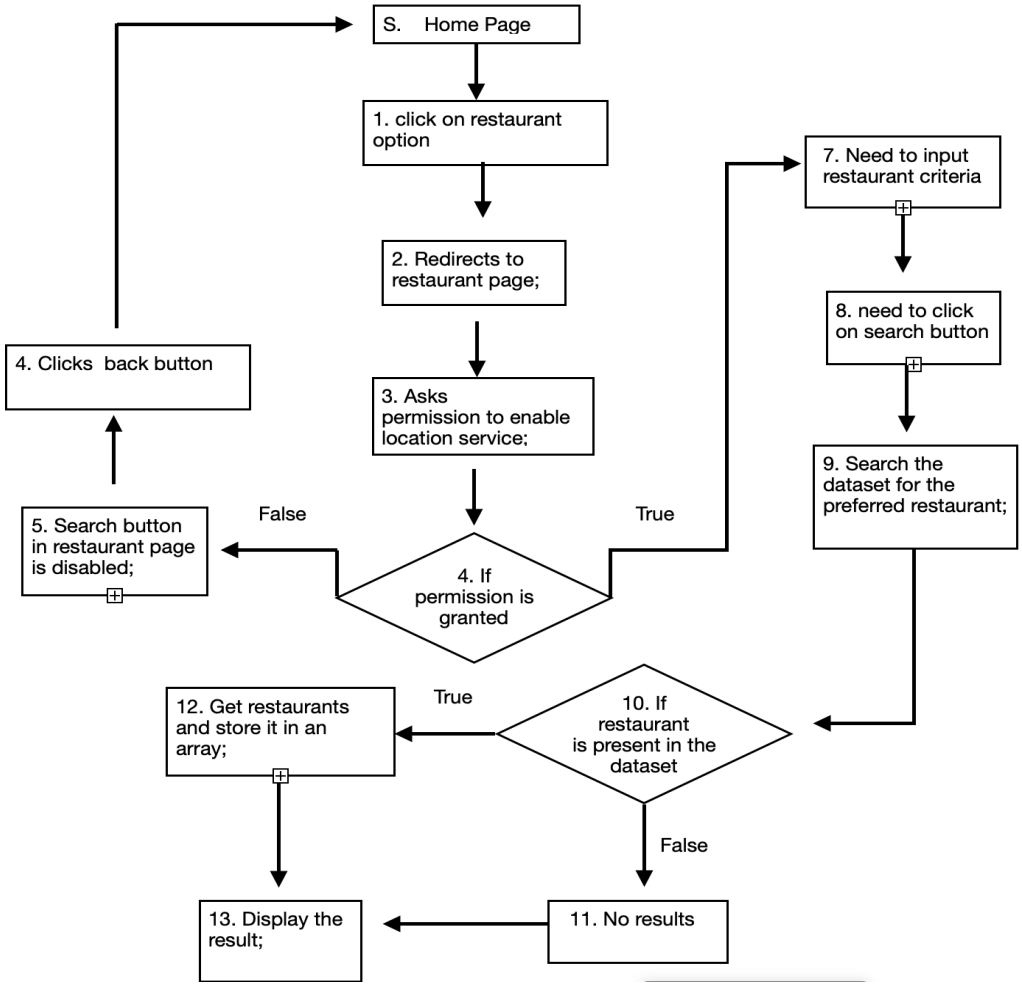
User clicks on sign in button when already Signed into google on device, displays the accounts already signed into the device

Execute Result:
Redirects to the app's home page when clicked on one of the accounts.

Path 2
User clicks on sign in button when not signed into google on device
Execute Result:
Redirects to an external webpage to sign in using google.

Use Case 2: Searching for restaurant

1. Control Flow Diagram



2. Cyclomatic Complexity - 2 + 1 = 3

3. Basis Paths
Path 1: S—1—2—3—4—7—8—9—10—11
Path 2: S—1—2—3—4—5—6—S
Path 3: S—1—2—3—4—7—8—9—10—12—13

4. Test Cases for Basis Paths

Path 1
User clicks the restaurant option in the home page, the app redirects the user to the restaurant page. Before proceeding the app asks permission from the user to enable location service. The user gives the permission.

User's Input:

Cuisine: French
Dietary Preference: Vegetarian, Healthy
Distance: 5 KM

Execute Result:
System searches the dataset for the preferred restaurants and finds that the preferred restaurant is not in the dataset. So it displays "no results found" in a new page

Path 2

User clicks the restaurant option in the home page, the app redirects the user to the restaurant page. Before proceeding the app asks permission from the user to enable location service. The user does not give the permission so the app disables the search button in the restaurant page, and the user clicks on the back button.

Execute Result:
Redirects to the app's Home Page.

Path 3
User clicks the restaurant option in the home page, the app redirects the user to the restaurant page. Before proceeding the app asks permission from the user to enable location service. The user gives the permission.

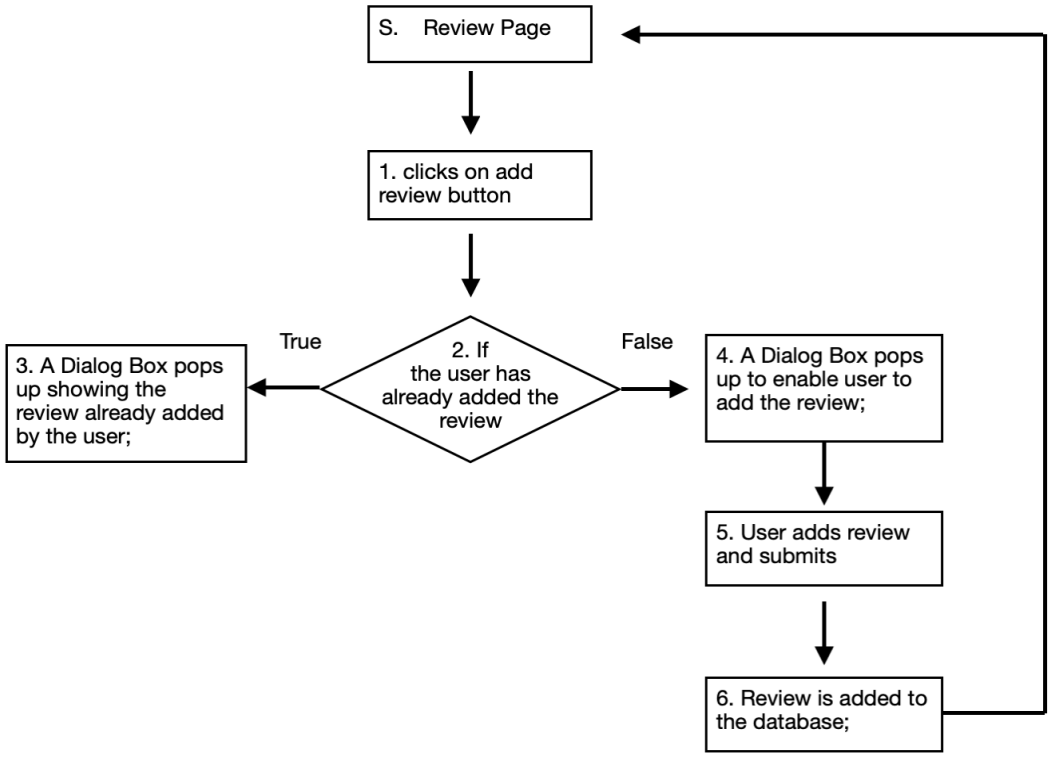
User's Input:

Cuisine: Chinese
Dietary Preference: Vegetarian
Distance: 10 KM

Execute Result:
System searches the dataset for the preferred restaurants and displays the restaurants' information as a result in a new page.

Use Case 3: Adding review for the restaurant

1. Control Flow Diagram



2. Cyclomatic Complexity - 1 + 1 = 2

3. Basis Paths

Path 1: S—1—2—4—5—6—s
Path 2: S—1—2—3

4. Test Cases for Basis Paths

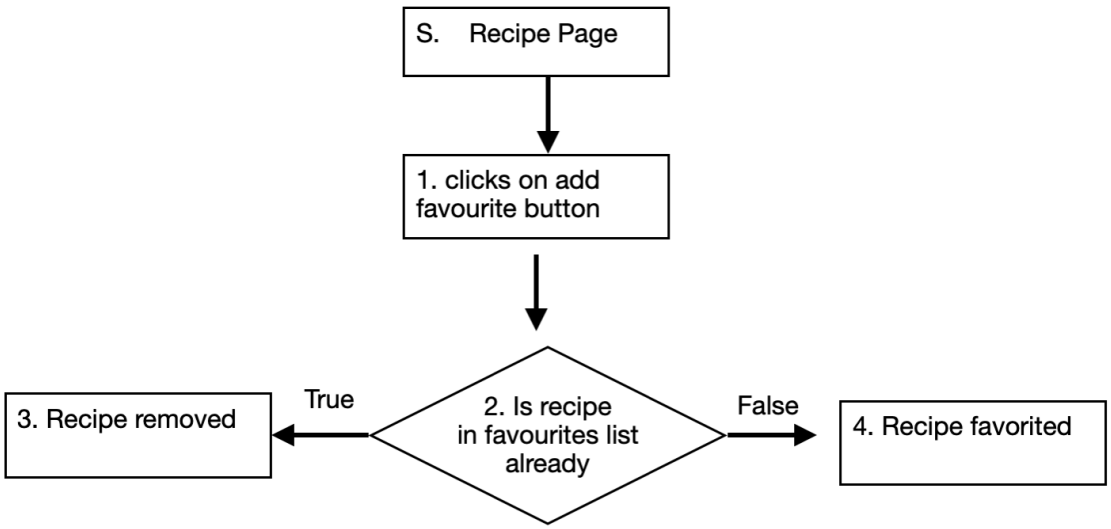
Path 1:
User clicks on the add review button, the user has not reviewed the restaurant before, the user adds the review and submits it.

Execute Result:
Review is added to the database and redirects the user back to the review page where the added review is displayed.

Path 2:
User clicks on the add review button, the user has already reviewed the restaurant.
Execute Result:
A dialog box pops up showing the already added review.

Use Case 4: Adding Favorite Recipes

1. Control Flow Diagram



2. Cyclomatic Complexity - 1 + 1 = 2

3. Basis Paths
Path 1: S—1—2—3
Path 2: S—1—2—4

4. Test Cases for Basis Paths

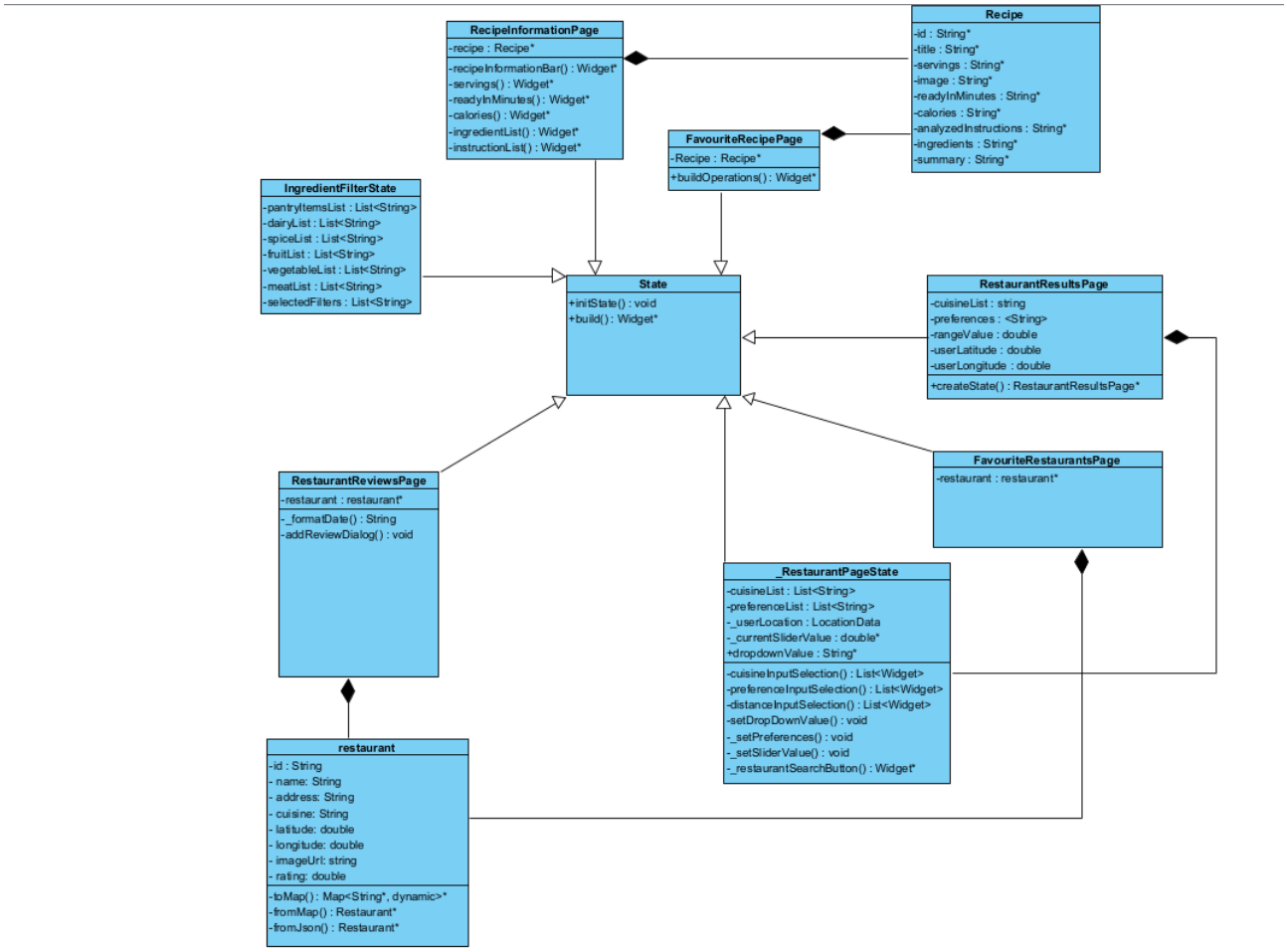
Path 1:
User clicks on favorite button, when the recipe is already in the favorite list
Execute Result:
Recipe is removed from the favorite list.

Path 2:
User clicks on favorite button, when the recipe in not there in favorite list
Execute Result:
Recipe is added to the favorite list.

7. Design Models

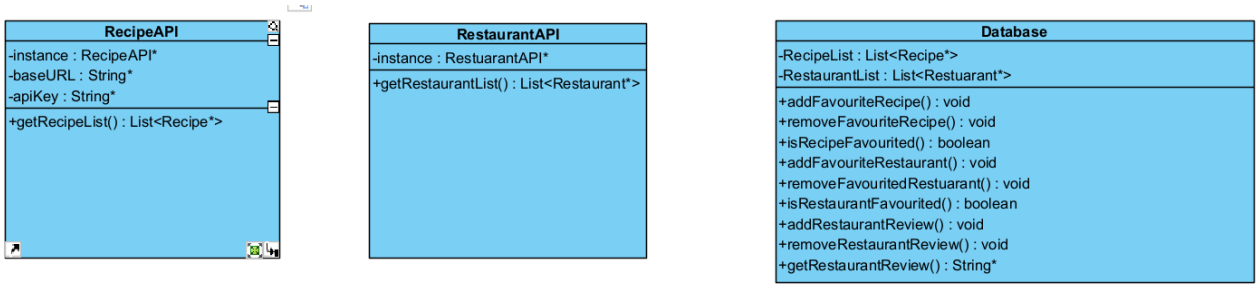
7.1 Conceptual Models - Class Diagrams

Class Diagram of Boundary Classes



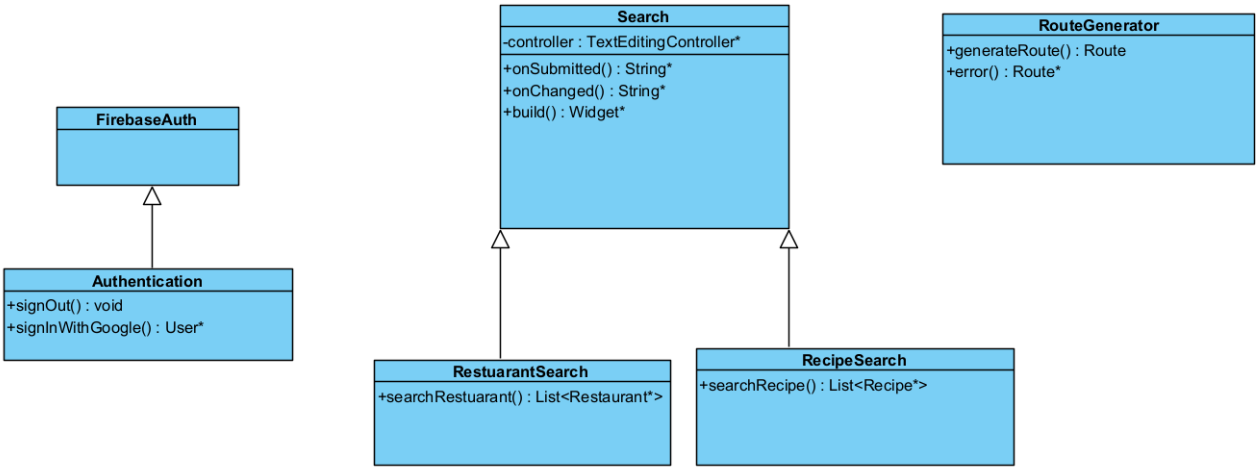
The boundary classes are classes that interface with the users of the app. They mainly consist of the different pages and user-interfaces of the app.

Class Diagram of Entity Classes



Entity classes represent the system data.

Class Diagram of Control Class

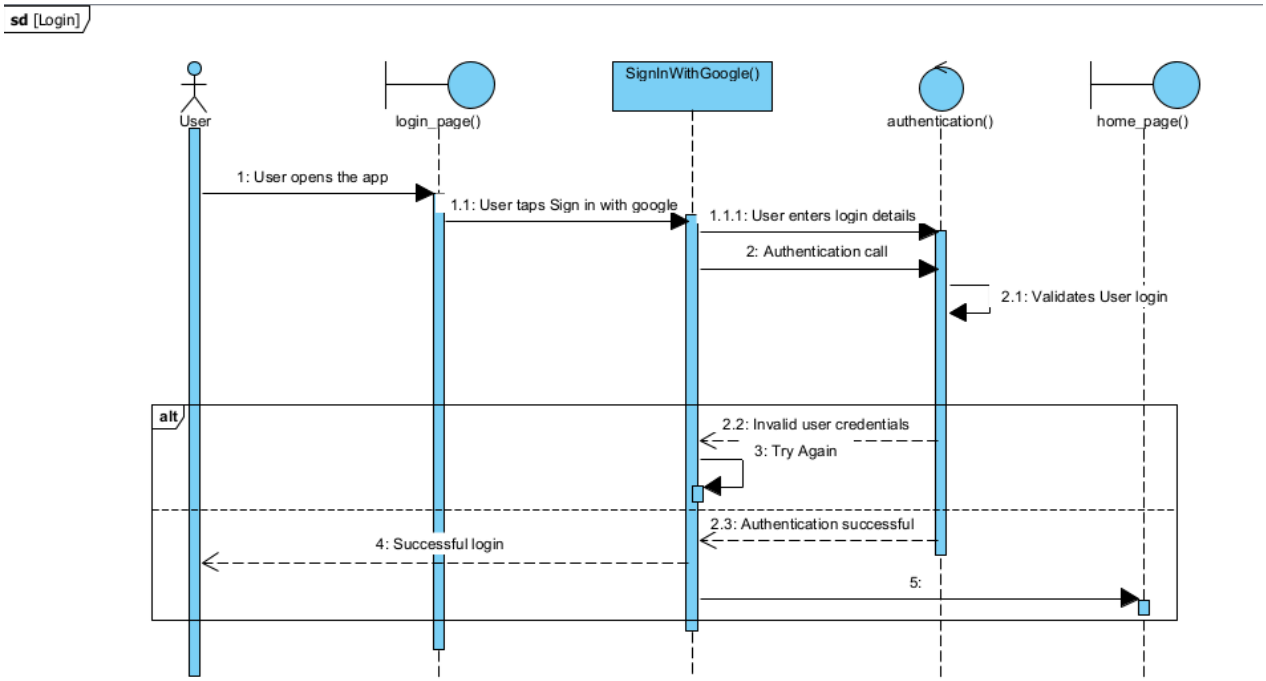


Control classes are responsible for implementing the logic and functions necessary for the app to work. They serve as the glue between the boundary class and the entity class.

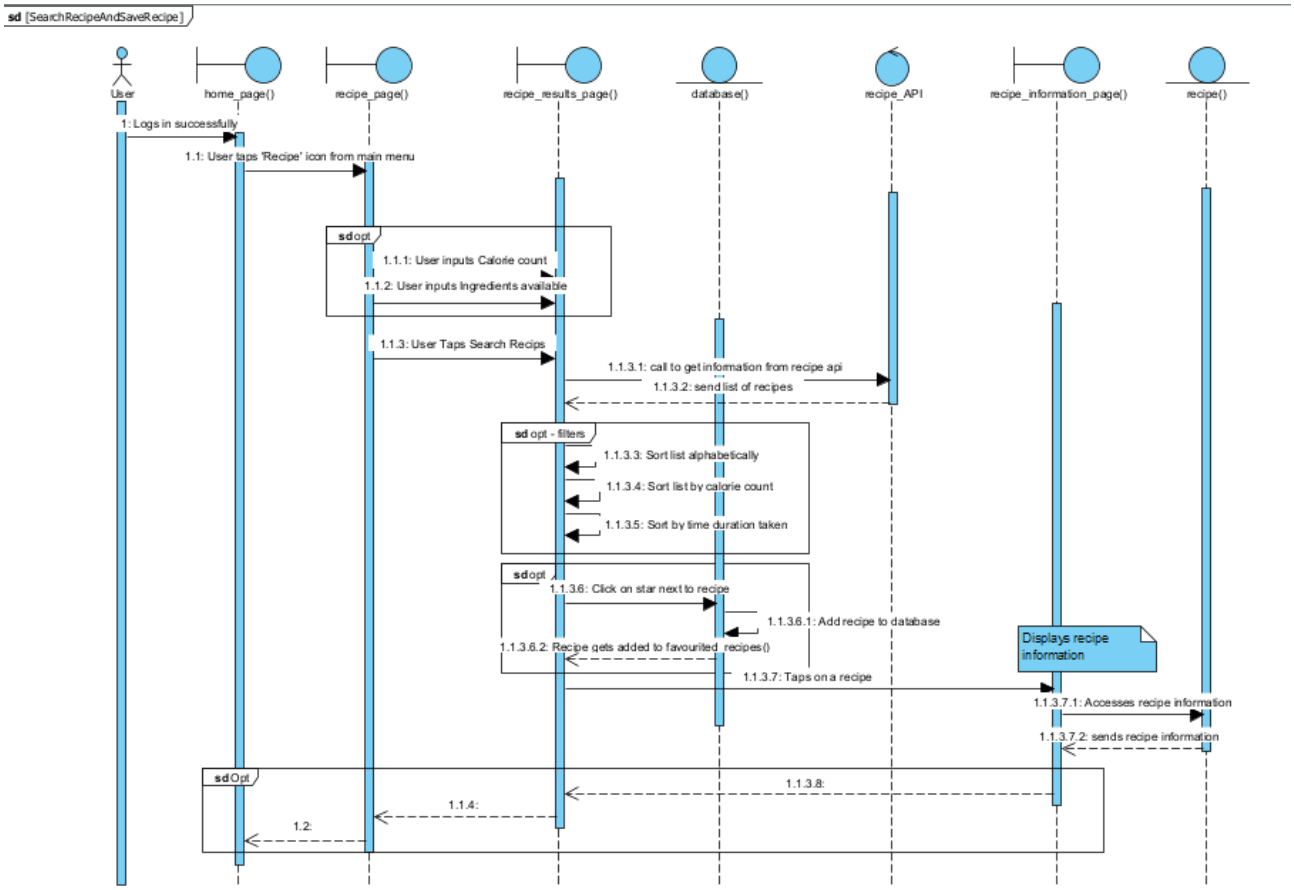
7.2 Dynamic Model - Sequence Diagrams

Below are sequence diagrams for some key uses cases:

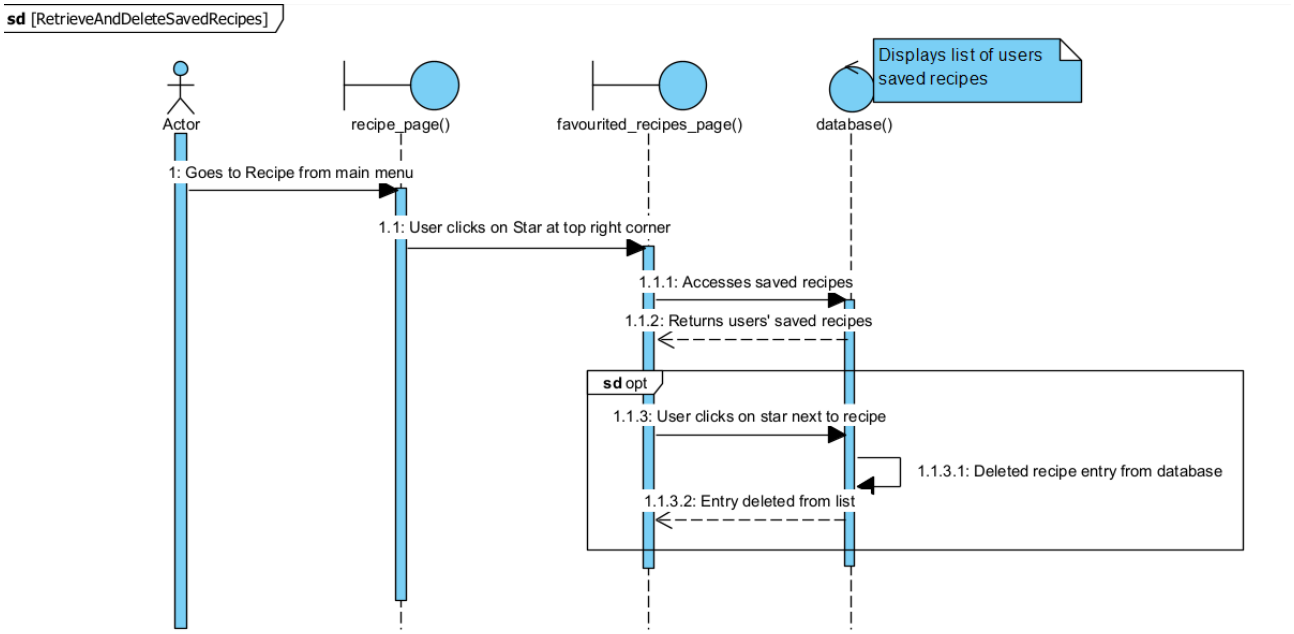
7.2.1 Login:



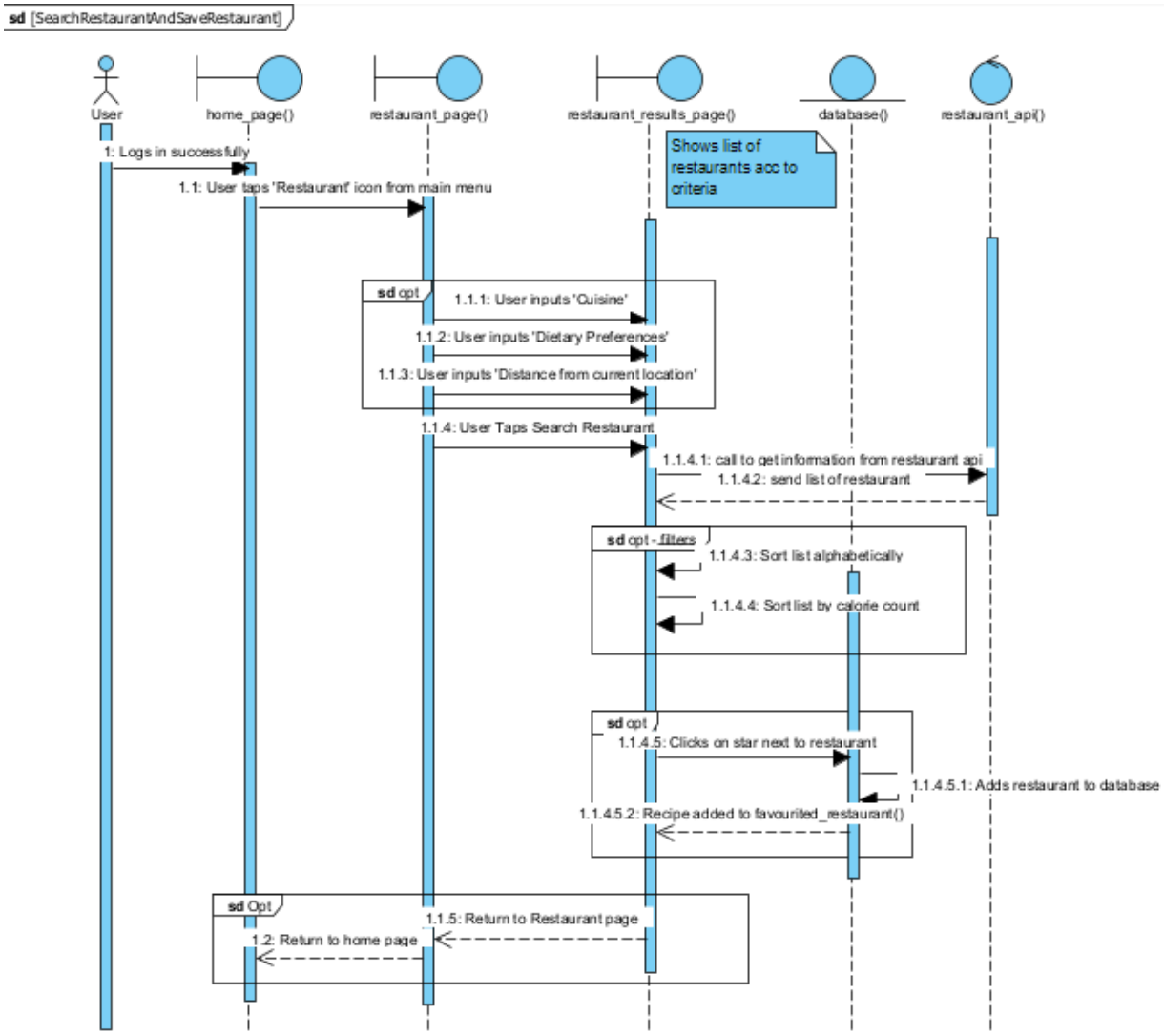
7.2.2 Recipe Search and Save:



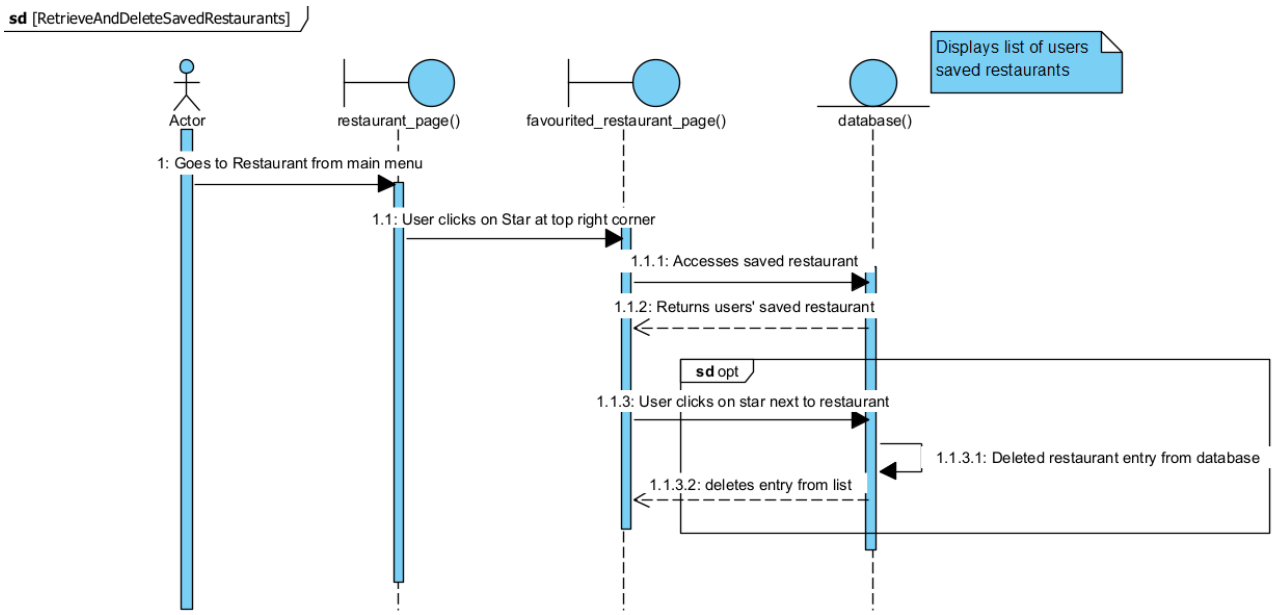
7.2.3 Retrieve and Delete Saved Recipes:



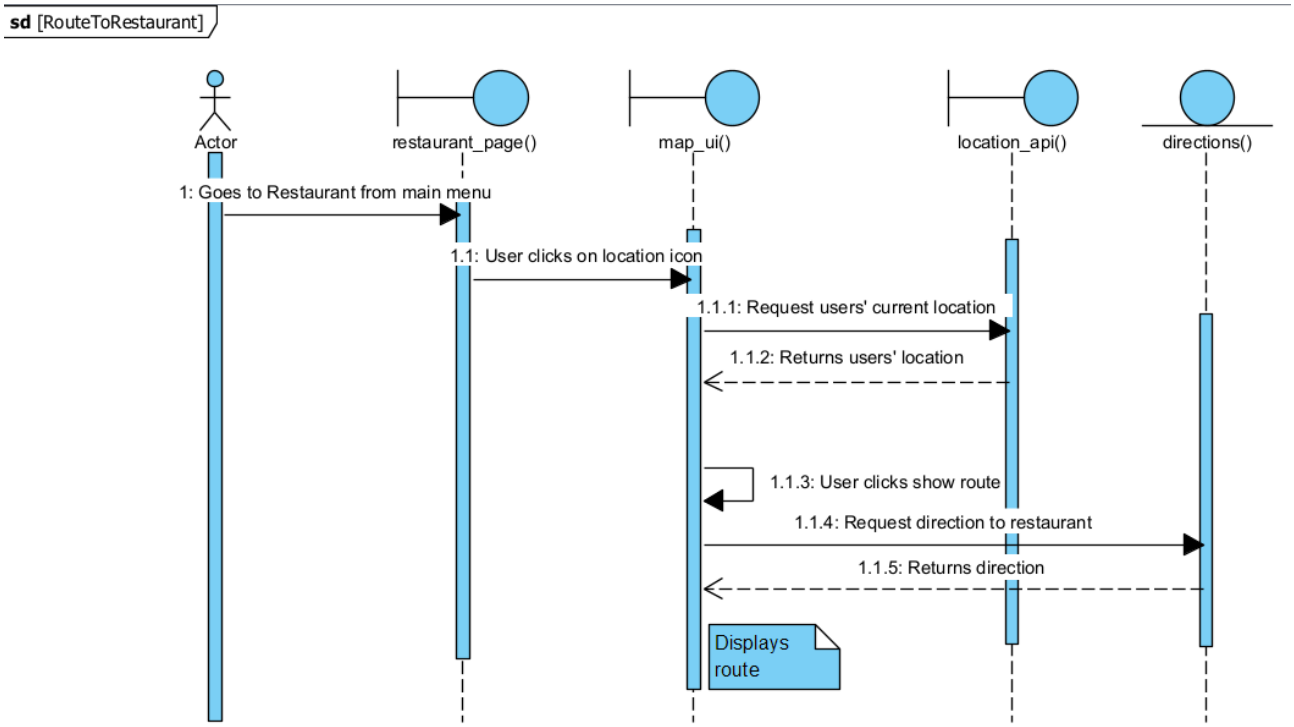
7.2.4 Restaurant Search and Save:



7.2.5 Retrieve and Delete Saved Restaurants:



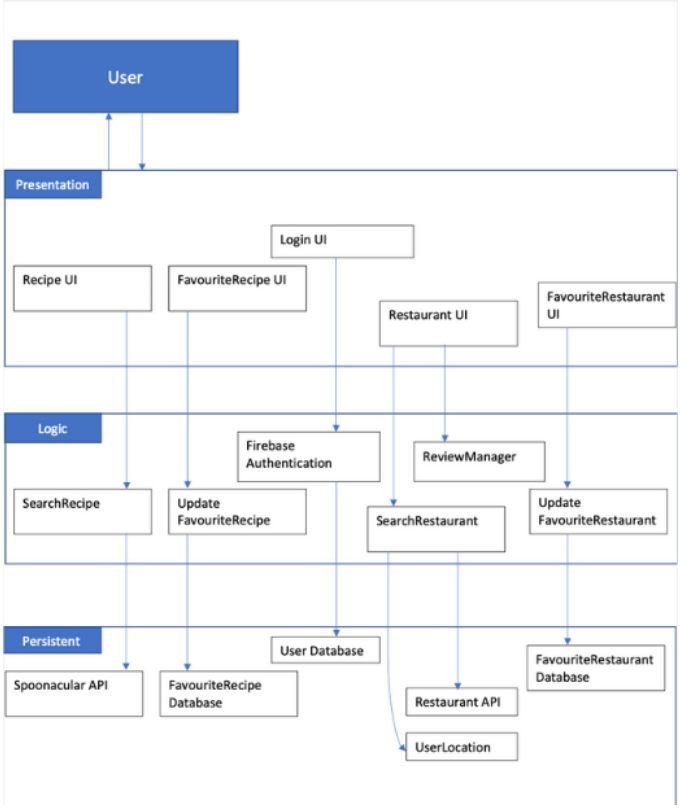
7.2.6 Route to Restaurant:



8. Design Concerns

8.1 System Architecture Diagram

3-layered Architecture



- Presentation Layer: UI and displays that user interacts with (search pages for recipe, restaurant, results pages and favorites)
- Logic layer: Implements the key functions (searching/sorting) and connects presentation layer to the persistent data layer
- Persistent data layer: Data is retrieved from external APIs. Internal database to store users' favorites.

Advantages of 3-layered architecture:

Having the overall system design split into three layers allows each layer to be independent of the other layers. This allows each layer to be developed simultaneously, and independent of each other, achieving loose coupling of the classes.

It also allows upgrades and modifications to be done to one layer without affecting the other layers in the architecture. For example, we can upgrade the Logic layer to implement new sorting criteria without affecting the Presentation layer or the Persistent Data layer.

9. Software Design Principles

9.1 Single Responsibility Principle (SRP)

“There should never be more than one class to change”

In the EasyFood app, we have factored the code in such a way that each class has only 1 responsibility. It allows loose coupling and high cohesion. Since each class has only one responsibility, the responsibilities are not coupled and there will not be more than one reason to make changes. When making changes, only one class needs to be modified due to the loose coupling.

It makes it easy for any developer to make changes to a particular class without having to worry about other dependencies. It thereby allows for easy extension of classes.

An example of this can be seen when we created separate classes for both RecipeResults and RecipeFavourites. This allowed us to make changes and modifications to the RecipeFavourites page and functionalities without touching our main results page.

9.2 Open-Closed Principle (OCP)

“A module should be open for extension but closed for modification”

We have arranged the code in such a way that we can change what the modules do, without changing the source code of the module. We achieved this by using abstraction.

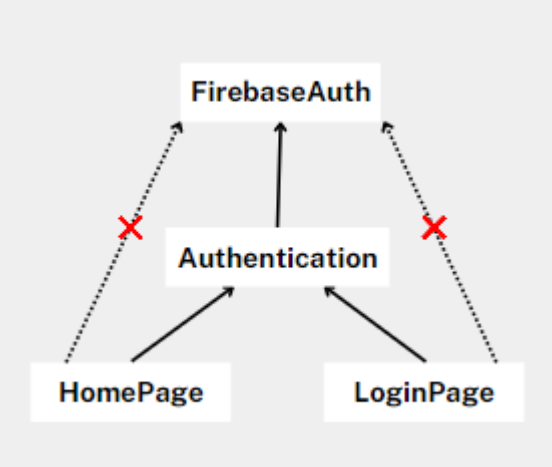
An example from our app is how each page inherits from a parent class. This allows the developers to create multiple pages by extending the parent class while preserving the fundamental functionalities of a page.

9.3 Dependency Inversion Principle (DIP)

This principle states that high level modules should not depend on low level modules. All such modules should instead depend on abstraction. These abstractions should not depend on details but rather the details should depend on abstractions. In our app, users sign in from the LoginPage, and sign out from the HomePage.

```
void _signOut() async {
  try {
    await Authentication.signOut();
    onSignOut();
  } catch (e) {
    debugPrint(e.toString());
  }
}
```

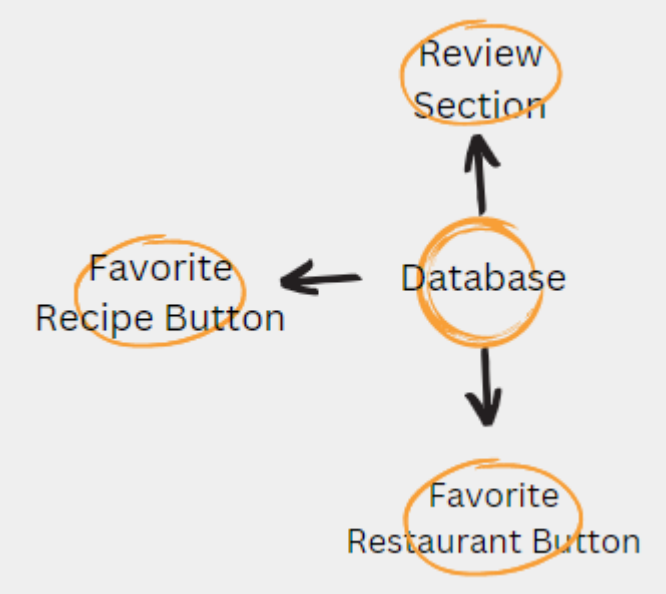
```
Future<void> _signInWithGoogle() async {
  try {
    setState(() => isLoading = true);
    final user = await Authentication.signInWithGoogle();
    setState(() => isLoading = false);
    widget.onSignIn(user);
  } catch (e) {
    debugPrint(e.toString());
    setState(() => isLoading = false);
  }
}
```



We did not directly link higher-level modules such as HomePage and LoginPage to the lower level FirebaseAuth class which stores authentication functionality for our app. Instead, we created the class Authentication to link both high and low level modules. This allows loose coupling of our modules. If we were to change our authentication provider, we would only need to modify the Authentication class and replace the FirebaseAuth class accordingly. This minimizes the number of changes we need to make to the code.

9.4 Observer Pattern

This technique is used when an object needs to receive an update (observer) when another object changes (Subject). This allows for there to be loose coupling between classes.



RestaurantReviewsPage listens for changes in the review database (when a user adds a new review, edits an existing review or deletes a review) of a particular restaurant, and updates the page accordingly.

```
body: StreamBuilder(
  stream: Database.restaurantReviewsStream(restaurant),
  builder: (context, snapshot) {
    return CustomScrollView(
      slivers: [
      _buildTopImage(),
      _buildReviews(snapshot),
      ],
    );  // CustomScrollView
  },
),  // StreamBuilder
```

Similarly, RecipeFavoriteButton and RestaurantFavoriteButton listens for changes in the favorite status of a particular recipe or restaurant and updates its icon accordingly.

```
return StreamBuilder(
  stream: Database.isRestaurantFavorited(restaurant),
  builder: (context, snapshot) {
    if (snapshot.hasData) {
      bool isFavorited = snapshot.data;
      return IconButton(
        onPressed: () => _toggleFavorite(isFavorited),
        icon: isFavorited
            ? const Icon(Icons.star)
            : const Icon(Icons.star_border),
      );  // IconButton
    }
    return const CircularProgressIndicator();
  },
);  // StreamBuilder
```

10. Other Requirements

10.1 Business Rule

10.1.1 Administrative Access to the System

Administrative access to the System shall only be granted to the Easy Food development team (5 members). Each administrator shall have their own administrative account, with sharing of accounts to be strictly prohibited.

Appendix A: Analysis Models

The following refer to the analysis models we have used in the documentation of our project (provided in submission folder):

1. System Architecture - “SystemArchitecture.jpg”
2. Dialog Map - “InitialDialogueMap.jpg”
3. Video Demo - “Demo Video.mp4”
4. Use Case Diagram - “EasyFood UseCaseDiagram - Final.pdf”
5. Sequence Diagrams - “EasyFood Sequence Diagrams.pdf”
6. Class Diagrams - “EasyFood Class Diagrams.pdf”
7. Source Code - “EasyFood Source Code.zip”

Appendix B: Application

To download the working model of the EasyFood Application - “app-release.apk” (file provided in submission folder)