FOOD RECIPE RELATED SOCIAL PLATFORM

BY

ANG YIK HANG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2021

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Food Recipe Related Social Platform

Academic Session: January 2021

I ANG YIK HANG

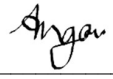
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



_____ _____

(Author's signature) (Supervisor's signature)

Address:

No 40, Lebuh Bercham (T) 1/3,

Taman Seri Bercham,

31400, Ipoh Perak.

Tan Joi San

_____

Supervisor's name

Date: 12th April 2021 **Date**: 15th April 2021

FOOD RECIPE RELATED SOCIAL PLATFORM

BY

ANG YIK HANG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

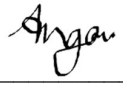
Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2021

DECLARATION OF ORIGINALITY

I declare that this report entitled "**Food Recipe Related Social Platform**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : Ang Yik Hang

Date : 12th April 2021

ACKNOWLEDGEMENTS

For this final year project, I would like to thank you for all the advice and guidance from my supervisor Dr. Tan Joi San, parents and friends. Thank you so much for offering the assistance to my food recipe social application development. I really appreciate that because this project could not be completed without these relevant advices and suggestions for improvement. When I encountered problems, the assistance that I've received is invaluable and it makes me getting better in project development and problem solving as well. I am grateful from the bottom of my heart.

ABSTRACT

In order to achieve long-distance communication, social platform has played a very important role in this modern era because it can connect people all over the world. For ordinary social networks like Instagram and Facebook, the information published on these channels usually has a high diversity. If the social network doesn't have a specific theme like food, it may cause problem for users that want to look for particular relevant content such as food recipes. In social network like Facebook, users face difficulty when searching for a recipe that only serves one people and have a less than twenty minutes cook time because Facebook is not specific for recipes sharing. Besides that, ordinary social network has made it more difficult for user to make friends that have same interest as well. Therefore, food social network is necessary to be proposed to solve such problem. The final deliverable of this project is a food-recipes social platform that can gather users with the same interest and share food-related information and recipes. By using the template provided in this food recipe social network, users can categorize the recipe, state the serve size and cook time before publishing to the community. Besides that, this social network also allow user to publish posts to reflect the food taste, service quality and sanitary condition of visited restaurant so other user can get basic understanding of that restaurant before visit there. Several food social networks have been reviewed in order to study the working logic of similar social network. Basic functionalities such as sign up, login, register and logout are implemented as well to make the application more systematic. This project is developed as an android native application using java and firebase as a backend for authentication and storing all data. Moreover, agile method is implemented during development which is prototype approach, the continuous improvements are implemented based on the initial project prototype until delivered a final product. The agile development approach can reduce cost and time required to develop a mobile application project. Although this project is completed, but improvements can still be done in the future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 Background Information

In the past two decades, the interaction between humanity across a long distance has always been a concern. As human beings are social animals, people need the communication to maintain healthy mind and strengthen the relationship with another individual. With such a demand, social network was born for connecting people all over the world. For example, Facebook, Instagram and Twitter are the most well know type of social network in current market. Not only provide a platform for connecting people all around the world, social network is also a great platform for information and knowledge sharing as well. For some particular community, social network may just design for a specific theme to focus on a specific knowledge and information sharing. Food-related information could be a social network theme because food lover is a large community that willing to share food-related knowledge and information such as food recipe. With food social network, food lovers who have the same interest are able to get in touch with each other easily. According to investigation, social network applications are the highest used mobile application over the world and it has been growing at exponential rates. It is a long story for social network development history. Some basic features of a social network such as build profile, build conversation, upload content was defined in this long journey.

The first created social media site was Six Degrees in 1997, this site allows user to create and set up the profile page, add other users into friend list and send messages via network. From 1997 to 2001, Six Degrees had reached a peak of about 1 million users. The major features of Six Degrees made it became a foundation of social application nowadays. The profile mechanism can greatly represent a user's background and information within the profile page is totally customizable. Moreover, the message features can help users to establish a connection with friends. The occurrence of Six Degrees had become an important milestone in social application development.

The first blogging sites, LiveJournal was proposed in year 1999. It is a platform that allow users to write blog in form format to keep updating daily lives. The blog format introduced by LiveJournal has become a social media sensation that is still popular nowadays where the Facebook status initially was designed base on blog format. After the popularization of blog site, social media channel began to increase in popularity. In

early 2000s, sites like Friendster and LinkedIn are launched and started to gain prominence. Friendster is a dating site that allows users to create personal profile, update daily status to reveal user's mood. For LinkedIn, it is a social media site that linked up with business as it is widely used in job finding and employee recruitment. Users of LinkedIn can post resume and send private message to other users for looking job opportunities.

In February 2004, the Facebook was launched by Mark Zuckerberg, along with his Harvard roommates. Initially, Facebook is only available for Harvard students but after months and years, the membership of Facebook is distributed worldwide. In 2012, Facebook announced that the number of users had reached a milestone – 1 billion. After 16 years, Facebook is still a leading giant in social network field and keep affecting the world. During these 16 years, the features of Facebook is getting better and complete eventually and it has become a sense of universality for having a Facebook account.

1.2 Problem Statement

In ordinary social network, the information published by user are usually not uniform. Different users can publish different contents to reflect daily status or interests. For food lover community, if the social network doesn't have a specific theme like food, it may cause problem when user try to look for particular relevant content such as food recipes and food-related posts. For example, users of general social network may face difficulty when search for a food recipe that only serves one people and have a less than twenty minutes cook time because the social network is not specific for recipes sharing, many features that can improve recipe sharing experience is not supported such as recipe template for easier recipe creation and recipe filter for easier search. Besides that, social networks without specific theme have made it more difficult for user to make friends that have same interest. Therefore, it is very necessary to build a food social network with specific features for food lover community to solve the problems above.

1) Features that improve recipe sharing experience are not supported by ordinary social networks.

For users who like to cook, recipe sharing is a very useful and interesting feature because through the recipe publish by others, user is able to learn how to cook cuisine more quickly and effectively. Although popular social networks nowadays are very mature, but unfortunately some problems may still occur when it comes to recipe sharing. That is because these social networks do not provide specific features that support recipe sharing such as recipe template and recipe posts filter. Therefore, it may affect food lover experience when sharing recipe or searching relevant recipes. Generally, a recipe is consisting of basic information such as recipe title, ingredient list and step by step tutorial. In ordinary social network, recipe post usually is written in a status format. Therefore, user may be confused when viewing the recipe because the recipe content layout that manually arranged by the author may not be intuitive. Moreover, for aged users, extra efforts are needed when arrange recipe content manually. Besides that, searching favourite recipe posts on ordinary social network can be inconvenient because it is lack of recipe filter feature. In this case, user cannot find any effective way when searching a recipe base on category, serve size and cook time.

2) Sharing among people with the same interests is difficult in social network without specific theme.

Food lover is an enormous community. There is always a group of people who are willing to share food information and knowledge with each other. The reason of why ordinary social network is not suitable for such community is because the information published on these social networks have a high diversity and not uniform. Different information such as international news, entertainment and sport can be found when using these social networks. In this case, difficulty may occur when user try to share information with others that have same interest.

1.3 Project Objectives

The main objective of this project is to develop a food social network for food lover community to support food-related information sharing such as food recipes and food posts become more effective and easily. According to the problem statements, two objectives are stated as below:

1) Implement features that can improve food recipes sharing experience.

When users publish a recipe in social network like Facebook or Instagram, the content of the recipe must be arranged manually which is not convenient and might confuse the reader when viewing the recipe. Hence, a prebuild template that support recipe sharing should be propose in this food social network. This recipe template should have a visual friendly and reasonable layout that allow user to fill in all recipe data such as recipe title, recipe description, serve size, estimate cook time, category, ingredient list and step by step tutorial. Besides that, this food social network should also support recipe filter so that all users are able to search interested recipes more quickly and effectively. User can filter food recipes base on category, serve size, cook time, popularity and timestamp. These rich filter options can enhance user experience when sharing or retrieving food recipes.

2) Create a social network with food theme and implement features that support social functionality to enhance usability and sharing effectiveness.

In general, a social network should support features such as register, login, reset password, user profile, chat, like, comment, and etc. Therefore, this project is going to implement these features as well. In addition, this food social network should also support user to write and publish food post for sharing daily meal status. Bookmark feature should also be implemented to let users save favourite recipes and food posts. After bookmark operation is done, the bookmarked items are saved into the user profile thus user can easily retrieve it.

1.4 Project Scope

A food social network known as “Tastiee” are delivered. It is a social application that design for food lovers to share food-related information and knowledge such as food recipes. The main motivation to develop Tastiee is because ordinary social network without theme does not support features that improve recipe sharing experience such as recipe template and recipe filter. Instead of just support recipe sharing, user can also write and publish food post to reflect daily meal status and interest. Besides that. “Tastiee” supports some social app general features such as authentication, profile management, chat system, bookmark feature, and etc.

First of all, user can register an account by using personal email and password. If the user accidentally forgot the password, user can choose the forgot password option in login interface and follow the instruction given to reset the password. With account authentication, the user identity is verified before login to Tastiee and the security of user’s data can be ensured. Besides that, users can customize their profile to reflect personal characteristic. Users are able to upload profile picture and modify personal information such as username, profession, bio and email. User can search another user by username and view other user’s profile. All created and bookmarked recipes and posts are displayed in user’s profile. Tastiee also supports chatting feature among the users. There is a channel for user to chat with others to establish social connection. Users can write post to reflect daily meal status as well. User can attach the food photo from phone gallery and write text to describe the post. For post about dining experience in a restaurant, other user who read the post can have a basic understanding of that restaurant before visit there.

For food recipe sharing feature, user can create recipes via the prebuild template to improve the content layout. The recipe content such as image, title, description, serve size, cook time, category, ingredient list and step by step tutorial are filled by user according to the template layout. User can then publish the recipe to the community, other user can leave like and comment on that recipes. The popularity of a recipe is depending on the likes number, if the recipe has the most likes, it is going to appear at the top when other user search food recipe base on popularity. By default, all recipes are displayed according to the timestamp, which is the latest on top. In addition, recipe filter provided can enable user to filter recipes base on category, serve size and cook time. Therefore, user can get the targeted recipe more quickly and easily.

1.5 Proposed Approach

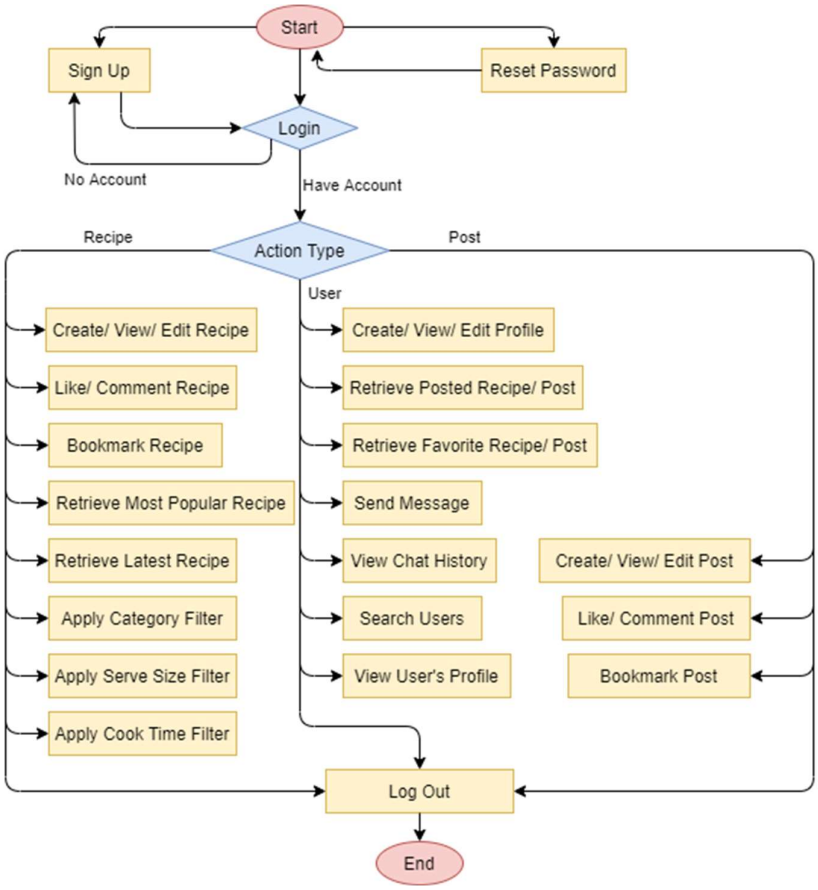


Figure 1.1 System Flow Chart

Firstly, user must register an account to have access to the social network. After profile setup is done, user is directed to the profile page. In this page, user can edit profile or access to posted recipes, bookmarked recipes, posted posts and bookmarked posts. User can go to the recipe page through the recipe tab on the bottom navigation menu bar. User can create and publish, modify or delete own recipes. User can also like, comment or bookmark interest recipes as well. User can click into each recipe to view the full details of that recipe. In addition, User can create own food post and publish in the post page. Similar with recipe, user is able to like comment, bookmark recipes as well. Modify and delete operation are only allow when user is the author of the post. Besides that, user can search other users by name, user can also send message to other user and view the chat history. Moreover, user can apply different filters when searching recipes such as category filter, serve size filter and cook time filter. Instead of filter, user can also sort recipes by popularity or latest on top. Lastly, user can sign out the account or reset password.

1.6 Highlight of What Have Been Achieved

Tastiee food social network is developed for food lover community that helps in recipe sharing and food-related information sharing. A lot of features have been implemented to achieve the project objectives. First of all, a complete user account mechanism is implemented to help in user identity authentication. Authentication and profile management is very essential in a social network as it is the way of how individuals are represented on the platform. Besides that, the recipe sharing feature has been successfully implemented as well. When user developed some unique recipes and want to share it out to benefit more people, user can make use of the prebuild recipe template to fill in every needed recipe data such as image, title, description, serve size, cook time, category, ingredients list, and step by step directions. With this template, user can save a lot of time as there is no need to manually arranged the recipe content. The number of ingredients field and directions field are totally managed by user as well. As there is no certain number of ingredient and direction for a recipe, user can click on add button to insert a new ingredient or direction field or click on delete button to remove ingredient and direction field, which make things easier to manage and modify.

After the recipe is published, edit and delete operation will only be enabled for author by using identity validation, thus no user can make change or delete other user's recipe. In addition, user can apply several filters such as category, serve size and cook time base on requirement. User can also sort the result by popularity or latest display on top. With help of filters, it can enhance user experience in recipe searching and save a lot of time for user to find the interest recipe. Instead of publish recipe, user can also write and publish food post to reflect daily meal status or share food-related information. For both recipes and food posts, user can perform operations such as like, comment or bookmark. Like numbers can reflect the popularity while comment feature can enable user to communicate or discuss with others. Bookmark feature helps user to save interested recipes into profile for easier retrieval in future.

In addition, user can search other users by username in users page. The search bar will detect the input text and search the user database for result generation. User can then click on image and enter to the user's profile page. The message feature has been implemented as well. When viewing another user's profile, user can click on send message button to enter the chat interface. After message is sent, sender can check the message status of whether the message has been seen by receiver. These implemented

features have completed this food social network system as the main goal of this food social network is to help food lover connect with others and share knowledge or information. The recipe sharing features such as recipe template and filter are also enhanced the recipe sharing experience as well.

1.7 Chapter Summary

In Chapter 1, the background information section has introduced the history of social sites development. An overview has been done from the very first recognized social site – Six Degrees, until today most popular social network – Facebook. The problem domain and motivation of this project is introduced with two problem statements. After that the project scope and 2 project objectives are discussed as well. The project scope described that the deliverable at the end of the project. After that, the proposed approach has been shown and achievements of this project have been highlighted and described as well.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview

In this chapter, three food-related applications are reviewed which are Yummly, FoodTribe, and I'm Hungry. Yummly and I'm Hungry are more emphasized on recipe feature while FoodTribe is more emphasized on sharing daily food-related post. For each of them, an account is registered to test the major features in order to point out the strengths and weaknesses.

2.2 Yummly

Yummly is a very popular recipe mobile app that provides recipe recommendation base on individual's taste. Users are not allowed to publish or share personal recipe in this app. It acts like a recipe library on hand as many recipes can be found in this app. It also supports create shopping list and one-hour grocery delivery service within particular country. This app is available on both ios and android device. In 2014, Yummly had 15 million active users in the US and has launched international websites in the UK, Germany and The Netherlands.

User can create account on Yummly easily through Google or Facebook login option. After registration is done, user is directed to the home page. Yummly app will display some trending recipes that may suit user's taste in this page. When user click on the interest recipe, the complete recipe content will load in a single page. Information such as image, title, nutrition, servings, preparation time, ingredient list and step by step tutorial are shown here. After user has tried out the recipe, user can write review message and these reviews will be shown at the bottom of recipe page. It also supports shopping feature where user can add all ingredients into shopping list and buy from local retailer, but this service is only available in some particular countries. To get step by step direction of some premium recipes, user can click on "Get Direction" button at the bottom of recipe page. Then, user is directed to a website that contains full tutorial guide.

User are not allowed to create and publish personal recipe on Yummly. There are many recipe categories available on Yummly app such as video guided, trending now, kid friendly, diets, and etc. User can search interest recipes according to the category provided. When user click into interested category, Yummly will display most relevant recipes recorded in recipe library. These recipes are managed and saved by Yummly so

every recipe has been reviewed critically to ensure the recipe quality. In profile page, the recipes that liked before by user are stored in several default recipe category tabs. User can also create a new collection by giving a collection name and description.

2.2.1 Strengths of Yummly

1. Rich recipes library, up to 2 million recipes
2. Up to 12 categories of recipes is sorted.
3. User can save recipe into collections.
4. Few steps to register an account.
5. The quality of recipe is high as all recipes is qualified before listed.
6. Does not support search recipes by serve size and preparation time.

2.2.2 Weaknesses of Yummly

1. User cannot publish personal recipe on this platform which means sharing of personal recipe is not allowed.
2. Profile only consist of name and profile image, which is not enough to reflect personal characteristic.
3. Does not support multilingual.
4. The ingredient shopping feature only supported in some countries.

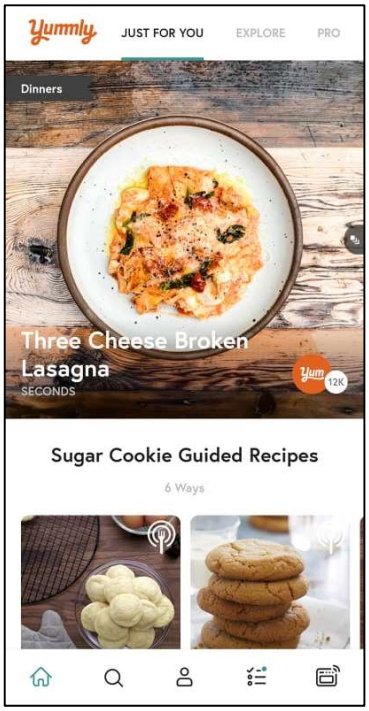


Figure 2.1 Home Page (Yummly)

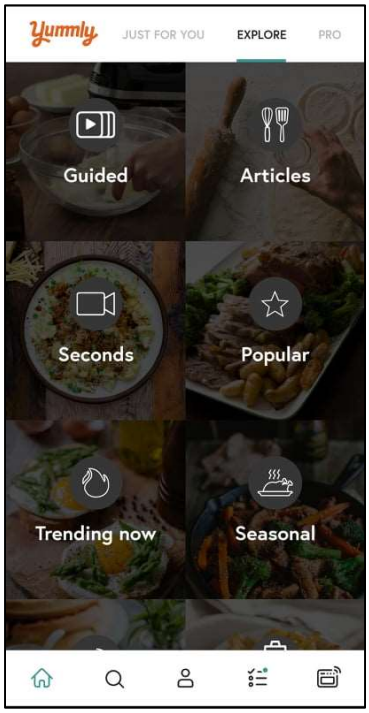


Figure 2.2 Explore Page (Yummly)

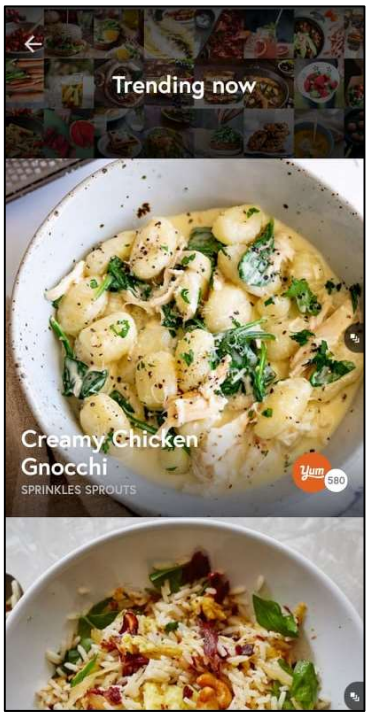


Figure 2.3 Categorized Page (Yummly)

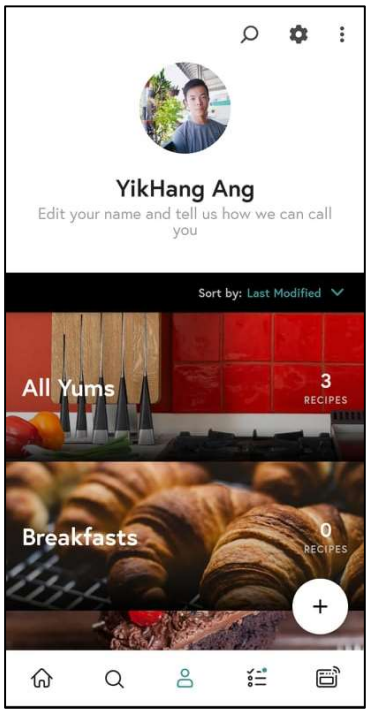


Figure 2.4 Profile Page (Yummly)

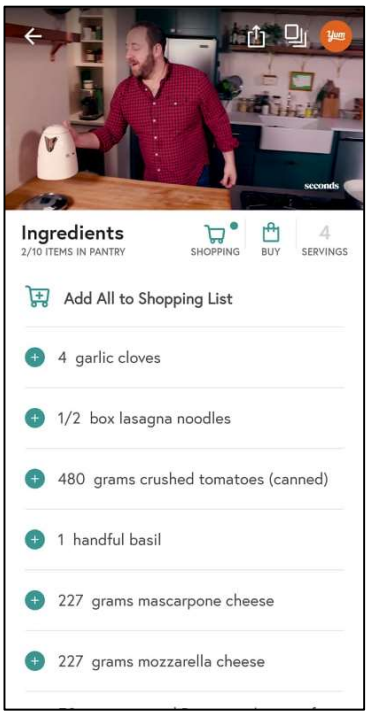


Figure 2.5 Ingredient List (Yummly)

Figure 2.6 Direction Details (Yummly)

2.3 FoodTribe

FoodTribe is a social media that design for food lovers. It is downloadable in google play store and it is totally free for use. The major feature of this application is allowed user share food-related status and information. Users are prompted to select the interest tribes when a new registration is done. There are many tribes available for user to follow such as Home Cooking, Sweet Treats, Meat, Vegan, Street Food etc. The tribes are divided into 49 categories and at least one of them should be choose by users. After followed the tribe, the tribe-related post published by others are shown on user's timeline. Moreover, tribes followed by user is going to display in user profile page hence users can know the interested tribes of each other. User can also modify the interest tribes whenever needed.

When found some interesting post, user can follow the publisher to receive the new post on time. Every user has a follower list and following list and it is totally transparent to others. After followed a user, the chat feature is enabled, and chat data is not public in order to ensure user privacy. Like general social applications, user is allowed to like and comment to any post publish by others. User can also choose whether publish the post on profile or tribes. User should at least attach a photo or video in every post and the caption can be write at template's bottom section. The overall layout of post in FoodTribe is quite simple. A thumbnail photo is displayed at the top of the post and following by the post title. The post content is arranged under the title as well but due to the text alignment, the content displayed is not neat as expected.

Moreover, there is another attractive feature in FoodTribe which is the quizzes feature. User can create quiz base on a particular tribe, the questions can be set by publisher and answer type can also be define as checkbox, radio button, short answer when needed. At the bottom of each quiz, the participants can comment to share opinion against the quiz question. The quiz feature in FoodTribe can help to spread knowledges and enhance the interactivity level among the users.

2.3.1 Strengths of FoodTribe

1. User can publish food-related post and categorize it into corresponding tribe easily.
2. Up to 49 tribes that can be subscribed by users.
3. Few steps to set up an account.

4. The overall design of user interface is neat and easy to view.
5. Quiz can be conduct by users to share food knowledges and ideas.
6. Completed social features such as like, comment and chat are available.

2.3.2 Weaknesses of FoodTribe

1. Does not support recipe sharing
2. Does not support multilingual.
3. The random advertisements that showed in application affects the user's experience.
4. Not friendly for new user as it is lack of tutorial and documentation, new user may not know how to perform some particular feature such as create new quiz.
5. Every post needs to attach at least one photo or video, it may cause problem when there is no suitable photo for use.
6. Text alignment of post is not neat.

Figure 2. 7 Tribes Page (FoodTribe)

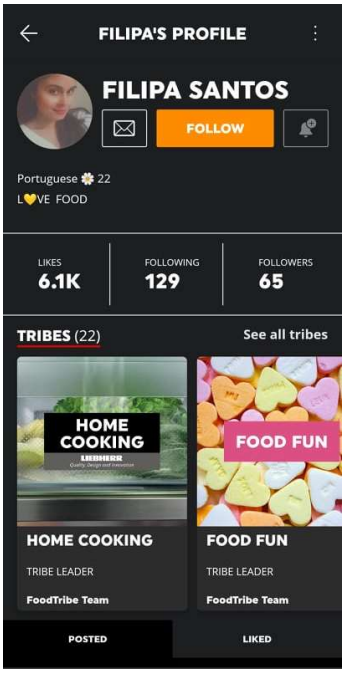


Figure 2. 8 User Page (FoodTribe)

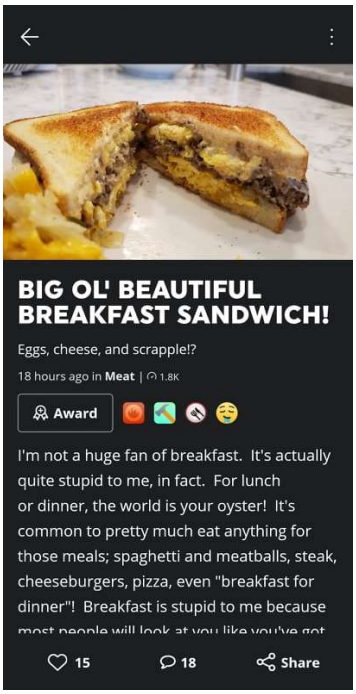


Figure 2. 9 Food Post (FoodTribe)

Figure 2. 10 Quiz Page (FoodTribe)

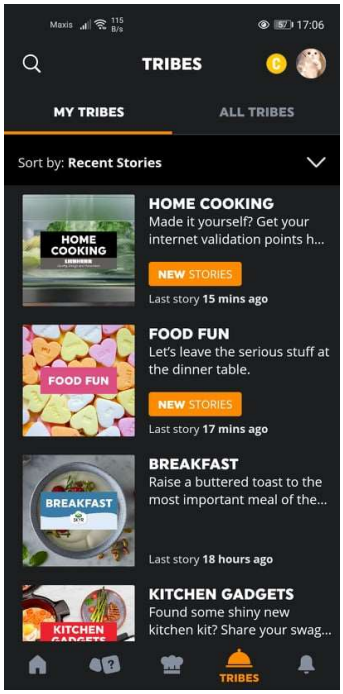


Figure 2. 11 My Tribe (FoodTribe)

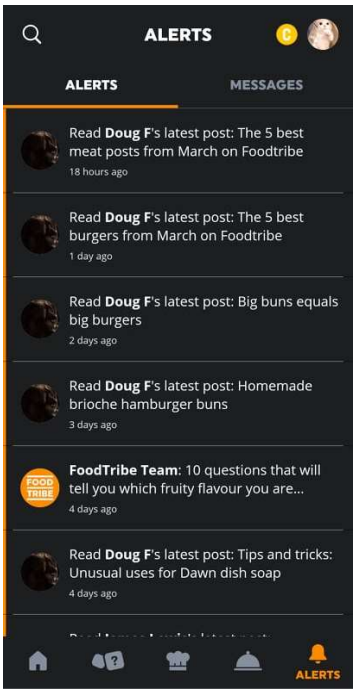


Figure 2. 12 Notification (FoodTribe)

2.4 I'm Hungry

I'm Hungry is an android app that helps people in discovering food recipes. User can quickly sign up an account by using google account sign up option. After registered, user is directed to the home page that displayed random food recipes. In this application, users are not allowed to create and share personal recipes. Every recipe displayed in this application is basically from website source. For each recipe, the website domain name is shown under the food recipe title, means that the food recipe is getting from this website.

When user click into interested recipe, user is directed to a recipe page that contains all recipe data such as recipe title, website source, category, preparation time, and ingredient list. At the bottom of the recipe page, user can click on view full recipe button and user is directed to the website that contain step by step direction tutorial. The application also provided search recipe feature. User can search recipes base on keyword or directly select the food category provided.

In addition, this recipe has supported meal plan feature. User can attach recipe to a particular day to have a more appropriate diet plan. This meal plan feature can help user organize the diet habit more easily as it can remind user of what should eat in the coming days. Moreover, this application is also vegan-friendly. User can choose to disable all meat recipes and allow only vegetarian recipes display in home screen by adjust the setting in profile page.

2.4.1 Strengths of I'm Hungry

1. Vegan-friendly, can disable all meat recipes from being displayed.
2. Number of recipes is large as most of the recipes are from internet source.
3. Search recipe by category can help user filter recipes more quickly.
4. Support meal plan feature.

2.4.2 Weaknesses of I'm Hungry

1. Lack of social functionality such as user profile, chat and search user are not available.
2. User cannot share personal recipe on this platform.
3. Does not support multilingual
4. Rough application user interface design
5. Does not provide search recipes by serve size and preparation time.

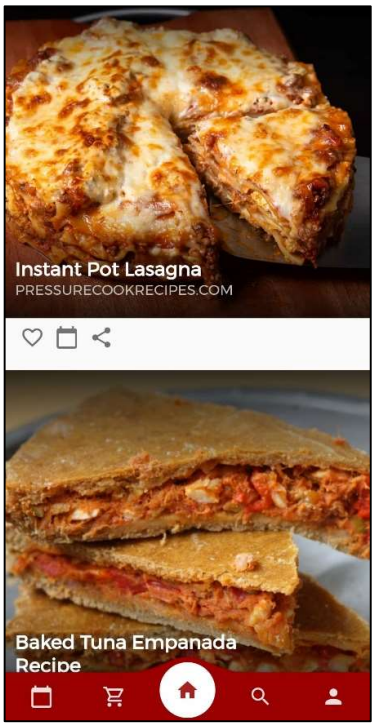


Figure 2.13 Home Page (I'm Hungry)

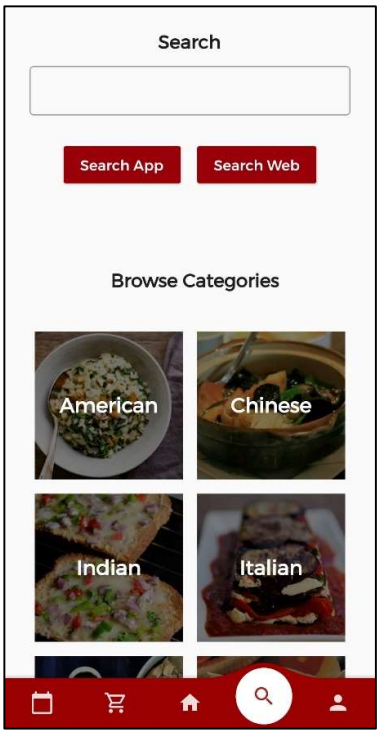


Figure 2.14 Search Page (I'm Hungry)

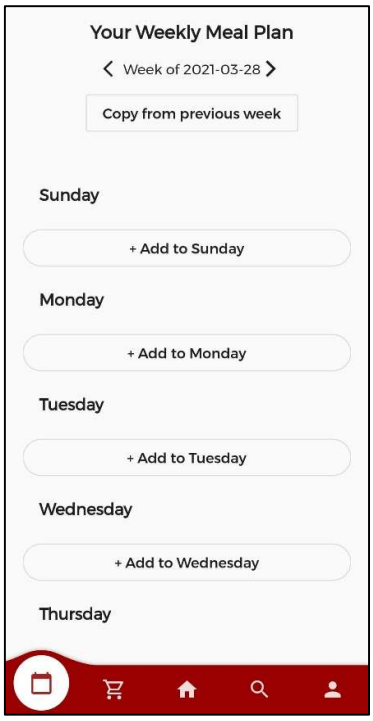


Figure 2.15 Meal Plan (I'm Hungry)

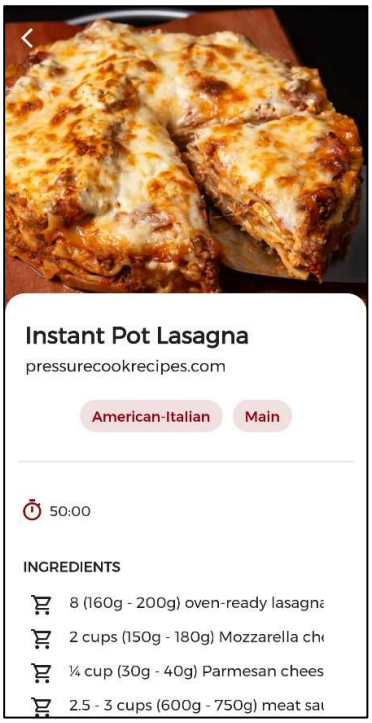


Figure 2.16 Recipe Page (I'm Hungry)

Figure 2.17 Ingredient List (I'm Hungry)

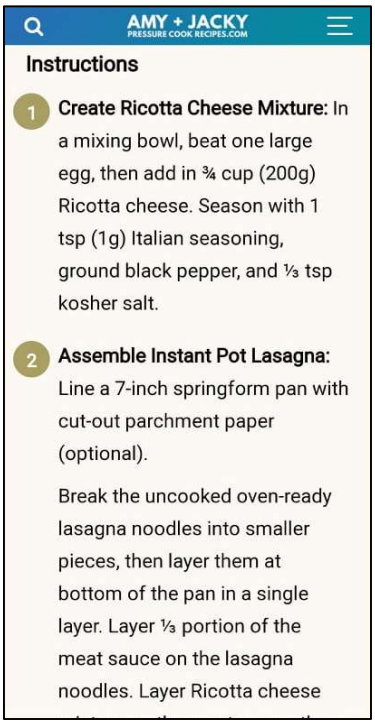


Figure 2.18 Direction Details (I'm Hungry)

2.5 Chapter Summary

In this Chapter, three food-related applications have been reviewed which are Yummly, FoodTribe, and I'm Hungry. First of all, the feature overview of application is written. Next, the strengths and weaknesses of these three applications are also listed in point form. Table 2.1 is briefly summarized the strengths and weaknesses of these applications.

Table 2. 1 Comparison of three applications

	Yummly	**FoodTribe**	**I'm Hungry**
Recipes Book Features	Available	Not Available	Available
Sharing of Personal Recipes	Not Available	Not Available	Not Available

Save Recipe Posts into Collection	Available	Not Available	Not Available
Sharing of Personal Posts	Not Available	Available	Not Available
Comment Features	Available	Available	Available
Notification Page	Not Available	Available	Not Available
Recipe Template	Not Available	Not Available	Not Available
Search Recipe by Attribute Such as Serve Size and Time	Available	Not Available	Available
Content Subscription of Interest Food Type	Not Available	Available, up to 49 food tribes can be subscribed.	Not Available
Support Multilingual	Not Available	Not Available	Not Available

CHAPTER 3 SYSTEM DESIGN

3.1 Overview

This chapter is mainly to describe the system design by showing the use case diagram and use case table. In addition, module and coding explanation are written to explain how the works are done. Coding screenshots are displayed and explained accordingly.

3.2 Use Case Diagram

Figure 3.1 System Use Case Diagram

3.3 Use Case Description

Table 3.1 Sign Up Use Case

Use Case Name: Sign Up	**ID:** 1	**Primary Actor:** User
Include use case: Create Profile		
Normal Flow of Event: 1. User fill in the email, password and confirmation password in registration page. 2. User click on register button. 3. System validate registration email and password. 4. System trigger **Create Profile** use case.		
Sub Flow: 1a. User check the "show password" checkbox. 1. The password and confirmation password become visible.		
Alternative Flow: 3a. Password not same with confirmation password. 1. System show error toast message. 3b. Empty field detected. 1. System show error toast message.		

Table 3.2 Create Profile Use Case

Use Case Name: Create Profile	**ID:** 1-1	**Primary Actor:** User
Normal Flow of Event: 1. User select profile image from device gallery 2. User fill in personal information (username, bio, profession, email). 3. User click on save profile button		
Sub Flow: Not Applicable		
Alternative Flow: 3a. Empty field detected 1. System show error toast message.		

Table 3.3 Sign In Use Case

Use Case Name: Sign In	**ID:** 2	**Primary Actor:** User
Normal Flow of Event: 1. User fill in the email, password in login page. 2. User click on sign in button. 3. System validate user. 4. User are directed to main page.		
Sub Flow: 1a. User check the "show password" checkbox. 1. The password become visible.		
Alternative Flow: 3a. Not valid email and password. 1. System show error toast message. 3b. Empty field detected. 1. System show error toast message.		

Table 3.4 Reset Password Use Case

Use Case Name: Reset Password	**ID:** 3	**Primary Actor:** User
Normal Flow of Event: 1. User click on forget password button. 2. User fill in an email to receive reset link. 3. User click on submit button 4. User reset password by the reset link in email.		
Sub Flow: Not applicable		
Alternative Flow: 3a. Not valid email 1. System show error toast message. 3b. Empty email detected. 1. System show error toast message.		

Table 3.5 Edit Profile Use Case

Use Case Name: Edit Profile	**ID:** 4	**Primary Actor:** User
Normal Flow of Event: 1. User click edit button on profile page. 2. User modify profile information. 3. User click on update profile button. 4. Changes are saved and updated on database.		
Sub Flow: Not applicable		
Alternative Flow: Not applicable		

Table 3.6 Search User Use Case

Use Case Name: Search User	**ID:** 5	**Primary Actor:** User
Normal Flow of Event: 1. User type username in search bar 2. The input text is used to filter user in database 3. Filtered result will display to user, 4. Extension point: **View User Profile**, **Send Message**		
Sub Flow: Not applicable		
Alternative Flow: Not applicable		

Table 3.7 View User Profile Use Case

Use Case Name: View User Profile	**ID:** 5-1	**Primary Actor:** User
Extend Use Case: Search User		
Normal Flow of Event: 1. User click the profile image from searched user row. 2. User are directed to target profile page.		

Table 3.8 Send Message Use Case

Use Case Name: Send Message	**ID:** 5-2	**Primary Actor:** User
Extend Use Case: Search User		
Normal Flow of Event: 1. User click the message icon from searched user row. 2. User are directed to 1 on 1 chat interface. 3. User type text in message box 4. User click send button 5. The message is sent to the receiver.		

Table 3.9 View Chat History Use Case

Use Case Name: View Chat History	**ID:** 6	**Primary Actor:** User
Normal Flow of Event: 1. User click on chat tab at the search user page. 2. User is directed to the chat history page. 3. The recent chat user is display along with last message.		
Sub Flow: 3a User click the chat row and enter the chat interface.		
Alternative Flow: Not applicable		

Table 3.10 View Profile

Use Case Name: View Profile	**ID:** 7	**Primary Actor:** User
Normal Flow of Event: 1. User click on profile tab at the bottom navigation menu bar 2. User is directed to the personal profile page 3. Extension point: **View Posted Recipe**, **View Bookmarked Recipe, View Posted Post, View Bookmarked Post**		
Sub Flow: Not applicable		
Alternative Flow: Not applicable		

Table 3.11 View Posted Recipe Use Case

Use Case Name: View Posted Recipe	**ID:** 7-1	**Primary Actor:** User
Extend Use Case: View Profile		
Normal Flow of Event: 1. User click the posted recipe tab on profile page. 2. User are directed to show posted recipe page. 3. The recipes posted by user are filtered and displayed in this page according to latest on top basis.		

Table 3.12 View Bookmarked Recipe Use Case

Use Case Name: View Bookmarked Recipe	**ID:** 7-2	**Primary Actor:** User
Extend Use Case: View Profile		
Normal Flow of Event: 1. User click the favourite recipe tab on profile page. 2. User are directed to show bookmarked recipe page. 3. The recipes bookmarked by user are filtered and displayed in this page according to latest on top basis.		

Table 3.13 View Posted Post Use Case

Use Case Name: View Posted Post	**ID:** 7-3	**Primary Actor:** User
Extend Use Case: View Profile		
Normal Flow of Event: 1. User click the posted post tab on profile page. 2. User are directed to show posted post page. 3. The posts published by user are filtered and displayed in this page according to latest on top basis.		

Table 3.14 View Bookmarked Post Use Case

Use Case Name: View Bookmarked Post	**ID:** 7-4	**Primary Actor:** User
Extend Use Case: View Profile		
Normal Flow of Event: 1. User click the favorite post tab on profile page.		

2. User are directed to show bookmarked post page. 3. The posts bookmarked by user are filtered and displayed in this page according to latest on top basis.

Table 3.15 Publish Food Post Use Case

Use Case Name: Publish Food Post	**ID:** 8	**Primary Actor:** User
Normal Flow of Event: 1. User click on add post button. 2. User write post according to the template given. 3. User click on submit button.		
Sub Flow: Not applicable.		
Alternative Flow: 3a. The image field or description field is missing 1. System show error toast message.		

Table 3.16 View Food Post Use Case

Use Case Name: View Food Post	**ID:** 9	**Primary Actor:** User
Normal Flow of Event: 1. User click on interest food post. 2. User is directed to a single food post page with full content. 3. Extension point: **Edit Food Post, Delete Food Post**		
Sub Flow: Not applicable		
Alternative Flow: Not applicable		

Table 3.17 Edit Food Post Use Case

Use Case Name: Edit Food Post	**ID:** 9-1	**Primary Actor:** User
Extend Use Case: View Food Post		
Precondition: User is the author of this food post.		
Normal Flow of Event:		

1. User click edit button on own food post. 2. User modify post content. 3. User click save button. 4. User is directed to previous page.

Table 3.18 Delete Food Post Use Case

Use Case Name: Delete Food Post	**ID:** 9-2	**Primary Actor:** User
Extend Use Case: View Food Post		
Precondition: User is the author of this food post.		
Normal Flow of Event: 1. User click delete button on own food post. 2. An alert dialog box will prompt user for the delete confirmation. 3. User click on yes button. 4. The post data is removed from database. 5. User is directed to previous page.		
Alternative Flow: 3a. User doesn't want to proceed delete operation 1. User click on cancel button. 2. User is directed to previous page.		

Table 3.19 Publish Recipe Use Case

Use Case Name: Publish Recipe	**ID:** 10	**Primary Actor:** User
Normal Flow of Event: 1. User click on add recipe button in recipe page. 2. User fill in recipe data according to the template given 3. User click on submit button.		
Sub Flow: Not applicable.		
Alternative Flow: 3a. Necessary recipe data field is missing. 1. System show error toast message.		

Table 3.20 View Recipe Use Case

Use Case Name: View Recipe	**ID:** 11	**Primary Actor:** User
Normal Flow of Event: 1. User click on interest recipe. 2. User is directed to a single recipe page that contain full recipe details. 3. Extension point: **Edit Recipe, Delete Recipe**		
Sub Flow: Not applicable		
Alternative Flow: Not applicable		

Table 3.21 Edit Recipe Use Case

Use Case Name: Edit Recipe	**ID:** 11-1	**Primary Actor:** User
Extend Use Case: View Recipe		
Precondition: User is the author of this recipe.		
Normal Flow of Event: 1. User click edit button on own recipe. 2. User modify recipe data according to the template given. 3. User click save button. 4. User is directed to previous page.		

Table 3.22 Delete Recipe Use Case

Use Case Name: Delete Recipe	**ID:** 11-2	**Primary Actor:** User
Extend Use Case: View Recipe		
Precondition: User is the author of this Recipe.		
Normal Flow of Event: 1. User click delete button on own recipe. 2. An alert dialog box will prompt user for the delete confirmation. 3. User click on yes button. 4. The recipe data is removed from database. 5. User is directed to previous page.		
Alternative Flow:		

3a. User doesn't want to proceed delete operation 1. User click on cancel button. 2. User is directed to previous page.

Table 3.23 Filter Recipe Use Case

Use Case Name: Filter Recipe	**ID:** 12	**Primary Actor:** User
Normal Flow of Event: 1. User click on filter icon at the top right corner in recipe page. 2. A section that contains filter options are expanded. 3. User select the filter properties base on category, serve size or cook time. 4. User click on filter button.		
Sub Flow: Not applicable.		
Alternative Flow: 4a. User want to sort the arrangement of filtered recipes. 1. User switch the sort mode between popularity and latest by clicking the button besides the filter icon. 2. The filtered recipes change the sort mode accordingly.		

Table 3.24 Bookmark Food Post Use Case

Use Case Name: Bookmark Food Post	**ID:** 13	**Primary Actor:** User
Normal Flow of Event: 1. User click on bookmark icon on the food post. 2. Bookmark icon become red color. 3. The bookmarked food post is saved to the profile as a collection.		

Table 3.25 Comment Food Post Use Case

Use Case Name: Comment Food Post	**ID:** 14	**Primary Actor:** User
Normal Flow of Event: 1. User click on comment button on the food post. 2. User is directed to the comment interface. 3. The previous comments of this food post are displayed accordingly.		

4. User type in comment message. 5. User click on send button. 6. The comment is saved into database and displayed accordingly.

Table 3.26 Like Food Post Use Case

Use Case Name: Like Food Post	**ID:** 15	**Primary Actor:** User
Normal Flow of Event: 1. User click on like button on the food post. 2. The color of like button become red. 3. The number of likes will increase and display in real time.		
Alternative Flow: 2a. User want to dislike the food post. 1. The red color of like button is removed and become white color. 2. The number of likes will decrease and display in real time.		

Table 3.27 Bookmark Recipe Use Case

Use Case Name: Bookmark Recipe	**ID:** 16	**Primary Actor:** User
Normal Flow of Event: 1. User click on bookmark icon on the recipe. 2. Bookmark icon become red color. 3. The bookmarked recipe is saved to the profile as a collection.		

Table 3.28 Comment Recipe Use Case

Use Case Name: Comment Recipe	**ID:** 17	**Primary Actor:** User
Normal Flow of Event: 1. User click on comment button on the recipe. 2. User is directed to the comment interface. 3. The previous comments of this recipe are displayed accordingly. 4. User type in comment message. 5. User click on send button. 6. The comment is saved into database and displayed accordingly.		

Table 3.29 Like Recipe Use Case

Use Case Name: Like Recipe	**ID:** 18	**Primary Actor:** User
Normal Flow of Event: 1. User click on like button on the recipe. 2. The color of like button become red. 3. The number of likes is increased and display in real time.		
Alternative Flow: 2a. User want to dislike the recipe. 1. The red color of like button is removed and become white color. 2. The number of likes will decrease and display in real time.		

Table 3.30 Sign Out Use Case

Use Case Name: Sign Out	**ID:** 19	**Primary Actor:** User
Normal Flow of Event: 1. User click on sign out button on the profile page. 2. User will be sign out and directed to the login page.		

3.4 Module and Coding Explanation

Before the project development, the first thing to do is open a new android project in Android Studio and create a google account for Firebase. After that, the project must connect to the Firebase in order to utilize Firebase services such as authentication, storage and real time database. According to the require module, dependencies must be added to project. Dependency is used to include external library to the project in order to achieve additional features. Below are the coding and explanation of different project module:

1) Register Account

```
reg_btn.setOnClickListener((view) → {
        String email = register_email_field.getText().toString();
        String pass = register_pass_field.getText().toString();
        String confirm_pass = register_confir_pass_field.getText().toString();

        if (!TextUtils.isEmpty(email) && !TextUtils.isEmpty(pass) && !TextUtils.isEmpty(confirm_pass)){
            if (pass.equals(confirm_pass)){
                mAuth.createUserWithEmailAndPassword(email,pass).addOnCompleteListener((task) → {
                        if (task.isSuccessful()){
                            sendtoProfile();
                        }else{
                            String error = task.getException().getMessage();
                            Toast.makeText(getApplicationContext(), text: "Error :"+error, Toast.LENGTH_LONG).show();
                        }
                });
            }else{
                Toast.makeText(getApplicationContext(), text: "Confirm password and password field doesn't match!", Toast.LENGTH_LONG).show();
```

Figure 3.2 Register Account

The figure above is the essential code that run the register operation. The on click listener is set on register button in the first line. First of all, the user inputs for registration such as email, password and confirmation password are retrieved from the corresponding EditText and store in variable email, pass and confirm_pass. After that, the system checks if there is any missing field using if condition statement. If no missing fields detected, system will make sure the password is same as confirmation password and run the **createUserWithEmailAndPassword()** function to register an user to the firebase. If the registration task is successful, user is directed to profile page.

2) Sign In

```
protected void onStart() {
    super.onStart();
    FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
    if (currentUser != null){
        Intent intent = new Intent( packageContext: Login.this, MainPage.class);
        startActivity(intent);
        finish();
```

Figure 3.3 Sign In (1/2)

The above code is run when the log in page is launched. It is used to check if the user has already log out from the application. If the current user variable is not null, means that there is a user in active state, then the login page won't launch, and user is directed to the home page.

```
loginbtn.setOnClickListener((view) → {
        String loginEmail = loginEmailtext.getText().toString();
        String loginpass = loginPasswordtext.getText().toString();
        if (!TextUtils.isEmpty(loginEmail) || !TextUtils.isEmpty(loginpass)){
            mAuth.signInWithEmailAndPassword(loginEmail,loginpass).addOnCompleteListener((task) → {

                    if(task.isSuccessful()){
                        sendtoMain();
                    }else{
                        String error = task.getException().getMessage();
                        Toast.makeText( context: Login.this,  text: "Error :"+ error, Toast.LENGTH_SHORT).show();
                    }
            });
        }else{
            Toast.makeText( context: Login.this,  text: "Please fill in all fields.", Toast.LENGTH_SHORT).show();
```

Figure 3.4 Sign In (2/2)

When user click on login button, the email and password are retrieved from corresponding field and store into variable. It validates the input email and password, if user account exists, the **mAuth.signInWithEmailAndPassword()** function sign user in and direct user to recipe page.

3) Reset Password

```
submitEmail.setOnClickListener((view) → {
        mAuth.sendPasswordResetEmail(resetEmail.getText().toString()).addOnCompleteListener((task) → {
            if(task.isSuccessful()){
                Toast.makeText( context: ResetPassword.this,  text: "Reset link sent to your email.", Toast.LENGTH_SHORT).show();
            }else{
                Toast.makeText( context: ResetPassword.this, task.getException().getMessage(), Toast.LENGTH_SHORT).show();
```

Figure 3.5 Reset Password

The function above is used to run reset password operation. Once user click on submit button, the on click listener will listen to this event and retrieved the input email and pass it to **mAuth.sendPasswordResetEmail()** function. If the input email is valid and has been registered as one of the user accounts, the password reset link is sent to that email. If it is not a valid email, system will show the error toast message.

4) Create Profile

```
String currentUserID = FirebaseAuth.getInstance().getCurrentUser().getUid();
storageReference = FirebaseStorage.getInstance().getReference().child("Profile Images");
databaseReference = FirebaseDatabase.getInstance().getReference().child("Users").child(currentUserID);
```

Figure 3.6 Create Profile (1/5)

Before create user profile and upload data to the database, the reference that point to the target storing location must be declared first. Firebase storage is used to store image data and the storage reference has declared the path which is “Profile Images”. Later when the profile image is uploaded successfully, that image is stored as a child under “Profile Images”. Firebase real time database is used to store user data and the database reference has declared the path which is **(“Users”).child(currentUserID)**. Later, the system store user data into this reference to upload user data under the user ID.

```
    imageView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(intent, PICK_IMAGE);
    }
});
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if(requestCode == PICK_IMAGE && resultCode == RESULT_OK && data != null && data.getData() != null){
        imageUri = data.getData();
        Picasso.get().load(imageUri).into(imageView);
```

Figure 3.7 Create Profile (2/5)

The code above is used to trigger the device gallery intent. When user click on image view, the on click listener brings user to the device gallery intent. If user choose image from gallery intent, that image data is loaded to the image view in form of imageUri.

Later when user click on the save profile button, the imageUri is uploaded to the user database and firebase storage.

```
if(task.isSuccessful()){
    Uri downloadUri = task.getResult();
    final HashMap profile = new HashMap();
    profile.put("name",name);
    profile.put("bio",bio);
    profile.put("profession",profession);
    profile.put("email",email);
    profile.put("url",downloadUri.toString());
    profile.put("privacy","Public");
    databaseReference.updateChildren(profile).addOnCompleteListener((task) → {
```

Figure 3.8 Create Profile (3/5)

The code above shows that how to use hash map to upload user data into a reference. If the upload task is successful, the hash map is used to save all user personal data by given corresponding field name. After that, **updateChildren()** function used this hash map to update all user data directly to the target reference declared before which is databaseReference.

Figure 3.9 Create Profile (4/5)

The figure above shows that the user profile hash map has been uploaded successfully to the firebase database. The hasp map is stored under a unique user id which can represent a user. Other users are also stored under the parent Users norm in this way.

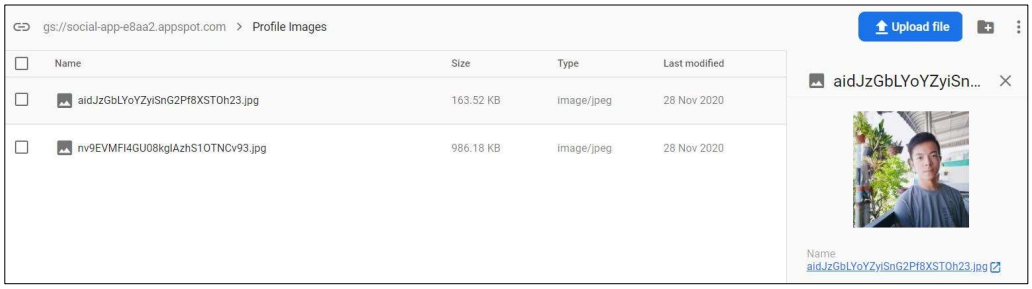


Figure 3.10 Create Profile (5/5)

The figure above shows that the profile image selected from device gallery has been uploaded and stored in Firebase Storage. Each image has an ID which in format "userID.jpg". Whenever the user change profile image, the new image is stored inside this image ID and replace the old profile image. It saved the storage because duplicated old profile image data will be erased.

5) Edit Profile

```
final HashMap profile = new HashMap();
profile.put("name",newnameData);
profile.put("bio",newbioData);
profile.put("profession",newprofessionData);
profile.put("email",newemailData);
profile.put("url",downloadUri.toString());
profile.put("privacy","Public");
databaseReference.updateChildren(profile).addOnCompleteListener((task) -> {
        if(task.isSuccessful()){
            Toast.makeText( context: EditProfile.this,  text: "Profile Updated", Toast.LENGTH_SHORT).show();
        }else{
            String message = task.getException().getMessage();
            Toast.makeText( context: EditProfile.this,  text: "Error: " + message, Toast.LENGTH_SHORT).show();
```

Figure 3.11 Edit Profile (1/2)

After user modify profile and click save button, the new profile data is retrieved from corresponding fields and store in variables. After that, a hash map is generated, and these variables are stored in this hash map. Finally, the hash map is used to update user database.

```
Query updatemyReview = ReviewsRef.orderByChild("uid").startAt(currentUserID).endAt(currentUserID + "\uf8ff");
updatemyReview.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        for(DataSnapshot ds : snapshot.getChildren()){
            String key = ds.getKey();
            ReviewsRef.child(key).child("name").setValue(newnameData);
            ReviewsRef.child(key).child("profileimage").setValue(downloadUri.toString());
```

Figure 3.12 Edit Profile (2/2)

After the user data is updated in database, the food post database and recipe database must be updated as well. (The food post and recipe function will be discussed later). Because both of recipe and food post in Firebase Database also contains two user data which are profile image and username. If these two attributes are not updated, when displaying the recipe or food post, the username and profile image shown will still remain the same as before profile modification. The query updatemyReview will filter all food posts to look for the review that published by current user. After that, the query used **addListenerForSingleValueEvent()** function to detect the datasnapshot. For each snapshot which is each food post, all name field and profile image field are updated in database. The same concept has applied on recipe database as well for the update operation.

6) Profile

```
if(task.getResult().exists()){
    String nameData = task.getResult().getString( field: "name");
    String bioData = task.getResult().getString( field: "bio");
    String professionData = task.getResult().getString( field: "profession");
    String emailData = task.getResult().getString( field: "email");
    String url = task.getResult().getString( field: "url");

    Picasso.get().load(url).into(imageView);
    nameTv.setText(nameData);
    bioEt.setText(bioData);
    professionEt.setText(professionData);
    emailEt.setText(emailData);
}else{
    Toast.makeText( context: Profile.this,  text: "No Profile Exist", Toast.LENGTH_SHORT).show();
    Intent intent = new Intent( packageContext: Profile.this,CreateProfile.class);
    startActivity(intent);
    finish();
```

Figure 3.13 Profile (1/2)

The above code executed when the profile page is launched. The user data is retrieved from database and stored into the variables. These variables are loaded to the corresponding field and display to users. If no data found in database, user is directed to create profile activity.

```
private void showRecipesNo() {
    DatabaseReference recipeRef = FirebaseDatabase.getInstance().getReference().child("Recipes");
    recipeRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            int i = 0;
            for(DataSnapshot dataSnapshot : snapshot.getChildren()){
                if ((dataSnapshot.child("uid").getValue().toString()).equals(currentUserID)){
                    i = i + 1;
                }
            }
            recipesNumber.setText(i+" Posted Recipes");
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```

Figure 3.14 Profile (2/2)

The code above is used to calculate and display the number of recipes posted by user. First of all, a database reference is created and pointed to a target database, after that the for loop has loop through this database to find the recipe that have current user id in the uid field. The value of integer i will be increased by 1 when one recipe was found in this loop. At the end, the number is displayed in bottom of profile page to indicate the number of users posted recipes. The same concept is applied on bookmarked recipe, posted food post and bookmarked food post as well. It is just the query codes are different when filtered the data. This is the place where user can retrieve the posted and bookmarked item easily.

7) Display Posted Recipes

```
private void ListMyRecipesAccordingCounter() {
    mRecipes = new ArrayList<>();
    reference = FirebaseDatabase.getInstance().getReference().child("Recipes");
    reference.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            mRecipes.clear();
            for(DataSnapshot dataSnapshot : snapshot.getChildren()){
                if((dataSnapshot.child("uid").getValue().toString()).equals(currentUserID)){
                    Recipe recipe = dataSnapshot.getValue(Recipe.class);
                    mRecipes.add(recipe);
                }
            }
            for (int i=0; i<mRecipes.size()-1; i++){
                for (int j=0; j<mRecipes.size()-1-i; j++){
                    if (Integer.parseInt(mRecipes.get(j).getCounter()) < Integer.parseInt(mRecipes.get(j+1).getCounter())){
                        Recipe temp = mRecipes.get(j);
                        mRecipes.set(j, mRecipes.get(j+1));
                        mRecipes.set(j+1, temp);
                    }
                }
            }
            recipeAdapter = new RecipeAdapter(getApplicationContext(),mRecipes);
            all_recipes_list.setAdapter(recipeAdapter);
        }
```

Figure 3.15 Display Posted Recipes

The code above is used to display all posted recipes according to counter attribute. First of all, an object array list is initiated for storing recipe object. After that a database reference has been declared and pointed to recipe database. Then, a for loop is executed to find all recipes posted by user. When a recipe is found in this for loop, the recipe is transformed into a recipe object and store into the array list. At this moment, the recipe objects in this array list are not being sorted according to the counter. To make the latest recipe display on top, every recipe contains a counter attribute when uploaded to the database. So, after obtaining the recipe array list, **bubble sort** is used to sort the recipe object according to the counter. After bubble sort, the final array list is passed into a recipe adapter, the adapter is inserted into a recycler view and display to user. Then similar concept is applied on display bookmarked recipe, display posted food post, and display bookmarked food post as well. It is just the query codes are different when filtered the data.

8) Update User Online and Offline Status

```
//update online offline status
private void status(String status){
    DatabaseReference ref = FirebaseDatabase.getInstance().getReference().child("Users").child(currentUserID);

    HashMap hashMap = new HashMap();
    hashMap.put("status", status);

    ref.updateChildren(hashMap);

    //update status in firebase firestore
    FirebaseFirestore fstore;
    fstore= FirebaseFirestore.getInstance();
    fstore.collection( collectionPath: "user").document(currentUserID).update( field: "status", status);
}

@Override
protected void onResume() {
    super.onResume();
    status("online");
}

@Override
protected void onPause() {
    super.onPause();
    status("offline");
}
```

Figure 3.16 Update User Online and Offline Status

The code above is used to update user online and offline status in database. When the current activity is in resume state, the **onResume()** method is invoked and run the **status()** function to update the user current status to online. Whenever user switch to another application or turn off the mobile device, the current activity is transformed to pause state, then the **onPause()** method is invoked to update the user to offline status. The online and offline status are used to indicate the user online and offline status in search user activity. This code is applied in all activities to keep track of user status.

9) Create and Publish Recipe

```
private void addIngredientRow() {
    final View ingredientRowView = getLayoutInflater().inflate(R.layout.add_ingredient_row,  root: null,  attachToRoot: false);

    TextView ingNum = (TextView)ingredientRowView.findViewById(R.id.ingnumbertv);
    ingNum.setText((ingredientLL.getChildCount()+1)+".");
    EditText ingET = (EditText)ingredientRowView.findViewById(R.id.ingredientET);
    ingET.setHint("Ingredient " + (ingredientLL.getChildCount() + 1));
    ImageView deleteIng = (ImageView)ingredientRowView.findViewById(R.id.dltingredient_btn);
    deleteIng.setOnClickListener((view) -> {
            deleteIngredientRow(ingredientRowView);
    });
    ingredientLL.addView(ingredientRowView);
```

Figure 3.17 Create and Publish Recipe (1/4)

The recipe template has supported manually add/delete data field feature. The code above shown the add ingredient row function. When user click on add button, a new ingredient row is created under the ingredient section. First of all, the layout inflator inflated the ingredient row xml into a single view, and store in a view variable. After that, the numbering of this new row is done according to the current total ingredient numbers. Besides that, the delete icon is initiated and invoked the delete row function when user click on it. The same concept is applied on add step row function as well.

```
private void deleteIngredientRow(View ingredientRowView) {
    ingredientLL.removeView(ingredientRowView);

    for(int i = 0; i < ingredientLL.getChildCount(); i++){
        View IngredientRow = ingredientLL.getChildAt(i);
        TextView ingNum = (TextView)IngredientRow.findViewById(R.id.ingnumbertv);
        ingNum.setText((i+1)+".");
        EditText ingET = (EditText)IngredientRow.findViewById(R.id.ingredientET);
        ingET.setHint("Ingredient "+(i+1));
    }
```

Figure 3.18 Create and Publish Recipe (2/4)

The ingredient row view is passed into this delete function when being invoked. First of all, layout list removed the view that passed into function. After that, a for loop is used to run through the parent layout to recalculate the numbering of each row. The same concept is applied on delete step row function as well.

```
private void ValidateRecipeInfo() {
    //check if have empty fields
    if(rcpImageUri == null || rcpTitleET.getText().toString().isEmpty() || rcpDescpET.getText().toString().isEmpty() ||
    serveET.getText().toString().isEmpty() || timeET.getText().toString().isEmpty() ){
        Toast.makeText( context: this,  text: "You cannot have empty field before submit the recipe.", Toast.LENGTH_SHORT).show();
    }
    else if ((ingredientLL.getChildCount() == 0) || (stepLL.getChildCount() == 0)){
        Toast.makeText( context: this,  text: "Your recipe has no ingredient or step.", Toast.LENGTH_SHORT).show();

    }
    else if (!wFoodcb.isChecked() && !aFoodcb.isChecked() && !spicycb.isChecked() && !vegancb.isChecked() && !dessertcb.isChecked() && !othercb.isChecked()){

        Toast.makeText( context: this,  text: "Please give your recipe at least a category", Toast.LENGTH_SHORT).show();
    }
    else if (ingredientLL.getChildCount() >= 1 || stepLL.getChildCount() >= 1 ){
        for(int i = 0; i < ingredientLL.getChildCount(); i++){
            View ingredientRow = ingredientLL.getChildAt(i);
            EditText etIngredient = (EditText)ingredientRow.findViewById(R.id.ingredientET);
            if (etIngredient.getText().toString().isEmpty()){
                Toast.makeText( context: this,  text: "The ingredient field is empty", Toast.LENGTH_SHORT).show();
                return;
            }
        }
        for(int j = 0; j < stepLL.getChildCount(); j++){
            View stepRow = stepLL.getChildAt(j);
            EditText etStep = (EditText)stepRow.findViewById(R.id.stepET);
            if (etStep.getText().toString().isEmpty()){
                Toast.makeText( context: this,  text: "The step field is empty", Toast.LENGTH_SHORT).show();
                return;
            }
        }
```

Figure 3.19 Create and Publish Recipe (3/4)

The code above is used to validate the recipe before perform upload operation. The reason of validation is because the system must ensure all recipe has already filled in the necessary data before upload to the database. If else statement is used to perform validation in this function. The user must upload a recipe image and fill in recipe title, description, serve size, cook time and category accordingly. The recipe should also contain at least one ingredient and one step, and the data must not be empty in ingredient row and step row as well. If any violation is found, user cannot publish the recipe and an error toast message is shown to indicate what's wrong during the validation process.

```
HashMap recipeMap = new HashMap();
recipeMap.put("rid",currentUserID + recipeRandomName + Integer.toString(nextInt));
recipeMap.put("uid",currentUserID);
recipeMap.put("date",saveCurrentDate);
recipeMap.put("time",saveCurrentTime);
recipeMap.put("title",rcpTitleET.getText().toString());
recipeMap.put("description",rcpDescpET.getText().toString());
recipeMap.put("serve",serveET.getText().toString());
recipeMap.put("cooktime",timeET.getText().toString());
recipeMap.put("recipeimage",downloadUrl);
recipeMap.put("profileimage",userprofileimg);
recipeMap.put("name",username);
recipeMap.put("counter",Long.toString(countRecipes));
recipeMap.put("ingredientnumber", Integer.toString(ingredientLL.getChildCount()));
recipeMap.put("stepnumber", Integer.toString(stepLL.getChildCount()));

DatabaseReference ingRef = FirebaseDatabase.getInstance().getReference().child("Recipes").child(currentUserID +
        recipeRandomName + Integer.toString(nextInt)).child("ingredients");
HashMap IngHM = new HashMap();
for (int i = 0; i < ingredientLL.getChildCount(); i++){
    View ingredientRow = ingredientLL.getChildAt(i);
    EditText etIngredient = (EditText)ingredientRow.findViewById(R.id.ingredientET);
    IngHM.put("ingredient"+(i+1), etIngredient.getText().toString());
}
ingRef.updateChildren(IngHM);

DatabaseReference stpRef = FirebaseDatabase.getInstance().getReference().child("Recipes").child(currentUserID +
        recipeRandomName + Integer.toString(nextInt)).child("steps");
HashMap StpHM = new HashMap();
for (int i = 0; i < stepLL.getChildCount(); i++){
    View stepRow = stepLL.getChildAt(i);
    EditText etStep = (EditText)stepRow.findViewById(R.id.stepET);
    StpHM.put("step"+(i+1), etStep.getText().toString());
}
```

Figure 3.20 Create and Publish Recipe (4/4)

The code above is used to upload recipe data into database according to recipe id. Firstly, a hashmap is created, after that all recipe data is insert into this hashmap along with the attribute name. The hashmap represent a recipe as it contains all data information of this recipe. Later, the hashmap is uploaded to a database reference that pointed to the recipe id inside the recipe database. Due to the ingredient data and step data is not directly a children data of a recipe, so the ingredient data and step data cannot be insert directly in this hashmap. New hashmap has to be created to handle the ingredient data and step data, once the ingredient and step hashmap is created and filled up correspondingly, these hashmaps are uploaded as a children of the recipe hashmap that created initially. Similar concept is applied in create food post function as well, the difference is the food post doesn't have that much complex data like recipe.

10) Recipe Adapter

```
@Override
public RecipeAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(mContext).inflate(R.layout.recipe_item, parent,  attachToRoot: false);
    return new RecipeAdapter.ViewHolder(view);
}
```

Figure 3.21 Recipe Adapter (1/6)

To display all recipes in a single page, recycler view is used to hold all recipes. The major component that used to fill in recipe data in each row of recycler view is call adapter. Recipe adapter is used to handle every recipe object that are going to display in recycler view. First of all, inflator is called and inflated the recipe item row design file into a view and store into a variable.

```
public class ViewHolder extends RecyclerView.ViewHolder{

    TextView userName, date, time, size, cooktime, title, likesNo, commentNo, ingtv, steptv;
    ImageView profileImage, recipeImage;
    ImageButton favbtn, likebtn;

    public ViewHolder(View itemView){
        super(itemView);

        userName = itemView.findViewById(R.id.recipe_profile_name);
        profileImage = itemView.findViewById(R.id.recipe_profile_image);
        recipeImage = itemView.findViewById(R.id.recipe_image);
        date = itemView.findViewById(R.id.recipe_date);
        time = itemView.findViewById(R.id.recipe_time);
        size = itemView.findViewById(R.id.sizett);
        cooktime = itemView.findViewById(R.id.timett);
        title = itemView.findViewById(R.id.recipe_title);
        likesNo = itemView.findViewById(R.id.noofLikes);
        commentNo = itemView.findViewById(R.id.noOfComments);
        favbtn = itemView.findViewById(R.id.favouritebtn);
        likebtn = itemView.findViewById(R.id.likerecipeImg);
        ingtv = itemView.findViewById(R.id.IngredientTV);
        steptv = itemView.findViewById(R.id.StepTV);
```

Figure 3.22 Recipe Adapter (2/6)

Next, every view component in this inflated view is defined using view id such as profile image field, name field, title field and etc. Now the view holder has been initiated and next step is focus on insert data corresponding into each field.

```
public void onBindViewHolder(@NonNull final RecipeAdapter.ViewHolder holder, int position) {
    final Recipe recipe = mRecipes.get(position);
    holder.userName.setText(recipe.getName());
    Picasso.get().load(recipe.getProfileimage()).into(holder.profileImage);
    holder.date.setText(recipe.getDate());
    holder.time.setText(recipe.getTime());
    Picasso.get().load(recipe.getRecipeimage()).into(holder.recipeImage);
    holder.size.setText(recipe.getServe() + " people ");
    holder.cooktime.setText(recipe.getCooktime() + " minute " );
    holder.title.setText(recipe.getTitle());
    holder.ingtv.setText(recipe.getIngredientnumber() + " Ingredients");
    holder.steptv.setText(recipe.getStepnumber() + " Steps");

    holder.recipeImage.setOnClickListener((view) -> {
            Intent intent = new Intent(mContext, ClickRecipe.class);
            intent.putExtra( name: "RecipeKey",recipe.getRid());
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            mContext.startActivity(intent);
    });

    holder.profileImage.setOnClickListener((view) -> {
            if (recipe.getUid().equals(currentUserID)){
                Intent intent = new Intent(mContext,Profile.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                mContext.startActivity(intent);
            }else{
                Intent intent = new Intent(mContext, UsersProfile.class);
                intent.putExtra( name: "UserKey",recipe.getUid());
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                mContext.startActivity(intent);
```

Figure 3.23 Recipe Adapter (3/6)

Under the onBindViewHolder() function, each recipe inside the recipe array list is took out and stored as a recipe object. The recipe class has already defined all getter and setter method that help recipe to retrieve data from the database. Now, every recipe data field has been filled in the data that get from the recipe database. The on click method can also be set into holder component view as well. When user click on recipe image, the recipe id is passed into a new intent and this recipe id is used to retrieve data from recipe database. So that whenever user click on recipe image, user is directed to a single recipe page that contains full recipe data. Same thing is happened when user click on profile image as well. User id is passed to a new intent and user is directed to the user profile page.

```
holder.favbtn.setOnClickListener((view) → {
        favChecker = true;
        final DatabaseReference RecipesRef = FirebaseDatabase.getInstance().getReference().child("Recipes");
        RecipesRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {

                if(favChecker.equals(true))
                {
                    if(snapshot.child(recipe.getRid()).hasChild(currentUserID)) {
                        RecipesRef.child(recipe.getRid()).child(currentUserID).removeValue();
                        favChecker = false;
                    }
                    else {
                        RecipesRef.child(recipe.getRid()).child(currentUserID).setValue("true");
                        favChecker = false;
                    }
                }
            }
```

Figure 3.24 Recipe Adapter (4/6)

The code above is used to bookmark a recipe and update the database. When user click on bookmark icon, the system checked if the recipe is already bookmarked by the user. If already bookmarked, then the bookmark status is removed. If not, user id is added into this recipe as a child in database to indicate the bookmark operation.

```
//like operation
holder.likebtn.setOnClickListener((view) → {
        likeChecker = true;
        final DatabaseReference LikesRef = FirebaseDatabase.getInstance().getReference().child("RecipeLikes");
        LikesRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
            {
                if(likeChecker.equals(true))
                {
                    if(snapshot.child(recipe.getRid()).hasChild(currentUserID)) {
                        LikesRef.child(recipe.getRid()).child(currentUserID).removeValue();
                        likeChecker = false;
                    }
                    else {
                        LikesRef.child(recipe.getRid()).child(currentUserID).setValue(true);
                        likeChecker = false;
                    }
```

Figure 3.25 Recipe Adapter (5/6)

The code above is executed when like button is being clicked by user. The like checker is used to check whether the user has liked the recipe before and update the database correspondingly. The colour of like button and the total like numbers are changed in real time as well.

```
//display comment number
DatabaseReference CommentRefs = FirebaseDatabase.getInstance().getReference().child("Recipes");
CommentRefs.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if(snapshot.child(recipe.getRid()).child("Comments").exists()){
            commentCounts = (int) snapshot.child(recipe.getRid()).child("Comments").getChildrenCount();
            holder.commentNo.setText(Integer.toString(commentCounts) + " Comments");
        }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {
    }
});

holder.commentNo.setOnClickListener((view) -> {
        Intent intent = new Intent(mContext, CommentRecipe.class);
        intent.putExtra( name: "RecipeKey",recipe.getRid());
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        mContext.startActivity(intent);
});
```

Figure 3.26 Recipe Adapter (6/6)

The code above is executed when comment button of a recipe is being clicked. The recipe id is passed into comment activity and user is directed to comment activity as well. Every comment of that recipe is displayed and shown in a recycler view. User can type in comment message and hit save button to comment a recipe. The comment data is uploaded to database and the total number of comments is increased as well. The adapter concept is also applied on user adapter, food post adapter and message adapter to handle user object, food post object and message object, it is just the function and structure of each adapter are different.

11) Filter Recipe

```
private void FilterRecyclerView() {
    mRecipes = new ArrayList<>();
    mRecipes2 = new ArrayList<>();
    mRecipes3 = new ArrayList<>();
    mRecipes4 = new ArrayList<>();
    DatabaseReference ref = FirebaseDatabase.getInstance().getReference().child("Recipes");
    ref.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            mRecipes.clear();

            for(DataSnapshot dataSnapshot : snapshot.getChildren()){
                Recipe recipe = dataSnapshot.getValue(Recipe.class);
                mRecipes.add(recipe);
            }

            // Category Filter
            for (int i=0; i<mRecipes.size(); i++){
                if (catspin.getSelectedItem().toString().toLowerCase().equals("category")){
                    mRecipes2 = mRecipes;
                }
                else if (snapshot.child(mRecipes.get(i).getRid()).child("category").hasChild(catspin.getSelectedItem().toString().toLowerCase())){
                    mRecipes2.add(mRecipes.get(i));
```

Figure 3.27 Filter Recipe (1/4)

The code above is used to filter the recipe when user applied category filter on recipe page. First of all, a database reference that pointed to recipe database was initiated. After that, a for loop is used to loop through the recipe database and store every recipe into a recipe object array list. After obtained the array list of all recipe, another for loop and if else statement is used to filter the recipes with category that match with category filer applied by user. The filtered recipe objects are stored into a new array list which is mRecipes2 for further filter operations.

```
// Serve Size Filter
if (sizespin.getSelectedItem().toString().equals("Serve Size")){
    mRecipes3 = mRecipes2;
}
else if (sizespin.getSelectedItem().toString().equals("> 4")){
    for (int i=0; i<mRecipes2.size(); i++){
        if (Integer.parseInt(mRecipes2.get(i).getServe()) > 4){
            mRecipes3.add(mRecipes2.get(i));
        }
    }
}
else{
    for (int i=0; i<mRecipes2.size(); i++){
        if (mRecipes2.get(i).getServe().equals(sizespin.getSelectedItem().toString())){
            mRecipes3.add(mRecipes2.get(i));
        }
    }
}
```

Figure 3.28 Filter Recipe (2/4)

The code above is used to filter recipes base on serve size attributes. After obtained the filtered list from category filter, the serve size filter is used to filter the list again to retrieve recipes that have the serve size that match with user selection. Whenever found a match recipe, the recipe is added into a new array list for further filter operations.

```
// Cooking Time Filter
if (timespin.getSelectedItem().toString().equals("Cook Time")){
    mRecipes4 = mRecipes3;
}
else if (timespin.getSelectedItem().toString().equals("1 to 10 minutes")){
    for (int i=0; i<mRecipes3.size(); i++){
        if (Integer.parseInt(mRecipes3.get(i).getCooktime()) > 0 && Integer.parseInt(mRecipes3.get(i).getCooktime()) <= 10){
            mRecipes4.add(mRecipes3.get(i));
        }
    }
}
else if (timespin.getSelectedItem().toString().equals("11 to 20 minutes")){
    for (int i=0; i<mRecipes3.size(); i++){
        if (Integer.parseInt(mRecipes3.get(i).getCooktime()) > 10 && Integer.parseInt(mRecipes3.get(i).getCooktime()) <= 20){
            mRecipes4.add(mRecipes3.get(i));
        }
    }
}
else if (timespin.getSelectedItem().toString().equals("21 to 30 minutes")){
    for (int i=0; i<mRecipes3.size(); i++){
        if (Integer.parseInt(mRecipes3.get(i).getCooktime()) > 20 && Integer.parseInt(mRecipes3.get(i).getCooktime()) <= 30){
            mRecipes4.add(mRecipes3.get(i));
        }
    }
}
else if (timespin.getSelectedItem().toString().equals("31 to 40 minutes")){
    for (int i=0; i<mRecipes3.size(); i++){
        if (Integer.parseInt(mRecipes3.get(i).getCooktime()) > 30 && Integer.parseInt(mRecipes3.get(i).getCooktime()) <= 40){
            mRecipes4.add(mRecipes3.get(i));
        }
    }
}
else if (timespin.getSelectedItem().toString().equals("> 40 minutes")){
    for (int i=0; i<mRecipes3.size(); i++){
```

Figure 3.29 Filter Recipe (3/4)

After the serve size filter, the obtained filtered list is proceeded to the cook time filter. Every cook time filter options are in range base. Therefore, conditional statements are used to filter the recipe objects if the recipe's cook time is within the selected cook time filter. Whenever found a matched recipe, the recipe is stored into a new array list for further sorting operation.

```
// Sorting
if (switchcounterlike.getText().equals("Sort by Latest")){
    for (int i=0; i<mRecipes4.size()-1; i++){
        for (int j=0; j<mRecipes4.size()-1-i; j++){
            if (Integer.parseInt(mRecipes4.get(j).getCounter()) < Integer.parseInt(mRecipes4.get(j+1).getCounter())){
                Recipe temp = mRecipes4.get(j);
                mRecipes4.set(j, mRecipes4.get(j+1));
                mRecipes4.set(j+1, temp);
            }
        }
    }
}
else if (switchcounterlike.getText().equals("Sort by Popularity")){
    for (int i=0; i<mRecipes4.size()-1; i++){
        for (int j=0; j<mRecipes4.size()-1-i; j++){
            if (Integer.parseInt(mRecipes4.get(j).getLikesno()) < Integer.parseInt(mRecipes4.get(j+1).getLikesno())){
                Recipe temp = mRecipes4.get(j);
                mRecipes4.set(j, mRecipes4.get(j+1));
                mRecipes4.set(j+1, temp);
            }
        }
    }
}

recipeAdapter = new RecipeAdapter(getApplicationContext(),mRecipes4);
all_recipes_list.setAdapter(recipeAdapter);
```

Figure 3.30 Filter Recipe (4/4)

Lastly, after obtained the filtered array list. Next procedure is performed sorting according to user selection. There are 2 sort modes provided which are sort by latest and sort by popularity. User can switch the sort mode by click on the switch mode button. Whenever click event is detected, the text of the button is changed, and sorting operation is triggered. The sort by latest is done by using the counter attribute and sort by popularity is done by using the total like numbers of each recipe. After obtained the final filtered array list. The array list is passed into the recipe adapter and finally the adapter is inserted into recycler view and display to user.

3.5 Chapter Summary

This chapter has discussed about the system design and overview. Use case diagram and use case table has been shown to describe the project scope and module with coding explanation is written to show how the work is done.

CHAPTER 4 METHODOLOGY

4.1 Overview

In this chapter, methodology that going to apply in system development and tools needed are described. In addition, a google form is used to collect user feedback and analysis is done regarding the user satisfaction towards the system.

4.2 Methodologies and General Work Procedures

The methodology that selected to develop this project is prototyping approach. With prototyping approach, first a prototype is built according to the initial requirements and after that the continuous improvements are implemented base on the application prototype. Eventually, the final prototype with fine function is delivered and the application is constructed base on it. Prototyping approach is a suitable methodology for mobile application development because it is considered as an agile development approach that can reduce time and cost effectively.

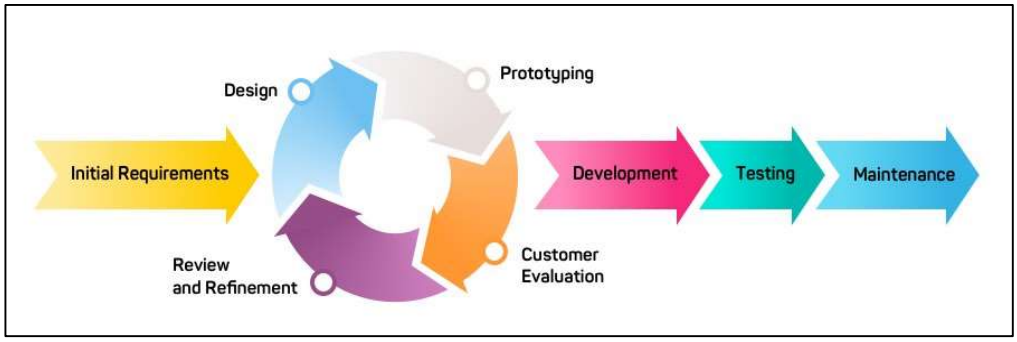


Figure 4.1 Prototype Approach of Mobile Application Development

In the beginning phase, a set of initial user requirements for this project are defined. In this case, the requirements are defined as the basic functions of a social application such as register, login, reset password, create profile, edit profile, logout and etc. Then, the next phase took a long interval by building an application prototype base on the user requirements. Although the prototype is built, it is still not completely functioning at this moment and a lot of improvements are gradually added based on the prototype. Once a prototype is released, it needs to be reviewed and evaluated by user to know what needs to be added and what should to be removed. After that, the changes are implemented, and next prototype is released. The steps are iterated until the prototype represent the final product desired and finally the application is constructed base on the final prototype. To ensure the application works properly, there are several criteria for

the application to achieve. First of all, the user should be able to register and login to the account, the password recovery function is also needed as well when user accidentally forget password. Next, user should be able to perform social action such as search user, view user profile, send message, comment, like and bookmark posted items. Lastly, user should be able to publish food post, publish recipe and search recipes.

4.3 Tools to Use

The hardware required in this project is a laptop and the software needed are Android Studio and Firebase. Android Studio is an official integrated development environment (IDE) for android base mobile application and this project is developed on this platform with JAVA programming language. Besides that, internet connection is also a requirement for this project as it is a social application that can connect users from anywhere. Moreover, the firebase is implemented in this project as well. There is a lot of features supported by firebase such as user authentication, real time database, firebase storage etc. The table for hardware and software specification are shown below:

Hardware Specification

Table 4.1 Components and Requirements of Hardware Specifications

Components	**Requirements**
Windows	Windows 10 64-bit Operating System
Processor	Intel ® Core ™ i7-8750H CPU @ 2.2GHz
RAM	12.0 GB

Software Specification

Table 4.2 Components and Requirements of Software Specifications

Components	**Requirements**
Android Studio	Version 4.0.1 (July 14,2020)
Firebase	Spark Plan (Free)
Firebase Real Time Database	Latest Version
Firebase Storage	Latest Version

4.4 Project Timeline

Task	Start Date	End Date	Duration (Days)
Food Recipe Related Social Application	26-10-2020	04-12-2020	40
Setup Android Project and Firebase (4 days)	26/10/2020-29/10/2020		
Learn how to manipulate android studio	26-10-2020	27-10-2020	2
Open project and connect to Firebase	28-10-2020	28-10-2020	1
Defining Objectives and Project Scope	29-10-2020	29-10-2020	1
Develop Profile Management Feature (9 days)	30/10/2020-07/11/2020		
Splash screen, register, login	30-10-2020	01-11-2020	3
Create profile, edit profile	02-11-2020	05-11-2020	4
Decide and enhance theme and UI design	06-11-2020	07-11-2020	2
Develop Food Post Sharing Feature (14 days)	08/11/2020-21/11/2020		
Develop create post feature	08-11-2020	11-11-2020	4
Study how to manipulate firebase database	12-11-2020	16-11-2020	5
Develop show posts and bookmark feature	17-11-2020	21-11-2020	5
Develop search user feature (6 days)	22/11/2020-27/11/2020		
Display user and search user	22-11-2020	27-11-2020	6
Testing before submit (6 days)	28/12-2020-03/12/2020		
Create Virtual data for testing	28-12-2020	28-12-2020	1
Debug and enhance UI	29-12-2020	01-12-2020	3
Performance testing	02-12-2020	03-12-2020	2
Project 1 Submission	04-12-2020		1

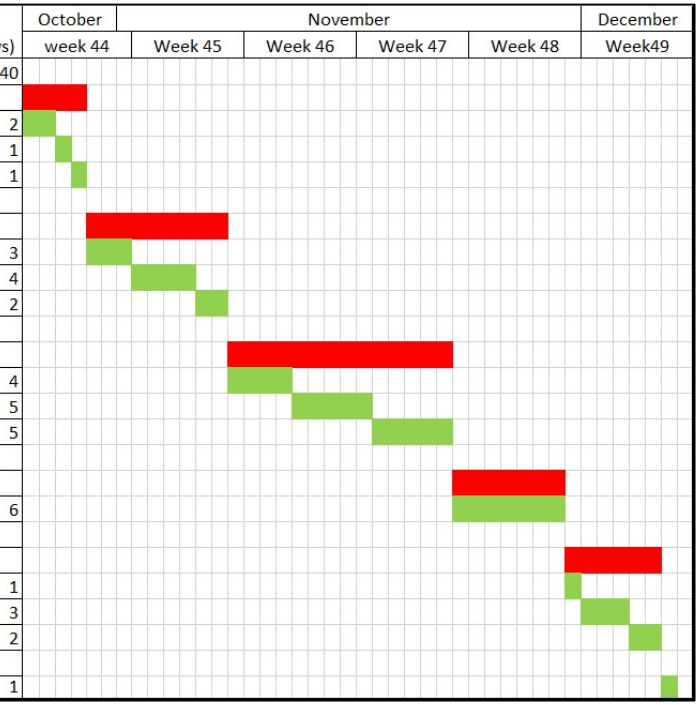


Figure 4.2 Previous Semester Timeline

Task	Start Date	End Date	Duration (Days)
Food Recipe Related Social Application	18-01-2021	20-04-2021	92
Planning Stage (3 days)	18/1/2021-21/1/2021		
Planned a to do list	18-01-2021	18-01-2021	1
Refine the previous work	19-01-2021	19-01-2021	1
Review project objectives	20-01-2021	21-01-2021	2
Learn how to use adapter and firebase query	22/1/2021-30/01/2021		
Watch tutorial online	22-01-2021	25-01-2021	4
Study the firebase adapter	26-01-2021	28-01-2021	3
Apply adapter and query into project	29-01-2021	30-01-2021	2
Design based on objectives (46 days)	31/01/2021-26/03/2021		
Develop features mentioned in objectives	31-01-2021	13-02-2021	46
Implementation Stage (30 days)			
Study the feasibility of features	18-03-2021	24-03-2021	10
Coding development phase	25-03-2021	01-04-2021	20
Testing before submit (3 days)			
Create Virtual data for testing	17-04-2021	17-04-2021	1
Debug and enhance UI	18-04-2021	18-04-2021	1
Performance testing	19-04-2021	19-04-2021	1
Project 2 Submission	20-04-2021		1

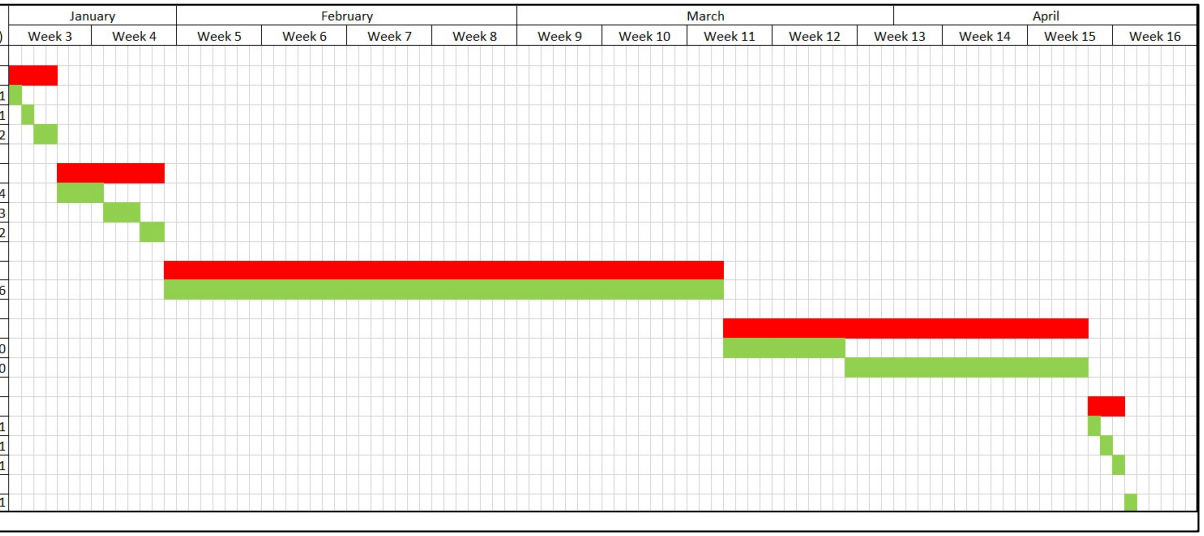


Figure 4.3 Current Semester Timeline

4.5 Chapter Summary

In this chapter, the system development methodology and tools needed for project development have been shown. At the end of this chapter, the project timeline is shown as well.

CHAPTER 5 IMPLEMENTATION AND TESTING

5.1 Overview

This chapter is mainly to test the system and justified features that have been implemented successfully. The screenshots are displayed and explanation about the screenshot are described accordingly to show the application working flow.

5.2 Project Screenshot and Explanation

Splash Screen and Login Page

Figure 5.1 Splash Screen

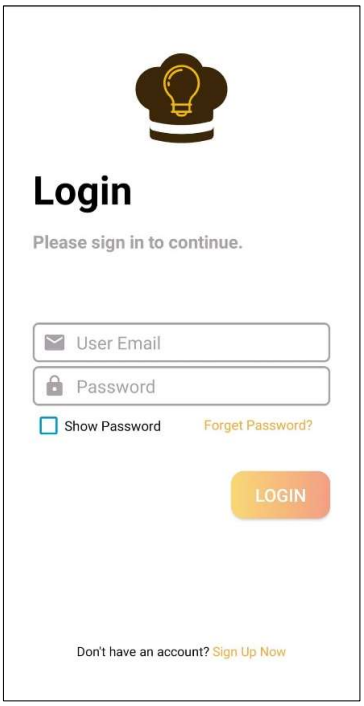


Figure 5.2 Login Page

After user click on the application icon, the first thing appeared is a three second duration animated splash screen. The splash screen has contained the application logo and name. After that, user is launched to the login page to access this application. For first time user, account registration page can be accessed by the “Sign Up Now” button at the bottom right corner.

Register Page and Password Reset Page

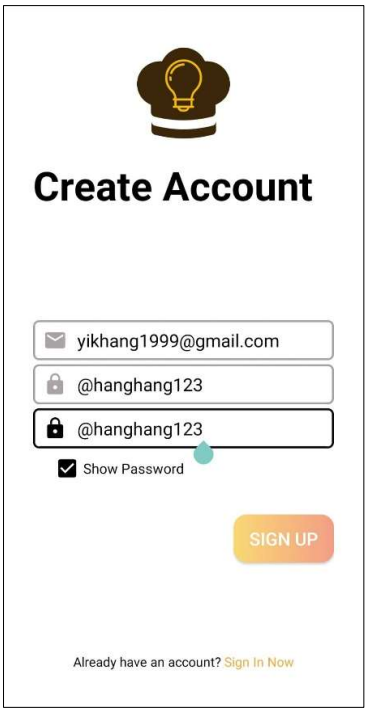


Figure 5.3 Register Page

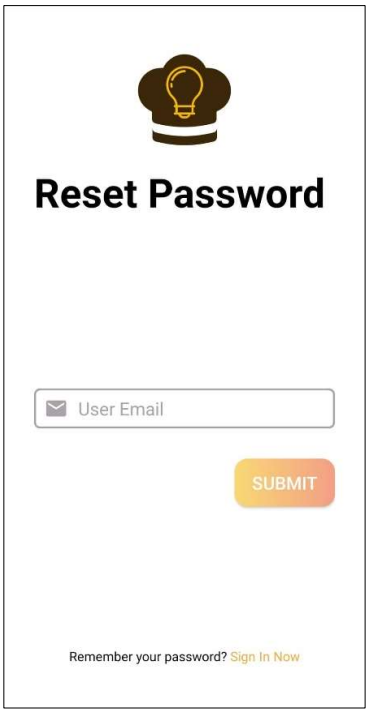


Figure 5.4 Password Reset Page

After user launched the register page, there are three fields need to be filled by the user to complete the account registration. The three fields are email, password and confirmation password. The email is used for login purpose and receive password reset link when user accidentally forget the password. The password and confirmation password must be same and there is a checkbox that can be used to ensure password before registration. At registration page, user can go back to login page by the button at bottom right corner.

If user accidentally forget the password, the password can be reset by submit an email that used to receive password reset link. This email must be the email that used to register account before. If the email is not valid, an error message is shown.

Profile Setup for First Time User

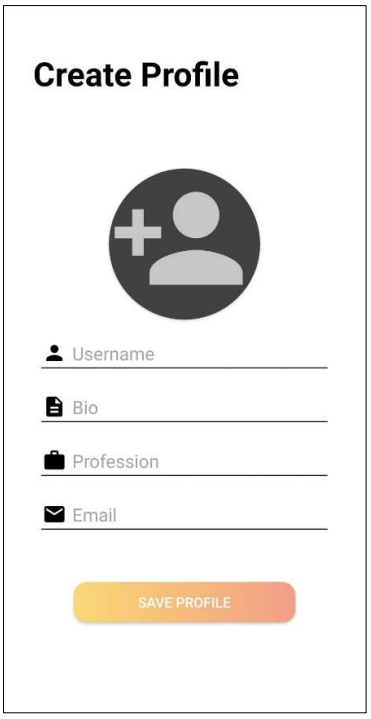


Figure 5.5 Profile Setup Page

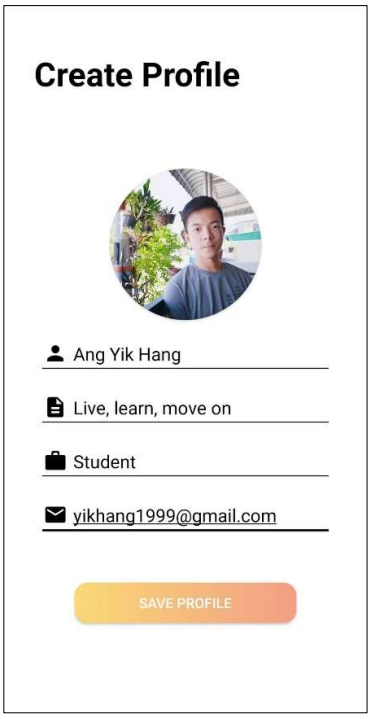


Figure 5.6 Setup Profile

After user register the account successfully, the application search for the user profile data stored in firebase database. For first time user, the profile data won't exist in firebase database. After detected this is a new user, user is directed to the profile setup page. User need to fill up five fields to complete the profile setup. When user click on add profile image button, the phone gallery is opened, and user can select profile picture from there. After user click on save profile button, the user data is stored in firebase database under a file name which named according to user id.

Profile Page and Edit Profile Page

Figure 5.7 Profile Page

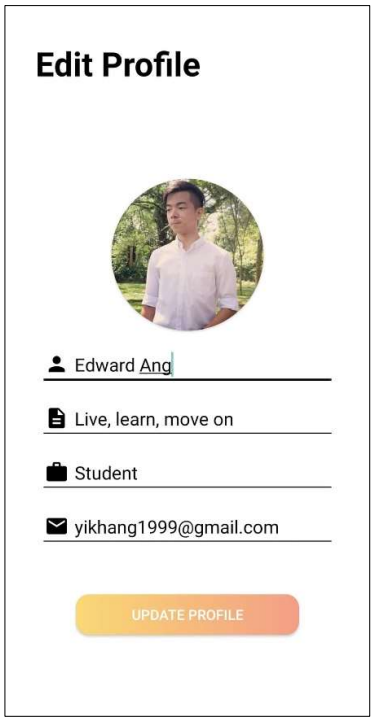


Figure 5.8 Edit Profile Page

After user created the profile, the profile page is shown to user. The button on top right corner can be used to sign out from the application. The user profile details are shown here as well, and the send message button is used to direct user to the chat page. At the bottom of the profile detail section, there are four tabs which are recipes, favourite recipes, posts and favourite posts. These tabs can help user to access posted or bookmarked item easily and quickly. For new user, all tabs are contained zero item at this moment. The blue floating action beside the profile details title is used to access to the edit profile activity, after user edit the profile and click update profile button, the changes are saved on database. The bottom of the screen is the menu bar which contained Recipe, Chat, Post and Profile tab.

All Recipes Page and Recipe Template

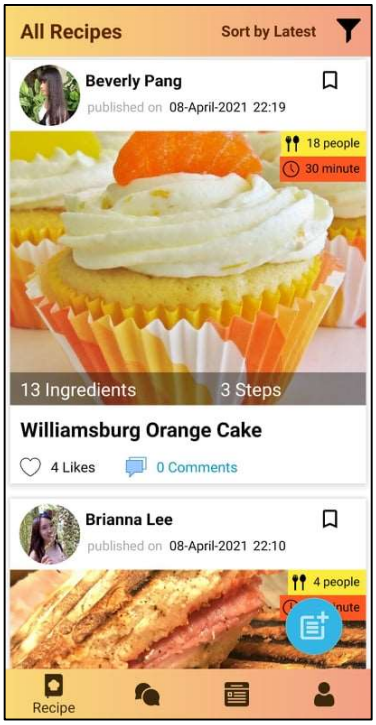


Figure 5.9 All Recipes Page

Figure 5.10 Recipe Template

User can navigate to the main recipe page through the recipe tab at the bottom menu bar. Every recipe published by other users are displayed here according to the recipe timestamp. For each recipe, basic information such as publisher image, publisher name, publish time, recipe image, cook time, serve size, number of ingredients and number of steps are displayed. Besides that, the like button, comment button and bookmark button allow user to like, comment and share the recipe if interested. If user want to publish recipe, user can click on the blue floating action button at the bottom right corner and user is directed to the recipe template page. The recipe template has already arranged the data fields needed and hints is provided to indicate what should be written in this field. Initially, there is no ingredients row and steps row in the recipe template. User has to add the rows accordingly base on the requirements.

Create and Publish Recipe

Figure 5.11 Create Recipe (1/2)

Figure 5.12 Create Recipe (2/2)

The first field in recipe template required user upload a recipe image. User can click on the image button and choose one image from device gallery. The second field is to give this recipe a title. The third field is the recipe description. After that user have to fill in the service size and cook time as well. Under category section, user can check on the category checkboxes to categorize the recipe. The ingredient rows and step rows are manually added by user according to the requirement. For each added row, user can perform delete operation as well by click on the trash button beside each row. After user filled in all recipe data field and click publish button on the top right corner, the progress bar is shown to indicate the upload progress and user is directed to the recipe page. Then a toast message is pop out to indicate the recipe has published successfully. If the recipe validation is failed, then progress bar is not going to displayed, and error toast message is pop out to indicate which data field has violated the validation. For example, user must attach an image and cannot leave title, description, serve size, cook time, ingredients and steps empty when creating recipe.

Posted Recipe and Click Recipe to View Full Content

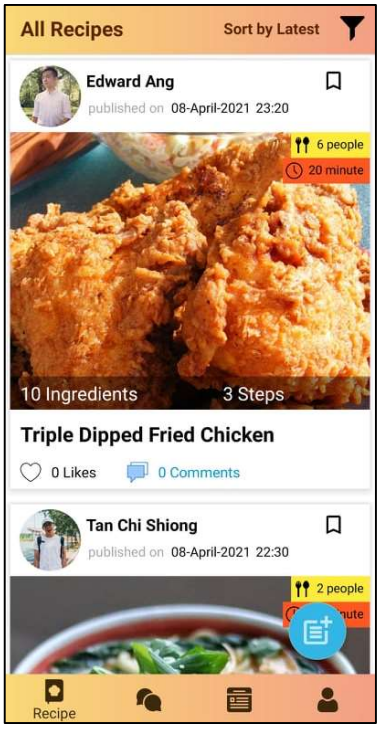


Figure 5.13 Posted Recipe

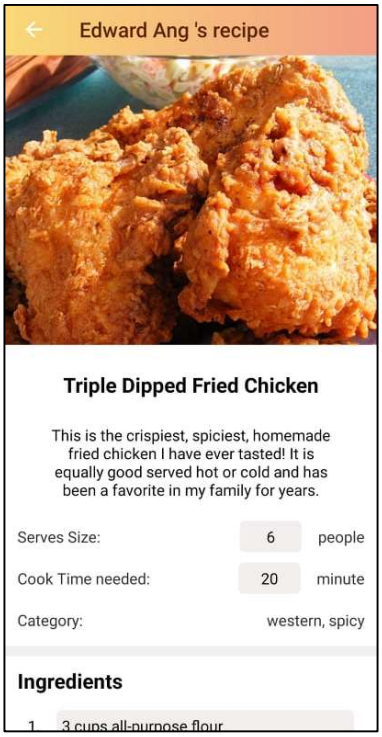


Figure 5.14 Click Recipe

After the recipe is published successfully, user is directed to the main recipe page and the posted recipe is displayed on top. Initially the like number and comment number are zero. When user click on the recipe image, user is directed to a single page that contained the complete information of this recipe. The top bar indicated the recipe author's name and the following content arrangement are same as the recipe template.

Edit and Delete Recipe

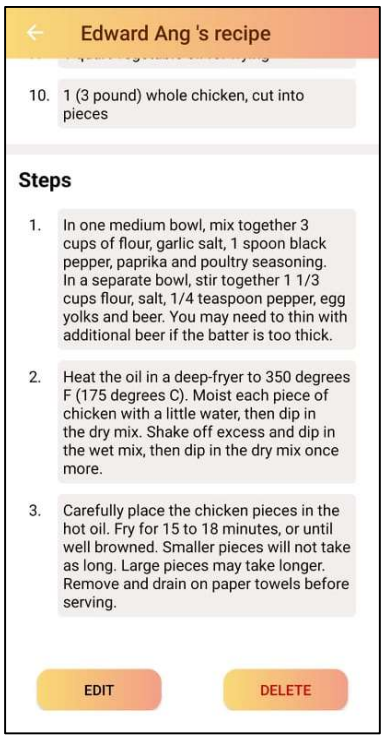


Figure 5.15 Edit and Delete Button

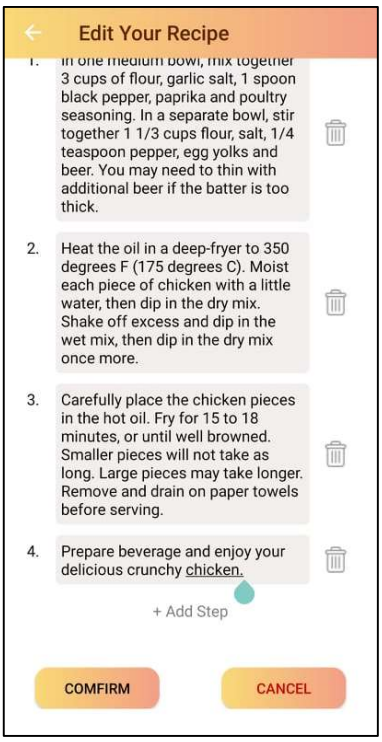


Figure 5.16 Edit Recipe

If the author wants to delete the recipe, user can scroll to the bottom of the recipe page and 2 buttons are displayed at the bottom which are edit button and delete button. The edit and delete button are only shown if the user is the author of this recipe. This can avoid user from deleting other user's recipe and protect other user's data. When user click on delete button, an alert dialog box is pop out for delete confirmation to ensure there is a second security layer when deleting the recipe. When user click on the edit button, user is directed to the edit recipe page. User can modify recipe content, add or remove ingredients and steps, or change the category of this recipe in this place. After changes are made, user can click on confirm button at the bottom of the edit recipe page to save the updated content, user can also click on cancel button to discard the edit operation.

Apply Recipe Filters

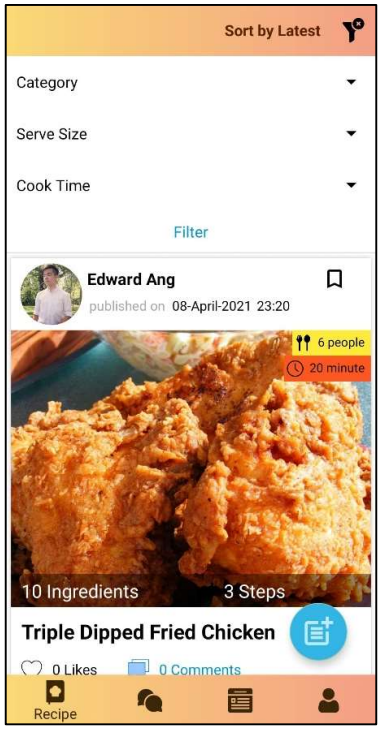


Figure 5.17 Recipe Filter Options

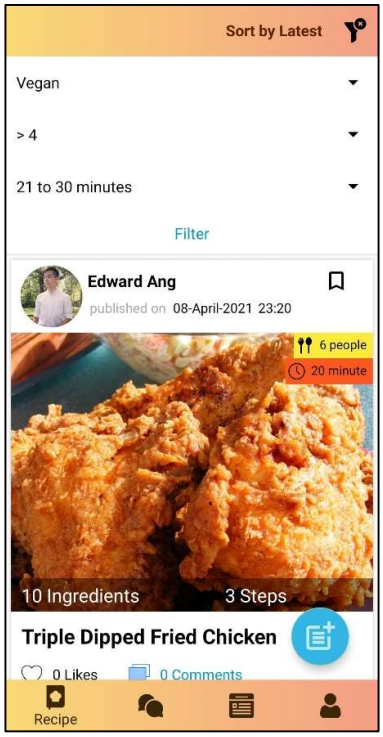


Figure 5.18 Apply Filters

When user click on the filter icon at the top right corner in main recipe page. The filter section is expanded and there are 3 filters that user can choose to filter the recipes. Each of the filter is a spinner, user can click and select the category, serve size and expected cook time base on requirement and click on filter button to trigger the filter operation.

Sort by Latest and Popularity

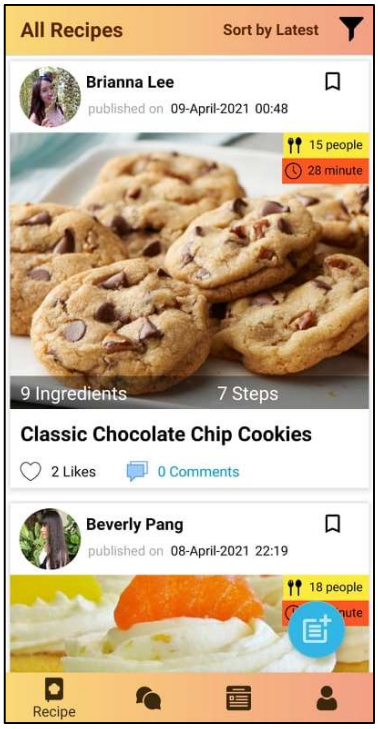


Figure 5.19 Sort by Latest

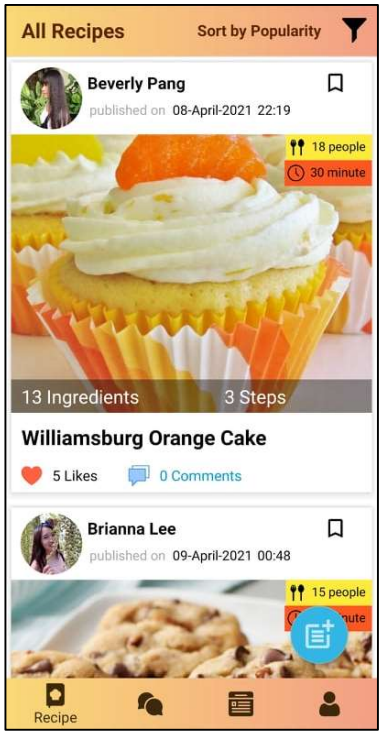


Figure 5.20 Sort by Popularity

After filter operation, the recipes are updated and displayed instantly in the main recipe page and the expanded filter section is collapsed. By default, the displayed recipes were sorted by latest. If user want to know which filtered recipe has the most likes, user can click on the sort by latest button beside the filter icon and the button text is changed to sort by popularity. The displayed recipes changed the display sequence according to the popularity, which is the most likes on top.

Search Users by User Name

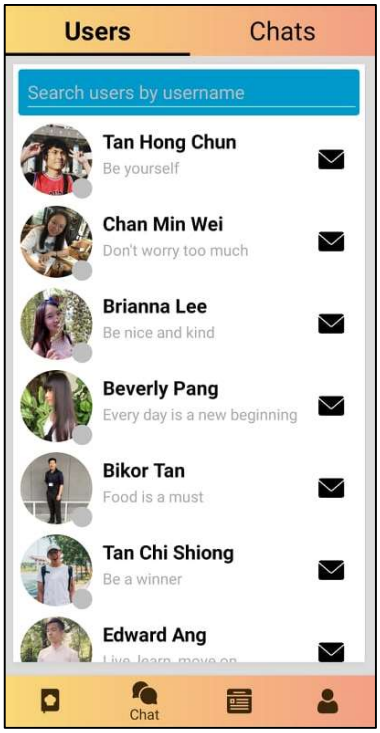


Figure 5.21 All Users

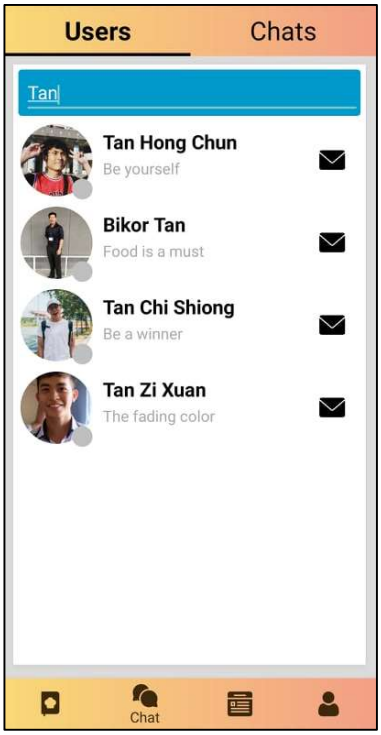


Figure 5.22 Search Users

When user want to chat and search other users, user can click on chat tab at the bottom navigation bar. User is directed to the page where all users are listed here. Each user row is used to display some basic profile information such as profile image, name, and bio. Moreover, the tiny circle icon located at the bottom right corner of each profile image can be used to indicate whether the user is in online or offline status. At the top search box, user can type in username that used to filter users. The filtered results are displayed in real time whenever the text in search box is changed. Whenever user want to send message to other user, user can directly click on the black message icon beside each row and be directed to the chat interface with that user.

Chat Interface and Chat History

Figure 5.23 Chat Interface

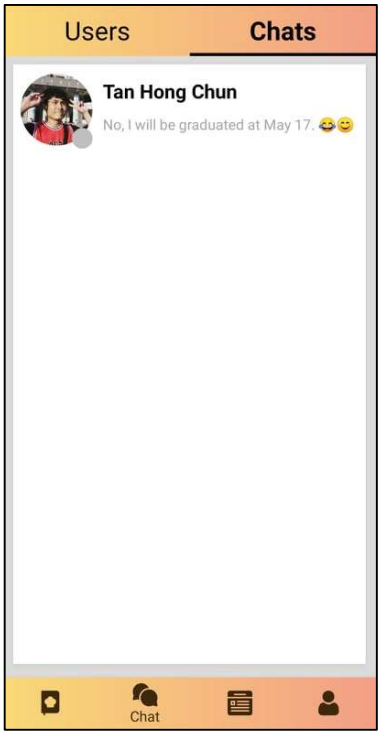


Figure 5.24 Chat History

When user click on the message icon, user is directed to a one on one chat interface. The user profile image and name are displayed on top of the page to indicate the message receiver. User can type in text message and click send button to send the message. After the message is sent, a “delivered” text is displayed at the bottom of the message. If the receiver has seen the message, then the “delivered” text is changed to “seen” text. If user want to view the recent chat, user can click on the chat tab at the top right corner to access the chat history page. The recent chat components are displayed along with the last message in the conversation.

Other User's Profile and Posted Content

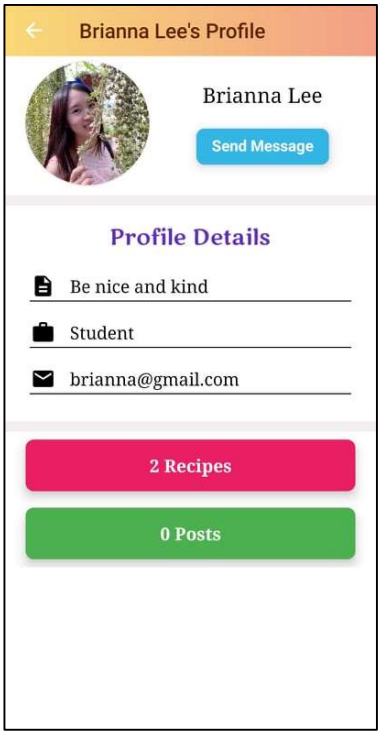


Figure 5.25 User's Profile

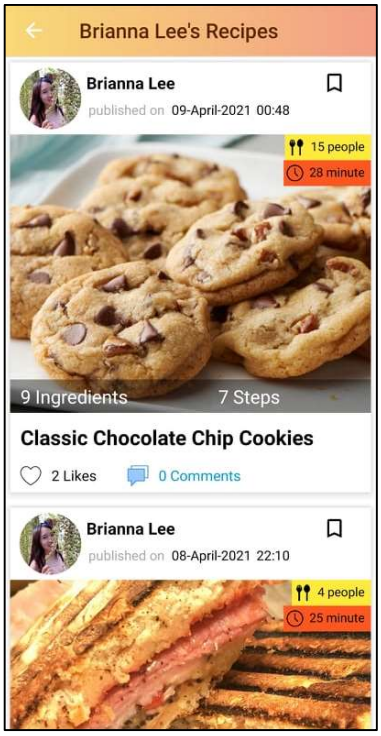


Figure 5.26 Posted Recipe

When click on profile image, user is accessed to other's account profile page. The basic profile information is displayed accordingly such as profile image, name, bio, profession and email. The send message button at the top is used to bring user to the chat interface. At the bottom of this page, user is able to know how many recipes and posts has been published by this account. Unlike personal profile page, the favorite items of other account cannot be seen in order to protect data privacy. When user click on the recipes tab or posts tab, user is directed to the page that contain the posted items by this account.

Main Post Page and Create Post Feature

Figure 5.27 All Posts

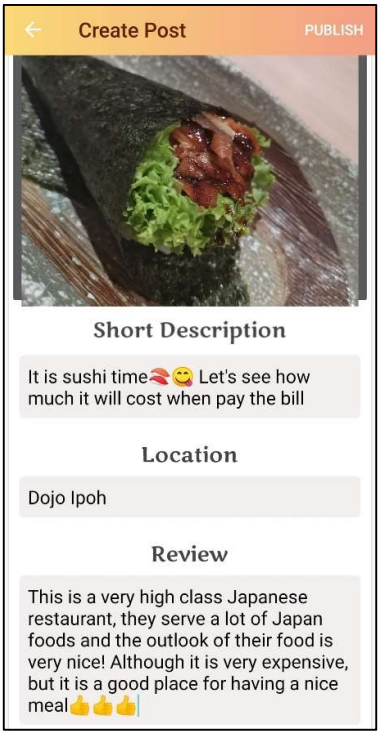


Figure 5.28 Create Post

User can navigate to main post page by clicking post tab at the bottom navigation bar. All posts published by other users are displayed according to the timestamp. Each post has supported bookmark, like and comment features as well. User can write own post by clicking on the blue floating action button at the bottom right corner. User is redirected to the create post page, user can attach photo and write post context inside this page. User can fill in content such as description, location of this place or leave review for this post as well. After that user can click on publish button to upload the post to the database. Then, user is directed to main post page.

Edit Post and Comment Post

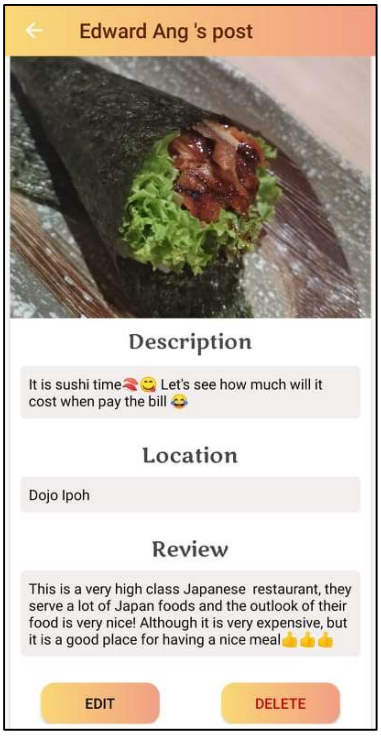


Figure 5.29 Edit Post

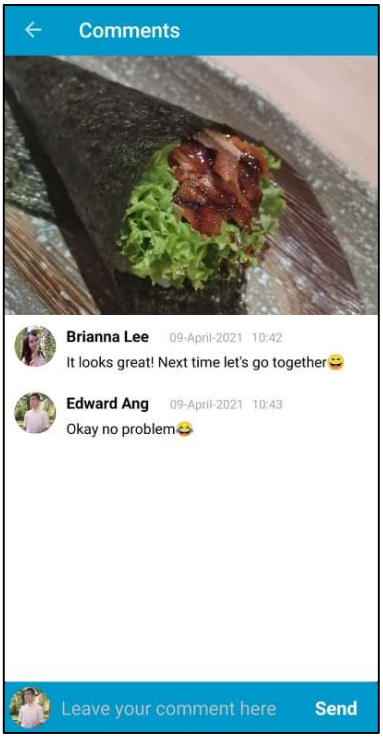


Figure 5.30 Comment Post

User can click on post image to perform the edit and delete operation. The edit and delete button are visible if the user is the author of this post. When click on edit button, user is directed to edit page. User can modify post content and save the changes. When user click on delete button, an alert dialog box is pop out for delete confirmation. This is a second security layer to avoid user delete data unconsciously. When user click on the comment button of each post, user is directed to the comment page. The post photo and previous comments are displayed on this page. User can comment the post and interact with other user regarding the post.

5.3 User Feedback Analysis

After the application is completed, the generated apk file of this application is distributed to 10 friends for testing purpose. These people have a common characteristic which is interested in cooking. First of all, the apk file is sent by Whatsapp and tester can directly download and install the application into an android device. After that, testers tested the application for 5 days and explored the functionalities inside the app. A google feedback form is created to collect user feedback regarding the application features. In this part, collected feedbacks are analysed to figure out the user satisfaction and what improvements are needed in the future.

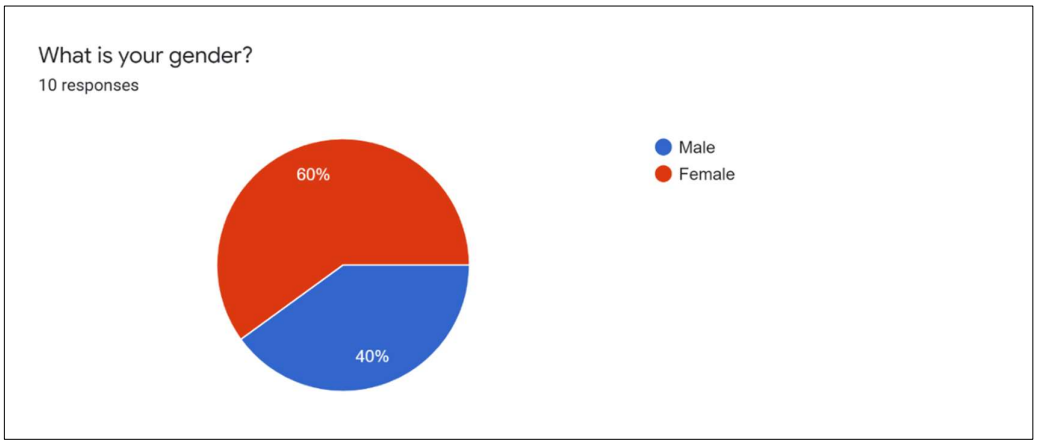


Figure 5.31 Respondent Gender

The first question of the feedback form is to ask respondent's gender. 10 respondents have participated in this survey after tested the application. According to the figure 5.31, there are 60% of female respondents and 40% of male respondent. Different opinions regarding the application are collected as usage habit may differ from different gender.

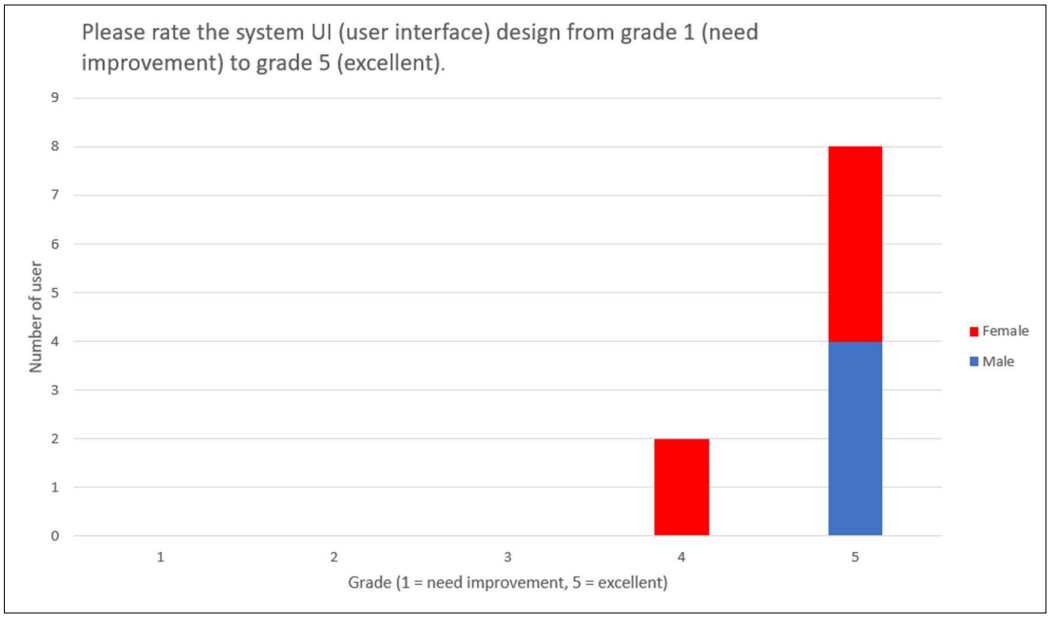


Figure 5.32 System UI Design Satisfaction

The second question of the feedback form is to ask respondents to rate the system user interface design from grade 1 (need improvement) to grade 5 (excellent). The reason for asking this question is because a good user interface design of a mobile application is very essential. One of the challenges when developing a mobile application is to design the user interface reasonably on the limited device screen. In addition, the design of a mobile application should also be attractive so that more and more users are interested in this application hence creating natural traffic. According to the figure 5.32, 20% of respondents have rated grade 4 and 80% of respondents have rated the highest grade which is grade 5. All male respondents have the highest satisfaction level with the design. In short, all respondents are very satisfied with the system user interface design.

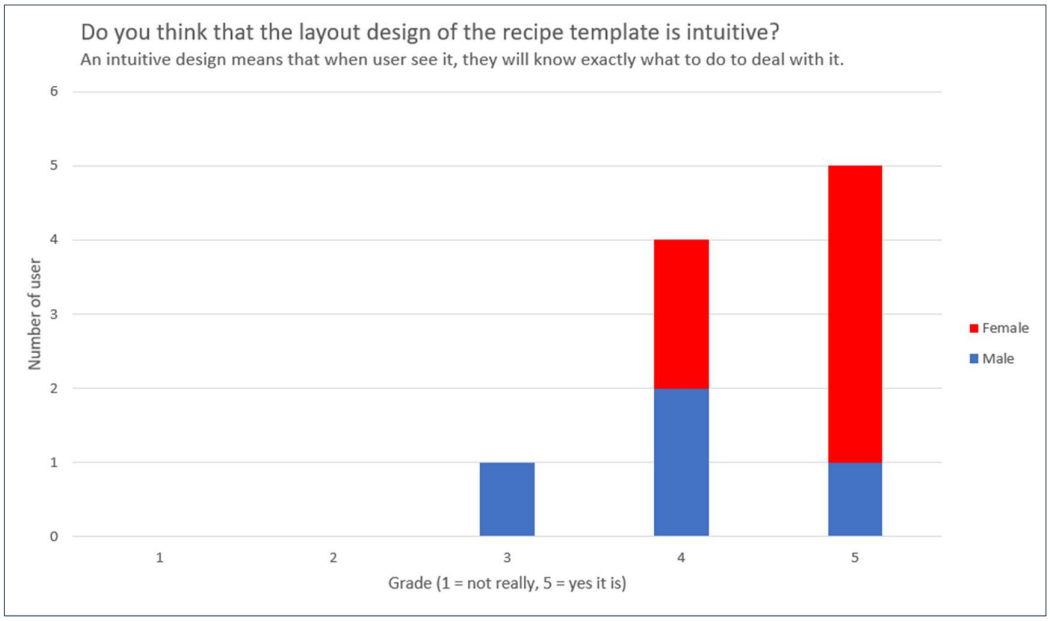


Figure 5.33 Recipe Template Satisfaction

The third question of the feedback form is to ask respondents to rate the recipe template from grade 1 to grade 5 if respondents think that the layout design of recipe template is intuitive. An intuitive means that when user see it, user is going to know exactly what to do to deal with it. The reason for asking this question is because it is necessary to investigate whether users have understanding on how to make use of recipe template to create own recipe. According to the figure 5.33, 10% of respondents rated grade 3, 40% of respondents have rated grade 4 and 50% of respondents have rated a grade 5. There is one male respondent rated a grade 3 and all female respondents' rate are mostly grade 4 and grade 5. The result get from this question shows that female respondents and most of the male respondents are know how to make use of recipe template but there also has male respondent think that the recipe template only has an average performance in layout design and it is not that intuitive. In short, most of the respondents are satisfied with the recipe template but the space of improvement is still existing.

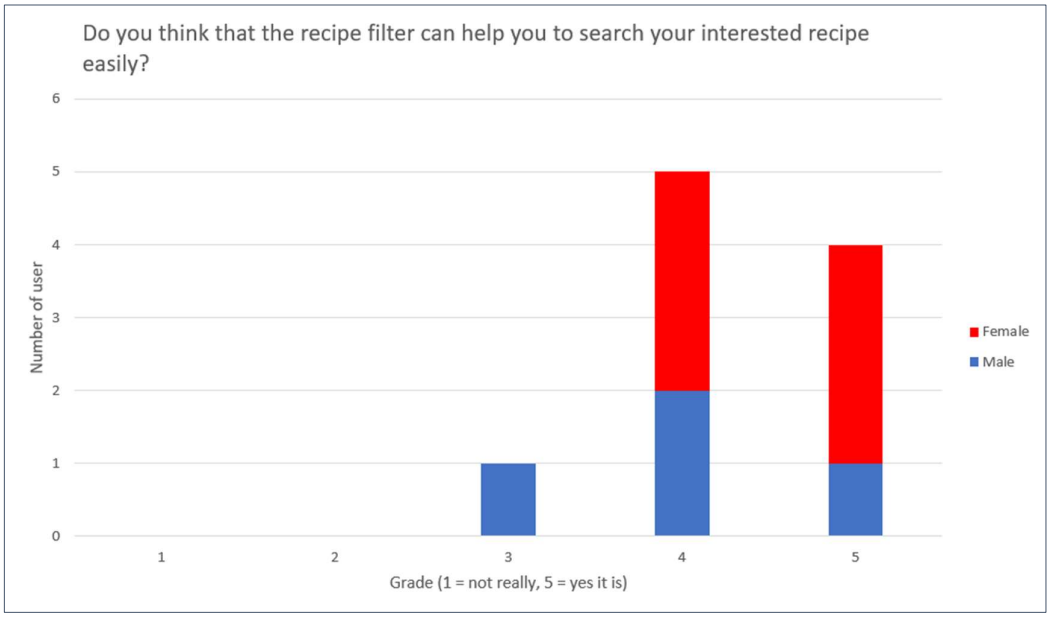


Figure 5.34 Recipe Filter Satisfaction

The fourth question of the feedback form is to ask respondents to rate the recipe filter from grade 1 to grade 5 if respondents think that the recipe filter is useful and can help user search recipes easily. The reason for asking this question is because recipe filter is one of the major features that improve recipe searching thus it is necessary to investigate whether users are satisfied with the recipe filter. According to the figure 5.34, 10% of respondents rated grade 3, 50% of respondents have rated grade 4 and 40% of respondents have rated a grade 5. There is one male respondent rated a grade 3 and other male respondents and all female respondents are rated grade 4 and 5.

The result observed from this question shows that 90% of the respondents think that recipe filter is helpful when searching relevant recipe. There is also a male respondent think that the recipe filter only has a normal performance. In short, most of the respondents has a higher satisfaction level regarding the recipe filter as it is useful for recipe searching.

Figure 5.35 User Experience Assessment

The fifth question of the feedback form is to ask respondents to rate the system from grade 1 to grade 5 if respondents think that the system can help to improve cooking skill through the recipes and cooking experiences shared by others. The purpose for assessing the user experience is because the ultimate goal of building this project is to help food lover to share and gain food related knowledge and information easily. According to the figure 5.35, 50% of respondents have rated grade 4 and 50% of respondents have rated a grade 5. For each grade, there are 2 male respondents and 3 female respondents, which is quite average.

Overall, the system has achieved a high satisfaction rate as all male and female respondents think that this system is able to provide help in improving user cooking skills. The experience shared by others is a useful and valuable information so that user can prevent from making unnecessary mistake when cooking.

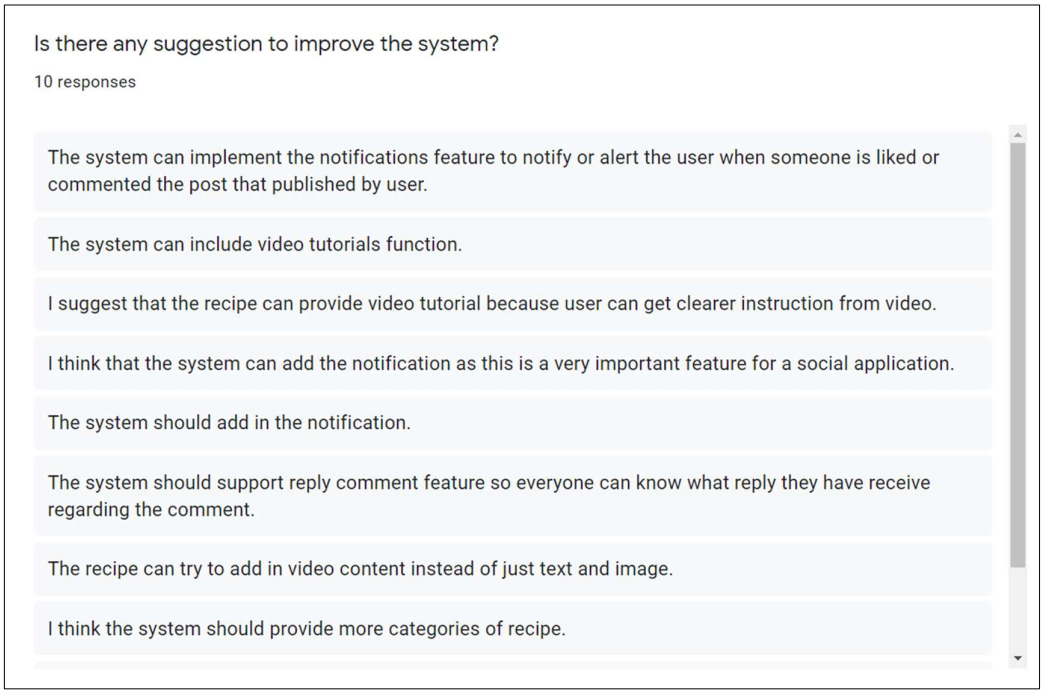


Figure 5.36 Suggestion for Improvement

The last question of the feedback form is to ask respondents to suggest if there is any improvement can be done to enhance the system performance. The purpose for asking this question is because it is necessary to know from user's point of view, what feature is lacking in system and what improvement should be done accordingly. This is an open-ended question and the answer is in short text form. According to the figure 5.36, the answers obtained can be divided into 4 improvements. First of all, respondent suggested that the system can support video tutorial in recipe because video content can help user in getting clearer instruction. Besides that, respondent also suggested to implement notification feature so that user is notified when others have liked or comment the content published by this user. In addition, there is a respondent that suggest that the system should support reply comment feature, so every user can know what reply has been received regarding the comment. Lastly, there is respondent think that the system should support more categories for recipe, so user can have more specified categories when search for relevant recipe.

5.4 Chapter Summary

In this chapter, the development result is shown via the application screenshot and explanation. In addition, a user feedback analysis that regarding the system satisfaction has been done.

CHAPTER 6 CONCLUSION

6.1 Project Review, Discussion and Conclusion

In ordinary social networks, the information published by users are usually not uniform. Different users can publish different contents to reflect daily status or interests. It is always better for food lover community to have a food social network that provided social functionality while allowed the sharing of food-related knowledge and information. Besides that, a social network without specific theme can make user have difficulties to meet friends that have same interest. Therefore, it is very necessary to build a food social network application with specific features for food lover community.

At the end of the project, a food social application known as "Tastiee" has been delivered successfully. It is a social application that design for food lovers to share food-related information and knowledge such as food recipes and food posts. The main motivation to develop Tastiee is because ordinary social network without theme does not support features that can improve recipe sharing experience such as recipe template and recipe filter. According to the problem statements, the recipe template has been successfully implemented to help user create recipe effortlessly. The recipe filter is successfully implemented as well. User can apply different filters to search for the relevant recipe more easily. User can search recipe base on category, serve size and cook time. As a food social network, Tastiee are also implemented features such as authentication, profile management, search users, chat system, bookmark, like and comment to enhance the social feasibility of this platform.

6.2 Novelties and Contributions

Sometimes, user may just want to look for a recipe that serves only one people and have a short cook time. With the recipe filter feature in this project, it can be easily achieved by just one fingertip. The flexibility of recipe filter in this project is very high therefore it can satisfy user's requirements under different scenarios. Like feature and comment feature can also enhanced recipe sharing experience because through the likes number, users are able to know which recipe has the higher popularity. Comment feature can enable user leave comments under a recipe and make discussion with another user who also interested in this recipe. Through the discussion, recipe can be learned easily as user can refer to the experiences share by others. In addition, instead of manually arrange the recipe content, users can reduce a lot of effort when creating a food recipe

because all user needs to do is just fill up the prebuild recipe template according to the format given. Whenever user views a food recipe, the content of recipe is arranged nicely as in the recipe template therefore user can read it more properly. The food social network also supported the bookmark feature, user can bookmark any interested food recipes and retrieve these recipes in user profile quickly and easily.

Whenever had a nice meal or visit a new restaurant, people like to give comment about the food and reflect daily meal status and interest. If the food and service is great, users can recommend the restaurant by create a food post on food social network. For restaurant with bad services or poor sanitary condition, the food post can also help to remind others about the situation. Information like this are very useful for others because writer can expose the important information such as how is the service quality, how does the food taste like, is the price worth it, and etc.

Nowadays, ordinary social networks did not provide a channel that share the uniform information, which there are many kinds of information shows up in user's newsfeed and somehow it may cause the user unable to retrieve the information needed. For food lovers, it will be better to have a food social network that is designed only based on food related content. "Tastiee" in this project is a food social network that aim to established connection between food lovers to perform food-related social activities. It is a platform that gathered food lovers who like to share food related information and knowledge.

6.3 Future Work

The Tastiee food social network still have improvements that could be done. After analysis of the collected user feedback, there are some improvements requested from users. One of the improvements is adding video content inside a recipe. Video instruction is a better option when it comes to knowledge sharing. If recipe video is supported, user can gain more clearer instruction by following the video when cook the dish. Besides that, viewer can play, pause and resume the video whenever needed, the learning barrier will be no longer exist.

Second improvement that can be done is adding notification feature to notify user whenever received a new message. A user could also be notified when someone comment, like or bookmark the content published by this user. It can enhance user experience and make this application more completed to a social network. In addition,

the recipe sharing feature can also be enhanced by adding more food categories because food can be sorted into more and more detailed categories. With more specified categories, users are able to search and retrieve targeted recipe easily.

BIBLIOGRAPHY

Ahmad, I. (2018). *The History of Social Media [Infographic]*. [online] Social Media Today. Available at: https://www.socialmediatoday.com/news/the-history-of-social-media-infographic-1/522285/.

Blystone, D. (n.d.). *The Story of Instagram: The Rise of the # 1 Photo-Sharing Application*. [online] Investopedia. Available at https://www.investopedia.com/articles/investing/102615/story-instagram-rise-1-photo0sharing-app.asp#:~:text=Instagram%20is%20a%20photo%20and.

Guru99.com. (2019). *Prototyping Model in Software Engineering: Methodology, Process, Approach*. [online] Available at: https://www.guru99.com/software-engineering-prototyping-model.html.

Hootsuite Social Media Management. (2018). *The History of Social Media: 29+ Key Moments*. [online] Available at: https://blog.hootsuite.com/history-social-media/.

Interestingengineering.com. (2018). *A Chronological History of Social Media*. [online] Available at: https://interestingengineering.com/a-chronological-history-of-social-media.

mdp.berkeley.edu. (n.d.). *Social Media and Development – Master of Development Practice*. [online] Available at: https://mdp.berkeley.edu/social-media-and-development/ [Accessed 7 Sep. 2020].

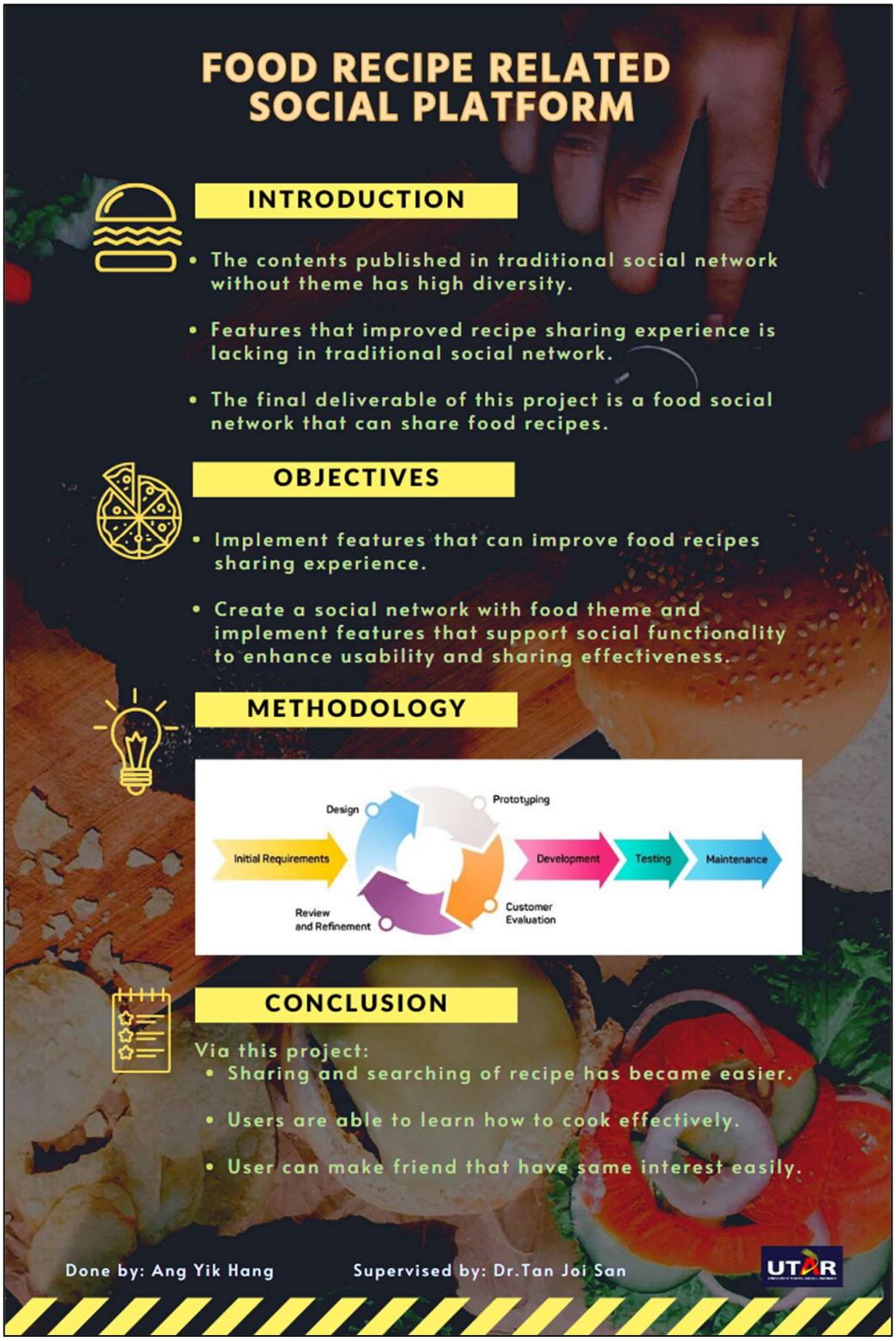
Media 42, D.H.I.S. (2013). *The Complete History of Social Media: Then And Now*. [online] Small Business Trends. Available at: https://smallbiztrends.com/2013/05/the-complete-history-of-social-media-infographic.html#:~:text=Internet%20relay%20chats%2C%20or%20IRCs [Accessed 7 Sep. 2020].

En.wikipedia.org. 2021. *Recipe - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Recipe> [Accessed 12 April 2021].

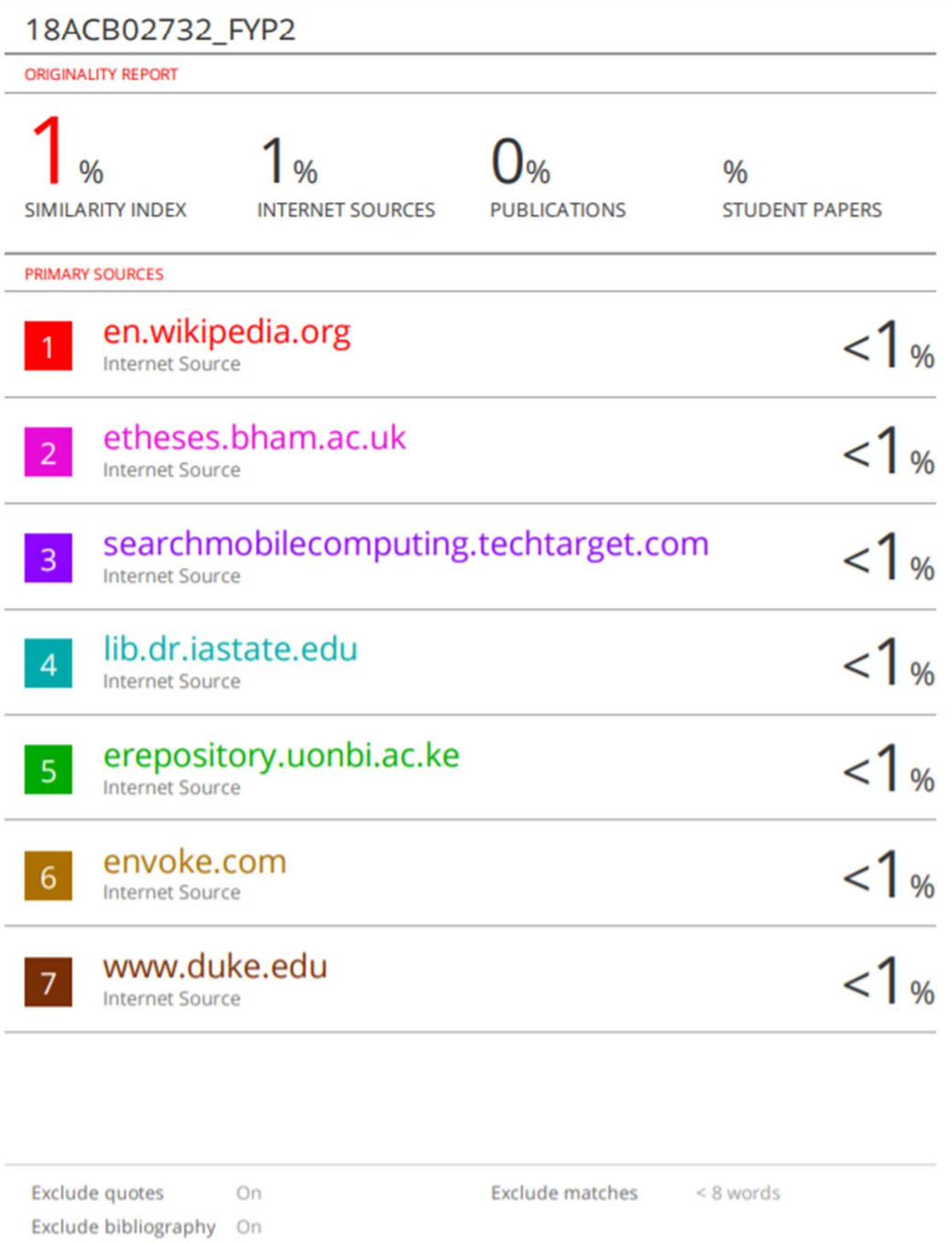
Tryqa.com. (2019). *What is Prototype model- advantages, disadvantages and when to use it?* [online] Available at: http://tryqa.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/.

En.wikipedia.org. 2021. *Yummly - Wikipedia*. [online] Available at: https://www.instagram.com/cookat_hk/?hl=en[Accessed 9 Sep. 2020].

POSTER


FOOD RECIPE RELATED SOCIAL PLATFORM
INTRODUCTION
• The contents published in traditional social network without theme has high diversity.
• Features that improved recipe sharing experience is lacking in traditional social network.
• The final deliverable of this project is a food social network that can share food recipes.
OBJECTIVES
• Implement features that can improve food recipes sharing experience.
• Create a social network with food theme and implement features that support social functionality to enhance usability and sharing effectiveness.
METHODOLOGY
Initial Requirements
Design
Prototyping
Development
Testing
Maintenance
Review and Refinement
Customer Evaluation
CONCLUSION
Via this project:
• Sharing and searching of recipe has became easier.
• Users are able to learn how to cook effectively.
• User can make friend that have same interest easily.
Done by: Ang Yik Hang
Supervised by: Dr.Tan Joi San
UTAR

Appendix A Turnitin Result



18ACB02732_FYP2

ORIGINALITY REPORT

1%	1%	0%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	en.wikipedia.org Internet Source	<1%
2	etheses.bham.ac.uk Internet Source	<1%
3	searchmobilecomputing.techtarget.com Internet Source	<1%
4	lib.dr.iastate.edu Internet Source	<1%
5	erepository.uonbi.ac.ke Internet Source	<1%
6	envoke.com Internet Source	<1%
7	www.duke.edu Internet Source	<1%

Exclude quotes	On	Exclude matches	< 8 words
Exclude bibliography	On		

Universiti Tunku Abdul Rahman			
Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	ANG YIK HANG
ID Number(s)	18ACB02732
Programme / Course	COMPUTER SCIENCE (CS)
Title of Final Year Project	FOOD RECIPE RELATED SOCIAL PLATFORM

Similarity	**Supervisor's Comment (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 1 % **Similarity by source** Internet Sources: 1 % Publications: 0 % Student Papers: - %	No comment
Number of individual sources listed of more than 3% similarity: 0	No comment
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, an** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Name: Tan Joi San

Date: 15th April 2021

Signature of Co-Supervisor

Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	18ACB02732
Student Name	ANG YIK HANG
Supervisor Name	DR. TAN JOI SAN

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Front Cover
✓	Signed Report Status Declaration Form
✓	Title Page
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
N/A	List of Symbols (if applicable)
N/A	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
N/A	Appendices (if applicable)
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. (Signature of Student) Date: 15th April 2021	Supervisor verification. Report with incorrect format can get 5 marks (1 grade) reduction. (Signature of Supervisor) Date: 15th April 2021