Transfer Learning on Inception ResNet V2 for Expiry Reminder: A Mobile Application Development

Ong Wi Yi

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Hons.) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

January 2020

ABSTRACT

The expiry date is a very common information that every product has. It represents the recommended period of time to use the product or the duration of time to use the product. However, it is hard to keep track of the expiry date when there are a lot of them. The project aims to solve the problem by using deep learning, Optical character Recognition (OCR) and also a smart and iterative user interface to help users record and track expiry dates fast and efficient. A Convolutional Neural Network (CNN), Inception ResNet V2 is trained with a newly created near reality synthetic expiry dates image dataset to recognize and capture the expiry date on products. The Inception ResNet V2 has achieved an accuracy of 0.9964 on synthetic data and an accuracy of 0.9612 on noisy reality data. The success in training and deploying the Inception ResNet v2 into the mobile application can significantly speed up the process of recording down new expiry dates and helps users track the expiry date efficiently. Moreover, various kind of alerts including push notifications, SMS notifications and email notifications are provided to reach and remind user more effectively than a normal application.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ ABBREVIATIONS

ANN	Artificial Neural Network
API	Application Programming Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRNN	Convolutional Recurrent Neural Network
DFD	Data Flow Diagram
DNN	Deep Neural Network
ERD	Entity Relationship Diagram
FCM	Firebase Cloud Messaging
GAN	Generative Adversarial Networks
GPU	Graphical Processing Unit
MB	megabytes
OCR	Optical Character Recognition
PWA	Progressive Web App
PXP	Personal Extreme Prototyping
RAM	Random Access Memory
ReLU	Rectifier Linear Unit
RNN	Recurrent Neural Network
SMS	Short Message Service
SDK	Software Development Kit
SDLC	Software Development Life Cycle
SSD	Solid State Drive
SSL	Secure Sockets Layer
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
WBS	Work Breakdown Structure

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Most of the things that consumers see in stalls have an expiry date, especially food. When something is expired, it means that the thing should no longer be used. There are so many things that have expiry dates, such as food, drinks, medicine, cosmetics products and many more. In technology aspects, many things will expire too. For example, a one-year license of software, a one-week free trial of a website, and even documents.

Sometimes, it is hard for consumers to keep track of everything when managing items that have expiry dates. It is bothering if the product expires before using it up. It is very wasteful when consumers need to discard something after they realize it is expired. Sometimes, people can't even remember the product that stores inside a cabinet for a very long time, especially for busy people.

People nowadays are living a hectic life. They are too busy to care about the expiry date of every product in their home. Although they are busy, there is one thing that humans nowadays will not forget to use daily, which is smartphones. As the mobile smartphone gets more advanced, people mostly stick to their smartphone the whole day. The usage of smartphones is getting more and more common and frequent for everyone. Without any doubt, smartphones have become important for human beings as in completing different tasks, entertaining themselves and many more. Phones are a communication tool for humans nowadays, but it is also a symbol of convenience to most of us.

The solution to reduce the problems and reduce the waste caused by the expired product is by developing a mobile application that reminds the user by SMS, notification or Email when a product is about to expire. It allows users to add a product by key in the name, type and the expiry date of the product by themselves or use OCR to scan the product details to fill it in automatically. Users can set the type of reminders such as SMS, E-mail or notification and the reminder period for 1 hour, 1 day or 1 week as per requirement.

This chapter will provide details about the project background, problem statement, objectives, scope, justification as well as the proposed solution.

1.2 Background

Most of the groceries and medicine in our household have their expiry dates. When something is expired, it is not recommended to be used or consumed anymore. The only solution for expired food and groceries is to throw them away. From the statistics shown by *Malaysian Food Waste* (2019), food waste took up 44.5% of Malaysia's waste composition then followed up with plastics waste which is 13.2%. Most of the food waste comes from household food waste, which is from the expired food and groceries that are thrown away by every household.

Besides, each of the cosmetics has its expiration date and shelf life. It is not recommended to use an expired cosmetic product. Repeated use of expired makeup can cause infection. But it is hard to keep track of a cosmetics product because the expiry date of the product depends on the opening date. For example, if according to the statistics shown by Jean Coutu (no date), mascara and eyeliner will expire after 3 months from the opening date, creams will expire after 6 to 9 months from the opening date and lipsticks can last for 2 years from the opening date. It is hard to keep track of each cosmetic's expiry dates when a user owns too many of them.

To keep track of the expiry date of every product's expiry date, a mobile application that can categorize products and reminds the user before the product is expired is indeed needed. An efficient and effective tool that helps users manage the expiry date of each product can effectively solve the problem of expired food and cosmetics that might cause waste and health threat.

This project aimed to develop a new Expiry Reminder Mobile Application, which has refined functionality and is more user friendly compared to the existing applications.

1.3 Problem Statement

The problem stated below has existed for a long time. These problems are revealed from direct observation to the targeted group and questionnaires survey. The problems are listed as follows:

1. Hard to trace the expiry dates of cosmetic products

   Cosmetic product expiry dates are normally determined by the date that the product is opened. There is only a manufacturing date on the product and the shelf life of the product. Most cosmetic products are considered expired after the shelf life or after some time after the product is opened. The period for the product being opened before expiration is normally written in an "opened container" symbol. It is normally 6M, 12M, 18M and 24M while "M" represents months. For example, 6M will be 6 months from the opened date before the product expired. With no exact expiry date on the product, it is hard to trace when the product is going to expire.

2. People tend to forget the product when they do not use it often

   Some of the product won't be used very often. According to Figure 1.1, most of the users tend to forget the product they do not use so often. The product that is forgotten will become a waste when it is expired. It is not only a waste of money

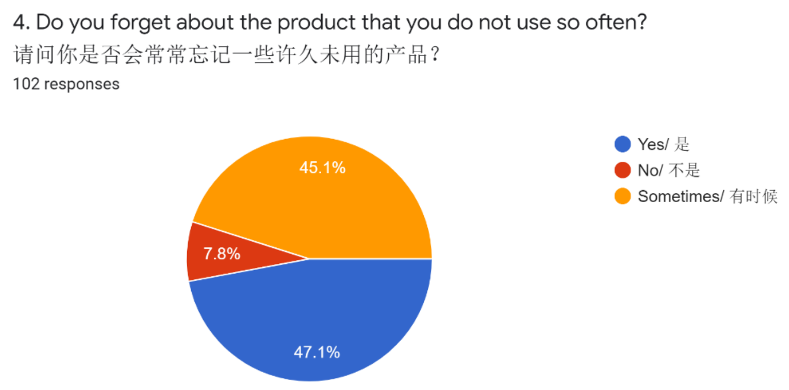


Figure 1.1: The percentage of users that tends to forget about the product they do not use so often

3. Inconvenient to record down the product manually

   It is very inconvenient for us to record down every product’s expiry date in a notebook or manually in a text document inside the mobile phone. This makes the user need to do redundant work and is very troublesome. Users need to key in the same thing multiple times when there are multiple same products of different expiry dates. This makes it inconvenient and not efficient. By recording down the expiry date manually, users are unable to store the product systematically and it makes it hard to read the expiry date of each product. Besides, no reminder will remind them when they record it down manually.

4. Fail to trace back the expiry date when users take out the product from the original container/package.

   Sometimes people will take out the product from the original package and store it somewhere else. It makes the user hard to trace back the product because the expiry date is normally written in the product package.

5. Always realized the product is expired when wanting to use the product

   There is no reminder to remind the user when the product is almost expired. According to Figure 1.2, most of the respondents encounter the problem which is realized when the product is expired when wanting to use it.

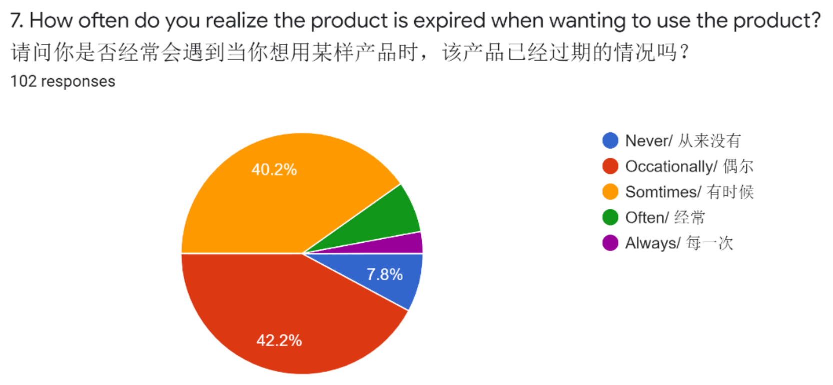

7. How often do you realize the product is expired when wanting to use the product?
请问你是否经常会遇到当你想用某样产品时，该产品已经过期的情况吗？
102 responses

Figure 1.2: The percentage of users that often realize the product is expired when wanting to use the product

1.4 Project Objectives

The project's main objective is to develop an expiry reminder app, which can assist users in managing every product's expiry dates more efficiently and effectively. The sub-objectives are as follows:

- To generate a dataset for recognizing expiry date using the Text Recognition Data Generator package.
- To train Inception Resnet V2 to recognize expiry dates.
- To develop a mobile application for the Android platform to detect expiry date using the trained deep learning model.
- To conduct usability testing for the mobile application.

1.5 Project Solution

The project solution is to create a mobile application that can efficiently overcome the problem stated above with the functionality stated below:

- Use the Deep Learning model to capture the expiry date on the packaging
  The mobile application should allow users to capture expiry dates on the product through their phone camera and produce an accurate result. Users can also choose to manually input the expiry date according to their needs.

- Reminds user through SMS, Push Notification, Email
  The mobile application should allow users to choose the preferred reminder method, which includes reminder through SMS, reminder through push notification and reminder through email so that the user will not miss any of the reminders. Users are also allowed to choose the period of the reminder. For example, 1 week before the expiry date, 1 month before the expiry date and many more.

- Allow user to use OCR to scan product details
  To ease users when inserting new products, the application should allow the user to use OCR to scan product details so that the user does not have to self-key any product details. This makes the add product more accessible and faster instead of typing word by word.

- Allow user to take photos of the product

The application should allow the user to take a photo of the product so that a user will not get confused when there are multiple similar products available. For example, a household has multiple brands of cooking oil. By taking a photo, the user can easily recognize it by the cooking oil packaging instead of reading the description of the product.

- Allow user to scan the barcode of the product for search and calculate stock left
  The application should also allow the user to manage the stock of the product. The application should allow users to scan the product barcode and search the product for users. From that, the user can choose to add the new stock with new expiry dates or delete the stock when the product is finished.

- Able to categorize product
  The application should allow users to categorize products, for example, groceries, medicine, documents, condiments, software license, passport, id, cosmetics, etc. This can make the application interface less confusing to users.

- Shows the total amount of expired, soon to expired and safe to use products
  The application is meant to allow users to manage the expiry date efficiently. So, the total amount of expired, soon to expire and safe to use products should be shown in the application so that the user can know how many products have expired and how many products are needed to be finished soon before expiration.

1.6 Project Approach

There are a variety of methodologies that can be used to develop a mobile application, for example, waterfall, agile, prototype and many more. Most of them are closely linked with the software development life cycle (SDLC), which includes the planning stage, analysis stage, design stage, construction stage, testing stage, implementation stage and support stage.

After going through some studies and comparing several methodologies, agile methodology is chosen. This is because mobile application development is different if compared to a desktop application. Mobile applications need a constant update and are limited by a small screen if compared to a desktop. Agile methodology is to be fast and able to make quick changes stated by HUB (2017). The use of the agile methodology in developing software makes it versatile enough for fast and easy adaption to the changing trends and technologies. Besides, the agile methodology also provides clients or users with a continuous feedback loop.

There are many frameworks in agile methodology. The framework that this project is going to use is Personal Extreme Prototyping (PXP) since the application is developed in a one-man team. According to Asri *et al.* (2018), PXP is an agile development methodology that combines Extreme Prototyping (XP) and Personal Software Process (PSP). It took the advantages from XP, a lightweight and problem-centered methodology, and the advantages from PSP, where a disciplined personal framework is used during the software development process. Unlike XP, PXP is for a single programmer instead of pair programmers. Agarwa and Umphress (2008) also state that PXP is a software development process for a single person team where XP values such as simplicity, communication, feedback and courage are kept and refined to be fit in the single programmer situation.

1.7 Scope of the Project

In this project, an expiry reminder application will be created on the mobile platform to solve the problems proposed.

1.7.1 User Scope

This project is targeted to housewives, working individuals who live alone, students who stayed in the hostel and anyone else who needs to manage expiry dates of any product.

1.7.2 System/Project Scope

System scope contains the features covered, the hardware used, and the software/API used.

1.7.2.1 Features Covered

This project contains features listed below:

1. Manage Products in the Application

   The application must allow users to add, edit and delete products in the application and add, edit and delete product categories in the application. For each product, the application must allow users to enter the name, category, expiry date and reminder period and reminder type. Users can choose to add a new category or select from the created category. The application should provide users with three types of reminder for users to choose from: SMS Reminder, Push Notification Reminder and E-mail Reminder. Apart from that, the application must allow users to take photos for every product and scan barcodes for every product. The application must allow users to use OCR to scan product details and fill in product details and description automatically. The application must allow users to enter the number of stocks and allow users to change it afterwards.

   In addition, the application must allow users to change and edit any details of the product after the product is added to the application. The application must also allow users to delete any products or category that the users want to delete.

2. Reminds User About the Product Before Expiry Date

The application must remind the user every product according to the reminder period and reminder type set by users. The application must be able to remind users through SMS Reminder, Push Notification Reminder and E-mail Reminder according to what users had set. The reminder must be according to what user set which includes “on that day”, “1 day before expire”, “3 days before expire”, “1 week before expire”, “2 weeks before expire”, “1 month before expire”, “2 months before expire”, “3 months before expired” and “6 months before expire” according to the expiry date of the product. The application should allow the user to set as many reminders as possible according to the user’s needs.

3. OCR and Barcode Scanning

   The application must allow users to use OCR (Optical Character Recognition) to identify products’ name and the description of the products when adding the product. The application should identify the name of the products and their description automatically and allow the user to edit them afterwards.

   The application must allow users to search for a product and perform stock-taking by scanning barcodes. If there is no barcode found in the application, the application must display an error message.

4. Expiry Date Recognizing

   The application must allow users to manually input the expiry date or use their camera to capture the expiry date for the deep learning model to recognize the expiry date for the user.

5. Stock Taking and Product Calculating

   The application must allow users to perform stock-taking by changing the number of stocks of every product. The application must also display the total number of expired products, soon to expire product and safe to use product on the main view and update it every day automatically.

6. Automated handling of expired or out-of-stock products

Expired products should be displayed as expired while out-of-stock products should be removed from the product page. Besides, there should be an automated process to add expired products and out-of-stock products into a shopping list. Users can also add items manually to the shopping list according to their needs.

1.7.2.2 Hardware Used

The hardware used will be the phone camera. The phone camera is needed for taking photos of each product, perform OCR scanning and barcode scanning.

1.7.2.3 Software Used

The software and API used in developing the application are listed below:

1. SMS Gateway API

The application needs the SMS Gateway API to send SMS Reminder to users.

2. Gmail API

The application needs the Gmail API to send E-mail Reminder to users.

3. Notification API

The application needs Notification API to send Push Notifications Reminder to users.

4. Text Recognition API

The application needs the Text Recognition API to recognize text when using OCR.

5. Barcode API

The application needs Barcode API to identify the barcodes on the product when scanning barcodes.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, several similar existing expiry reminder applications are studied and their features and user interfaces are evaluated and discussed. There are three applications from the Android system and two applications from the iOS system that have a higher rating and special features are chosen to perform the studies and evaluation.

Besides, different system development methodologies are studied and compared to find the best that is suitable for this project. Next, this chapter also includes the literature review of the technology involved to develop the software.

2.2 Similar Mobile Application Study and Evaluation

Five mobile applications are being studied for this section. Three of the applications are from the Android platform while two of the applications are from the iOS platform. All of the applications are downloaded from Google Play Store for the Android platform and Apple App Store for the iOS platform and are tested on a real mobile phone.

2.2.1 Expiry Wiz by Terrific Mobile

Adopted from
<https://play.google.com/store/apps/details?id=com.terrificmobile.expirywiz&hl=en>

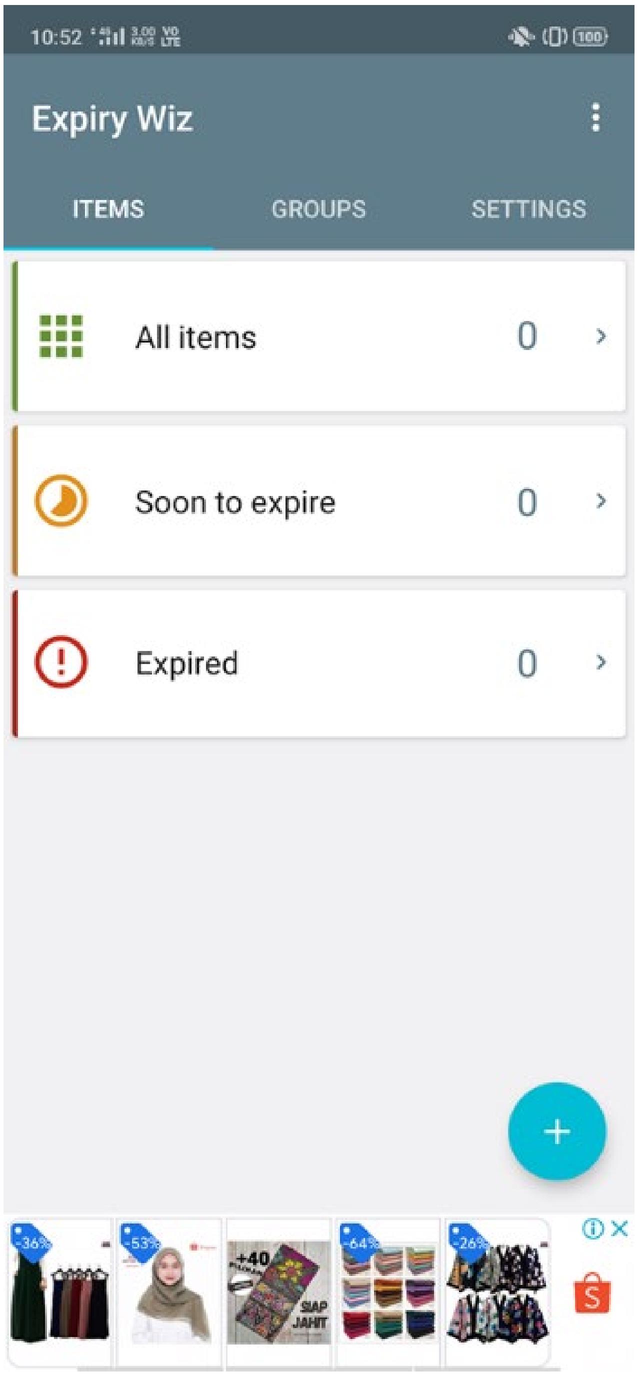
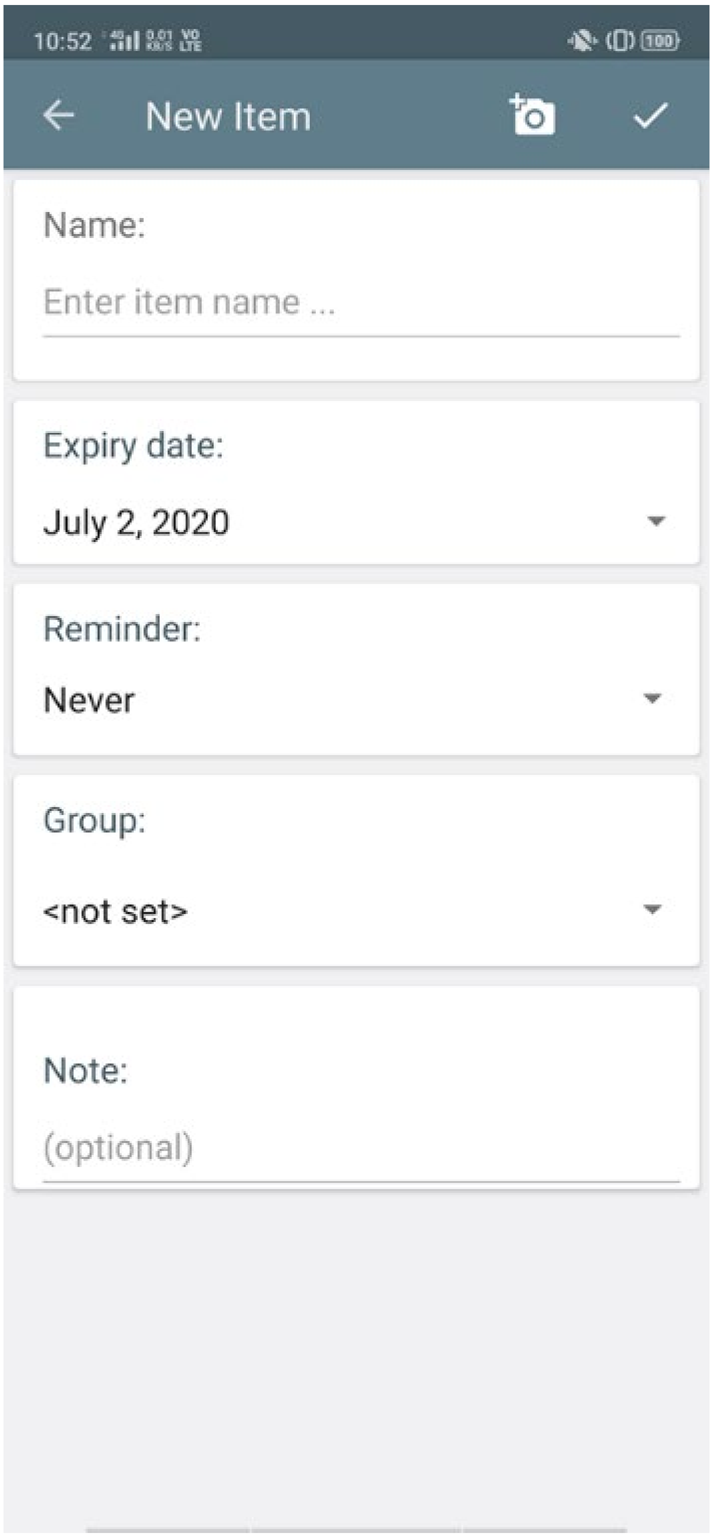


Figure 2.1: Main View of the Expiry Wiz Application


10:52

New Item

Name:

Enter item name ...

Expiry date:

July 2, 2020

Reminder:

Never

Group:

<not set>

Note:

(optional)

Figure 2.2: Add Product View of The Expiry Wiz Application

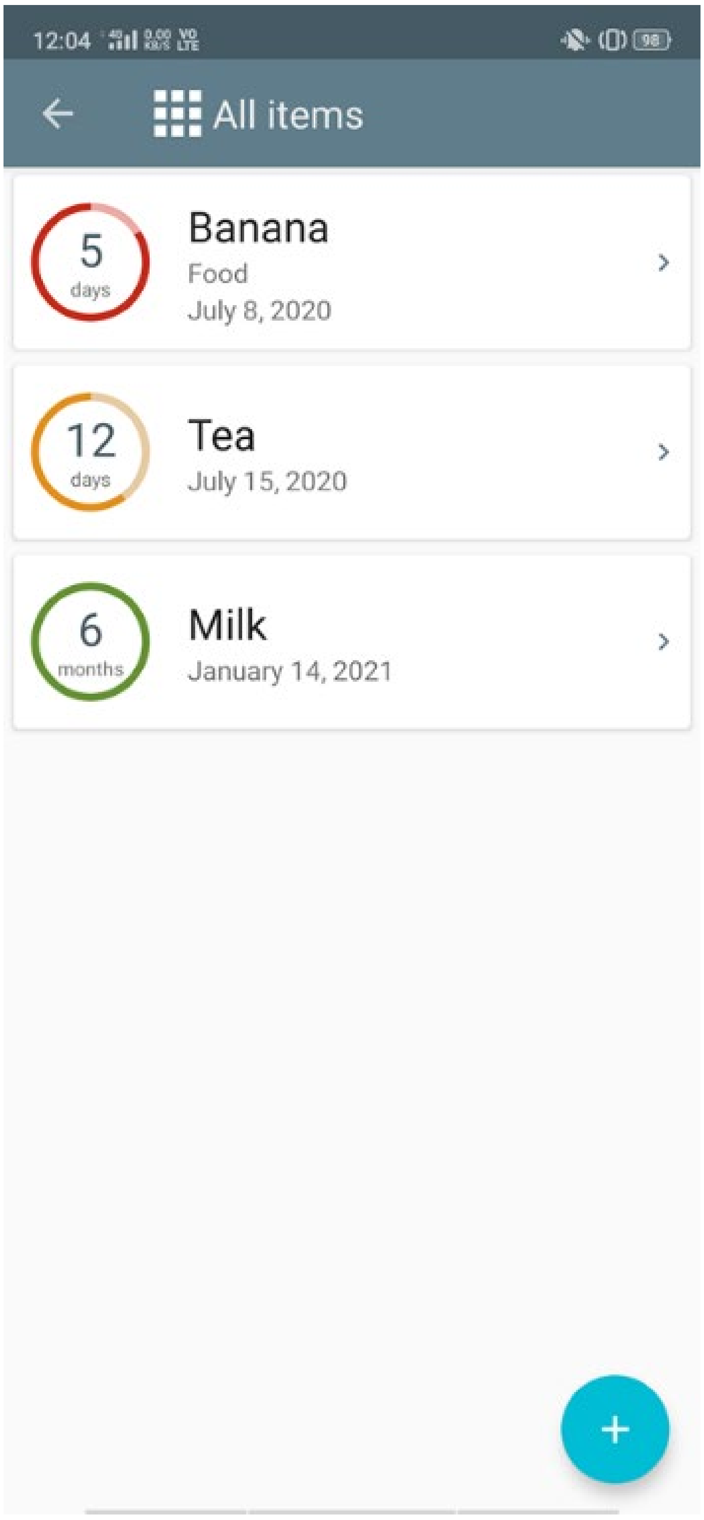


Figure 2.3: All Product View of The Expiry Wiz Application

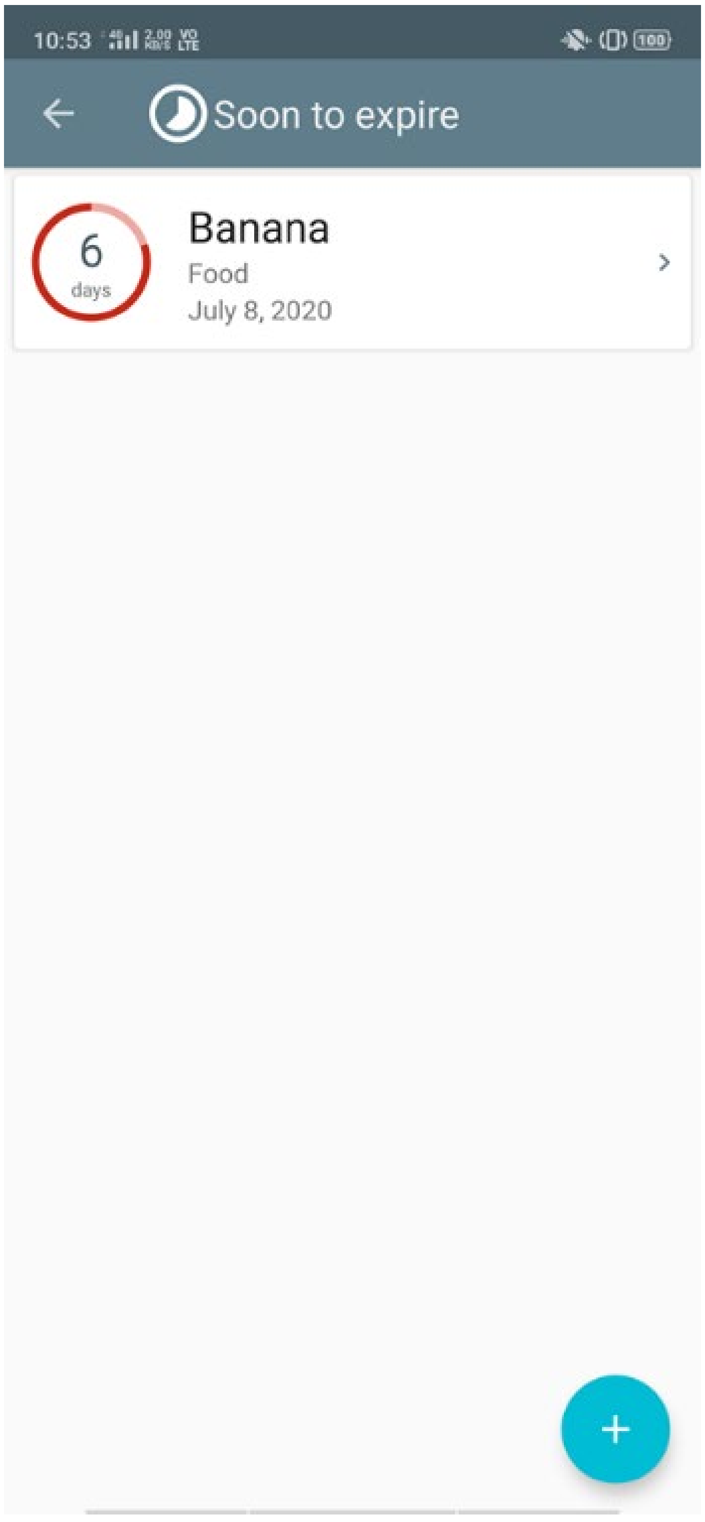
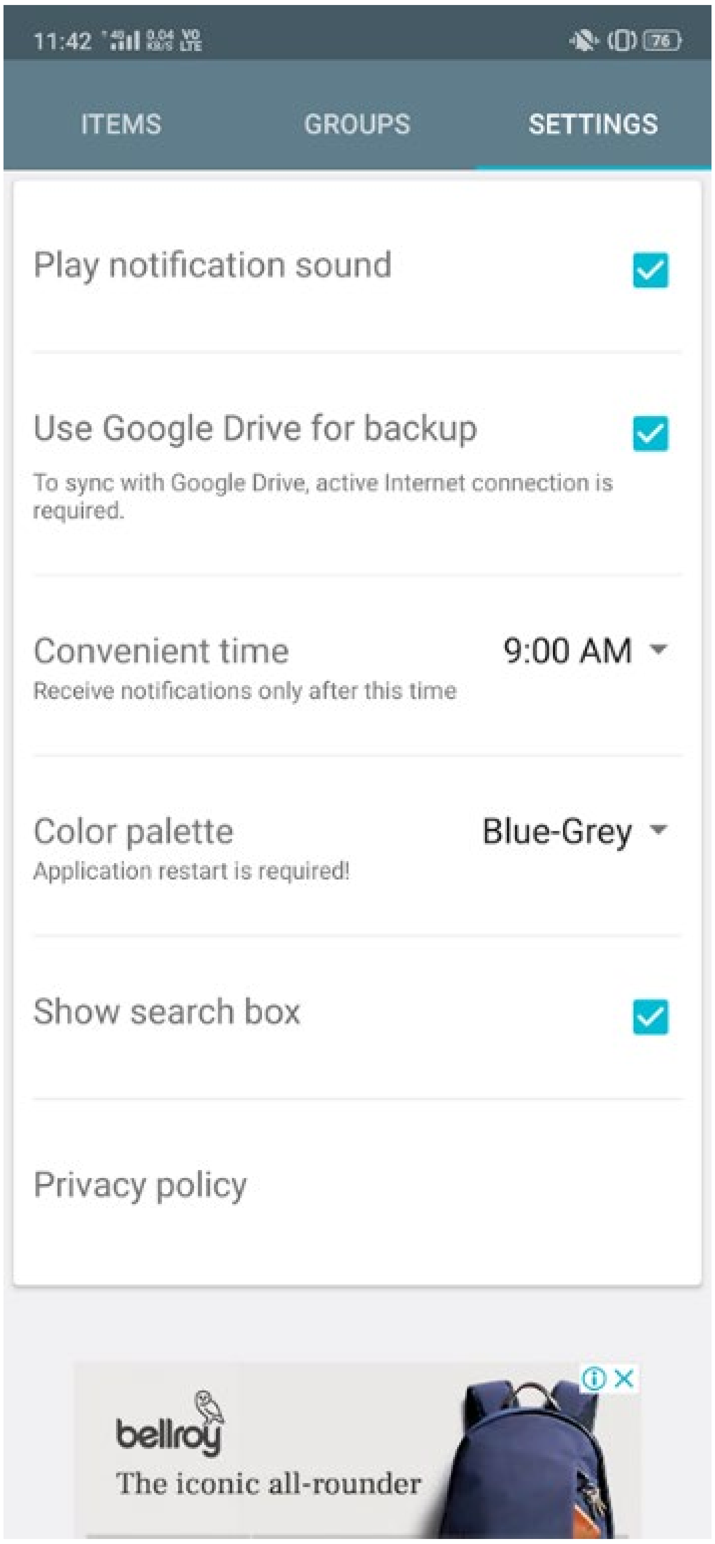


Figure 2.4: Soon to Expire View of The Expiry Wiz Application


11:42

ITEMS GROUPS SETTINGS

Play notification sound ✓

Use Google Drive for backup ✓
To sync with Google Drive, active Internet connection is required.

Convenient time 9:00 AM
Receive notifications only after this time

Color palette Blue-Grey
Application restart is required!

Show search box ✓

Privacy policy

Figure 2.5: Setting Page of the Expiry Wiz Application

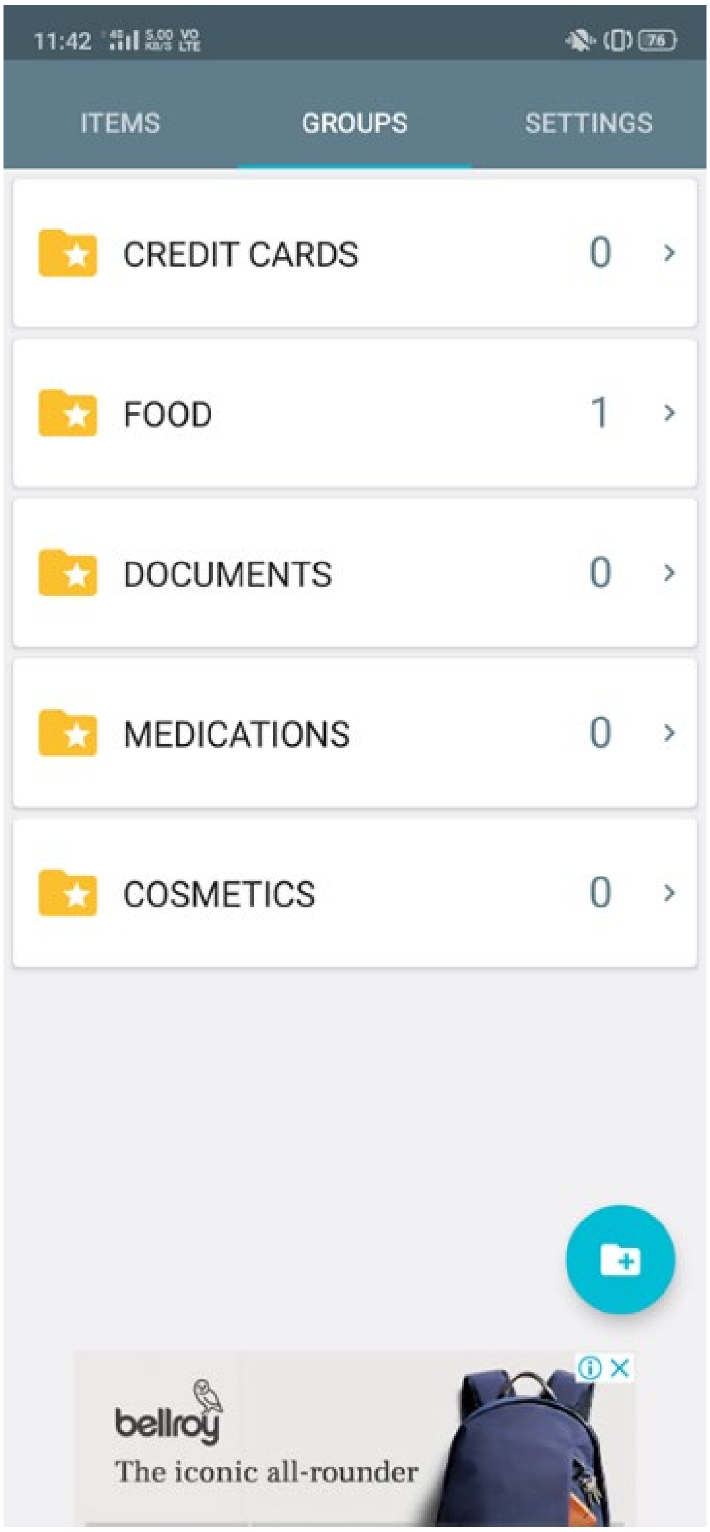


Figure 2.6: Group View of The Expiry Wiz Application

2.2.1.1 Main Features of Expiry Wiz

1. Add New Product

   Referring to Figure 2.2, users can add a new product, set reminders, and set the expiry date. Users can also set the group of the product such as credit cards, food, documents, medications, and cosmetics so that users can view products according to the group. Apart from that, users can choose a specific date of the reminder if there is no suitable reminder period that is suitable. Besides, users are able to create a new group to store the product. In addition, users are able to add notes and photos of each product.

2. View product in different categories

   As shown in Figure 2.1, users can view products in different categories. Users can view all products, products that are going to be expired, and products that are expired. Plus, as shown in Figure 2.3 and Figure 2.4, users can view products by the group. Each category will show the product name, product group, expiry date, and day to expire.

3. Notification/ Reminder

   The application will remind users of the reminder date of the product added to the application. Figure 2.5 shows that users can change the convenient time for receiving notifications and reminders from the application.

4. Edit or manage product and product's group

   The application allows users to edit all added products and delete added products. Users can modify the product's name, expiry date, reminder date, group, and annotation. Besides, users are allowed to add, edit or delete the product's group.

5. Backup and restore function

   Since it is an android application, it allows users to back up data to the user's Google account and restore data from the user's Google account. So that users can backup data and restore it when the user changes a new mobile phone or wipe out data on the mobile phone.

2.2.1.2 The User Interface Design

1. Neat and clean design

   The application has a very clean and neat design. Every part of the application is grouped and arranged accordingly. The application has three groups, Products that show products by the expiry date, Group that shows products by group, and Settings that allow users to set their preference. On the main page, the application shows the Product tab that containing All products, Soon to expire products and Expire products and the number of products in each category on the right side. Users can know how many products they have, how many products are going to expire, and how many products are already expired with just one glance. By navigating to the Group tab, it shows products by group. It allows users to search for the product they want according to the group the user assigned it to. The Settings tab contains user preference, like allowing users to choose the colour palette for the application, on and off the notification sound and choose the convenient time for the user to receive the notification.

2. Add Button that is easy to find

   The add product button is at the Product tab and is indicated with a big round blue button with a '+' sign. The add button is easy to understand because there is a '+' sign on it. Besides, it is on the right bottom of the screen which allows users to click it easily because the position is closed to the user's finger. On the Group tab, the add product button is changed into the add group button. The '+' sign is changed into a folder symbol with a '+' sign inside the folder. It is easy to understand by users because the application indicates the group using a folder symbol.

3. Use colour to distinguish the level of importance

   First of all, the main page shows "All Products" in green colour fonts, and below "All Products" is "Soon to expire" which is indicated in orange colour fonts. Then, the last row "Expired" shows in red colour fonts. Red normally means warning and importance, orange is normally used as a transient warning while green normally means safe and encouraged. In this application, red indicates the product is expired or almost expired, orange indicates the product is going to expire while green indicates the product is still far from its expiry date. The same technique is used in displaying the days to expire of each product. A product that is almost expired or already expired will be shown in a red colour circle, a product that is closed to its expiry date or soon to expired will be shown in an orange colour circle while a product still far from its expiry date will be shown in a green colour circle.

4. Using a symbol to indicate the meaning of the label

   Every label has a symbol that indicates its meaning on the left side.

2.2.1.3 Hardware Used

1. Phone Camera

   The camera is used to capture the image of a product.

2.2.2 Aladdinpro-expiry reminders by Aladdinpro

Adopted from:

<https://play.google.com/store/apps/details?id=com.aladdin.pro&hl=en>

Application Official Website:

<https://www.aladdinpro.com/>

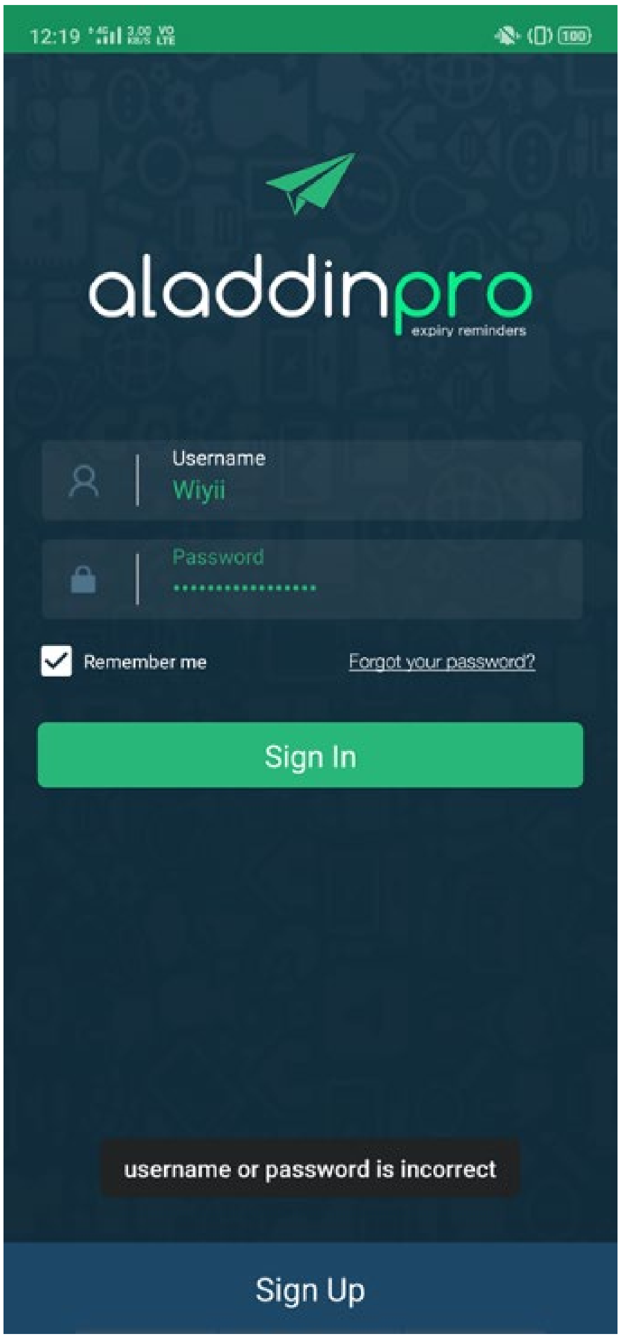


Figure 2.7: User Login Interface of The Aladdinpro-expiry Reminder Application

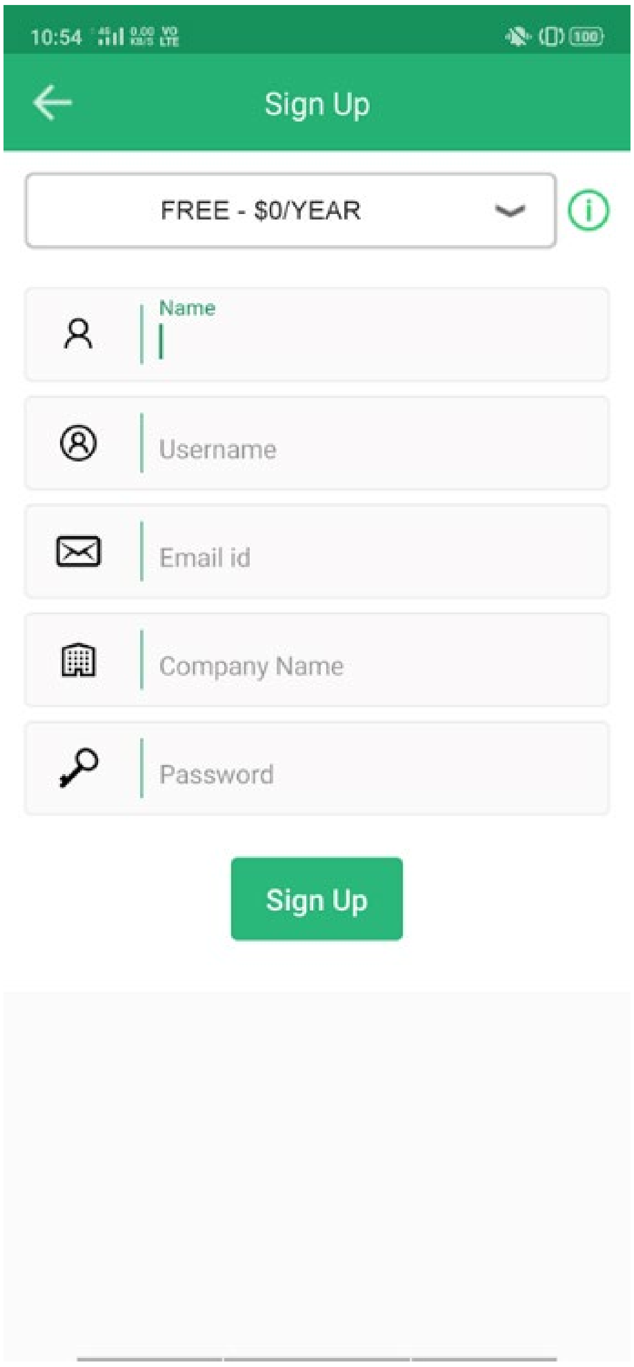


Figure 2.8: Sign Up Interface of The Aladdinpro-expiry Reminder Application

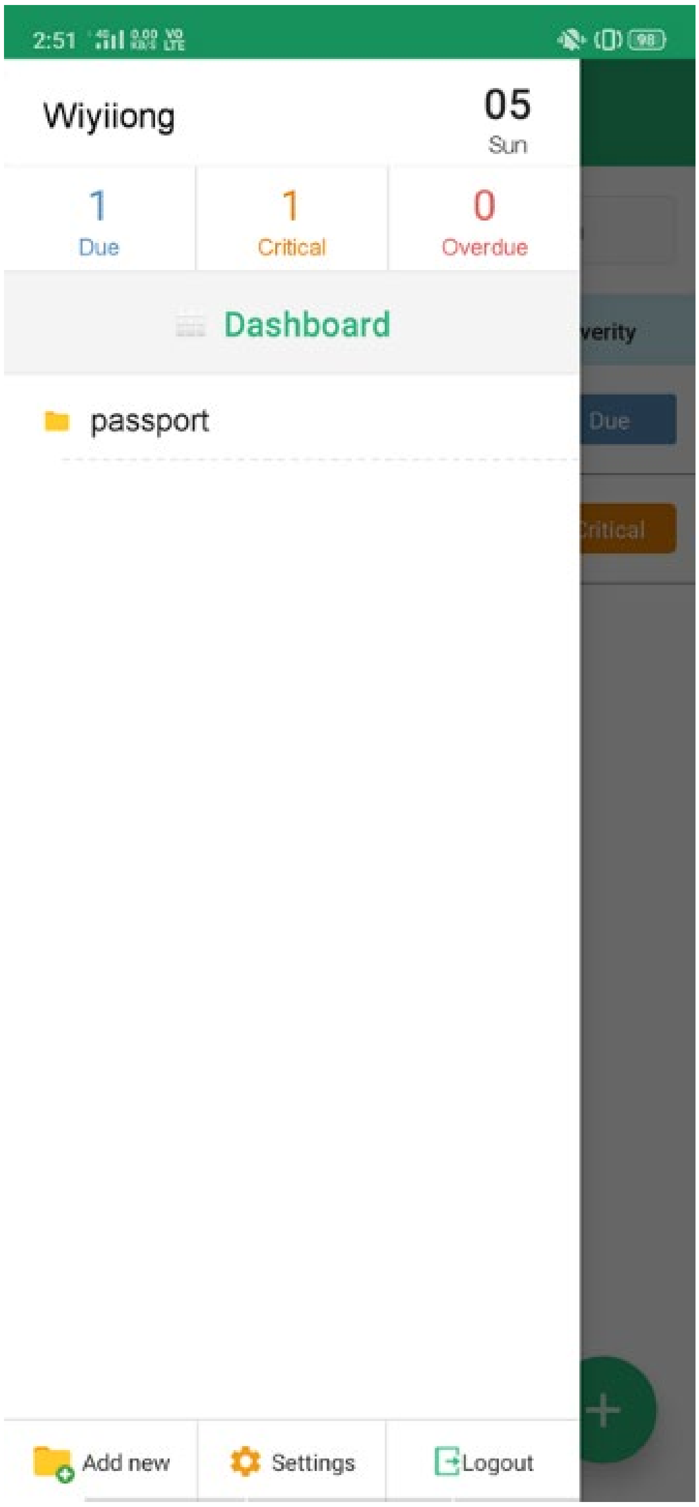
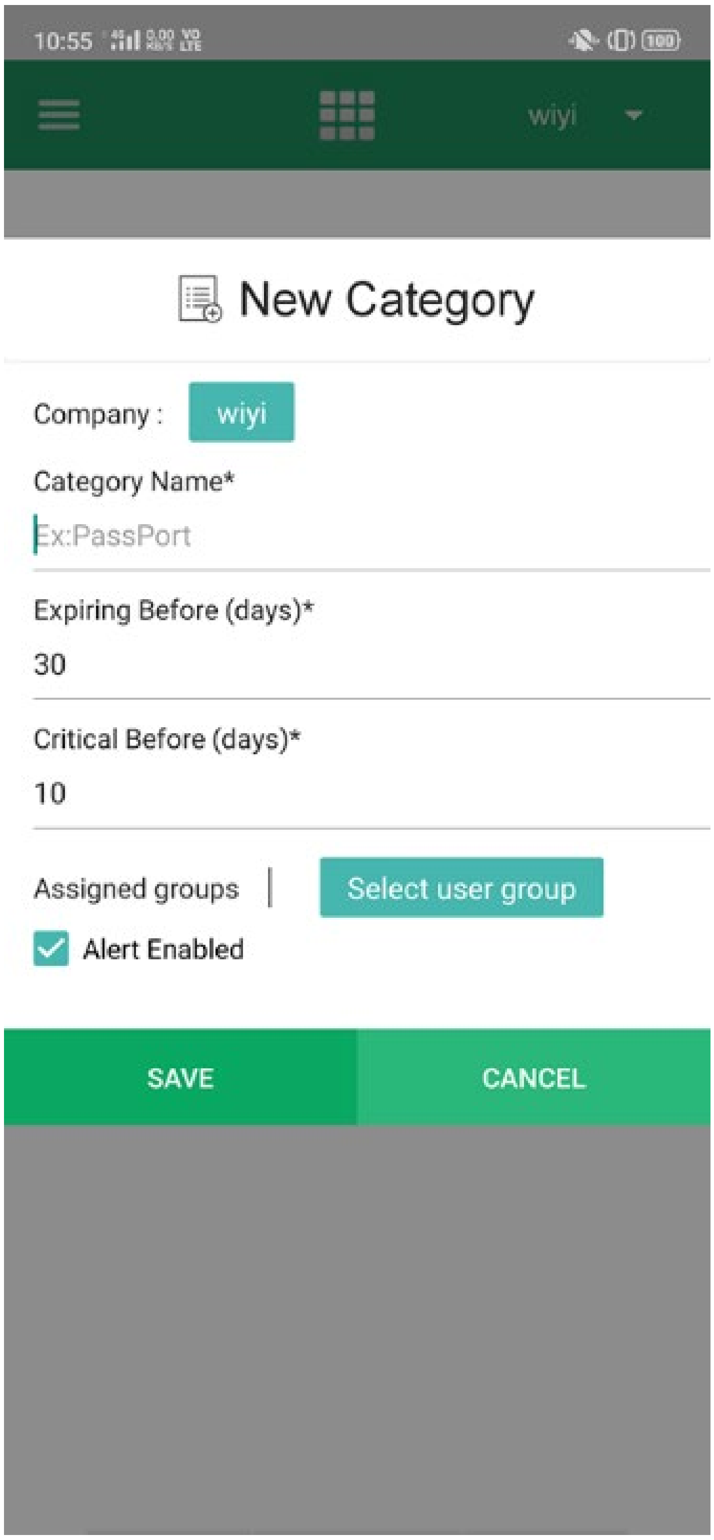


Figure 2.9: Dashboard of The Aladdinpro-expiry Reminder Application


New Category

Company : wiyi

Category Name*

Ex:PassPort

Expiring Before (days)*

30

Critical Before (days)*

10

Assigned groups | Select user group

✓ Alert Enabled

Figure 2.10: Add New Category View of The Aladdinpro-expiry Reminder Application

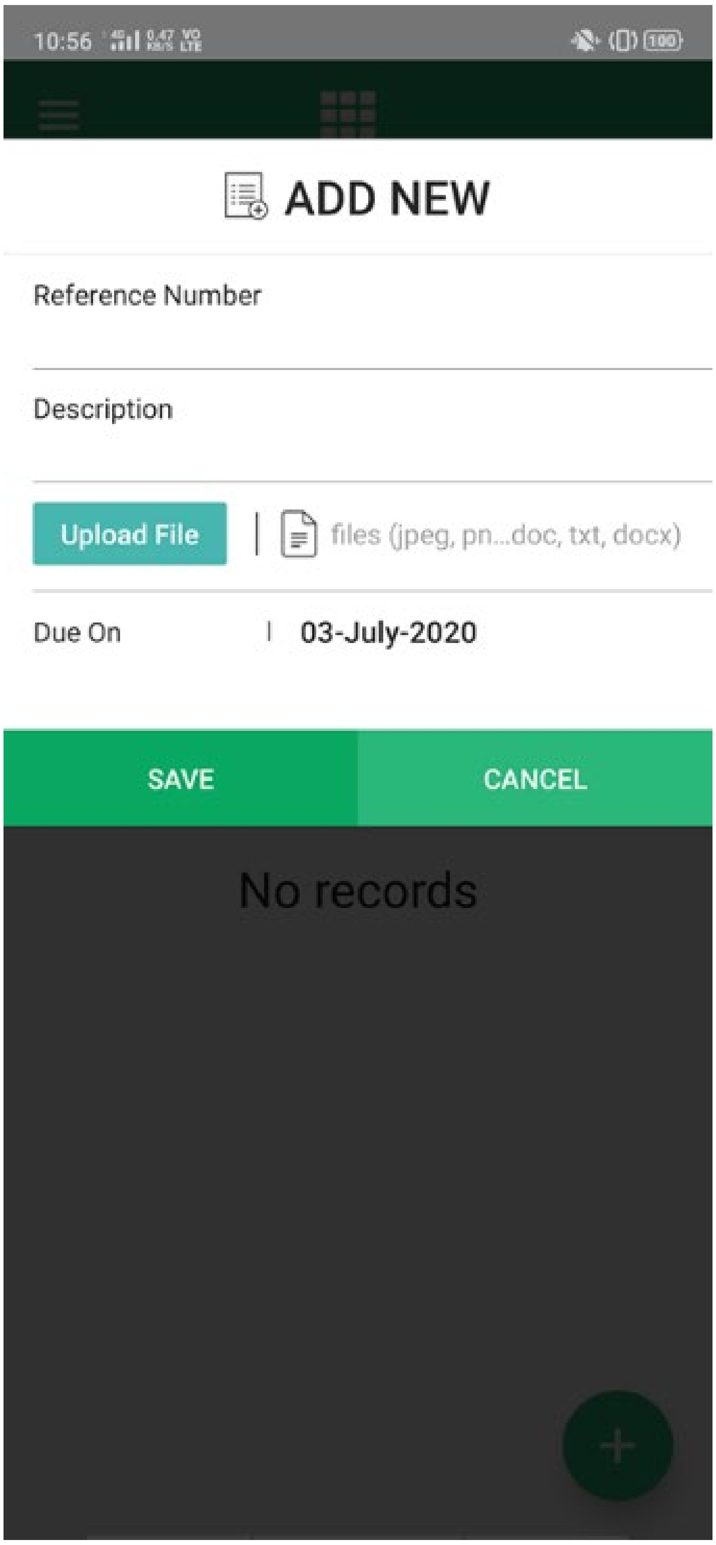
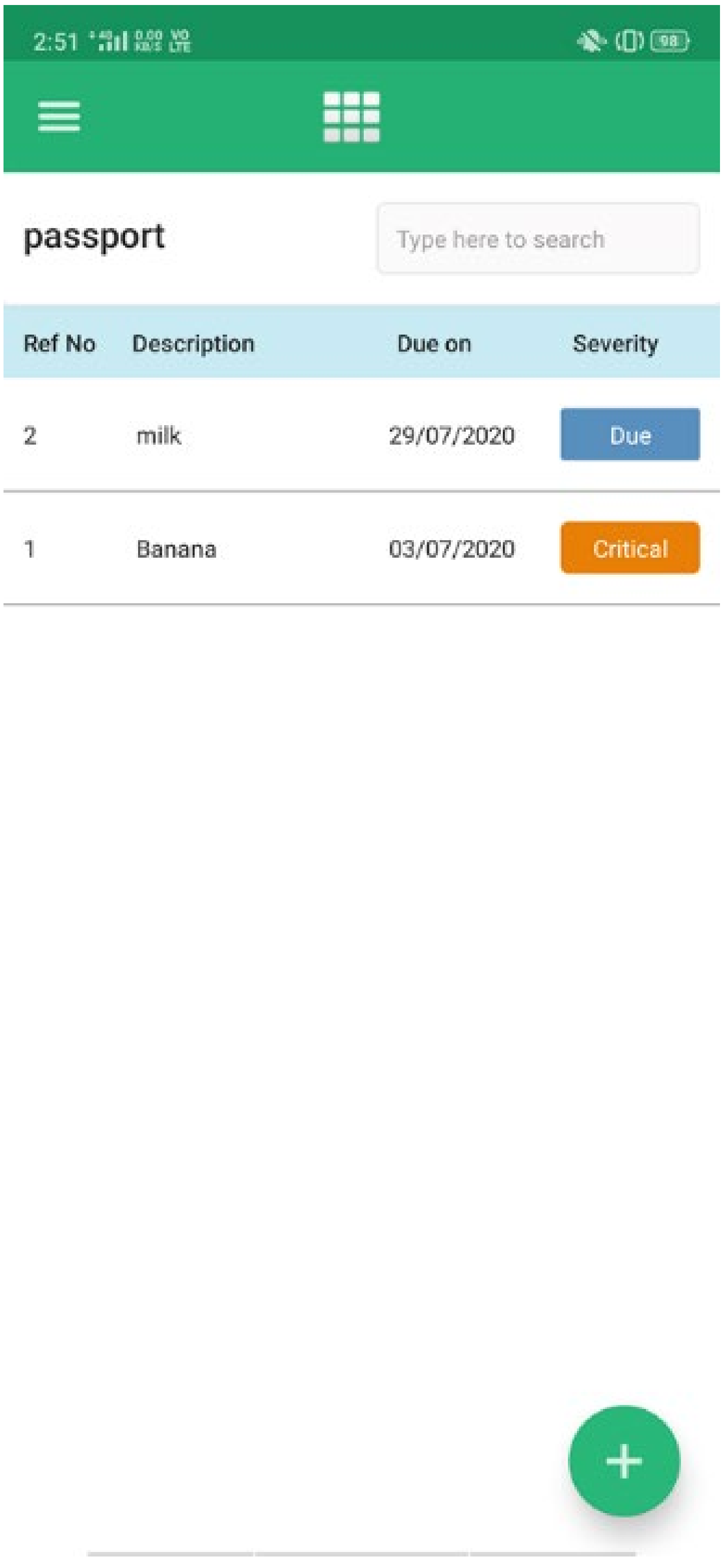


Figure 2.11: Add New Product View of The Aladdinpro-expiry Reminder Application



passport

Type here to search

Ref No	Description	Due on	Severity
2	milk	29/07/2020	Due
1	Banana	03/07/2020	Critical

Figure 2.12: View Product by Group Page of The Aladdinpro-expiry Reminder Application

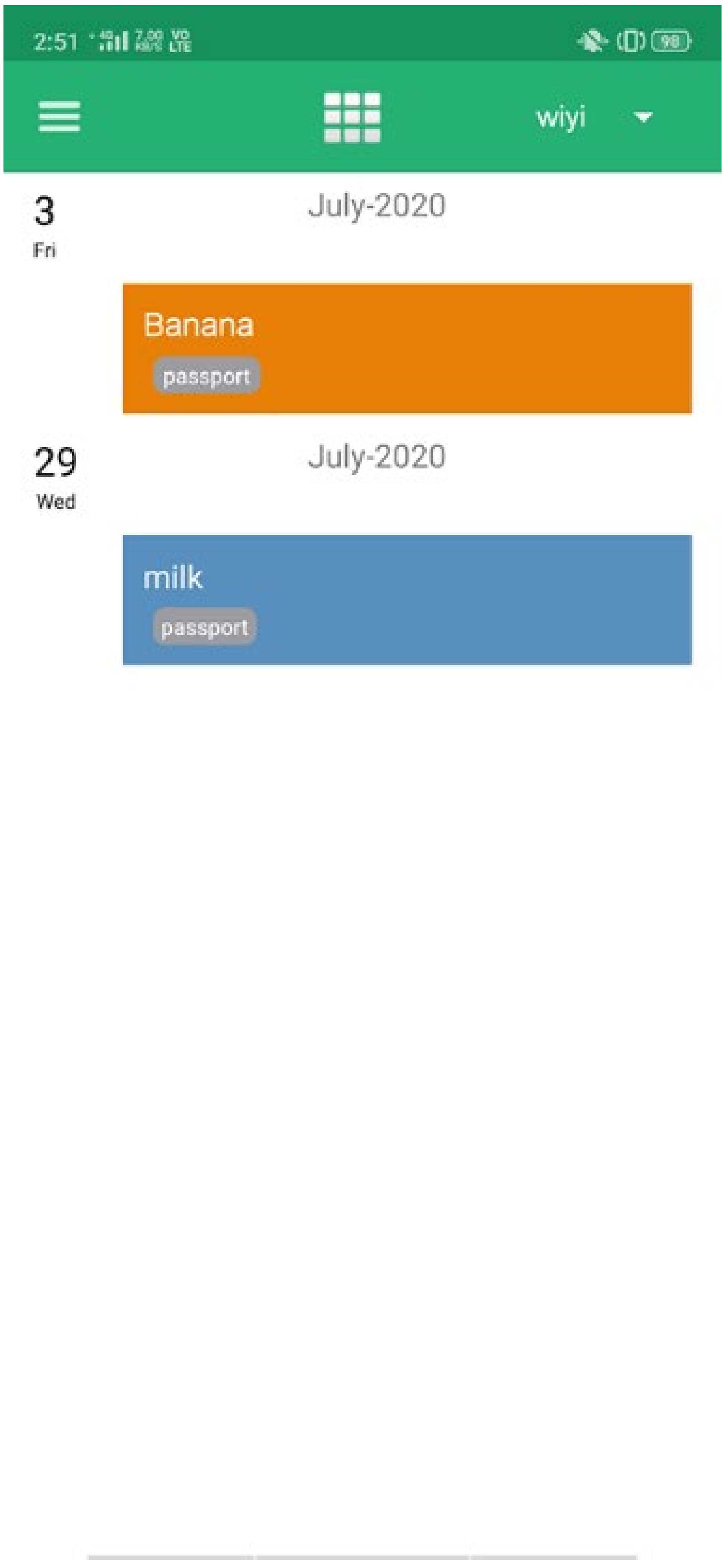


Figure 2.13: Main View of The Aladdinpro-expiry Reminder Application

2.2.2.1 Main Features of Aladdinpro-expiry

1. Add Category and Product

From Figure 2.10, users can add categories like passport, credit card, food, medicine and many more. Users need to enter the number of days from the expiry date to allow the application to show the product as expiring or critical. Based on Figure 2.11, users can add products with the details of the product such as reference number, description of the product, photos of the product and the expiry date in each category.

2. Manage product

   Users can edit and delete products in each category. Users are only able to change the description and the uploaded photo for each product only.

3. Search product in a category

   As shown in Figure 2.12, users can search for the product they want in every category.

4. Cross-Platform Notifications

   The application sends reminder through email, push notifications if the user uses the application on a mobile device, and a notification app if the user uses the application's web version on the user's Personal Computer.

5. Multi-company/ multi-user support

   The application allows user to create multiple users (called "company" in the application) in one account and assign a certain category to other users. This is convenient if the user needs to organize products for many departments.

6. View products by category or by the expiry date

   Users are able to view products by category or by the expiry date. In the category view, shown in Figure 2.12, products are arranged by the date the product is added. The product added later will be on top. In the main/ dashboard view as shown in Figure 2.13, products are arranged by the expiry date. The product that is going to expire comes first.

7. Data is backed up by account

   The application data is backed up by the account. Users can use the application on multiple mobile devices by login into the same account.

2.2.2.2 The User Interface Design

1. Minimalistic user interface

The main page or dashboard page of the application shows only the expiry date and the name of the product going to expire. Another function of the system is hidden in the side navigation bar on the left side. In the navigation bar, there is the username on the top-left side and today date on the top-right side. Below the username and date is the total number of due products, total number of critical products and the total number of overdue products. Then, there is a 'Dashboard' button that navigates back to the main page. Then, each category will be listed down. At the bottom of the navigation bar, there are three functions: adding a new category, navigating to settings and logout. On each category page, products will be listed down with their reference number, description, expiry date and level of importance in a tabular form.

2. Use different colours to distinguish the level of severity
   The application uses different colours to represent the level of severity. In the main page or dashboard page, the application shows due products in blue colour box, critical products in orange colour box and overdue product in red colour box. In the side navigation bar, the application shows the number of due products in blue colour fonts, the number of critical products in orange colour fonts and the number of overdue products in red colour fonts. On the category page, the level of severity is shown in different colours. Blue colour box for the due tag, orange colour box for the critical tag and red colour for the overdue tag.

2.2.2.3 Technology and Hardware Used

1. Linode's SSD Servers
   The application is hosted on Linode's SSD Servers. According to (Linode, no date), Linode is the largest independent open cloud provider. The application is hosted on Solid State Drive (SSD) server provided by Linode.
2. Using SSL secure data communication
   The application uses a Secure Sockets Layer (SSL) protocols to encrypt the data communication between the client and the server. According to (Team, 2019), SSL is a protocol to create authenticated and encrypted connections between networked computers. It works by binding the identities of entities including websites and companies to encryption key pairs via digital documents known as X.509 certificates (Team, 2019). Every key pair features

a private key and a public key, which the private key is kept secure while the public key is spread widely through a certificate.

2.2.3 Expired – Grocery Reminder & Alerts App by Mobile Farm

Adopted from:
<https://play.google.com/store/apps/details?id=com.mobilefarm.expired>

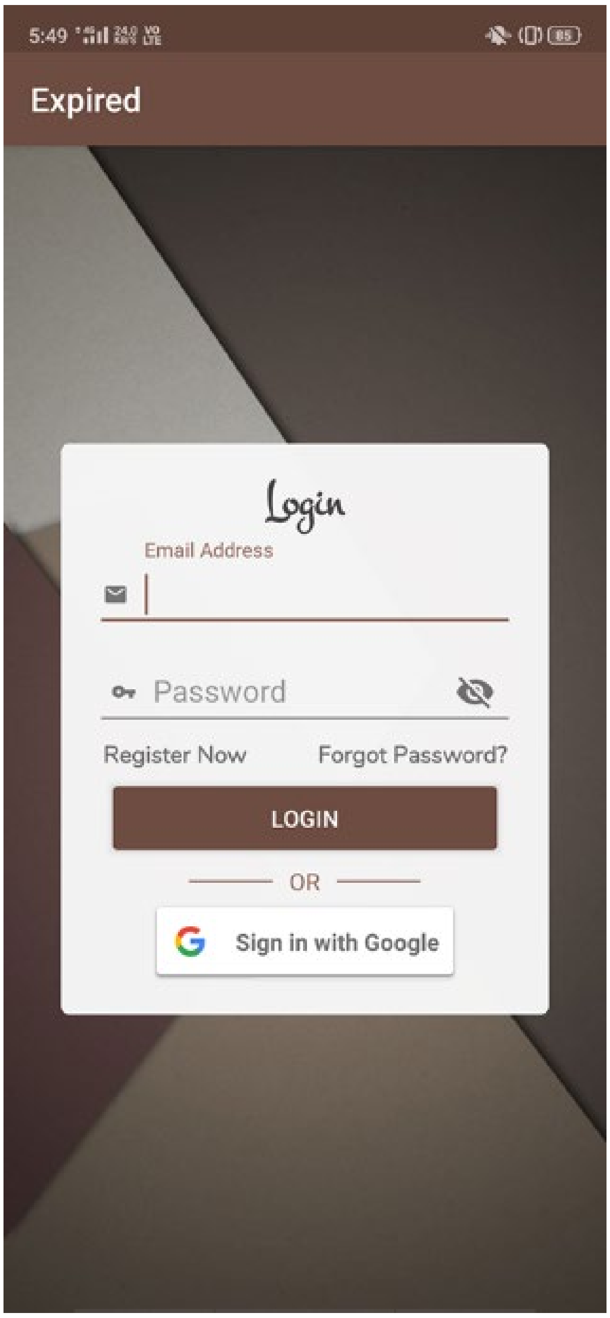
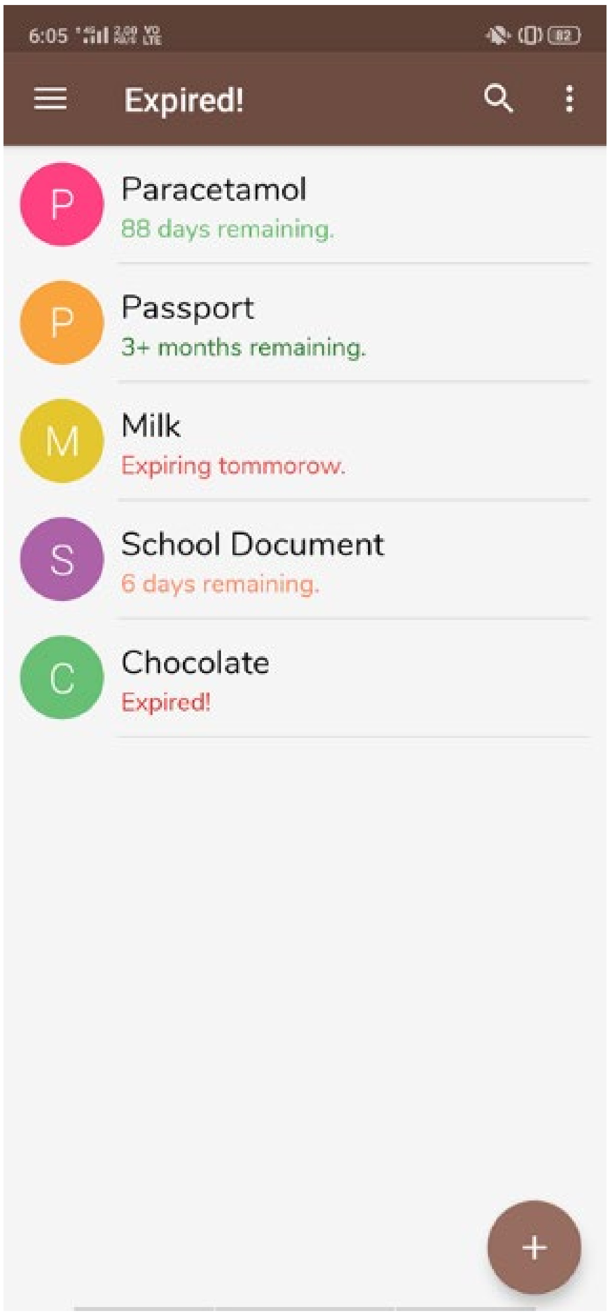


Figure 2.14: Login View of The Expired! Application



Paracetamol
88 days remaining.

Passport
3+ months remaining.

Milk
Expiring tommorow.

School Document
6 days remaining.

Chocolate
Expired!

Figure 2.15: Main View of The Expired! Application

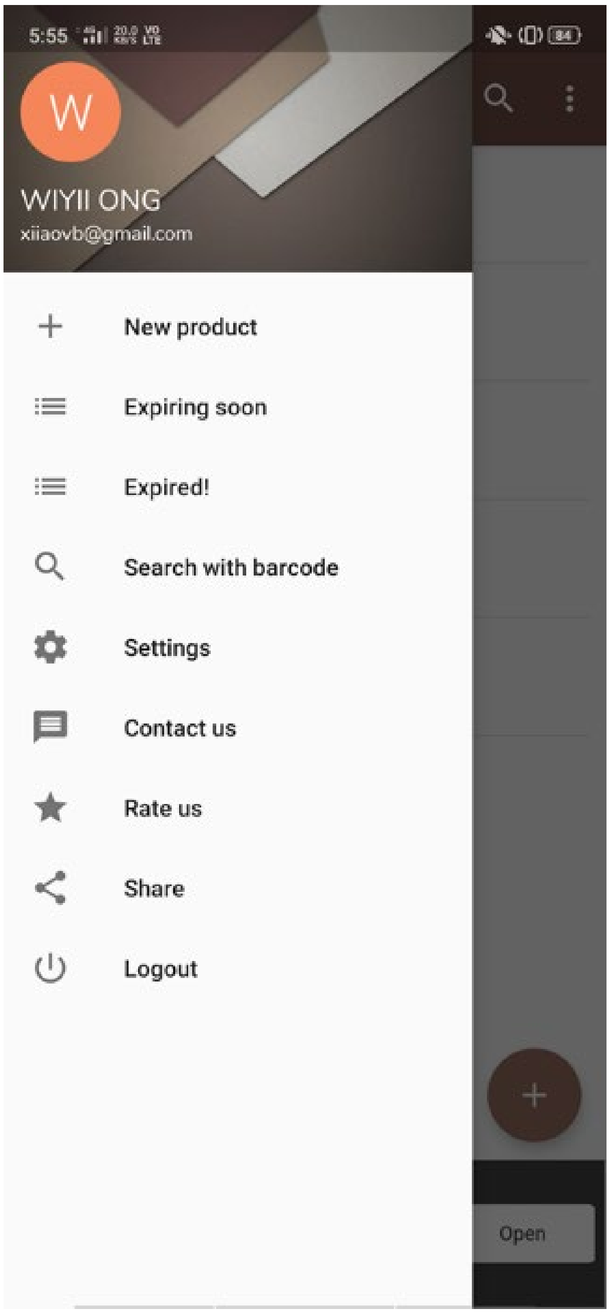
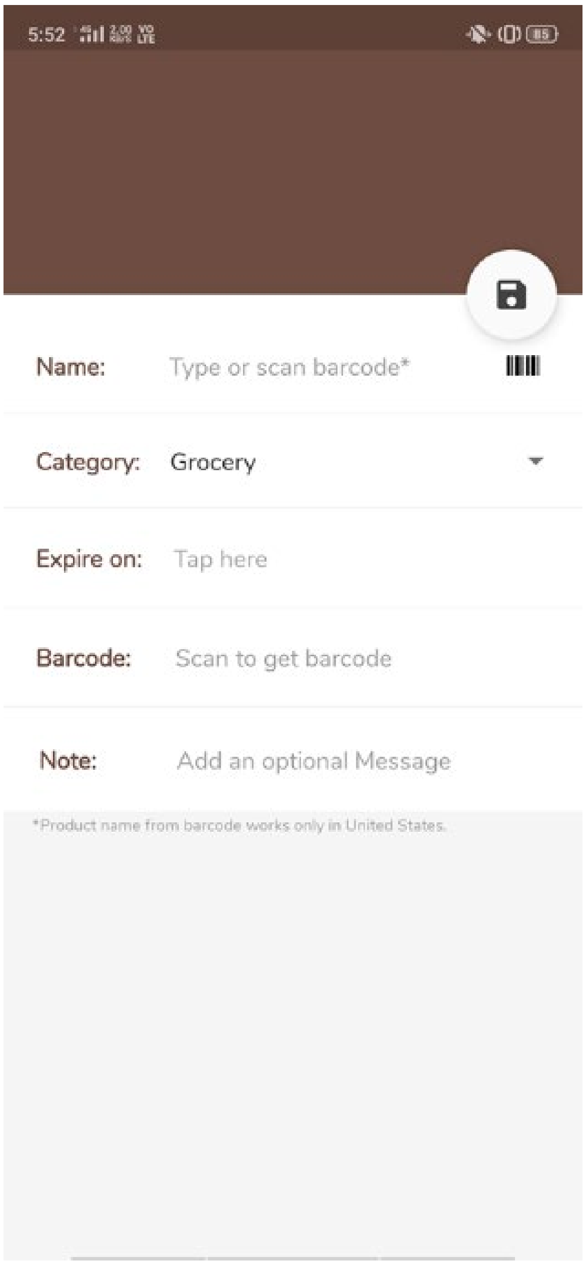


Figure 2.16: Side Navigation View of The Expired! Application


5:52

Name: Type or scan barcode*

Category: Grocery

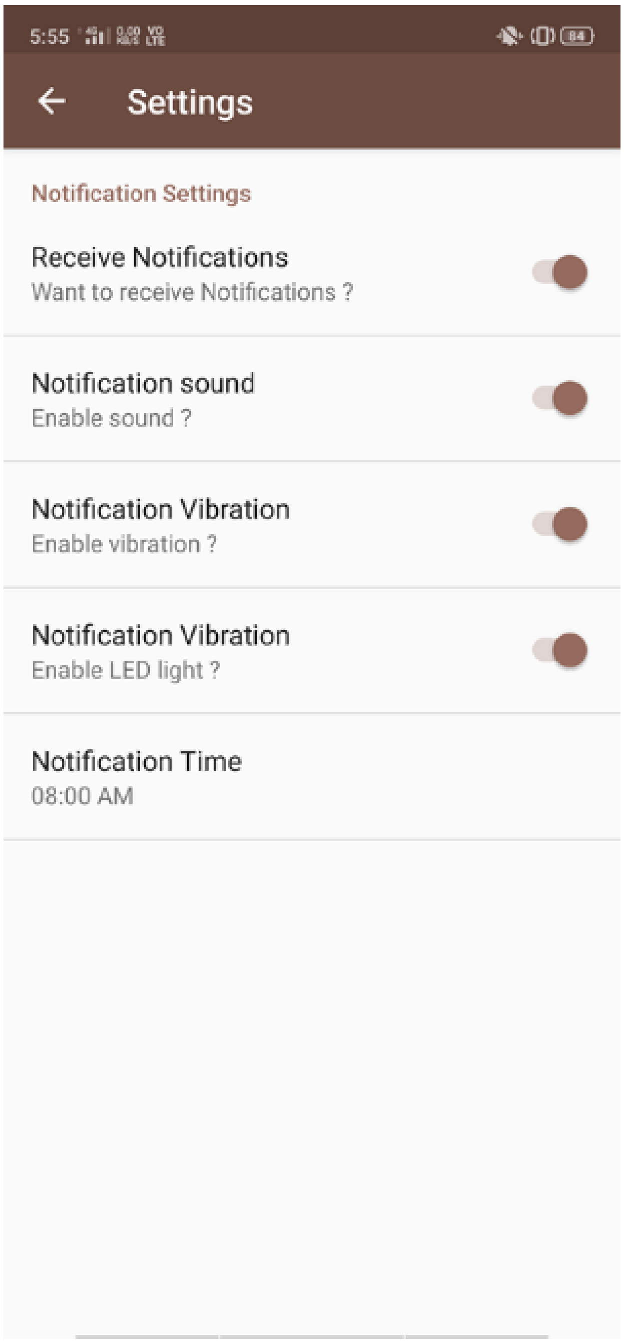
Expire on: Tap here

Barcode: Scan to get barcode

Note: Add an optional Message

*Product name from barcode works only in United States.

Figure 2.17: Add Product View of The Expired! Application


5:55

Settings

Notification Settings

Receive Notifications
Want to receive Notifications ?

Notification sound
Enable sound ?

Notification Vibration
Enable vibration ?

Notification Vibration
Enable LED light ?

Notification Time
08:00 AM

Figure 2.18: Setting View of The Expired! Application

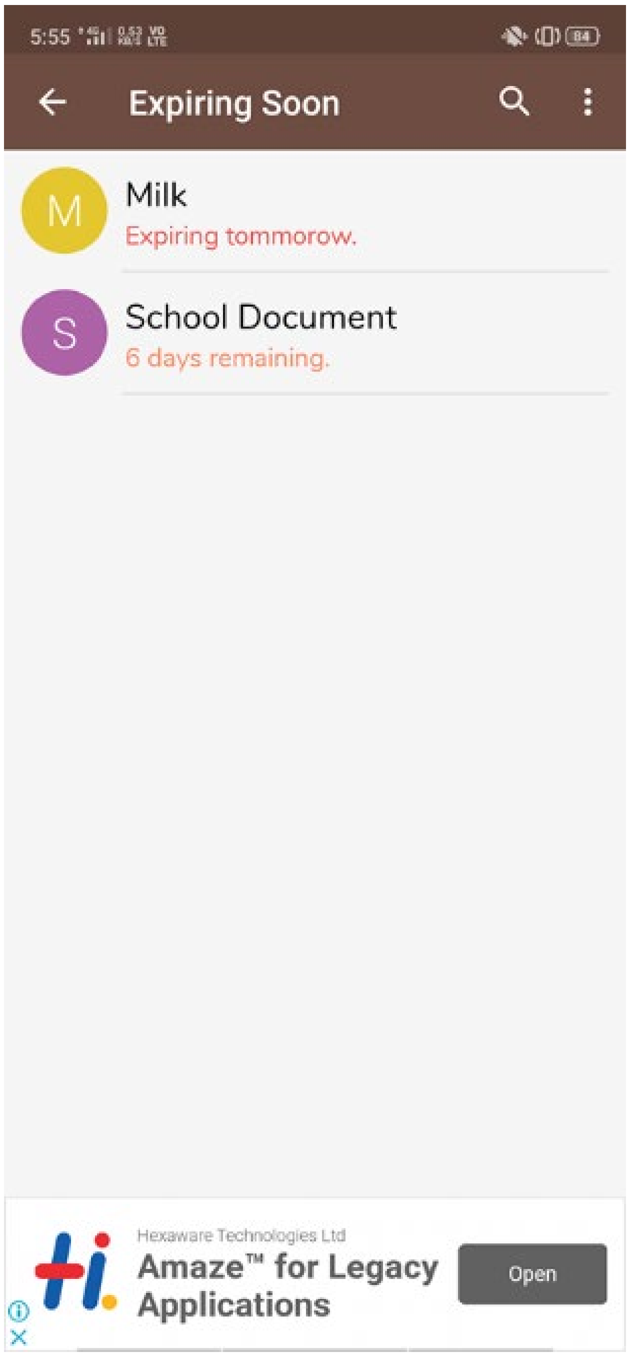


Figure 2.19: Expiring Soon Product View of The Expired! Application

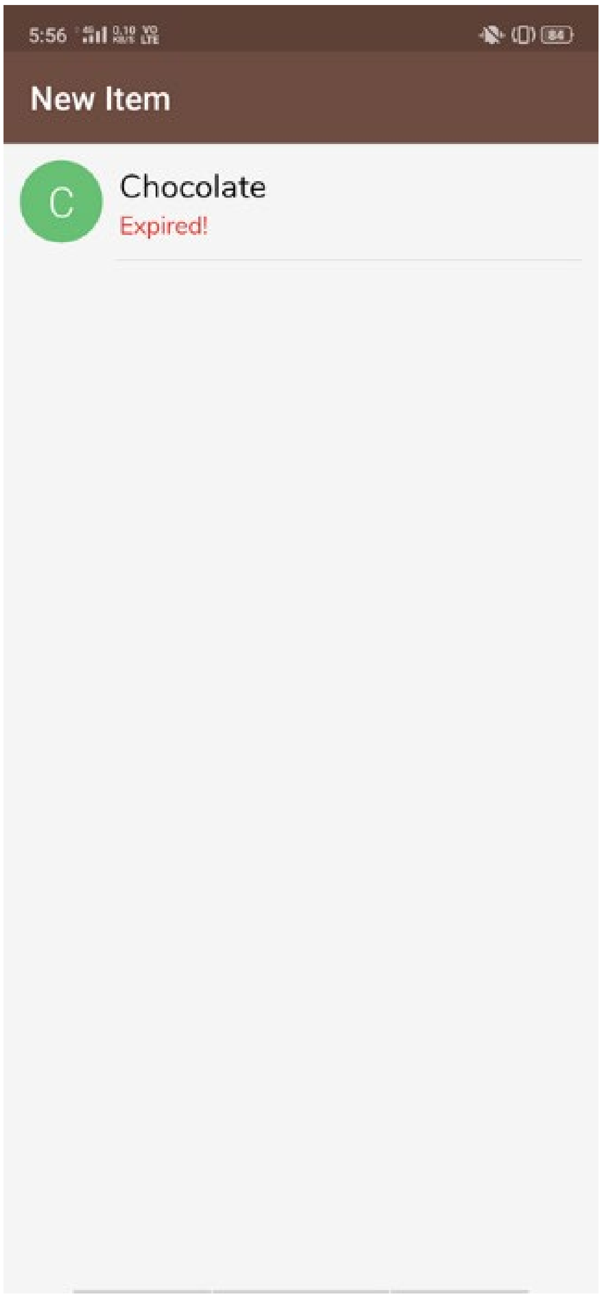


Figure 2.20: New Product View of The Expired! Application

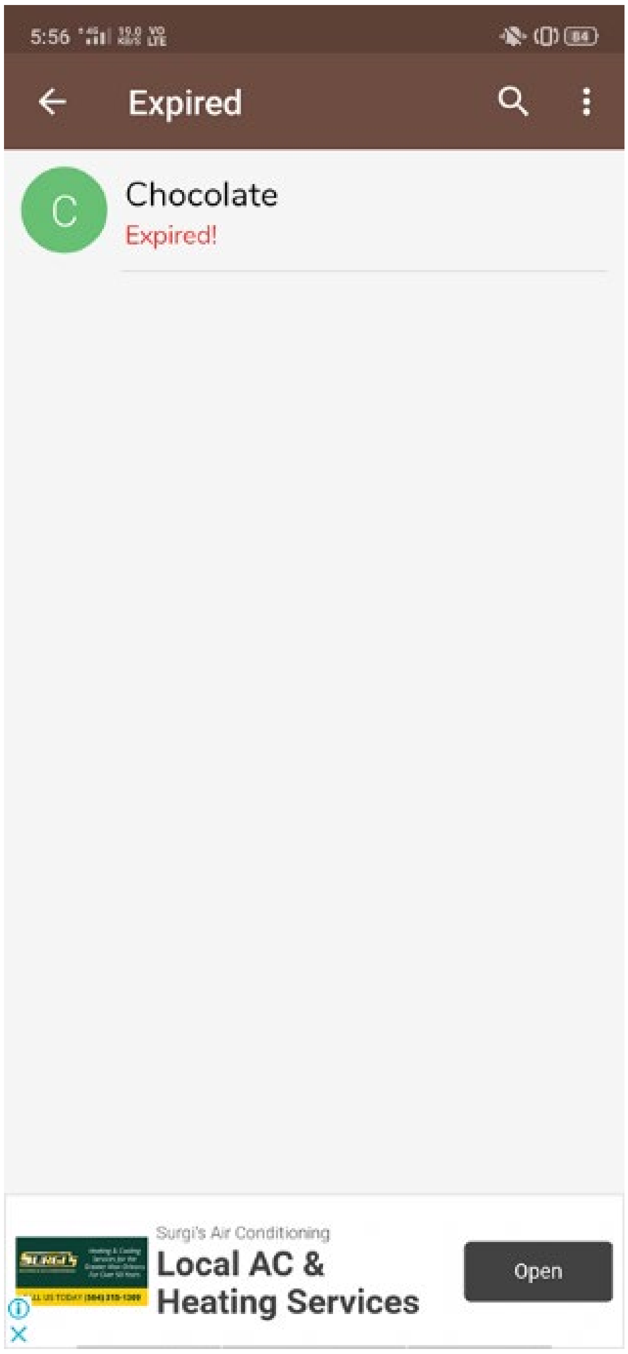


Figure 2.21: Expired Product View of The Expired! Application

2.2.3.1 Main features of Expired - Grocery Reminder & Alerts App

1. Scan barcode of a product

   As shown in Figure 2.16, the side navigation shows a function named “Search with Barcode”. By referring to Figure 2.17, users can scan the barcode of a product when adding a product. The barcode information will be stored and users can search product using barcode later.
2. Add a product

According to Figure 2.17, users can add a product with the details of the product's name, category, expiry date, barcode and notes. Users can select the category of a product from grocery, medical, bills, cards and others. Users can add a new category. Users need to select others if the product does not belong to any category from the selection. As shown in Figure 2.15, the add product function can be found in the '+' button on the main page and 'New product' button at the side navigation bar on the left side.

3. Manage products

   Users can delete a product by long-press a product and edit a product by pressing a product. Users can change the name, category, expiry date, barcode and notes for the product they want to edit.

4. View soon expire product and expired product

   Users can view products that are expiring soon and products that are already expired on a different page as shown in Figure 2.19 and Figure 2.21. Both of the features can be navigated through the side navigation bar on the left side shown in Figure 2.16.

5. Push notification reminder

   The application will send a push notification on the mobile device to reminds users about the product that is going to expire. Users can change the notification time in Settings as shown in Figure 2.18.

6. Sync data with all devices of the same account

   Users can sync the application data with all devices that are logged into the same account. This means that the application data is backed up.

7. Use a Google account to log in

   According to Figure 2.14, the application allows users to log in using their Google account instead of going through a series of registration steps to create an account to use the application.

2.2.3.2 The User Interface Design

1. Simple interface that looks alike as Google Application

   The user interface of the application is very simple and clean. The developer uses the design of old UI of Google applications like Google Photo and Google Drive on as a reference.

2. Using different colour on the description of each product

The application uses 3 different colours to show the level of severity. The red colour font is for the expired product or the product that will expire on the next day, the orange colour font is for the product that will expire, and the green colour font is for the product that is far from its expiry date.

3. Navigation bar available

   The main page only shows the product added and the add product button. Other functions and views are placed inside the side navigation bar to make sure the main page is simple and clean.

2.2.3.3 Technology and Hardware used

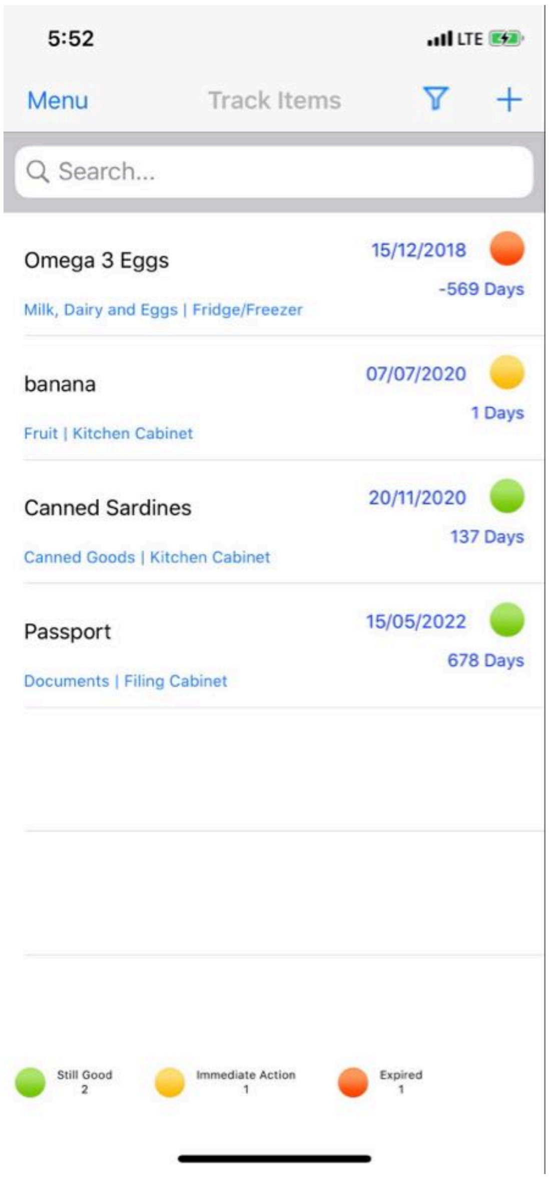
1. Camera & Barcode API (The Vision API)

   The camera is used to scan the barcode and the Barcode API is used to detect and identify the barcode. In this case, 1D barcodes are used. This API also able to detect 2D barcodes such as QR Codes. According to (Google, 2017), the barcode API can detect barcodes in any orientation, in real-time. It can detect several barcodes at one time too.

2.2.4 Before Expiry by Law Gie joo

Adopted from: <https://apps.apple.com/us/app/id1384262435>

Official Websites: <https://appadvice.com/game/app/before-expiry/1384262435>



5:52

Menu Track Items

Search...

Omega 3 Eggs	15/12/2018	-569 Days
Milk, Dairy and Eggs | Fridge/Freezer		
banana	07/07/2020	1 Days
Fruit | Kitchen Cabinet		
Canned Sardines	20/11/2020	137 Days
Canned Goods | Kitchen Cabinet		
Passport	15/05/2022	678 Days
Documents | Filing Cabinet		

Still Good 2 Immediate Action 1 Expired 1

Figure 2.22: Main View of The Before Expiry Application

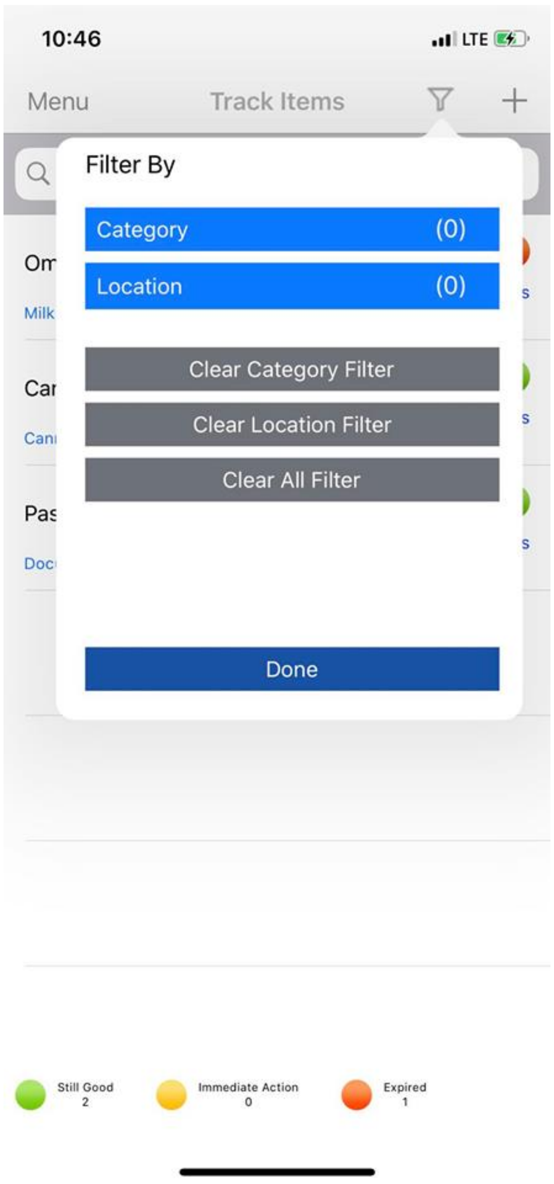


Figure 2.23: Filter Function of The Before Expiry Application

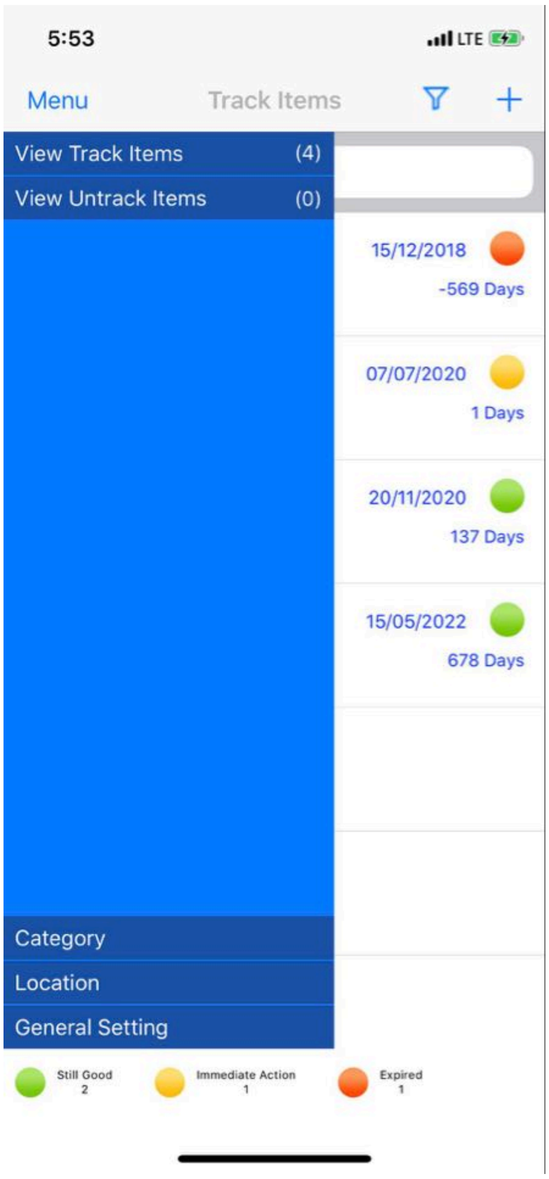
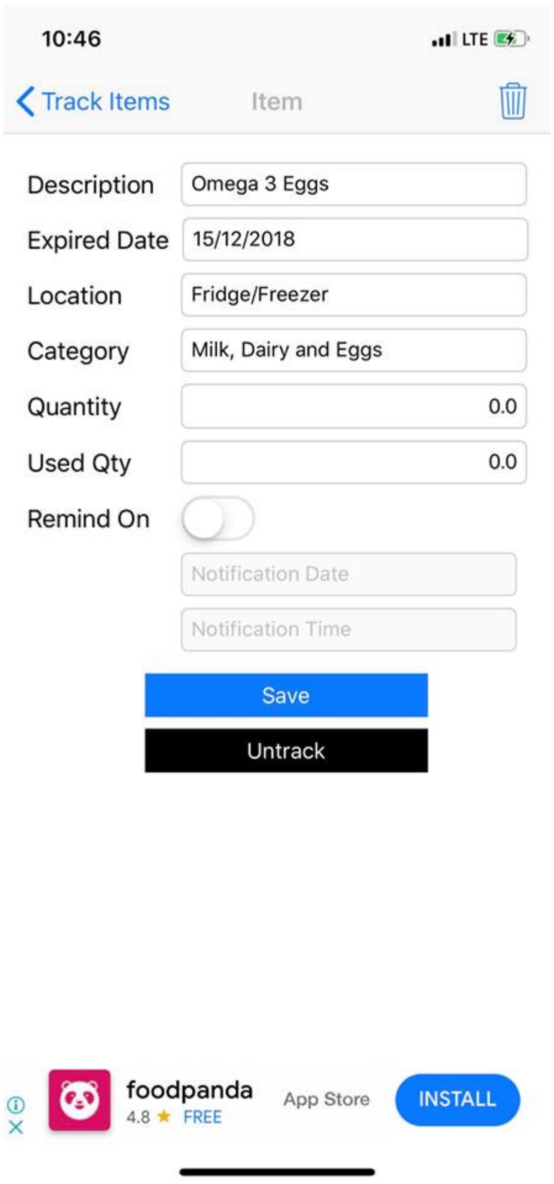


Figure 2.24: Side Navigation Bar of The Before Expiry Application


10:46

Track Items Item

Description: Omega 3 Eggs

Expired Date: 15/12/2018

Location: Fridge/Freezer

Category: Milk, Dairy and Eggs

Quantity: 0.0

Used Qty: 0.0

Remind On

Notification Date

Notification Time

Save

Untrack

foodpanda 4.8 ★ FREE App Store INSTALL

Figure 2.25: Manage Product View of The Before Expiry Application

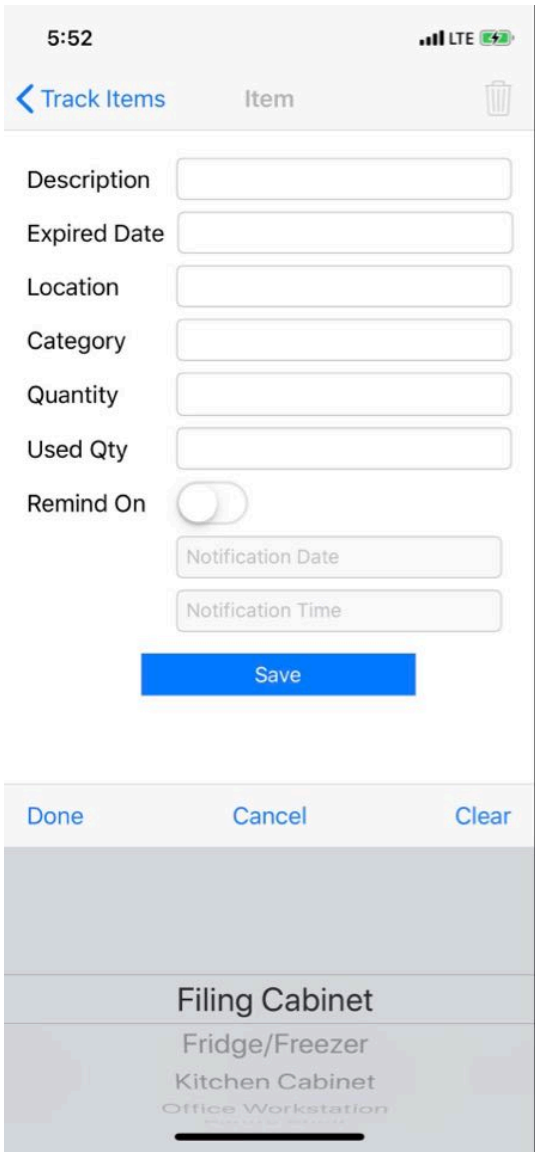


Figure 2.26: Add Product View of The Before Expiry Application

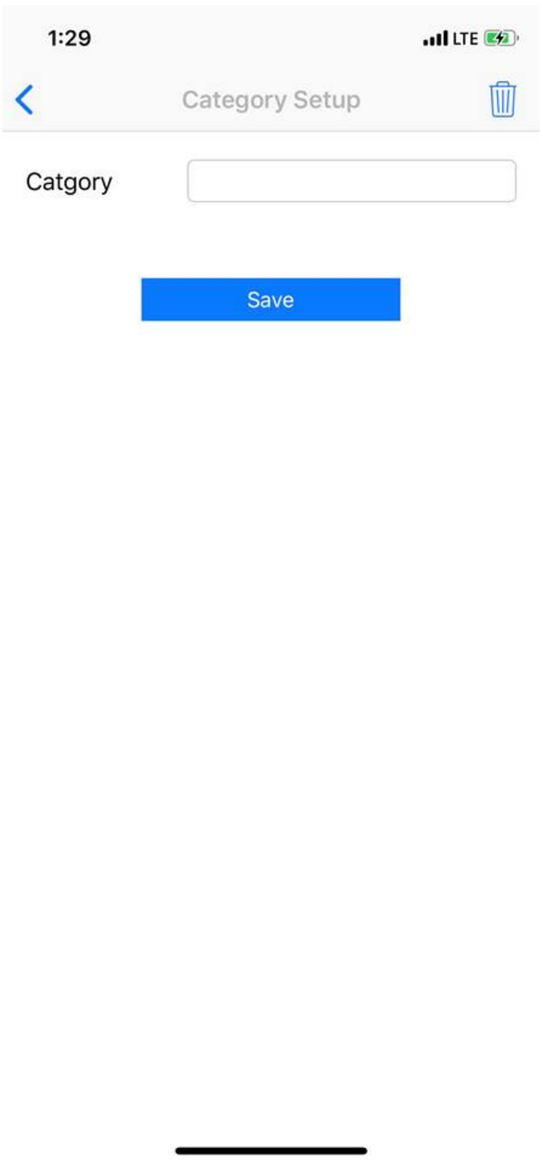


Figure 2.27: Add Category View of The Before Expiry Application

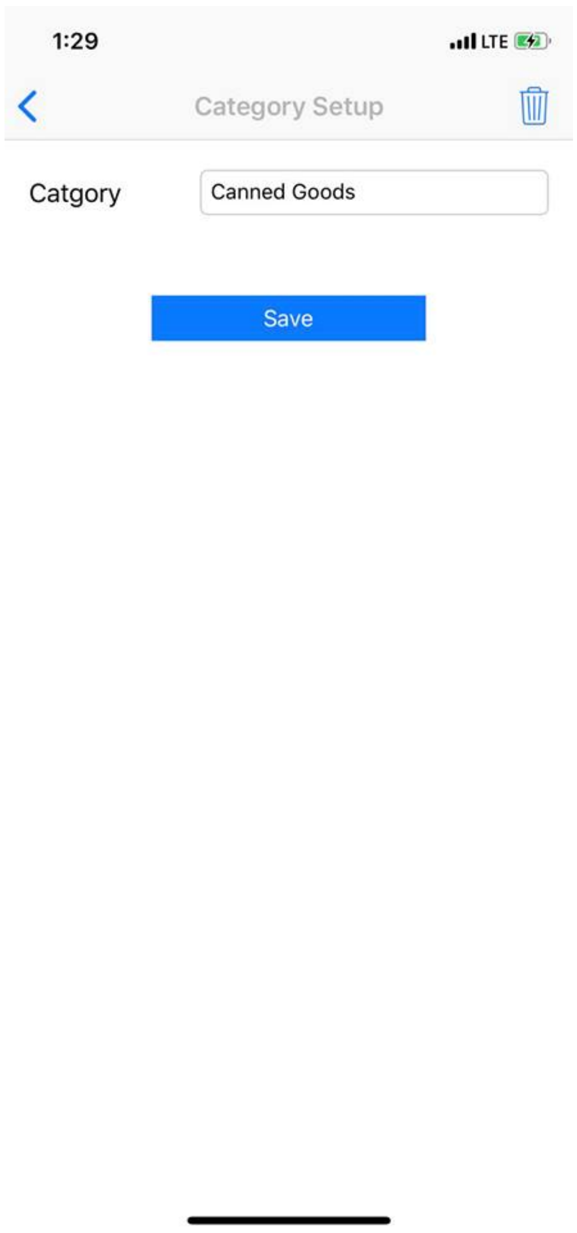


Figure 2.28: Mange Category View of The Before Expiry Application

2.2.4.1 Main features of Before expiry

1. Add products, category and location

   Users can add products, category and location as their need. The add products function can be found on the top right of the main page as shown in Figure 2.22 while the add category function and add location function can be navigated through the menu on the right side as shown in Figure 2.24.

   In add products view shown in Figure 2.26, users need to enter product name (Description), quantity and used quantity manually. The expiry date and notification date is selected by using the Apple date picker, while notification time is selected by using the Apple time picker. Users can only select from the available categories and locations by using the Apple picker.

To add a category or location, users need to navigate to its page through the side menu bar shown in Figure 2.24. In the category page and the location page, the add button can be found on the top right side of the screen. Users only need to type in the category name or the location to add them, as shown in Figure 2.27.

2. Manage products, category and location

   Users can edit or delete a product, a category or a location in the application. As shown in Figure 2.25, users can change the product name (Description), expiry date, quantity, used quantity, location, category, notification date and notification time of a product. As shown in Figure 2.28, users can delete or edit the category in the category page and the location in the location page.

3. Apply filter when viewing product

   On the main page, users can filter the products by category and by location as shown in Figure 2.23. Users can clear the filter after they do not need the filter.

4. Search product

   On the main page, there is a search bar on top that allows users to search product.

5. Have an 'untrack' option

   Users can 'untrack' a product by clicking the 'untrack' button. After a product is 'untrack', the product will no longer be shown on the main page. If users want to view the 'untrack' product, users can navigate to the page through the side menu bar as shown in Figure 2.24.

6. Receive alert notification based on date and time selected

   Users can receive the push notification based on the notification date and notification time selected by the users when adding a product.

2.2.4.2 The User Interface Design

1. Consistent UI Design

   The UI design is consistent. For example, the add button and delete button are the same in the main page, location page and category page. Plus, the colour used for the button is the same.

2. Use a different colour to represent the level of severity

In the application, there are 3 colours to represent severity. The red colour circle means the product is expired, the yellow colour circle means the product needs immediate action, and the green colour circle means the product is still good. The application also shows the meaning of the colour circles at the bottom of the screen and the total amount of product in each level of severity.

3. Menu bar available

   A menu bar is available on the left-hand side that allows users to navigate through different pages. It makes sure the main page remains simple and clean.

2.2.5 Expiry Reminder by AppNextDoor Labs

Adopted from: <https://apps.apple.com/us/app/expiry-reminder/id1231732904>

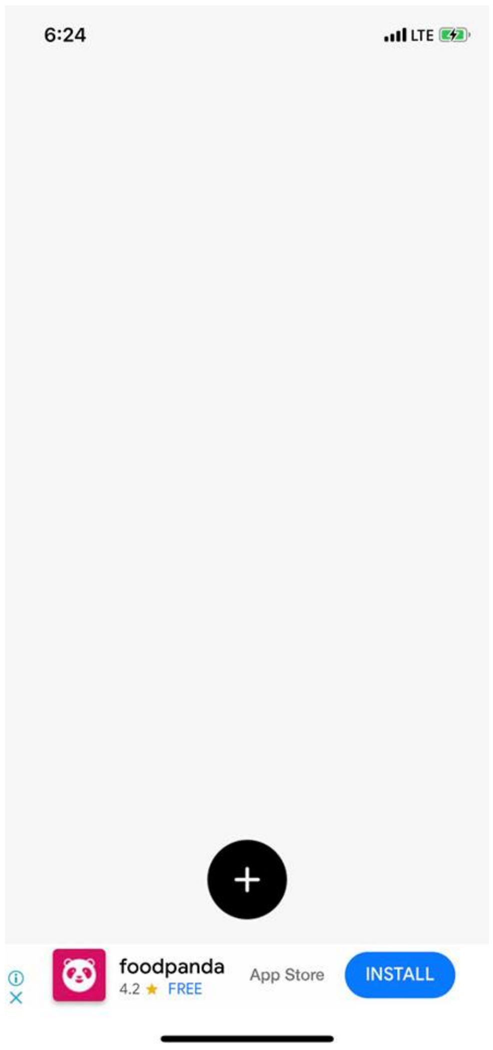
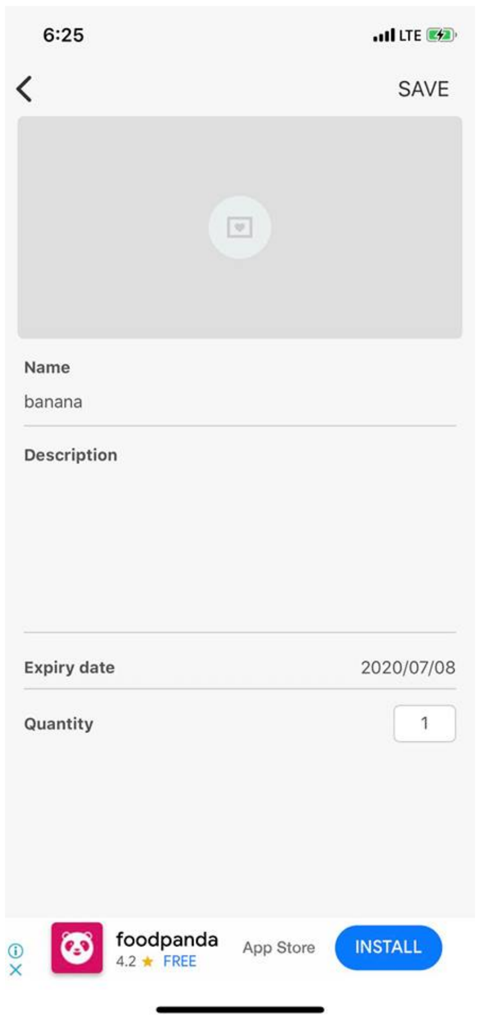


Figure 2.29: Main View of Expiry Reminder Application When User First Use It



Name
banana

Description

Expiry date 2020/07/08

Quantity 1

Figure 2.30: Add Product View of Expiry Reminder Application

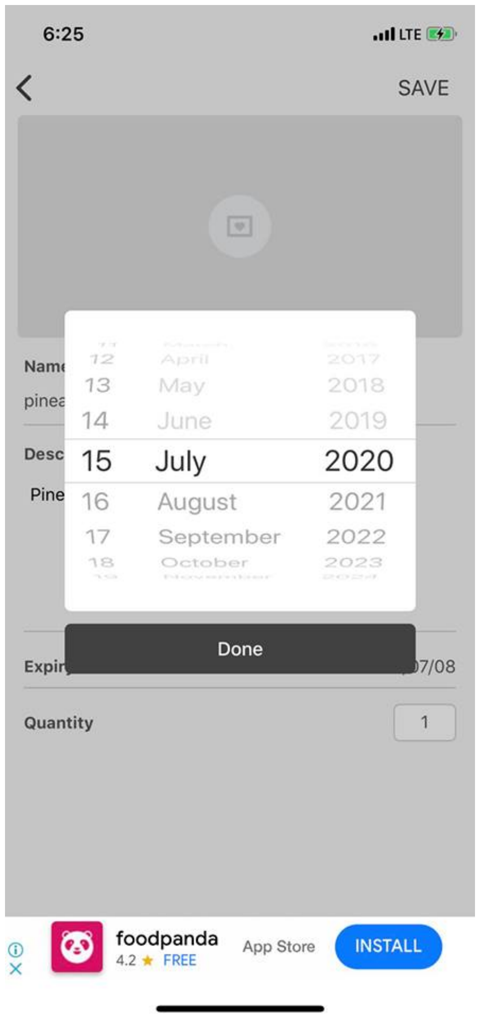


Figure 2.31: Date Picker View of Expiry Reminder Application

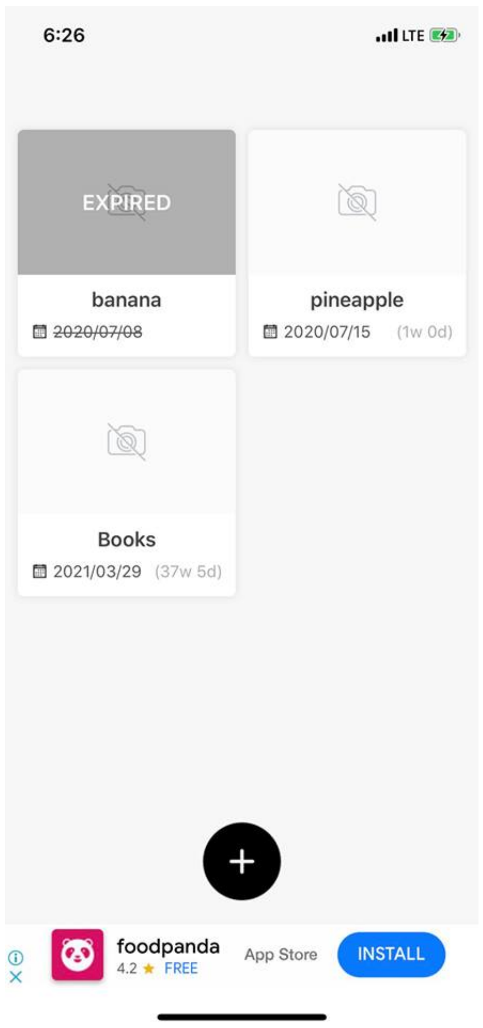


Figure 2.32: Main View of Expiry Reminder Application After Adding Products

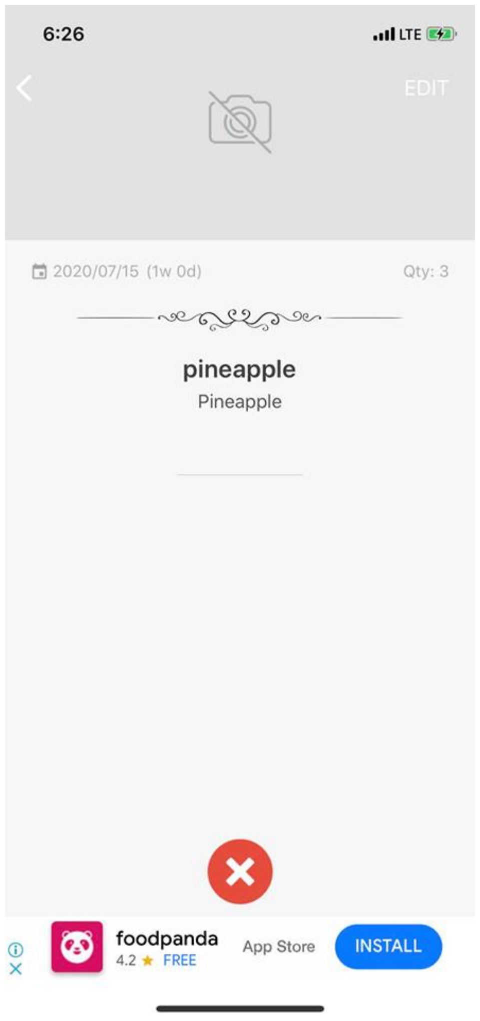


Figure 2.33: Detail View of a Product of Expiry Reminder Application

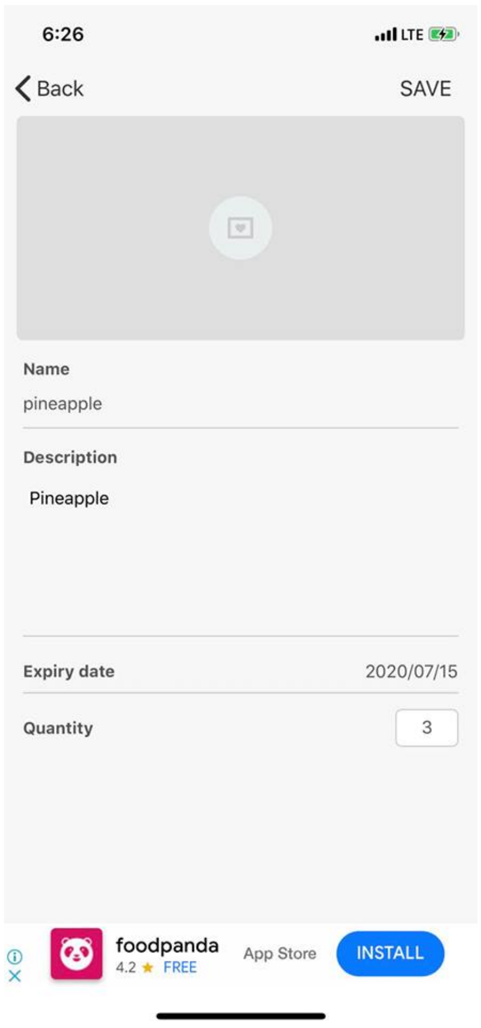


Figure 2.34: Edit Product View of Expiry Reminder Application

2.2.5.1 Main Features of Expiry Reminder

1. Add product

   Users can add products with the details of name, description, expiry date and quantity by pressing the '+' button in the bottom-centre of the main page. Users are also able to add photos for the product.

2. View product

   Users can list view products on the main page and details at the view product page as shown in Figure 2.32.

3. Edit product

   According to Figure 2.34, users can edit the name, description, expiry date and quantity of a product by clicking the edit button on the top-right side of the screen. Users are also able to change the photo of the product.

4. Delete product

Users can delete a product by pressing the red ‘x’ button in the view product page as shown in Figure 2.33.

5. Push notification

   Users will receive a push notification that reminds users about the product in the application.

2.2.5.2 The User Interface Design

1. The interface design is too simple

   When the user first opens the application, only an add button and a white screen are shown in Figure 2.29. There is no tutorial or guide to brief the user about the system. After adding the product, the application only distinguishes expired product by adding a half-transparent block that writes ‘expire’ on the product photo as shown in Figure 2.32. There is no warning for products that are going to expire. Besides, the date format is very hard to read and the font is very small.
2. The placing of add and delete buttons

   The add and delete buttons are placed at the bottom centre of the screen. In most IOS applications, the add and the delete button are mostly placed on the top right of the screen.
3. Not following the human interface design guidelines of Apple

   The application does not follow the human interface design recommended by Apple. If follow the Apple Human Interface guidelines, the add button should be a ‘+’ button on the top-right of the screen and the delete button should be a dustbin button on the top-right.

2.2.5.3 Technology and Hardware Used

1. Camera

   User use the camera to capture the photo of a product.

2.2.6 Comparison Table of The Application Studied

In this section, the basic features, user interface, advantage and disadvantage of the five applications listed above are tabulated.

2.2.6.1 Basic Features

Table 2.1: Comparisons of Basic Features of Five Studied Applications

	Expiry Wiz	**Aladdinpro-expiry reminders**	**Expired - Grocery Reminder & Alerts App**	**Before Expiry**	**Expiry Reminder**
Platform	Android	Android	Android	iOS	iOS
Basic Features:					
Add, Edit, Delete product	✓	✓	✓	✓	✓
Categorize product	✓	✓	✓	✓	-
Manage category: Add	✓	✓	-	✓	-
Manage category: Edit, Delete	✓	-	-	✓	-
Push Notification	✓	✓	✓	✓	✓
Email Notification	-	✓	-	-	-
Set reminder date & time	✓	-	✓	✓	-
Search product	-	✓	✓	✓	-
Back up & sync data	✓	✓	✓	-	-
Additional Features:					
	N/A	Multi-user support	Search product by barcode	Categorized in Category and Location	N/A
		Cross-platform available	Third party login (Google acc)	Apply Filter when viewing products	
				Untrack product	

2.2.6.2 User Interfaces

Table 2.2: Comparisons of User Interfaces of Five Studied Applications

	Expiry Wiz	**Aladdinpro-expiry reminders**	**Expired - Grocery Reminder & Alerts App**	**Before Expiry**	**Expiry Reminder**
Overall design	Neat and clean	Minimalist but complicated	Simple and Easy	Consistent but sloppy	Minimalist and monotonous
Important buttons are placed in somewhere noticeable	Add button > bottom-right of the screen	Add button > bottom-right of the screen	Add button > bottom-right of the screen	Add button > top-right of the screen	Add button > top-right of the screen
Use colour to distinguish level of severity	Red, orange, green	Red, orange, blue	Red, orange, green	Red, orange, green	N/A
Using meaningful symbol to indicate the meaning of labels	✓	-	✓	-	-
Using navigation bar	-	✓	✓	✓	-

2.2.6.3 Advantages and Disadvantages

Table 2.3: Comparisons of Advantage and Disadvantage of Five Studied Applications

	Expiry Wiz	**Aladdinpro-expiry reminders**	**Expired - Grocery Reminder & Alerts App**	**Before Expiry**	**Expiry Reminder**
Advantages:					
	Able to back up and restore using google drive	Able to use in multiple devices & cross-platform	Able to login using third party (Google acc)	Able to untrack products when do no need	Very simple interface
		Can have multiple user	Store product with barcode	Have filter functions	
	Categorized product by the severity	Email notification	Search product with barcode	Display total number by the severity at the bottom of the screen	
		Show total number of products by severity			
Disadvantages:					
	Cannot set multiple reminders	Cannot edit or delete the category	Do not sort according to expiry date	The interface looks sloppy and old-fashion	There is no back up and sync feature available
		Users need to define category by themselves	User cannot add, edit, delete category	There is no back up and sync feature available	The interface is too simple and without any tutorial
	Can set reminder date before the date of the product is added (Logical error)	Cannot login by using third party account	User cannot view product by category	User needs to add a category first before adding the product into the category	
		Cannot use email as username to login	User cannot upload photos of a product		Cannot categorized product
			The interface seems very messy because too many colours are shown in one screen		
		Cannot set reminder date	Cannot set multiple reminders	Cannot set multiple reminders	Do not have search feature

2.2.6.4 Overall Comparisons

Overall, the basic features for an expiry reminder application, which including adding, editing and deleting products are available in all five of the applications studied. Most of the applications studied allowed users to categorize each product and some of them even provide the freedom to users to manage the category. Most of the application reminds users through push notification and only Aladdin-pro provide email reminder. Most applications allow users to set the reminder date and time and search the product by name. Expired - Grocery Reminder & Alerts App allow users to search product using barcodes, which ease users in searching the product. Most of the Android platform expiry reminder application provides backup and sync functions while most the iOS platform expiry reminder application does not provide the backup and sync functions. The applications studied above uses a simple and minimalist user interface which is the current app design trend. The applications also use three different colours to represent the severity of the product.

2.3 System Development Methodologies

When developing software, the whole software developing process will be separated into several phases to ensure the whole software developing process progress smoothly and the development team can deliver quality software on time. This process is also known as the Software Development Life Cycle (SDLC). According to Innovative Architects (2017), the SDLC phases are planning, system analysis & requirements, systems design, development, integration & testing, implementation, operations & maintenance. There are many different software development methodologies in the current industry. In this section, we are going to study several famous methodologies, which include the waterfall model and agile methodology.

2.3.1 The Waterfall Model

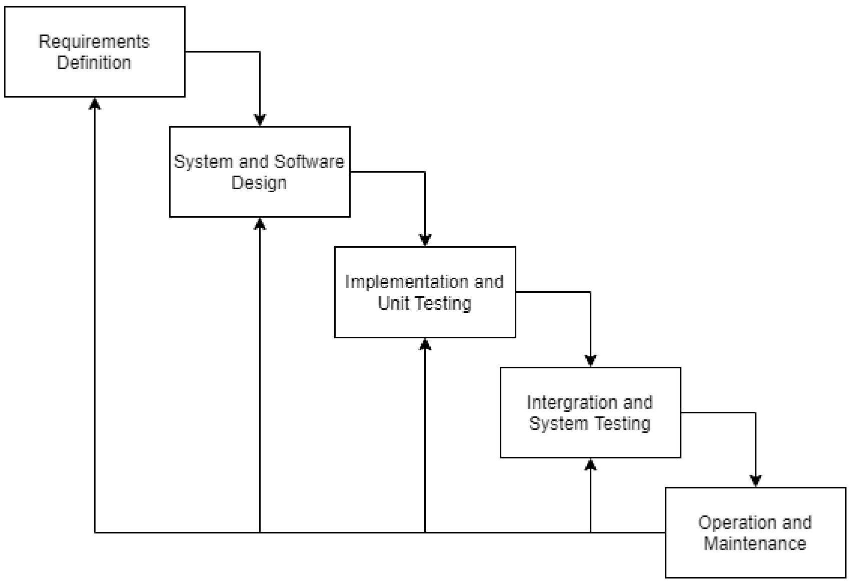

Requirements Definition
System and Software Design
Implementation and Unit Testing
Intergration and System Testing
Operation and Maintenance

Figure 2.35: Waterfall Model, adopted from ‘The waterfall model’ (Sommerville 2011, Fig. 2.1, p. 30)

The waterfall model is a plan-driven methodology, it needs to be carried out strictly according to the plan shown in Figure 2.35, where requirements are defined in the early stage of the model, then followed by system and software design, implementation and unit testing, integration and system testing then lastly operational and maintenance. The waterfall model does not change the requirement specifications and the high-level design of the project in the early stage of the development life cycle, precedence over engaging in more detailed design and implementation work.

The waterfall model normally needs a longer time in requirement gathering and project planning to identify the full set of project requirement (Shaydulin and Sybrandt, 2017). Because of the commitments made at such an early stage, it is difficult to adapt to changes in requirements. (Sommerville, 2011). This methodology is suitable when there are well-defined requirements and do not have any changes afterwards.

According to Pressman (2010), when using the waterfall model, a development team may find it difficult to follow the sequential flow that the model proposes when it comes to real projects, difficult to accommodate the natural ambiguity that emerges at the beginning of the projects, and a working version of the software can only be delivered to the customer at the end of the projects where most of the projects are completed. However, Pressman (2010) also states that the waterfall model still a useful process model when the requirements are fixed and the work is proceeding linearly.

In conclusion, the waterfall model is suitable when the project has well-defined requirements that are fixed and clear and the work need to proceed linearly while according to the project plan and time constraints set on the early stage of the project. Although this methodology ensures the quality of the projects, it does not adapt to changes due to the strict project plan and time constraints that need to be followed and deliver a workable program very slow which is close to the end of the project.

2.3.2 The Agile Methodology

Agile methodology has a characteristic of iterative development and mainly focus on interaction and communication among each other in a development team and among stakeholders. Agile methods also focus on short iterative cycles with feature planning and dynamic prioritization (Casteren, 2017). Some famous methodologies used in the market are based on agile techniques, including Extreme Programming, Rapid Application Development and Personal Extreme Prototyping.

2.3.2.1 Extreme Programming

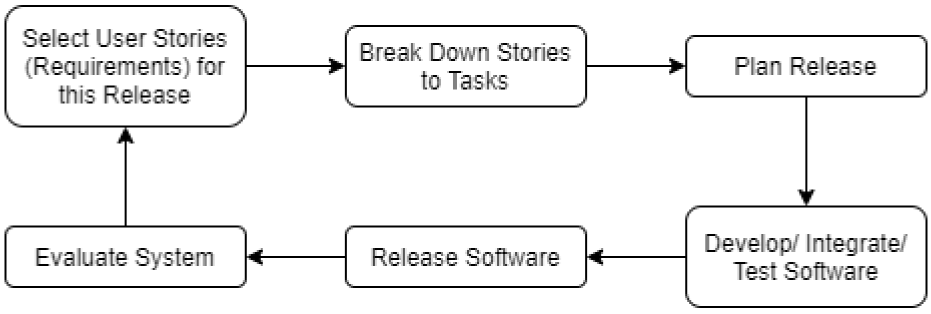

Select User Stories (Requirements) for this Release
Break Down Stories to Tasks
Plan Release
Develop/ Integrate/ Test Software
Release Software
Evaluate System

Figure 2.36: The extreme programming release cycle, adopted from ‘The extreme programming release cycle’ (Sommerville 2011, Fig. 3.3, p. 65)

Extreme Programming (XP) is one of the agile development methods widely used in the industry currently. In extreme prototyping, programmers will work in pairs and develop each task. The system will be released with a new feature in a very short time after the previous release. The whole process of a release is shown in Figure 2.36. The model starts with selecting user stories or user requirements for the current release, then the user breaks down the stories in many small tasks and plans for the release. After that, the requirements selected is developed, integrate and test to make sure no errors occur in the whole software. When all tests are successfully executed, the development team will release the software and evaluate it based on the feedback from the client.

When using this model, customers and developers need to work together to decide the requirements for the next release. Customers will engage with the development team continuously and the customer representative needs to define acceptance tests for the system. This model allows changing in requirements and each change is addressed by a regular system release to the customers. Code will be refactored to avoid code degeneration and improves code quality.

According to Wells (1999), XP is applicable when requirements change dynamically, a high project risk due to using new technology in specified project time, small development team, and automated unit tests and functional tests. It solves the main problem of the waterfall models, where changes are hard to make caused by its plan-driven characteristics. However, XP focuses more on the code than the design which may result in bad design software that is hard to be sold in the software market. Additionally, XP is normally lacked defect documentation which may lead to the existence of bugs that are similar in the future. This methodology also not suitable when the programmers are separated geographically due to its pair programming characteristic.

2.3.2.2 Personal Extreme Prototyping (PXP)

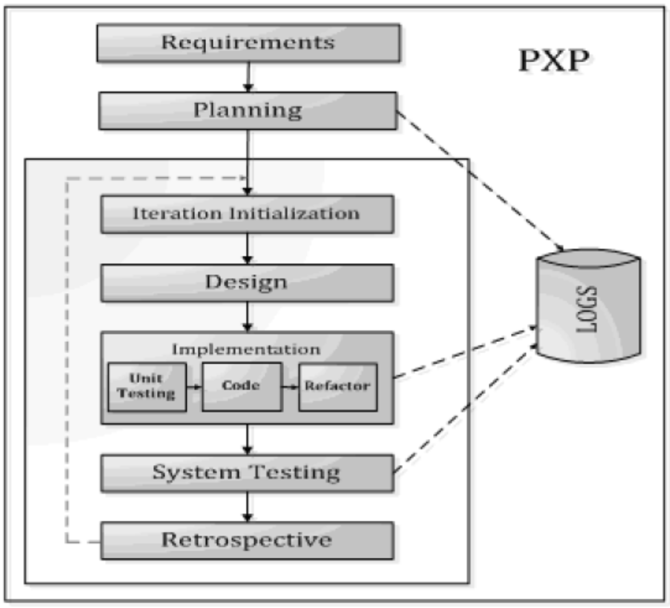


Figure 2.37: PXP Process Phase, adopted from ‘PXP process phase’ (Dzhurov, Krasteva and Ilieva, 2009, Figure 1, p. 255)

PXP is a lightweight agile framework that is used for the one-man team by combining XP and Personal System Processes (PSP). This methodology aims at improving PSP by keeping the basic principles of PSP but reducing the efforts in documentation and maintenance. Hence, the methodology can improve programmer performance and the delivery interval can be shortened if the developer is autonomous. The disadvantage of this methodology is it is heavily relying on the automation of most of the developer works.

PXP methodology is based on the 6 principles listed below:

- PXP relies on a self-disciplined developer who is accountable for adopting and executing the PXP cycle.
- Developers need to measure, monitor and analyze their work regularly.
- Developers should learn from the performance difference and target to enhance the process using the collected project data
- Continuous testing is essential for PXP methodology
- Bugs and defects should be fixed early in the development cycle because the cost is lower.
- Developers should aim to automate as much of their everyday work as possible.

There are several phases in PXP methodology as shown in Figure 2.37. The developer that implements PXP methodology starts at requirement gathering where the client and other stakeholders address functional and non-functional requirements to produce a requirements document. Each requirement is then broken down into smaller tasks in the planning process, where user stories are generated based on the user requirement and the approximate time to produce and user storey is measured. Significant design decisions, such as the programming language to be used, the development framework, the application model, and several others, will be made during this phase.

According to Dzhurov, Krasteva and Ilieva (2009), the iteration initialization process marks the start of each iteration. Each iteration will have a task assigned to it, and the timeframe of each iteration will range from one to three weeks, based on project scope. Each iteration release will be either a release candidate or a released version of the product. After initializing the iteration, the developer will move on to the design phase where modules or classes to be implemented are created. During the implementation phase, the code is developed. In this phase, the developer must maintain three baselines during this phase: development, refactoring, and output, says Agarwa and Umphress (2008). Upon completion in development, the code is refactored after completion, and unit, integration, and acceptance testing are performed before moving to the production baseline for release. A retrospective is done at the end of the iteration to determine whether the iteration was completed on schedule or not. If this is not the case, the causes of project delays should be investigated, and a solution to address the situation should be released as soon as possible to avoid project failure. A new iteration will be started, and the same process will be repeated until all of the tasks have been completed.

Based on the research from Asri *et al.*, (2018), PXP has been effective in producing a device in a reasonably short time. By applying PXP, it becomes easier to determine the length of the development cycle and helps the designer to develop code directed by user stories. To suit in an alone programmer scenario, PXP retains and refines the principles of XP, which are simplicity, communication, and feedback. However, in the PXP methods, a developer must strike a balance between the "too heavy" and "too light" methodologies such that the methodology injects the appropriate level of rigour into the situation without overburdening the project.

2.3.3 Overall Comparisons

The waterfall model normally needs a long time for requirement gathering and project planning to identify the full set of project requirement. According to Andrei *et al.* (2019), the majority of the projects that use waterfall are small projects that have teams of less than 10 people and a project duration that is fewer than 6 months. Andrei *et al.* (2019) also state that waterfall is suitable for small projects by small teams when there are well-defined requirements. Undeniable, waterfall methodology is good when the requirement is well-defined, and no changes will be made.

Agile Methodology, on the other hand, is good when it comes to a new programming environment, uncertain requirements and a small team. According to Shaydulin and Sybrandt (2017), agile methodology is applicable in small teams' problems while for large and extensive projects, the over emphasizes many person-to-person interactions do not work out well. Andrei *et al.* (2019) state that agile is more flexible and preferred when continuous feedback is important.

In conclusion, not a single methodology can address all important aspects of the software design process. The methodology should be chosen according to the project nature and the working style of the development team and customers to produce high-quality software on time.

2.4 Technology Used

Nowadays, there are many different technologies that are available in the industry that allows a programmer to code for mobile application. There are three types of mobile application, native apps, web apps and hybrid apps.

A native app is an application that is developed for a single OS platform. For example, an iOS native app cannot be used in an Android platform. Each of them is developed using their own language. For instance, iOS application uses Objective-C or swift to develop, Android application uses Java or Kotlin to develop, Windows Phone use Net to develop. Native apps are good on performance at the same time ensuring a good user experience. However, these kinds of mobile application require a higher cost compared to other types of application. If there is a need to create the same application on a different platform, support and maintenance costs will be very high because it is separated for each platform.

A hybrid app is a web application disguised in a native wrapper that is built for multi-platform. A hybrid app can be built using different kinds of tools and framework, which include React Native, Flutter, Xamarin and many more. This kind of application is relatively fast and easy to develop and the cost to build and maintain the application is low. There are many APIs available for developers to use. However, hybrid apps are generally lacking in performance if compared to native apps. Plus, the design will not look the same way on different platforms.

A web app is a web application that uses a browser to run on a mobile device. Not like a native app and hybrid app, the web app does not have to download and install on a mobile device. One of the benefits of a web app is it does not need to customize to fit in a platform, and this cuts down the development costs. Moreover, users do not need to download the application; hence it saves up more space on the mobile device. The maintenance of a web app is easier than a native app. The development team only need to push the update over the web, and users will be able to use the latest version on their mobile device through a browser. However, some functionalities may work on a certain browser, while the same functionalities may not work on the other browser. This affects the users' experiences when the functionalities do not work. Plus, a web app will not work completely if the device does not connect to an internet connection.

In this section, we will look at the framework available in the hybrid app.

2.4.1 Framework for Creating a Hybrid App

There are several different frameworks available in the industry to allow developers to create a hybrid app effortlessly. Xamarin, Flutter and React Native will be the framework to be studied in this section.

2.4.1.1 Xamarin

Xamarin is a commercial cross-platform development tool owned by Microsoft. It is built based on the .Net Framework and uses C# to create hybrid or cross-platform mobile applications. Xamarin projects need to be run on a Windows computer with Visual Studio and Xamarin installed. Xamarin developers use Visual Studio to code and debug the application.

Xamarin is known for creating applications that have almost native-like performance levels. Plus, Xamarin allows developers to use Xamarin, iOS, and Android to manually customized the applications if needed. However, there will be a delay when a new feature of a platform is coming out. This may cause issues to the app. Besides, Xamarin apps have larger app sizes because of the libraries used to convert C# calls into native calls. It can be up to 5 MB for releases build and 20 MB for debug builds, which is often larger than a native application. A developer needs some knowledge of languages used to write some platform-specific code when the developer wishes to customize the UI in the application. This makes the Xamarin app not an optimal tool for an application that has a complex interface like mobile games. A developer will need to spend a lot of time writing the platform-specific code, making it not an effective tool to be used.

2.4.1.2 Flutter

Flutter is a cross-platform mobile app development tool produced by Google and it is open source. Flutter uses Dart as its programming language and Dart is a modern language created by Google in 2011 (Thomas, 2019).

Flutter is supported by Android Studio and Visual Studio Code. Flutter provides a Software Development Kit (SDK) that help developers in compiling the code into native machine code and a framework that contains a series of reusable UI elements like buttons, text inputs, sliders and many more for developers to create stunning applications. Flutter also allows developers to customize the UI element and improve the UI of the application. In addition, Flutter also provides a hot reload to allow developers to change the codes and see the real-time results. This increase the productivity of developers in debugging the application. However, Flutter apps are normally large in size due to the libraries and packages used by the developers. Plus, it does need a third-party package to use the native APIs. The programming language of Flutter, Dart, is not widely used. Hence, there is a smaller community available online. A developer will also need to take more time to get familiar with Dart if the developer is unfamiliar with it.

2.4.1.3 React Native

React Native is an open-source development framework for the cross-platform mobile app developed by Facebook. It uses JavaScript as its programming language, which is familiar to many experienced web application developers.

React Native contains many native UI components for both iOS and Android Platforms, making the application feels like a native application by using JavaScript only. React Native framework also can access native functionalities, such as a camera. React Native also provide a hot reload that allows developers to change the app in real-time. React Native framework also allows user to customize the application with platform-specific code to further enhance the UX when using the application on a different platform. Unfortunately, React Native app developers struggle when building complex apps with composite animation and transitions. React Native also demands extensive platform-specific code to ensure the UI work seamlessly on different platforms.

2.4.1.4 Overall Comparisons

Xamarin is really convenient when it comes to reusing the code and creating a near-native application. However, the support is limited and is not friendly for the developer that is new to the framework. The availability of React Native is higher than Xamarin due to the popularity of the frameworks (Avdic, 2018). The market of React Native framework is increasing while Xamarin is decreasing. On the other hand, Flutter and React Native are open-source frameworks. The language Dart used by Flutter is a new language. Hence, it has a smaller community because it is not widely used yet (Fentaw, 2020). While for React Native, JavaScript is used. It is easier to pick up as if the developer has a strong web development background. According to the study from Fentaw (2020), the overall CPU, GPU and Memory performance is nearly equal for both Flutter and React Native frameworks.

In a nutshell, Flutter is more suitable when the project has sufficient resource and budget. React Native framework is recommended when the project is small and simple. Xamarin is preferred if the project is big and the budget and time given are sufficient.

Table 2.4: Comparisons Between Xamarin, Flutter and React Native

	Xamarin	**Flutter**	**React Native**
License	Commercial, Free for individuals and small companies	Open-Source	Open-Source
Language	C#	Dart	JavaScript
Tools	Code, Run and Debug in Visual Studio	Code in Android Studio/ IntelliJ/ Visual Studio Code, Run and Debug in DevTools	Code in Atom, Nuclide, Visual Studio Code, Run and Debug in Expo, Redux
Created By	Microsoft	Google	Facebook
Performance	Near-Native app	Close to Native code	Near-Native app
Popularity	Large userbase before 2017	Been raising recently	Surpassing Xamarin in 2017
Code Reuse	Reusable	Reusable on UI widget tree	Partially reusable
Support	Limited support	Robustly growing community base	Strong community base

2.5 Deep Learning

Deep learning is a part of machine learning where the algorithms, such as artificial neural networks, are influenced human brains' structure and function. According to Kim (2019), deep learning is a form of machine learning that allows machines to learn from experience and comprehend the reality through the use of a hierarchy of concepts. Recently, deep learning is widely used by big companies in helping them to perform daily tasks. Fraud detection, virtual assistance, investment modelling and face recognition system are examples of the deep learning application.

There are many types of neural network algorithm in deep learning, each with a different structure and functions. In this section, three types of neural networks, Artificial Neural Network, Convolutional Neural Network, Recurrent Neural Network and Convolutional Recurrent Neural Networks will be discussed.

2.5.1 Artificial Neural Network (ANN)

Psychologist Frank Rosenblatt proposes the ideas of ANN in 1958, where it intends to model how human brains processed visual data to learn how to recognize objects. Same as the human brain, neurons are the fundamental unit of an ANN. Neural Network is made up by creating connections between processing elements, and each processing element is represented as a neuron. A neuron will receive many inputs and produce an output based on the weighting system and its bias. According to Uhrig (1995), ANN normally contains three or more tightly connected layers, which includes an input layer, an output layer and one or more hidden layers. The structure of ANN is shown in the figure below.

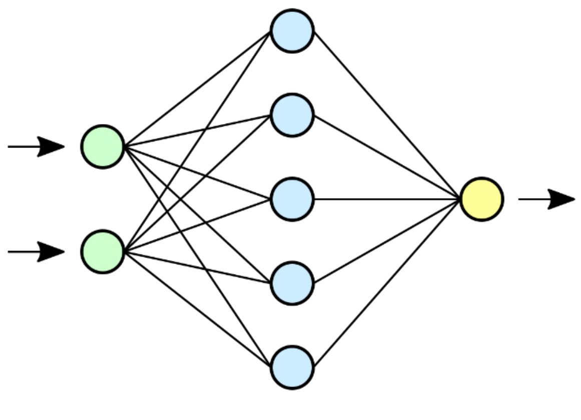


Figure 2.38: Structure of ANN

2.5.2 Convolutional Neural Network (CNN)

CNN is a type of feed-forward neural networks based on the idea of ANN. The difference between CNN and ANN shows in the hidden layer. According to Yu *et al.* (2021), the hidden layer of a CNN comprises three layers: convolutional layer, sub-sampling layer or pooling layer, and fully connected layer. The convolutional layer will use filters to perform feature extraction. Each filter will contain a bias, and the activation function like ReLU will be applied. A pooling layer is used to down-sampling and reduce the dimension of the feature map. A max-pooling layer can take the highest value and assist in extracting low-level features such as edges, while an average pooling layer is good in extracting smooth features by taking the average value. After performing several rounds of features extracting and down-sampling, a fully connected layer is used to flatten the features into a one-dimensional layer. Lastly, the features are classified at the output layer by using Softmax as the activation function.

Each CNN layer (Convolutional layers + Pooling layers) learns something different in each layer. Basic feature detection filters, such as edges, are learned in the first few layers. In the middle layers, feature detection filters such as eyes and mouths are learned. The feature detection filters in the final layers learn and identify the entire object in various shapes and positions.

Many pre-trained networks, including ResNet, GoogleNet, EfficientNet, uses CNN as their architecture, and these pre-trained networks are performing well in classifying images. According to Kranthi Kumar *et al.* (2021), CNN can be used in image classification, feature extraction and recognition. Its simple architecture allows us to use its simplicity to create an efficient neural network with high accuracy or combine it with other neural networks to become a modular neural network that can conduct a more complex task.

2.5.3 Recurrent Neural Networks (RNN)

Unlike feed-forward neural networks, which have separate inputs and outputs, RNN uses the output from the previous step as input to the current step. According to Navamani (2019), RNN handles sequential data, including text encoding, speech recognition and DNA sequences very well. However, RNN faces issues, such as vanishing gradient problems, which occur when the network receives little or no training, and exploding gradient problems, which occur when the gradient becomes extremely high due to backpropagation, both of which can result in poor performance.

To overcome vanishing gradient problems, long short-term memory units (LSTMs) and gated recurrent units (GRU) are proposed.

According to Yu *et al.* (2019), RNN consists of standard recurrent cells which can be sigma cells or tanh cells. These standard recurrent cells cannot handle long term dependencies and become more difficult to learn when the gap between the inputs grows. To solve the problem of long-term dependencies in RNN, LSTM cell is introduced. LSTM cell has a higher remembering capacity than the standard recurrent cell, and it is achieved by a forget gate. The forget gate will determine what information will be discarded from the cell states, resulting in an increase in memory space.

The advantages of RNN are it remembers information through time and this characteristic makes it useful to predict time-series data. However, training RNN is relatively difficult. It requires a huge number of computational resources especially when using LSTM cells due to its capability in remembering and handling long short-term memory.

2.5.4 Overall Comparisons

ANN is a feedforward neural network that is having the simplest structure comparing to CNN and RNN. It is built up of three layers: an input layer, an output layer, and one or more hidden layer. However, its simple structure limits its performance on complicated problems.

CNN is also a feedforward neural network that consists of an input layer, an output layer and several hidden layers. The hidden layers in CNN are convolutional layers and pooling layers. Convolutional layers are to apply filters for feature extraction, and pooling layers is to down sampling the image. After several rounds of feature extraction, the features extracted will be flattened by a fully connected layer to be sent to the output layer for classification. CNN is a more powerful neural network than ANN in performing tasks like facial recognition, image classification and object detection. However, CNN requires large training data, which is costly and hard to obtain.

RNN has a different structure compared to ANN and CNN. RNN uses recurrent cells that save the output of the previous step and feed it as input. RNN is more complex than ANN and CNN and is outperformed than the other two neural networks in time series prediction. The most concern problems in RNN is the gradient vanishing and gradient exploding problem. However, the problem can be solved by using LSTM cells and GRU cells. Another problem faced when training RNN is it is computationally expensive than CNN and ANN. The recurrent cells will store more and more memory after each step, and the RAM, CPU and GPU used to train the model will be increasing after each step.

In conclusion, no one model fits all. Each of the neural networks has its role in helping human being to solve their everyday task, but choosing the best neural network that fits the situation is very crucial because it will affect the performance of the neural network. ANN is simple and easy to train when it comes to simple problems like predicting tabular data. CNN has higher accuracy when predicting image data, while RNN is made for time series and sequence data.

Table 2.5: Comparisons for ANN, CNN and RNN

	ANN	**CNN**	**RNN**
Structure	An input layer, one or many hidden layers and an output layer	An input layer, several feature extraction layers (convolutional layers + pooling layers) and an output layer	An input layer, several recurrent cells, an output layer
Data Type	Suitable for tabular data	Suitable for image data	Suitable for time series and sequence data
Advantages	Fault tolerance, simple structure	High accuracy in solving image recognition problem	Remembers every information and can predict time-series data
Disadvantages	Unexplained behaviour of the network	Requires a large amount of data to train	Gradient vanishing and exploding problems, computationally expensive to train

2.5.5 Application of Deep Learning in Actual Problems

Deep learning mimics the structure of human brains to allow machines to interpret data in a manner close to that of humans. In contrast to machine learning, which learns from data supplied by humans, deep learning does not require an individual to tell it what to do with the data. Bernard (2018) has listed out 10 applications of Deep Learning in different industries, which are:

1. Customer service automated chatbox
2. Translation from image
3. Adding colour to black and white images and videos
4. Language dialects recognition
5. Autonomous vehicles
6. Computer vision such as image classification, object detection, image restoration, image segmentation, handwriting recognition
7. Text generation with proper spelling, grammar and style
8. Image caption generation
9. News aggregator based on sentiment
10. Deep learning robots like housekeeping robots

In the research from Lake *et al.* (2015), a comparison between humans and Deep Neural Network (DNN) is made, and DNN is seen to model human typical ratings on a range of products. Apart from that, Geirhos *et al.* (2017) also made a thorough comparison between human and deep learning models, where humans and DNNs are needed to classify degraded images, which includes grayscale images, images with reduced contrast, images with uniform white noise and image with distortion. The result shows that DNNs outperformed human for non-distorted and coloured images. Although DNNs outperformed or having the same performance as human observers in most of the experiment, DNNs' accuracy significantly dropped in the white noise and distorted image experiment. The author then concludes that DNNs are more fragile to adaptive noise when training on normal images. However, there is a technique called Data Augmentation, where noise is added to the image when training on the DNN, which is widely used to prevent CNNs from overfitting. Nazaré *et al.* (2018) has compared the accuracy of various DNNs training on original image, noisy image and restored image. The research then concludes that training DNNs with noisy images put the networks at an advantage in dealing with different levels of noisy images by improving the resilience of the network to noises.

While deep learning cannot yet outperform humans in certain situations, such as classifying noisy images, it does work well in the majority of them. As a result, the implementation of deep learning to real-world problems can be accurate much of the time with limited human supervision.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this chapter, the selected software development methodology, work plan and Gantt chart will be discussed.

3.2 System Development Methodology

The system development methodology chosen is Personal Extreme Prototyping (PXP). PXP is a software development methodology that is made perfectly for a one-man's team. In this section, the workflow of the PXP framework will be discussed.

3.2.1 Personal Extreme Prototyping (PXP)

PXP is a software development methodology that uses the basic PSP principle and the advantage of extreme prototyping to ensure it is agile and lightweight. Same as many other software development methodologies, PXP starts with a requirement gathering where user requirements are gathered and a document is created. The user requirement in this project is gathered through the survey questionnaire to the targeted user and study existing similar application. Once the requirements are gathered, functional requirements and non-functional requirements will be drafted out.

Moving on to the planning phase, a list of tasks is created based on the requirements gathered. Each task is then split into smaller tasks, and the software developer calculates the time needed to complete each task. During this process, the most crucial design choices, such as the programming language used, the software architecture, tools for developing the system, and many more, will be made. Each task in PXP needs to go through iterations of initialization, design, implementation, testing, and retrospective. Each iteration's output will be logged for future reference.

On each iteration starts, there will be an iteration initialization phase. At this phase, task selection is performed to select the most crucial task and the dead for completing the iteration is set. Each iteration will usually last between one to three weeks depending on the complexity of the project. After iteration initialization, module and classes will be modelled in the design phase to meet user requirement.

In the implementation phase, the objects and classes defined in the design phase will be applied into the actual code. The code would be subjected to unit testing, code generation (versioning), and code refactoring to ensure that it is both functional and consistent. The code must compile without errors and pass all unit tests to exit the implementation phase. System testing will be conducted at last before delivering the output for the iteration.

Retrospective will be conducted at last for the iteration. Here is where the developer identifies and analyses problems faced during the iteration, tracks down its root cause and comes out with a solution to address the problem. The developer should also evaluate whether the iteration is completed on time or not. A new iteration will begin until all of the user requirements are fulfilled.

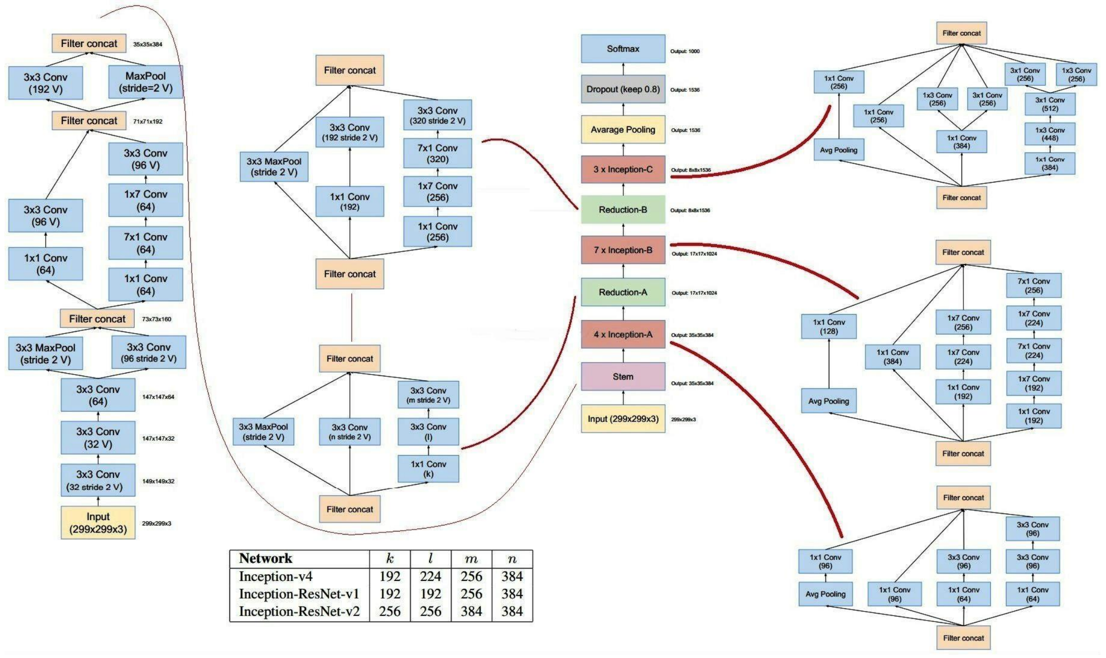
3.3 Deep Neural Network

In this project, a DNN will be used to recognize expiry dates on the product. The architecture of the DNN and methodology on training the DNN will be discussed in detail in this section.

3.3.1 Deep Neural Network Architecture

To recognize the expiry date on the product, a DNN with good image feature extraction should be used. In this case, CNN is selected because of its high precision in image recognition and its window sliding technique for extracting features from images. The CNN architecture chosen for the application is Inception ResNet V2.

Inception ResNet V2 is a hybrid Inception that has improved performance. One of the interesting architectures in Inception-ResNet-V2 is the residual connections. According to Szegedy *et al.* (2017), residual connections will greatly increase the architecture's training speed. Residual Connection was introduced by He *et al.* in 2016, and the author proved that by utilizing additive merging of signals, the model will be having good performance in image recognition and object detection. The figure below shows the full architecture of the Inception ResNet V2.



Network	k	l	m	n
Inception-v4	192	224	256	384
Inception-ResNet-v1	192	192	256	384
Inception-ResNet-v2	256	256	384	384

Figure 3.1: Architecture of Inception ResNet V2, adopted from (Lynnandwei, 2016)

3.3.2 Training Deep Neural Network

3.3.2.1 Generate Dataset

A synthetic dataset is created to train the Inception ResNet V2 to recognize the expiry dates. The synthetic dataset aims to mimic real image that could be captured on a real product. A python library called Text Recognition Data Generator (TRDG) created by Belval (2019) aims to create an image dataset to train the text recognition model. The library helps developers model real-world situations that may be captured by humans.

To improve the model's accuracy, various noisy backgrounds are used for the image produced by TRDG to simulate actual product packaging, which may contain several patterns or be in different colours. Developers can also select specific fonts that meet the situation. For example, for identifying product name at packaging, exaggerate and fancy fonts should be used to generate the dataset. In this case, various digital fonts and dotted fonts are used to mimic the fonts of expiry dates on products are used to generate the dataset. Besides, random blurred, random distorted and random rotation is applied to ensure the diversity of the dataset generated so that the deep learning model can identify slanted and blurry images in the future.

All dates from March 2021 to December 2025 in a specific format are created, and the dataset contains a total of 5359 classes. There are 696670 images in total, with 130 images in each class and each image is 80 pixels in height, 215 pixels in width. Different symbols and spaces are positioned between day, month, and year in each of the date formats created in the dataset to simulate different types of data used in actual product packaging. Only a few common date formats used on product packages are generated due to limited training time and computational resources. The table below shows the format of the dates created for the dataset.

Table 3.1: Date Format Included in the Dataset

Date Format	**Format Included**	**Example**
ddMMyy	- ddMMyy	- 011221
	- dd MM yy	- 01 12 21
	- dd MM yy	- 01 12 21
	- dd-MM-yy	- 01-12-21
	- dd - MM - yy	- 01 - 12 - 21
	- dd/MM/yy	- 01/12/21
	- dd / MM / yy	- 01 / 12 / 21
	- dd.MM.yy	- 01.12.21
	- dd . MM . yy	- 01 . 12 . 21
ddMMyyyy	- ddMMyyyy	- 01122021
	- dd MM yyyy	- 01 12 2021
	- dd MM yyyy	- 01 12 2021
	- dd-MM-yyyy	- 01-12-2021
	- dd - MM - yyyy	- 01 - 12 - 2021
	- dd/MM/yyyy	- 01/12/2021
	- dd / MM / yyyy	- 01 / 12 / 2021
	- dd.MM.yyyy	- 01.12.2021
	- dd . MM . yyyy	- 01 . 12 . 2021
yyyyMMdd	- yyyyMMdd	- 20211201
	- yyyy MM dd	- 2021 12 01
	- yyyy MM dd	- 2021 12 01
	- yyyy-MM-dd	- 2021-12-01
	- yyyy - MM - dd	- 2021 - 12 - 01
	- yyyy/MM/dd	- 2021/12/01
	- yyyy / MM / dd	- 2021 / 12 / 01
	- yyyy.MM.dd	- 2021.12.01
	- yyyy . MM . dd	- 2021 . 12 . 01
MMyyyy	- MMyyyy	- Dec2021
	- MM yyyy	- Dec 2021

3.3.2.2 Hyperparameter Tunning

To train a deep learning model with improved efficiency, hyperparameter tuning is critical and cannot be overlooked. There are two methods for tuning hyperparameters. The first is to tune the hyperparameter manually, and the second is to computerize the tuning process to find the best hyperparameter. At the start of the experiment, hyperparameters are manually tuned to see how each parameter interacts with the model and easily generate a set of suitable hyperparameters for later hyperparameter tuning with Grid Search.

The hyperparameter to be tuned is the batch size, the learning rate and the dropout rate. Batch size is the number of training example that is fed to the model in an iteration. The larger the batch size, the more data will be fed in on iteration and the more computing resources will be needed. The learning rate determines how much the model can adjust to the expected error each time the model weights are changed. The model learns more as its learning rate increases, and vice versa. However, the learning rate cannot be too high or too small since a large learning rate can result in overfitting, while a small learning rate will prolong the training process or cause the model to become stuck at the same accuracy. Dropout is a method often used in neural network training. It is used to regularise the network to minimise overfitting and enhance deep neural network generalisation by dropping nodes from the network. The dropout rate represents the probability of nodes being dropped. A dropout rate of 1.0 indicates that no dropout occurred between the layers, while a dropout rate of 0.0 indicates that no output was transmitted to the next layers.

After selecting a range of hyperparameter from manual hyperparameter tuning to filter most of the undesirable range, Grid Search is used to identify the best hyperparameter among the others. It will automatically train the model with ranges of hyperparameter using trial and error methods and return the best hyperparameter that produces the best accuracy. After finding the best hyperparameter for the model, the training process can proceed with the best hyperparameter obtained from the Grid Search.

3.3.2.3 Data Augmentation

Data augmentation is a commonly used approach for training deep learning image recognition. Data augmentation aims to extend the original dataset’s diversity by generating synthetic data from the training dataset. Many data augmentation strategies are available, including random rotation, random contrast, random translation, random zoom, random brightness, etc. Dates in real life might be taken at various angles and contrast levels. Hence, random rotation and random contrast are used for the data augmentation to improve deep learning performance when handling the noise mentioned above.

3.3.2.4 Prevent Overfitting

Overfitting is a typical issue that most Artificial Intelligence researchers will encounter while training an algorithm. Overfitting happens when a learning algorithm predicts well on the testing dataset but incorrectly on a new dataset. Overfitting is induced mainly by an overly complex learning algorithm and the distribution of too many features across a small collection of training samples. Figure 3.2 shows the learning curve of a high bias (underfit) and high variance (overfit) deep learning model. We can see from the learning curve that when a model is overfitted, the validation loss is much higher than the training loss. It implies that the model has a high variance and will not perform as well as it did in the trained data while predicting new, unknown data.

Inception ResNet V2 is a complex deep neural network with several hidden layers that are used to capture the feature. So, the model will be easily overfitted when the training data is too small. Hence, to prevent overfitting, the number of images for each class should be sufficient for the complexity of the model and features. This is because the training data has lots of features, and a complex model is needed to identify the image correctly. So, a total of 100 images per class is used to train the deep learning model to prevent overfitting.

Furthermore, the dropout strategy is used to avoid overfitting. The model complex-ity can be minimized by randomly removing nodes at each iteration so that not all neurons are trained during the iteration. This ensures that the network is not over-trained, resulting in the network is not picking up noise rather than features from the image dataset, which causing the deep learning algorithm to have a high variance.

Apart from that, early stopping is used to avoid overfitting. When the deep learn-ing algorithm learns all of the features of each class, it will begin to overfit by captur-ing unwanted noise as features. Early stopping can be used to track the validation loss at each epoch and stop the model training if the validation loss begins to rise, which indicates overfitting.

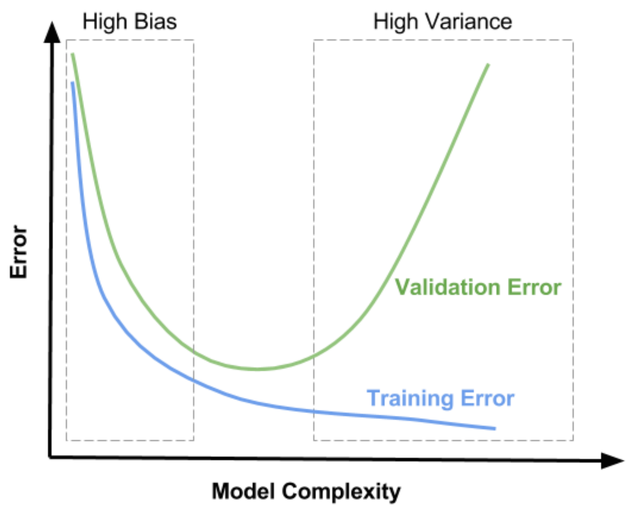


Figure 3.2: The learning curve for high bias and high variance model, adopted from (Mallick, 2017)

3.3.2.5 Training, Validation and Testing

The dataset should be divided into three parts to train a deep learning model: the training dataset, the validation dataset, and the testing dataset. To train Inception ResNet V2, the training dataset, validation dataset, and testing dataset are used in the following proportions: 70:15:15, which means that 70 per cent of the dataset is used for training, 15 per cent for validation, and 15 per cent for testing. The training dataset will be used to train the deep learning algorithm to learn the features of each class, while the validation dataset will be used to evaluate model performance at each epoch. After training, the model will be tested on unknown data to evaluate its accuracy. The rationale for selecting such a ratio is that the dataset is large, with a total of 696670 images. The training dataset contain 487669 training instances, while the validation and testing datasets will each contain 104500 examples. Also, with such a ratio, the validation and testing instances would be more than adequate.

3.4 Work Plan

In this section, a Work Breakdown Structure (WBS) and Gantt Chart is created to further clarify the work plan for this project.

3.4.1 Work Breakdown Structure (WBS)

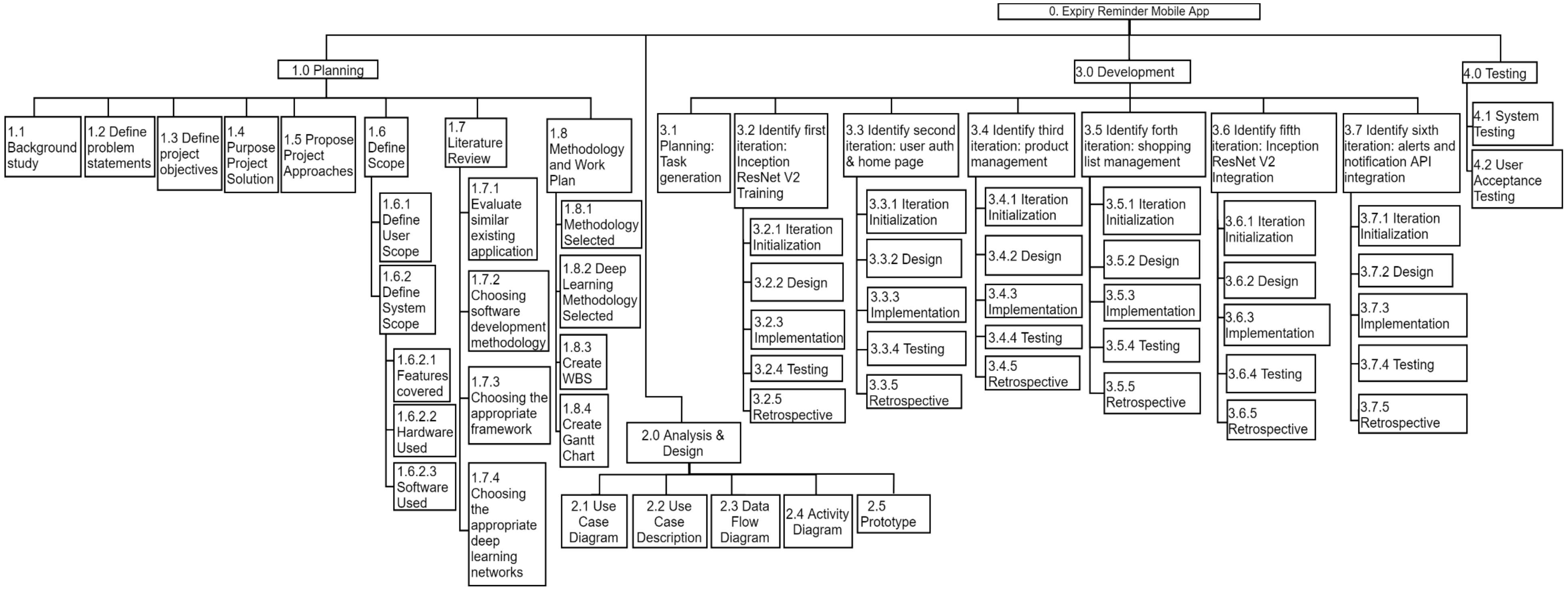
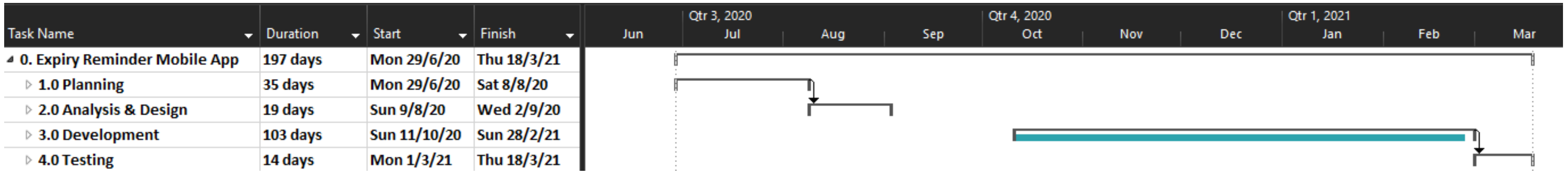

0. Expiry Reminder Mobile App
1.0 Planning
1.1 Background study
1.2 Define problem statements
1.3 Define project objectives
1.4 Purpose Project Solution
1.5 Propose Project Approaches
1.6 Define Scope
1.6.1 Define User Scope
1.6.2 Define System Scope
1.6.2.1 Features covered
1.6.2.2 Hardware Used
1.6.2.3 Software Used
1.7 Literature Review
1.7.1 Evaluate similar existing application
1.7.2 Choosing software development methodology
1.7.3 Choosing the appropriate framework
1.7.4 Choosing the appropriate deep learning networks
1.8 Methodology and Work Plan
1.8.1 Methodology Selected
1.8.2 Deep Learning Methodology Selected
1.8.3 Create WBS
1.8.4 Create Gantt Chart
2.0 Analysis & Design
2.1 Use Case Diagram
2.2 Use Case Description
2.3 Data Flow Diagram
2.4 Activity Diagram
2.5 Prototype
3.0 Development
3.1 Planning: Task generation
3.2 Identify first iteration: Inception ResNet V2 Training
3.2.1 Iteration Initialization
3.2.2 Design
3.2.3 Implementation
3.2.4 Testing
3.2.5 Retrospective
3.3 Identify second iteration: user auth & home page
3.3.1 Iteration Initialization
3.3.2 Design
3.3.3 Implementation
3.3.4 Testing
3.3.5 Retrospective
3.4 Identify third iteration: product management
3.4.1 Iteration Initialization
3.4.2 Design
3.4.3 Implementation
3.4.4 Testing
3.4.5 Retrospective
3.5 Identify forth iteration: shopping list management
3.5.1 Iteration Initialization
3.5.2 Design
3.5.3 Implementation
3.5.4 Testing
3.5.5 Retrospective
3.6 Identify fifth iteration: Inception ResNet V2 Integration
3.6.1 Iteration Initialization
3.6.2 Design
3.6.3 Implementation
3.6.4 Testing
3.6.5 Retrospective
3.7 Identify sixth iteration: alerts and notification API integration
3.7.1 Iteration Initialization
3.7.2 Design
3.7.3 Implementation
3.7.4 Testing
3.7.5 Retrospective
4.0 Testing
4.1 System Testing
4.2 User Acceptance Testing

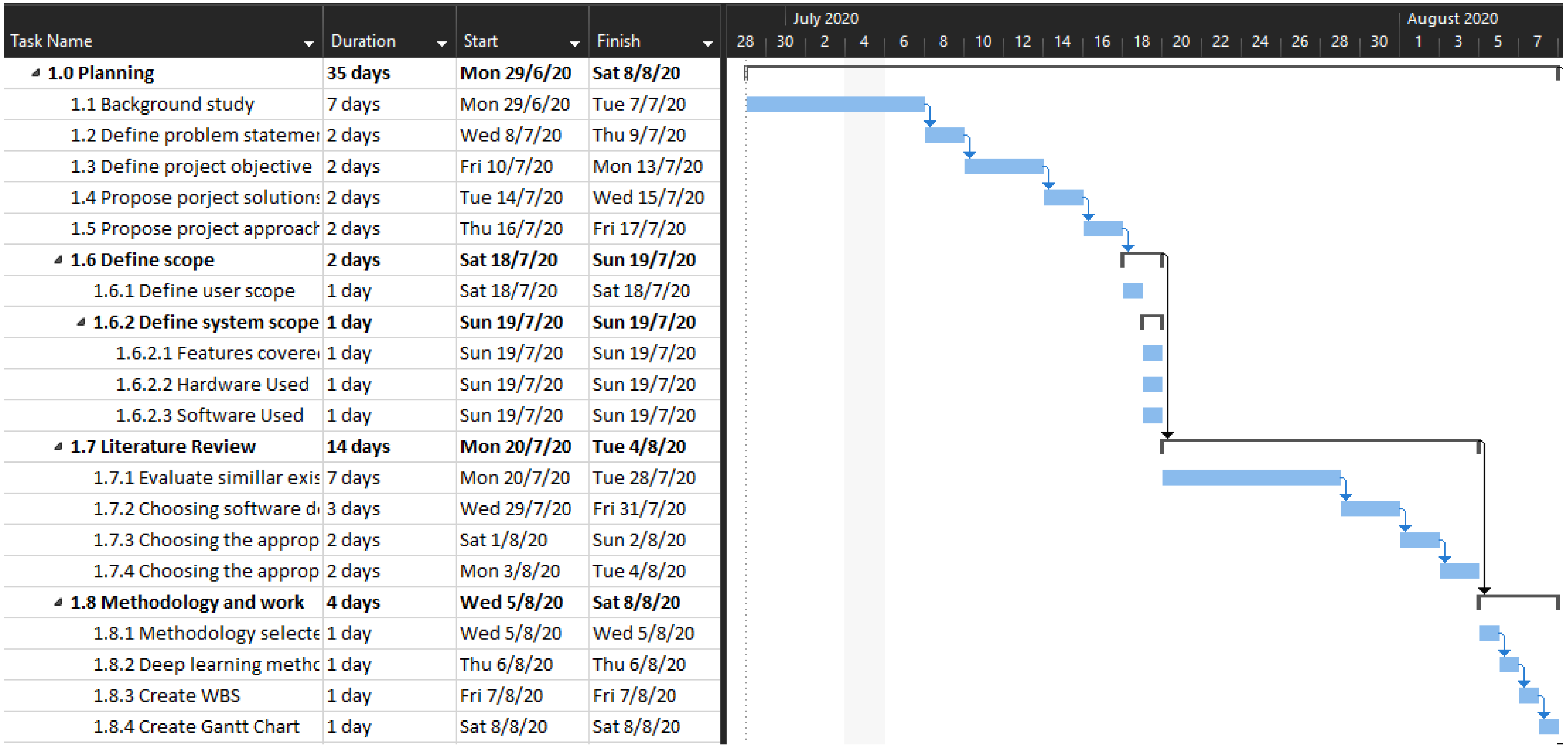
Figure 3.3: Work Breakdown Structure of the project

3.4.2 Gantt Chart



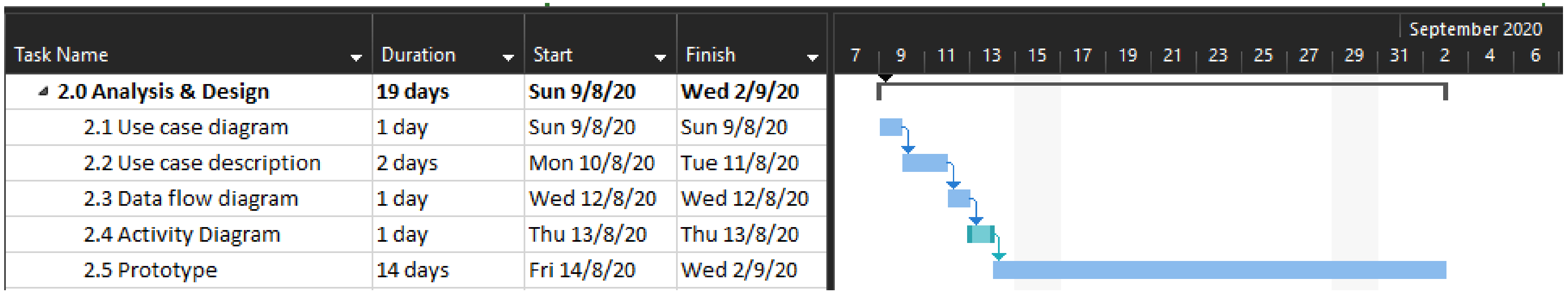
Task Name	Duration	Start	Finish
0. Expiry Reminder Mobile App	197 days	Mon 29/6/20	Thu 18/3/21
1.0 Planning	35 days	Mon 29/6/20	Sat 8/8/20
2.0 Analysis & Design	19 days	Sun 9/8/20	Wed 2/9/20
3.0 Development	103 days	Sun 11/10/20	Sun 28/2/21
4.0 Testing	14 days	Mon 1/3/21	Thu 18/3/21

Figure 3.4: Overview Gantt Chart of The Whole Project



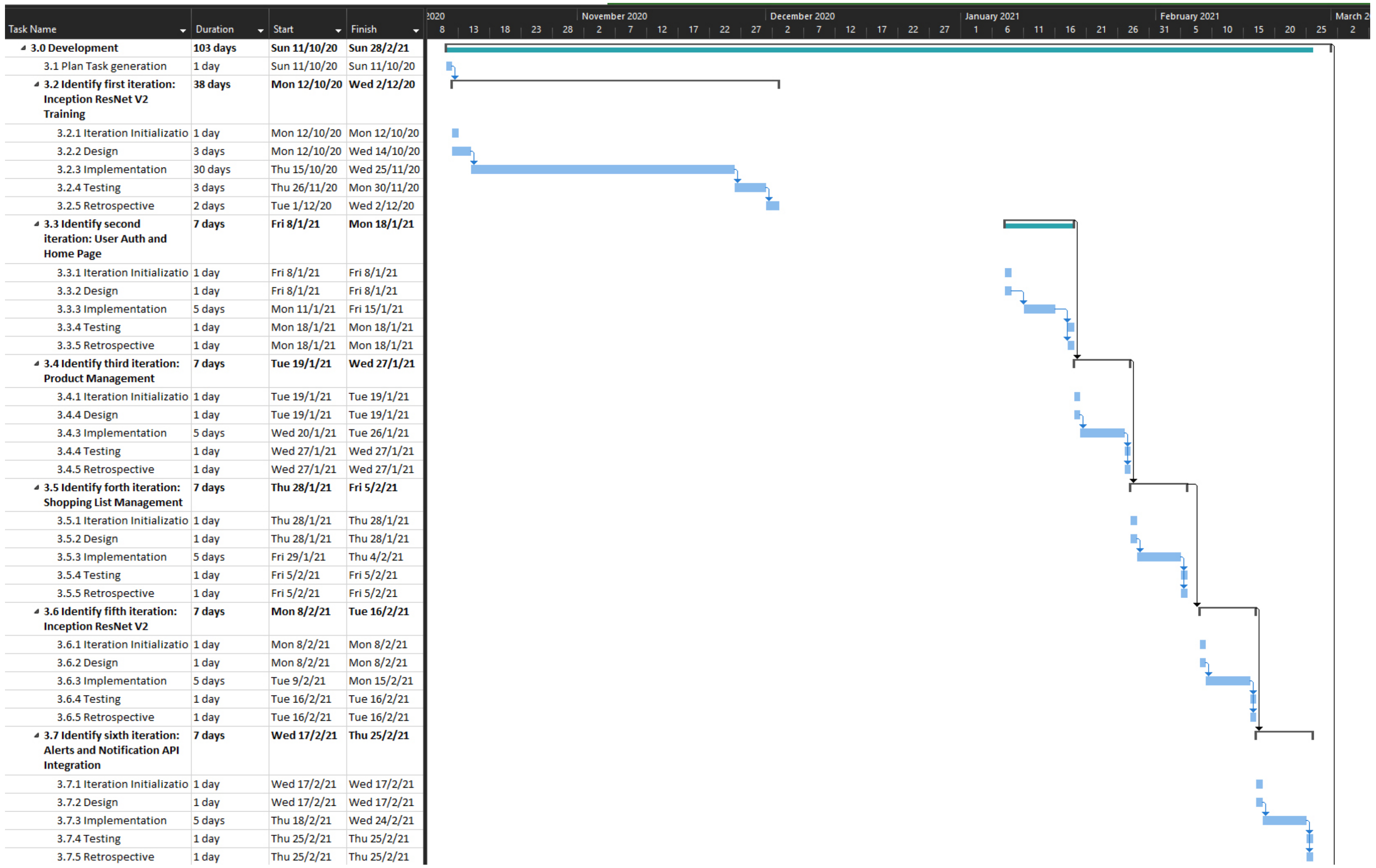
Task Name	Duration	Start	Finish
1.0 Planning	**35 days**	**Mon 29/6/20**	**Sat 8/8/20**
1.1 Background study	7 days	Mon 29/6/20	Tue 7/7/20
1.2 Define problem statemer	2 days	Wed 8/7/20	Thu 9/7/20
1.3 Define project objective	2 days	Fri 10/7/20	Mon 13/7/20
1.4 Propose porject solutions	2 days	Tue 14/7/20	Wed 15/7/20
1.5 Propose project approach	2 days	Thu 16/7/20	Fri 17/7/20
1.6 Define scope	**2 days**	**Sat 18/7/20**	**Sun 19/7/20**
1.6.1 Define user scope	1 day	Sat 18/7/20	Sat 18/7/20
1.6.2 Define system scope	**1 day**	**Sun 19/7/20**	**Sun 19/7/20**
1.6.2.1 Features covere	1 day	Sun 19/7/20	Sun 19/7/20
1.6.2.2 Hardware Used	1 day	Sun 19/7/20	Sun 19/7/20
1.6.2.3 Software Used	1 day	Sun 19/7/20	Sun 19/7/20
1.7 Literature Review	**14 days**	**Mon 20/7/20**	**Tue 4/8/20**
1.7.1 Evaluate simillar exis	7 days	Mon 20/7/20	Tue 28/7/20
1.7.2 Choosing software d	3 days	Wed 29/7/20	Fri 31/7/20
1.7.3 Choosing the approp	2 days	Sat 1/8/20	Sun 2/8/20
1.7.4 Choosing the approp	2 days	Mon 3/8/20	Tue 4/8/20
1.8 Methodology and work	**4 days**	**Wed 5/8/20**	**Sat 8/8/20**
1.8.1 Methodology selecte	1 day	Wed 5/8/20	Wed 5/8/20
1.8.2 Deep learning methc	1 day	Thu 6/8/20	Thu 6/8/20
1.8.3 Create WBS	1 day	Fri 7/8/20	Fri 7/8/20
1.8.4 Create Gantt Chart	1 day	Sat 8/8/20	Sat 8/8/20

Figure 3.5: Overview Gantt Chart of The Planning Phase of Expiry Reminder Mobile Application



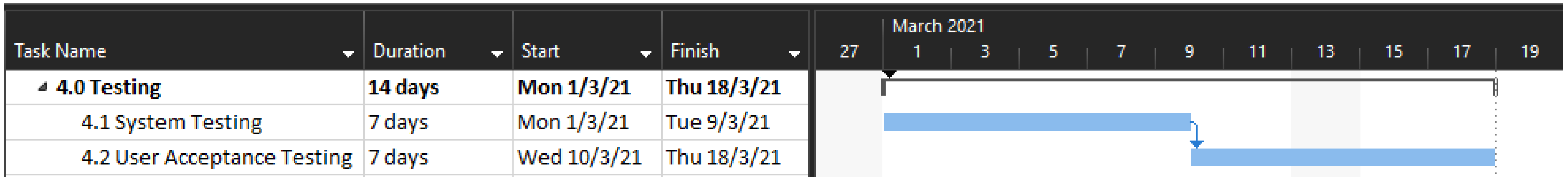
Task Name	Duration	Start	Finish
2.0 Analysis & Design	**19 days**	**Sun 9/8/20**	**Wed 2/9/20**
2.1 Use case diagram	1 day	Sun 9/8/20	Sun 9/8/20
2.2 Use case description	2 days	Mon 10/8/20	Tue 11/8/20
2.3 Data flow diagram	1 day	Wed 12/8/20	Wed 12/8/20
2.4 Activity Diagram	1 day	Thu 13/8/20	Thu 13/8/20
2.5 Prototype	14 days	Fri 14/8/20	Wed 2/9/20

Figure 3.6: Overview Gantt Chart of The Analysis and Design Phase of Expiry Reminder Mobile Application



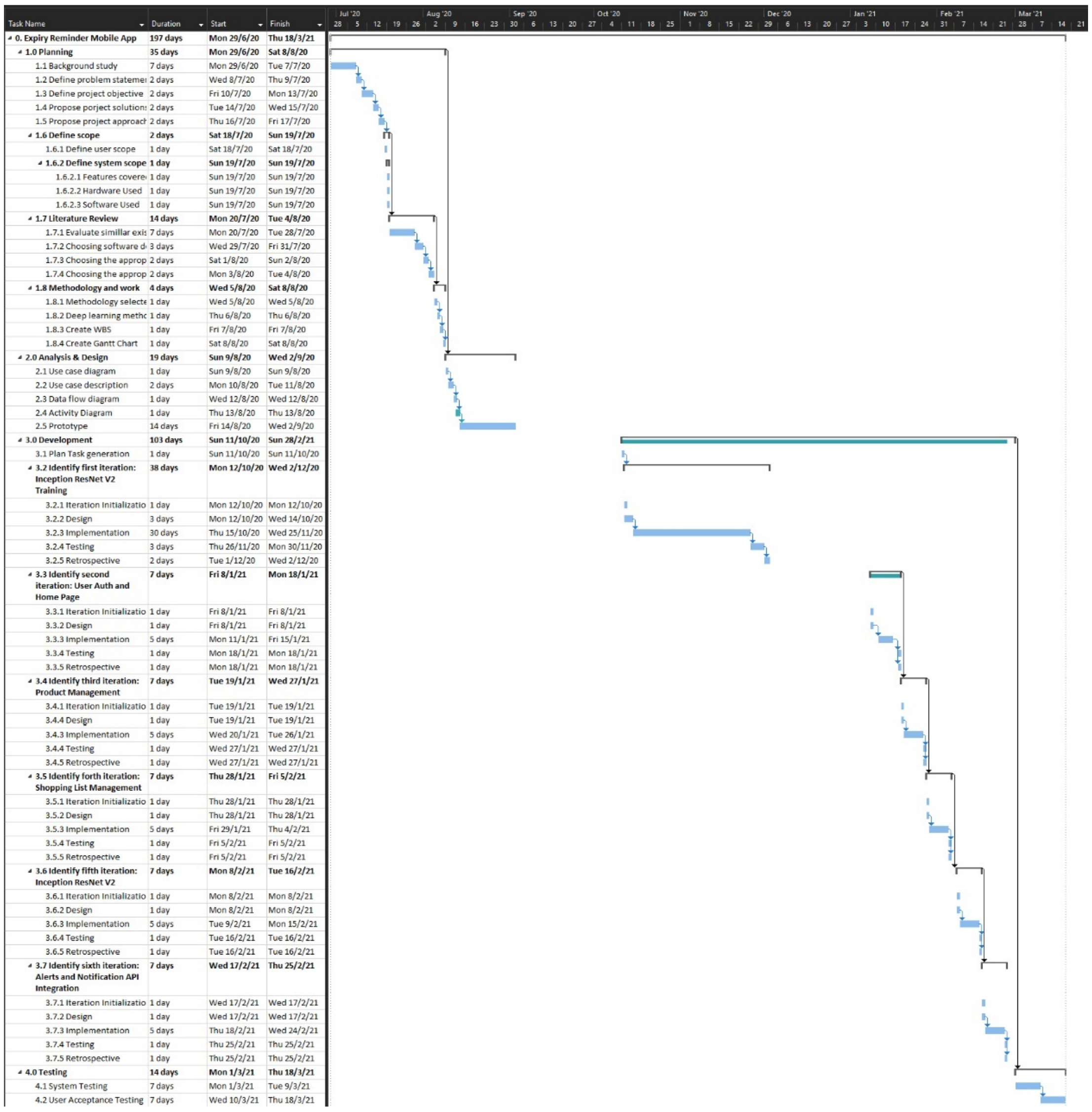
Task Name	Duration	Start	Finish
3.0 Development	103 days	Sun 11/10/20	Sun 28/2/21
3.1 Plan Task generation	1 day	Sun 11/10/20	Sun 11/10/20
3.2 Identify first iteration: Inception ResNet V2 Training	38 days	Mon 12/10/20	Wed 2/12/20
3.2.1 Iteration Initializatio	1 day	Mon 12/10/20	Mon 12/10/20
3.2.2 Design	3 days	Mon 12/10/20	Wed 14/10/20
3.2.3 Implementation	30 days	Thu 15/10/20	Wed 25/11/20
3.2.4 Testing	3 days	Thu 26/11/20	Mon 30/11/20
3.2.5 Retrospective	2 days	Tue 1/12/20	Wed 2/12/20
3.3 Identify second iteration: User Auth and Home Page	7 days	Fri 8/1/21	Mon 18/1/21
3.3.1 Iteration Initializatio	1 day	Fri 8/1/21	Fri 8/1/21
3.3.2 Design	1 day	Fri 8/1/21	Fri 8/1/21
3.3.3 Implementation	5 days	Mon 11/1/21	Fri 15/1/21
3.3.4 Testing	1 day	Mon 18/1/21	Mon 18/1/21
3.3.5 Retrospective	1 day	Mon 18/1/21	Mon 18/1/21
3.4 Identify third iteration: Product Management	7 days	Tue 19/1/21	Wed 27/1/21
3.4.1 Iteration Initializatio	1 day	Tue 19/1/21	Tue 19/1/21
3.4.4 Design	1 day	Tue 19/1/21	Tue 19/1/21
3.4.3 Implementation	5 days	Wed 20/1/21	Tue 26/1/21
3.4.4 Testing	1 day	Wed 27/1/21	Wed 27/1/21
3.4.5 Retrospective	1 day	Wed 27/1/21	Wed 27/1/21
3.5 Identify forth iteration: Shopping List Management	7 days	Thu 28/1/21	Fri 5/2/21
3.5.1 Iteration Initializatio	1 day	Thu 28/1/21	Thu 28/1/21
3.5.2 Design	1 day	Thu 28/1/21	Thu 28/1/21
3.5.3 Implementation	5 days	Fri 29/1/21	Thu 4/2/21
3.5.4 Testing	1 day	Fri 5/2/21	Fri 5/2/21
3.5.5 Retrospective	1 day	Fri 5/2/21	Fri 5/2/21
3.6 Identify fifth iteration: Inception ResNet V2	7 days	Mon 8/2/21	Tue 16/2/21
3.6.1 Iteration Initializatio	1 day	Mon 8/2/21	Mon 8/2/21
3.6.2 Design	1 day	Mon 8/2/21	Mon 8/2/21
3.6.3 Implementation	5 days	Tue 9/2/21	Mon 15/2/21
3.6.4 Testing	1 day	Tue 16/2/21	Tue 16/2/21
3.6.5 Retrospective	1 day	Tue 16/2/21	Tue 16/2/21
3.7 Identify sixth iteration: Alerts and Notification API Integration	7 days	Wed 17/2/21	Thu 25/2/21
3.7.1 Iteration Initializatio	1 day	Wed 17/2/21	Wed 17/2/21
3.7.2 Design	1 day	Wed 17/2/21	Wed 17/2/21
3.7.3 Implementation	5 days	Thu 18/2/21	Wed 24/2/21
3.7.4 Testing	1 day	Thu 25/2/21	Thu 25/2/21
3.7.5 Retrospective	1 day	Thu 25/2/21	Thu 25/2/21

Figure 3.7: Overview Gantt Chart of The Implementation and Testing Phase of Expiry Reminder Mobile Application



Task Name	Duration	Start	Finish
4.0 Testing	**14 days**	**Mon 1/3/21**	**Thu 18/3/21**
4.1 System Testing	7 days	Mon 1/3/21	Tue 9/3/21
4.2 User Acceptance Testing	7 days	Wed 10/3/21	Thu 18/3/21

Figure 3.8: Overview Gantt Chart of The Testing Phase of Expiry Reminder Mobile Application



Task Name	Duration	Start	Finish
0. Expiry Reminder Mobile App	**197 days**	**Mon 29/6/20**	**Thu 18/3/21**
1.0 Planning	**35 days**	**Mon 29/6/20**	**Sat 8/8/20**
1.1 Background study	7 days	Mon 29/6/20	Tue 7/7/20
1.2 Define problem statemer	2 days	Wed 8/7/20	Thu 9/7/20
1.3 Define project objective	2 days	Fri 10/7/20	Mon 13/7/20
1.4 Propose porject solution:	2 days	Tue 14/7/20	Wed 15/7/20
1.5 Propose project approach	2 days	Thu 16/7/20	Fri 17/7/20
1.6 Define scope	**2 days**	**Sat 18/7/20**	**Sun 19/7/20**
1.6.1 Define user scope	1 day	Sat 18/7/20	Sat 18/7/20
1.6.2 Define system scope	**1 day**	**Sun 19/7/20**	**Sun 19/7/20**
1.6.2.1 Features covere	1 day	Sun 19/7/20	Sun 19/7/20
1.6.2.2 Hardware Used	1 day	Sun 19/7/20	Sun 19/7/20
1.6.2.3 Software Used	1 day	Sun 19/7/20	Sun 19/7/20
1.7 Literature Review	**14 days**	**Mon 20/7/20**	**Tue 4/8/20**
1.7.1 Evaluate simillar exis	7 days	Mon 20/7/20	Tue 28/7/20
1.7.2 Choosing software d	3 days	Wed 29/7/20	Fri 31/7/20
1.7.3 Choosing the approp	2 days	Sat 1/8/20	Sun 2/8/20
1.7.4 Choosing the approp	2 days	Mon 3/8/20	Tue 4/8/20
1.8 Methodology and work	**4 days**	**Wed 5/8/20**	**Sat 8/8/20**
1.8.1 Methodology selecte	1 day	Wed 5/8/20	Wed 5/8/20
1.8.2 Deep learning methc	1 day	Thu 6/8/20	Thu 6/8/20
1.8.3 Create WBS	1 day	Fri 7/8/20	Fri 7/8/20
1.8.4 Create Gantt Chart	1 day	Sat 8/8/20	Sat 8/8/20
2.0 Analysis & Design	**19 days**	**Sun 9/8/20**	**Wed 2/9/20**
2.1 Use case diagram	1 day	Sun 9/8/20	Sun 9/8/20
2.2 Use case description	2 days	Mon 10/8/20	Tue 11/8/20
2.3 Data flow diagram	1 day	Wed 12/8/20	Wed 12/8/20
2.4 Activity Diagram	1 day	Thu 13/8/20	Thu 13/8/20
2.5 Prototype	14 days	Fri 14/8/20	Wed 2/9/20
3.0 Development	**103 days**	**Sun 11/10/20**	**Sun 28/2/21**
3.1 Plan Task generation	1 day	Sun 11/10/20	Sun 11/10/20
3.2 Identify first iteration: Inception ResNet V2 Training	**38 days**	**Mon 12/10/20**	**Wed 2/12/20**
3.2.1 Iteration Initializatio	1 day	Mon 12/10/20	Mon 12/10/20
3.2.2 Design	3 days	Mon 12/10/20	Wed 14/10/20
3.2.3 Implementation	30 days	Thu 15/10/20	Wed 25/11/20
3.2.4 Testing	3 days	Thu 26/11/20	Mon 30/11/20
3.2.5 Retrospective	2 days	Tue 1/12/20	Wed 2/12/20
3.3 Identify second iteration: User Auth and Home Page	**7 days**	**Fri 8/1/21**	**Mon 18/1/21**
3.3.1 Iteration Initializatio	1 day	Fri 8/1/21	Fri 8/1/21
3.3.2 Design	1 day	Fri 8/1/21	Fri 8/1/21
3.3.3 Implementation	5 days	Mon 11/1/21	Fri 15/1/21
3.3.4 Testing	1 day	Mon 18/1/21	Mon 18/1/21
3.3.5 Retrospective	1 day	Mon 18/1/21	Mon 18/1/21
3.4 Identify third iteration: Product Management	**7 days**	**Tue 19/1/21**	**Wed 27/1/21**
3.4.1 Iteration Initializatio	1 day	Tue 19/1/21	Tue 19/1/21
3.4.4 Design	1 day	Tue 19/1/21	Tue 19/1/21
3.4.3 Implementation	5 days	Wed 20/1/21	Tue 26/1/21
3.4.4 Testing	1 day	Wed 27/1/21	Wed 27/1/21
3.4.5 Retrospective	1 day	Wed 27/1/21	Wed 27/1/21
3.5 Identify forth iteration: Shopping List Management	**7 days**	**Thu 28/1/21**	**Fri 5/2/21**
3.5.1 Iteration Initializatio	1 day	Thu 28/1/21	Thu 28/1/21
3.5.2 Design	1 day	Thu 28/1/21	Thu 28/1/21
3.5.3 Implementation	5 days	Fri 29/1/21	Thu 4/2/21
3.5.4 Testing	1 day	Fri 5/2/21	Fri 5/2/21
3.5.5 Retrospective	1 day	Fri 5/2/21	Fri 5/2/21
3.6 Identify fifth iteration: Inception ResNet V2	**7 days**	**Mon 8/2/21**	**Tue 16/2/21**
3.6.1 Iteration Initializatio	1 day	Mon 8/2/21	Mon 8/2/21
3.6.2 Design	1 day	Mon 8/2/21	Mon 8/2/21
3.6.3 Implementation	5 days	Tue 9/2/21	Mon 15/2/21
3.6.4 Testing	1 day	Tue 16/2/21	Tue 16/2/21
3.6.5 Retrospective	1 day	Tue 16/2/21	Tue 16/2/21
3.7 Identify sixth iteration: Alerts and Notification API Integration	**7 days**	**Wed 17/2/21**	**Thu 25/2/21**
3.7.1 Iteration Initializatio	1 day	Wed 17/2/21	Wed 17/2/21
3.7.2 Design	1 day	Wed 17/2/21	Wed 17/2/21
3.7.3 Implementation	5 days	Thu 18/2/21	Wed 24/2/21
3.7.4 Testing	1 day	Thu 25/2/21	Thu 25/2/21
3.7.5 Retrospective	1 day	Thu 25/2/21	Thu 25/2/21
4.0 Testing	**14 days**	**Mon 1/3/21**	**Thu 18/3/21**
4.1 System Testing	7 days	Mon 1/3/21	Tue 9/3/21
4.2 User Acceptance Testing	7 days	Wed 10/3/21	Thu 18/3/21

Figure 3.9: Gantt Chart of Expiry Reminder Mobile Application Project

CHAPTER 4
PROJECT SPECIFICATION

4.1 Introduction

In this chapter, the functional requirement and non-functional requirement will be discussed. In addition, a use case diagram and use case description will be used to show how users can interact with the system. The screenshots of the prototype will also be shown in this chapter.

4.2 Functional Requirement

The functional requirement is a definition of the features a software must provide. It can be input, system behaviour and output. The functional requirements of this project are listed as follow:

1. The mobile application must ask and validate the user's phone number and E-mail for first time users.
2. The mobile application must allow the user to sign up.
3. The mobile application must allow the user to log in.
4. The mobile application must allow the user to use a third-party account (e.g. Google Account, Facebook Account) to log in.
5. The mobile application must display products added and their details according to the expiry dates.
6. The mobile application must display products added and their details according to the category.
7. The mobile application must display the total number of expired, soon to expire and safe to use products.
8. The mobile application must allow the user to add a new product together with the product name, product description, product category, expiry date, reminder date, reminder method, product photo, number of stock and barcode.
9. The mobile application must allow the user to add a new category.
10. The mobile application must allow the user to edit the product name, product description, product category, expiry date, reminder date, reminder method, product photo, number of stock and barcode.
11. The mobile application must allow the user to edit the category.

12. The mobile application must allow the user to delete the product added.
13. The mobile application must allow the user to delete the category added.
14. The mobile application must allow the user to search the product by product name or barcodes.
15. The mobile application must notify the user according to the reminder method (e.g. Push Notification, SMS Reminder, E-mail reminder) chosen by the user.
16. The mobile application must allow the user to use OCR (Optical Character Recognition) to scan the product name and product details when adding a product.
17. The mobile application must disable reminder automatically of a product when the stock of the product becomes zero.
18. The mobile application must move the product to the restock list when the product's stock becomes zero.
19. The mobile application must allow the user to detect dates using a camera.

4.3 Non-Functional Requirement

The non-functional requirement will specify different criteria that can be used to measure the system's operation and quality. In this section, adaptability requirements, availability requirements, development requirements, performance requirements, responsiveness requirements, reliability requirements, security requirements, usability requirements will be listed.

4.3.1 Adaptability Requirements

1. The mobile application shall able to render its layout in different screen sizes.

4.3.2 Availability Requirements

1. The mobile application shall be accessible by the user with the condition that they have access to their mobile phone and internet connection.

4.3.3 Development Requirements

1. The mobile application shall be built on the Android Platform.
2. The methodology used in the development is Personal Extreme Prototyping.

3. The programming language used for the mobile application is Dart.
4. The app data is stored in the Firebase Firestore database.
5. The APIs are deployed in Firebase Cloud Functions.

4.3.4 Performance Requirements

1. The mobile application shall process a user request and return the result within 3 seconds.
2. The mobile application shall load within 3 seconds.
3. The mobile application shall handle 99% of the exception thrown during runtime and display readable error messages.
4. The mobile application shall be able to recognize the barcode and character within 5 seconds.

4.3.5 Responsiveness Requirements

1. The mobile application shall save the state and return to the same state when it got interrupted.

4.3.6 Reliability Requirements

1. The mobile application shall display a confirmation message when the user performs some important action like delete a product.

4.3.7 Security Requirements

1. The mobile application shall verify the user’s phone number and E-mail using OTP (One-time password) whenever the user adds or updates the phone number and E-mail.
2. The mobile application shall ask for the user’s permission to gain access to all authentication token.

4.3.8 Usability Requirements

1. The UI of the mobile application shall be simple and easy to understand.
2. The buttons and icon of the mobile application shall be understandable and simple.

4.4 Use Case

This section will show how a user can interact with the system. There are 12 use cases identified and are displayed on the use case diagram. Apart from that, 5 use case descriptions are used to clarify how each use case works.

4.4.1 Use Case Diagram

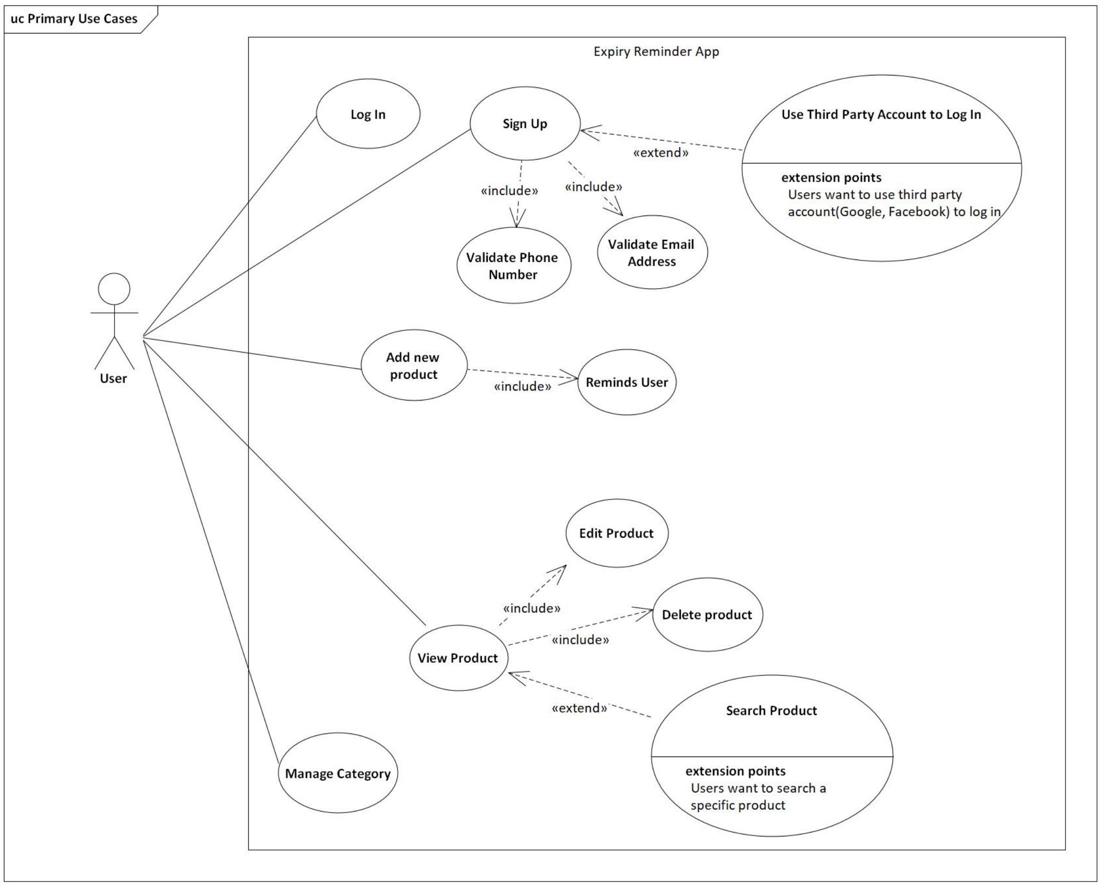


Figure 4.1: Use Case Diagram of Expiry Reminder Mobile App

4.4.2 Use Case Description

Table 4.1: Use Case Description of Sign Up

Name: Sign Up	**ID:** 1	**Priority:** High
Actor: User	**Type:** Detail, Essential	
Stakeholders and Interests: User – New user will sign up and create an account.		
Brief Description: This use case describes the steps needed for the user to create an account for the mobile application.		
Trigger: First time user that does not have an account and does not want to use a third-party account to log in.		
Relationships: Association: User Include: Validate Phone Number, Validate Email Address Extend: Use a third-party account to log in		
Normal Flow of Events: 1. The user clicks on the register button in the application to enter the registration page. If the user wishes to use a third-party account to log in, the user can perform sub-flow 1.1. 2. The user fills in details which include username, password, email and phone number. 3. The system verifies the password to check whether the password has fulfilled the requirement. 4. The system sends an OTP code to the user's phone number. 5. The user enters the OTP code to verify the phone number. If the OTP code is expired, perform sub-flow 5.1. 6. The system sends an activation link to the user's email address to activate the account and verify the email address. 7. The user taps the activation link to verify the email address. If the verification link is expired, perform sub-flow 1.2. 8. The system displays a success message when the account is activated.		
Sub-Flows: 1.1. The user wants to log in using third party accounts. 1.1.1 The user clicks the login button according to the third-party account the user wants. 1.1.2 The user logs in to the third-party account.		

1.1.3 The user authenticates the mobile app to use the third-party account to log in.

1.1.4 The system verifies the user’s phone number and email address.

5.1. The system resends the OTP code to the phone number.

7.1 The system resends the activation link to the email address.

Alternate/Exceptional Flows:

3.1. If the password does not fulfil the requirement, the system displays an error message. The user needs to fill in the password again.

3.2. If the phone number is used, the system displays an error message. The user needs to enter another phone number.

3.3. If the email address is used, the system displays an error message. The user needs to enter another email address.

3.4. If the input fields are empty, the system displays an error message. The user needs to fill in the empty fields.

Table 4.2: Use Case Description of Log In

Name: Log In	**ID:** 2	**Priority:** High
Actor: User	**Type:** Detail, Essential	
Stakeholders and Interests: User – The user who wants to use the app and already has an account.		
Brief Description: This use case describes the steps needed for the user to log in to the account for the mobile application.		
Trigger: A user wishes to log in to the account using the created account.		
Relationships: Association: User Include: - Extend: -		
Normal Flow of Events: 1. The user clicks on the login button in the application to enter the login page. 2. The user fills in the email address and password. If the user forgets the password, perform sub-flow 2.1. 3. The system verifies the email address and the password. 4. The system retrieves user data. 5. The system navigates the user to the home page.		
Sub-Flows: 2.1. The user forgets the password. 2.1.1 The user clicks forget password button. 2.1.2 The user fills in the email address. 2.1.3 The system sends the reset password link to the user's email. 2.1.4 The user reset the passwords by entering a new password. 2.1.5 The system updates the old password to the new password.		
Alternate/Exceptional Flows: 3.1 If the email address or password is incorrect, the system displays an error message. The user needs to fill in the email address and the password again. 3.2 If the input fields are empty, the system displays an error message. The user needs to enter an email address and password.		

Table 4.3: Use Case Description of Add New Product

Name: Add New Product	**ID:** 3	**Priority:** High
Actor: User	**Type:** Detail, Essential	
Stakeholders and Interests: User – The user who wants to add a new product.		
Brief Description: This use case describes the steps needed for the user to add a new product.		
Trigger: The user wishes to add a new product to the mobile app.		
Relationships: Association: User Include: Reminds user Extend: -		
Normal Flow of Events: 1. The user clicks the add button to add a new product. 2. The user fills in the name, description, category, expiry date, reminder date, number of stocks, reminder type and reminder time for the product. If the user wishes to add photos, perform sub-flow 2.1. If the user wishes to add a barcode, perform sub-flow 2.2. If the user wishes to use OCR to add a product name, perform sub-flow 2.3. If the user wishes to use the camera to detect the expiry date, perform sub-flow 2.4. 3. The user clicks the save button to add the product. 4. The system adds the product to the database. 5. The system navigates back to the home page. 6. If the reminder date and time of a product are the same as the current date and time, the system will remind the user according to the reminder type the user select. a. If the user selects push notification reminder as the reminder type, perform sub-flows 6.1. b. If the user selects email reminder as the reminder type, perform sub-flows 6.2. c. If the user selects SMS reminder as the reminder type, perform sub-flows 6.3.		
Sub-Flows: 2.1. The user wants to add a photo of the product. 2.1.1. The user taps the image placeholder. 2.1.2. The system navigates to the camera app. 2.1.3. The user takes one photo of the product. 2.1.4. The user crops the photo. 2.1.5. The user taps the done button.		

2.1.6. The system saves the photo to the product and navigates back to the add product page. 2.2. The user wants to add a barcode to the product. 2.2.1. The user taps the scan barcode button. 2.2.2. The system navigates to the barcode scanner page. 2.2.3. The user scans the barcode of the product. 2.2.4. The system retrieves the barcode and saves the barcode to the product. 2.2.5. The system navigates back to the add product page. 2.3.The user wants to add a product name with OCR. 2.3.1. The user taps the Scan Product Name with the OCR button. 2.3.2. The system navigates to the OCR Scanner page. 2.3.3. The system detects all the text on the product. 2.3.4. The user taps on the detected text that the user wants to use as the product name. 2.3.5. The system adds the selected product name into the product name field and navigates back to the add product page. 2.4.The user wants to add the expiry date with the camera. 2.4.1. The user taps the scan date button. 2.4.2. The system navigates to the camera app. 2.4.3. The user takes a photo of the expiry date. 2.4.4. The user crops the expiry date. 2.4.5. The user taps the done button. 2.4.6. The system identifies the expiry date captured and navigates back to the add product page 2.4.7. The system inserts the expiry date into the expiry date fields 6.1. The system will display a notification on the reminder date and time. 6.2. The system will send an email to the user's email address on the reminder date and time. 6.3. The system will send an SMS to the user's phone number on the reminder date and time
Alternate/Exceptional Flows: 3.1. If the input fields are empty, the system displays an error message and the user needs to fill in the empty fields. 3.2 If the expiry date is before the current date, the system displays an error message and the user needs to reselect the expiry date. 3.3 If the reminder date is before the current date or after the expiry date, the system displays an error message and the user needs to reselect the reminder date. 3.4 If the number of stocks is less than 1, the system displays an error message and the user needs to enter the number of stocks again.

Table 4.4: Use Case Description of View Product

Name: View Product	**ID:** 4	**Priority:** High
Actor: User	**Type:** Detail, Essential	
Stakeholders and Interests: User – The user who wants to view the added product.		
Brief Description: This use case describes the steps needed for the user to view the products added.		
Trigger: The user wishes to view the products available in the mobile app.		
Relationships: Association: User Include: Edit Product, Delete Product Extend: Search Product		
Normal Flow of Events: 1. The user can choose the view products according to expiry dates or according to categories. a. If the user wishes to view products according to expiry dates, perform sub-flow 1.1. b. If the user wishes to view products according to categories, perform sub-flow 1.2. 2. The system retrieves data and displays it to users. 3. The user clicks into a product to view the details of the product. If the user wishes to search for a specific product, perform sub-flow 3.1. 4. The system navigates to the product detail page. 5. The user can choose to edit or delete a product. a. If the user wishes to edit a product, perform sub-flow 5.1. b. If the user wishes to delete a product, perform sub-flow 5.2.		
Sub-Flows: 1.1. View products according to expiry dates. 1.1.1. The user selects the home page from the navigation menu. 1.1.2. The system navigates back to the home page and displays all products according to the expiry dates. 1.2. View products according to categories. 1.2.1. The user selects view product by category from the navigation menu. 1.2.2. The system navigates to the category page. 1.2.3. The user selects the category the user wants. 1.2.4. The system displays all products of the category. 3.1. Search product 3.1.1. Search product by name 3.1.1.1. The user clicks on the search bar and enters the name of the product.		

3.1.1.2. The system searches for the data and displays the search results. 3.1.2. Search product by barcode 3.1.2.1. The user clicks search by barcode in the navigation menu. 3.1.2.2. The system navigates to the barcode scanner page. 3.1.2.3. The user scans the barcode of the product the user wants to search. 3.1.2.4. The system retrieves the product with the same barcode. 5.1. Edit product 5.1.1. The user can edit the name, description, category, expiry date, reminder date, number of stocks, photo, barcode, reminder type and reminder time for the product. 5.1.2. When the user has done editing the product, the user clicks save. 5.1.3. The system will display a success message when the product is updated. 5.1.4. If the number of stocks of a product is equal to 0, perform alternate flow 5.1.4.1. 5.2.Delete Product 5.2.1. The user clicks the delete buttons to delete the product. 5.2.2. The system displays a confirmation message to the user. 5.2.3. If the user clicks confirm, the product is deleted. A success message will be shown. If the user clicks cancel, the system navigates back to the product detail page.
Alternate/Exceptional Flows: 5.1.4.1 The system will move the product to the restock list and disable the reminder of the product.

Table 4.5: Use Case Description of Manage Category

Name: Manage Category	**ID:** 5	**Priority:** High
Actor: User	**Type:** Detail, Essential	
Stakeholders and Interests: User – The user who wants to add, edit and delete a category.		
Brief Description: This use case describes the steps needed for the user to manage the category.		
Trigger: The user wishes to add, edit and delete a category.		
Relationships: Association: User Include: - Extend: -		
Normal Flow of Events: 1. The user clicks manage category in the navigation menu. 2. The system navigates to the manage category page. 3. The user can choose to add, edit and delete the category. a. If the user wishes to add a new category, perform sub-flow 3.1. b. If the user wishes to edit a category, perform sub-flow 3.2. c. If the user wishes to delete a category, perform sub-flow 3.3.		
Sub-Flows: 3.1. Add new category 3.1.1. The user clicks the add button to add a category. 3.1.2. The user enters the name of the category. 3.1.3. The user clicks the save button to save the category. 3.1.4. The system adds the category to the category list. 3.2. Edit category 3.2.1. The user clicks on the category the user wishes to edit. 3.2.2. The user clicks the edit button. 3.2.3. The user enters a change category name. 3.2.4. The user clicks the save button to save the category. 3.2.5. The system updates the category into the category list. 3.3. Delete category 3.3.1. The user clicks on the category the user wishes to delete. 3.3.2. The user clicks the delete button. 3.3.3. The system displays a confirmation message. 3.3.4. If the user pressed confirm, the system will delete the category and set all the products in the category to none. If the user pressed cancel, the system closes the confirmation message.		

Alternate/Exceptional Flows: 3.4. If the user edits or deletes the none category, an error message will be displayed.

CHAPTER 5
SYSTEM DESIGN

5.1 Introduction

This chapter discusses the system design, database design and also interface design.

5.2 System Design Models

In this section, system architecture design, Data Flow Diagram and activity diagram will be used to illustrate how users interact with the system and how the system handles users' request.

5.2.1 System Architecture Design

The system architecture design used for the mobile application is Model View Controller (MVC) architecture and microservice. The model contains the application's data structure from the database and directly manages the data. The view is any representation of information that accepts user input and display output. The Controller will process the user input before passing back to the model or manipulate data before passing it to view. On the other hand, cloud microservices are also used to automate backend services. Both architectures are used together to provide a good user experience and at the same time ensure the structure of the code can be easily modified in the future. The overall system architecture design is shown in Figure 5.1 below.

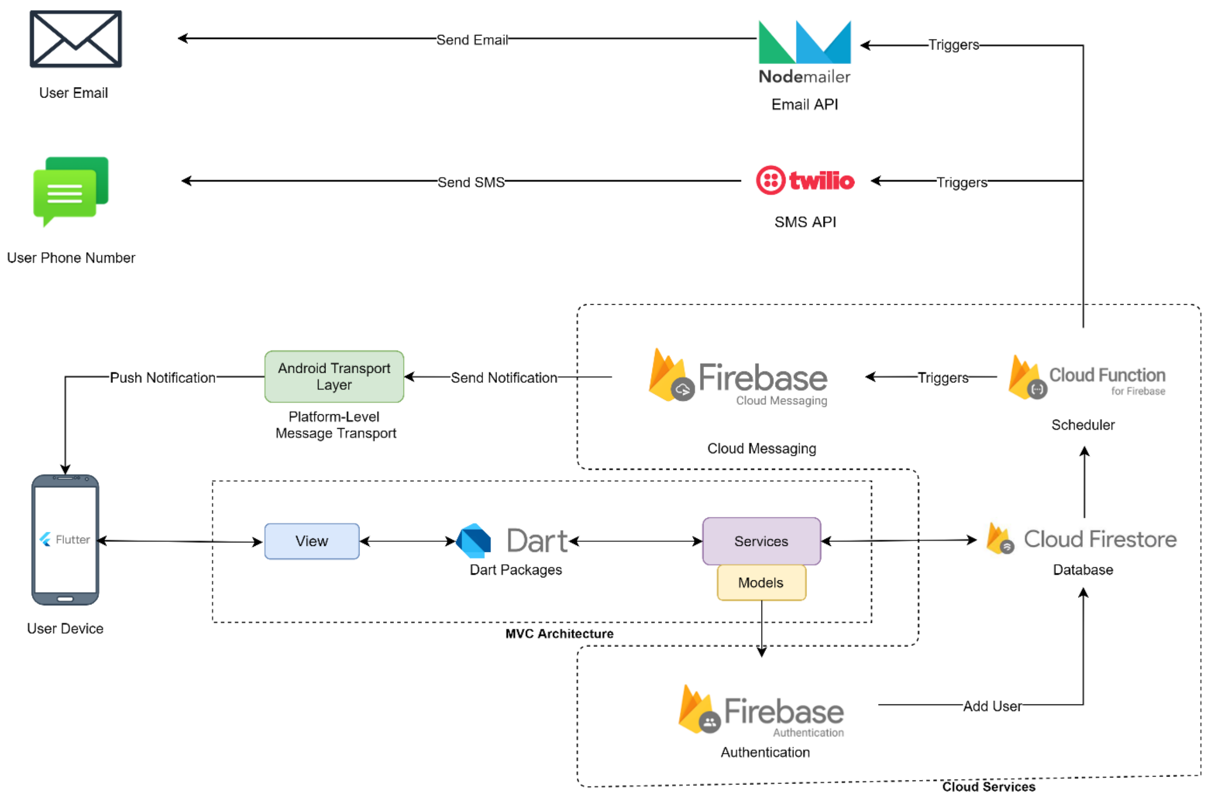


Figure 5.1: System Architecture Design

5.2.1.1 Model View Controller Architecture

Model in MVC architecture is the component representing the shape of the data. It is the lowest level of the pattern which is used to maintain data in the database. The controller in MVC architecture is the part of the application that handles user requests from the view, processes and manipulates data from the model according to the user request. It is mostly a request handler that converts user requests into commands for model or view. View in MVC architecture is the part of the application that is responsible for presenting data or information and mainly acts as a user interface that displays information to the end-user. The view accepts user input to be sent to the controller and the controller will render the output back to the view to be displayed to the end-user.

In the flutter application, the view is used to accept user input through forms. Then, the input is passed to the Dart packages. Dart packages will pass the input to the controller, which is also named as services. Services will process and store the data received into Firebase Firestore through the model. Then, if the output is needed to be displayed to the end-user, services will retrieve data from Firebase Firestore through the model and process the data according to the needs from the view. Then, the data will be passed to the Dart Packages to be rendered into the widget in the view to be displayed to the user. The detailed MVC Architecture is illustrated in Figure 5.2 below.

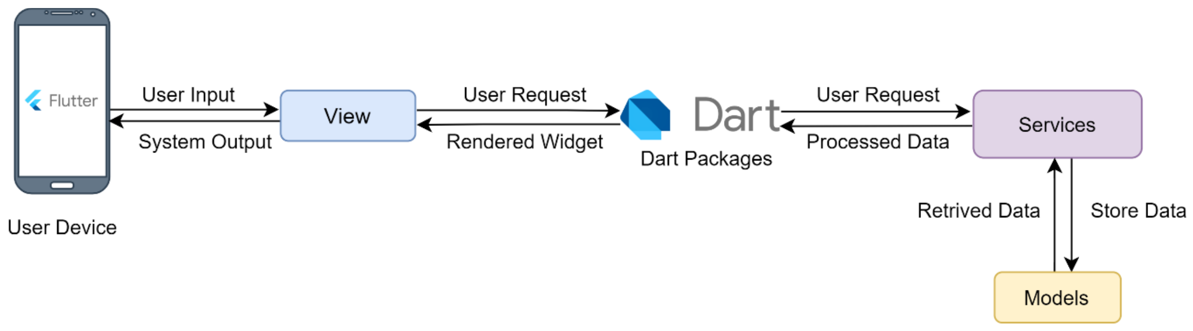


Figure 5.2: MVC System Architecture Design

5.2.1.2 Microservices Architecture

Microservices architecture is developing a monolithic application, a single logical executable and deployed to the cloud. Each microservice is a small function that is independent and loosely coupled and is deployed independently. This ensures agility in developing and deploying the services because one can update a service without redeploying the whole system.

The microservices for the application are deployed to the Firebase Cloud Function, where the service will constantly check the database in the Firestore every minute, and trigger an alert according to the alert type the user chose when necessary. If the alert type is a push notification, the service will send a request to Firebase Cloud Messaging to send a push notification to the user device. If the alert type is an SMS notification, the service will send a request to Twilio SMS API to send an SMS to the user’s phone number. If the alert type is email notification, the service will send a request to Node Mailer API to send an email to the user’s email address. The whole process is automated, and the alert messages are customized according to the situation. This can improve user experiences when the notification can clearly state out useful information to the user, such as product name and expiry date.

5.2.1.3 Firebase as the Backend Services

As mentioned before, Firebase is used as a backend for the application. Firebase Firestore acts as the database of the application where each data is stored on the cloud. The reason for choosing Firestore as the database is because by storing information on the cloud, users can access the same data even on multiple devices without synchronizing the data or transferring the data to another device.

Besides, Firebase Authentication is also used to authenticate users. Firebase Authentication provides various types of authentication methods: Email and Password authentication, Facebook Account authentication, Google Account authentication, and Phone authentication. This cloud API smoothens the developing process because the API will handle all of the validation, password hashing and generate a random user id.

On the other hand, Firebase Cloud Messaging is used to send push notification to the user. Apart from local push notification, sending notification from cloud services enables more flexibility and customization for a greater user experience. Plus, notification messages can be changed frequently without having to redeploy the application.

Last but not least, Firebase Cloud Function is used to deploy the microservices developed. By deploying microservices to the cloud, the maintenance and update progress can be done easily by updating the services deployed on the cloud. This also ensures scalability when putting on the cloud where a development team can scale up and down according to the needs.

5.2.2 Data Flow Diagram (DFD)

The data flow diagram is commonly used to display the data input and output of a system. In this section, the context diagram and the Level 1 data flow diagram are displayed to provide a clear vision of how the system handles inputs and outputs.

5.2.2.1 Context Diagram

Context diagram displays systems as a single high-level process that visualizes the movement of data in and out from the user to the system and vice versa. Figure 5.3 below shows the context diagram of the Expiry Reminder System.

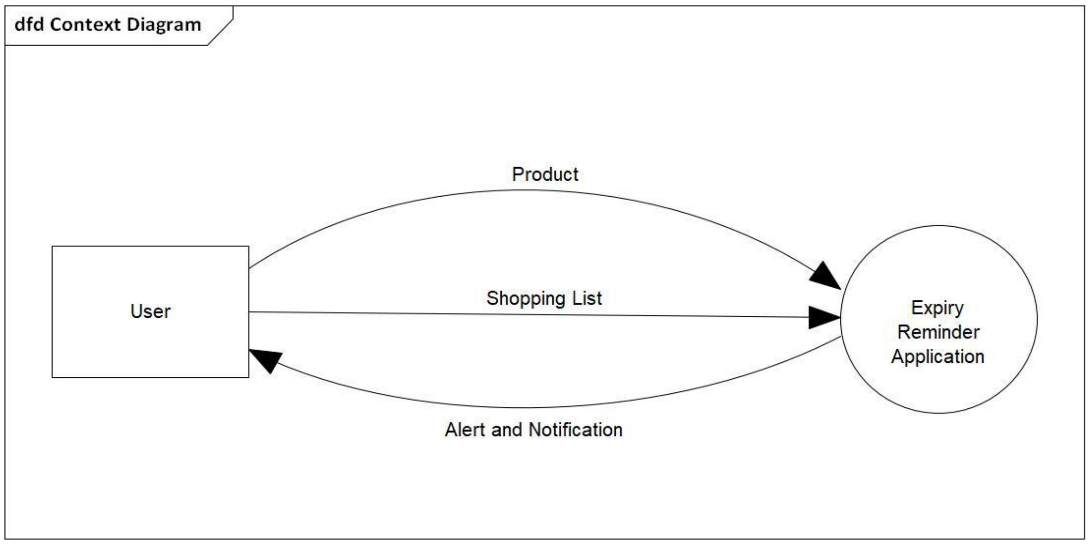


Figure 5.3: The Context Data Flow Diagram

5.2.2.2 Level-1 Data Flow Diagram (DFD)

Level-1 DFD is a more specific DFD broken down from the context diagram. It provides a more comprehensive view of the context diagram. Figure 5.4 shows the Level-1 DFD of the Expiry Reminder System.

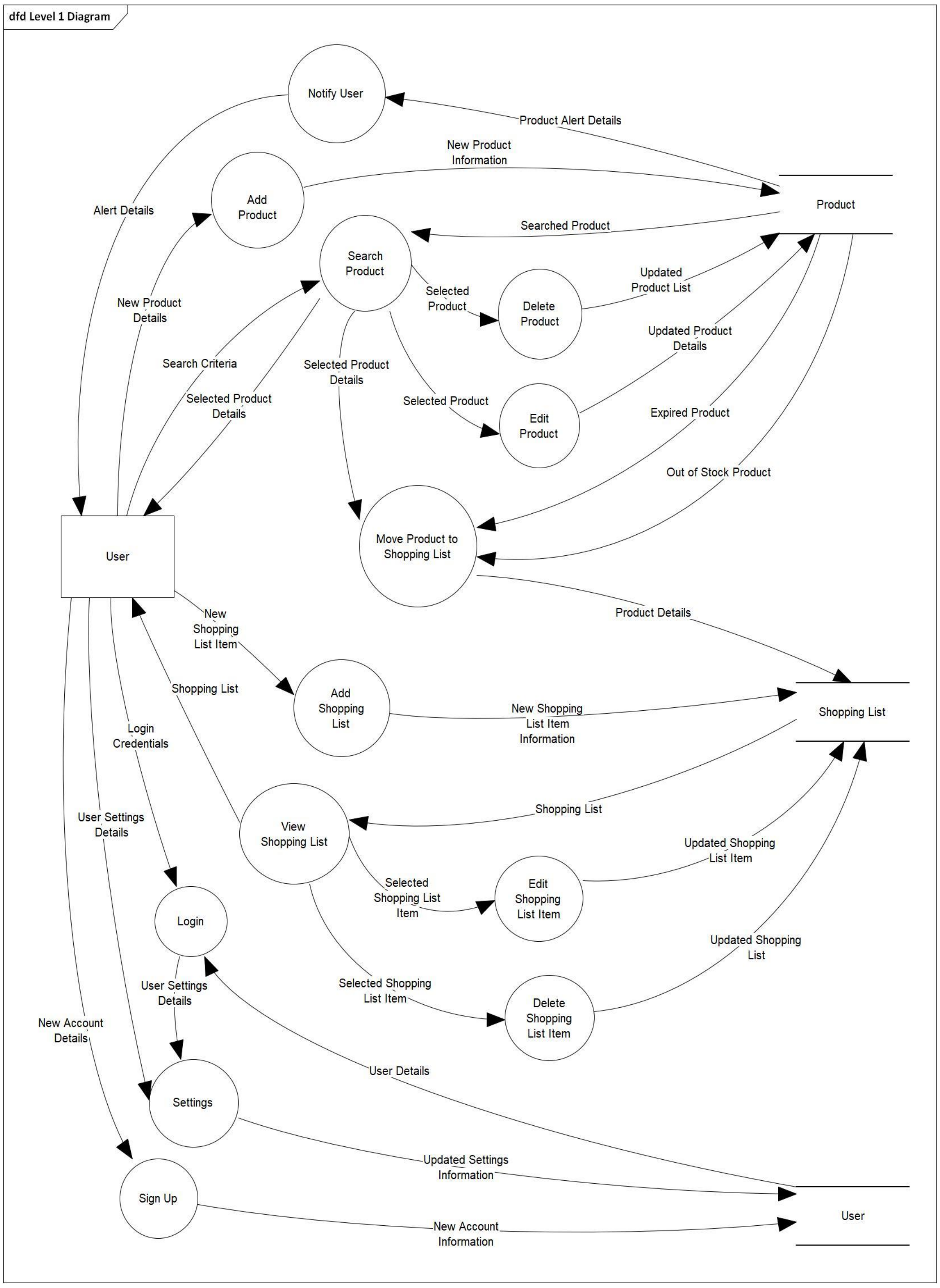


Figure 5.4: The Level-1 Data Flow Diagram

5.2.3 Activity Diagram

5.2.3.1 Login

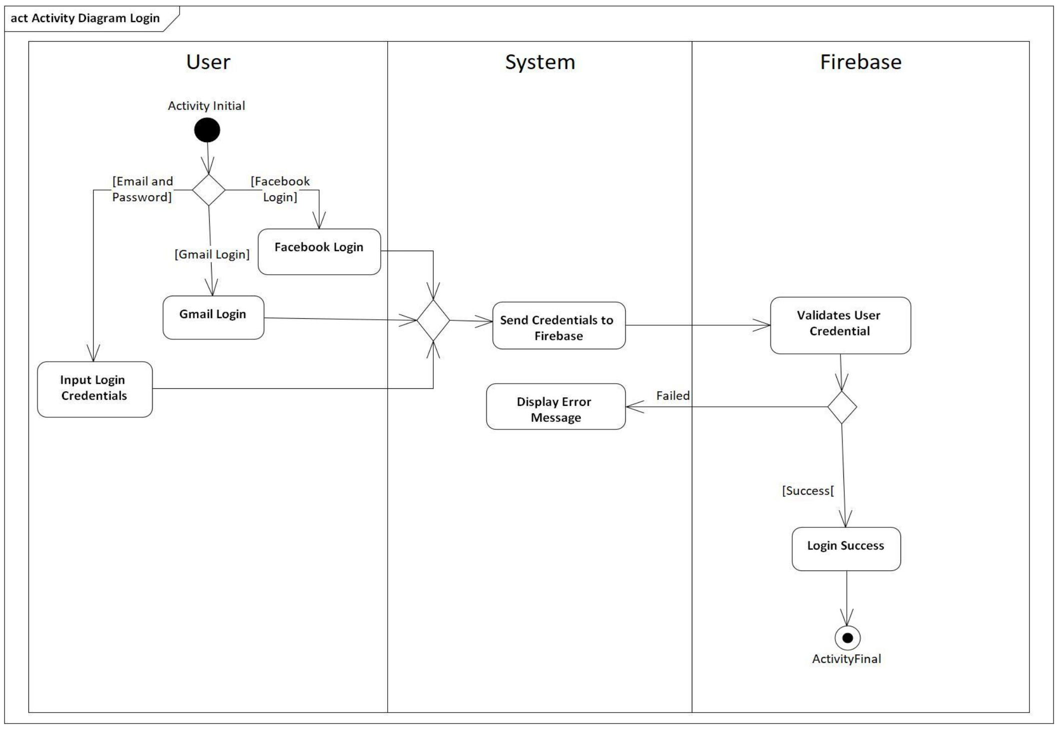


Figure 5.5: The Activity Diagram for Login

5.2.3.2 Sign Up

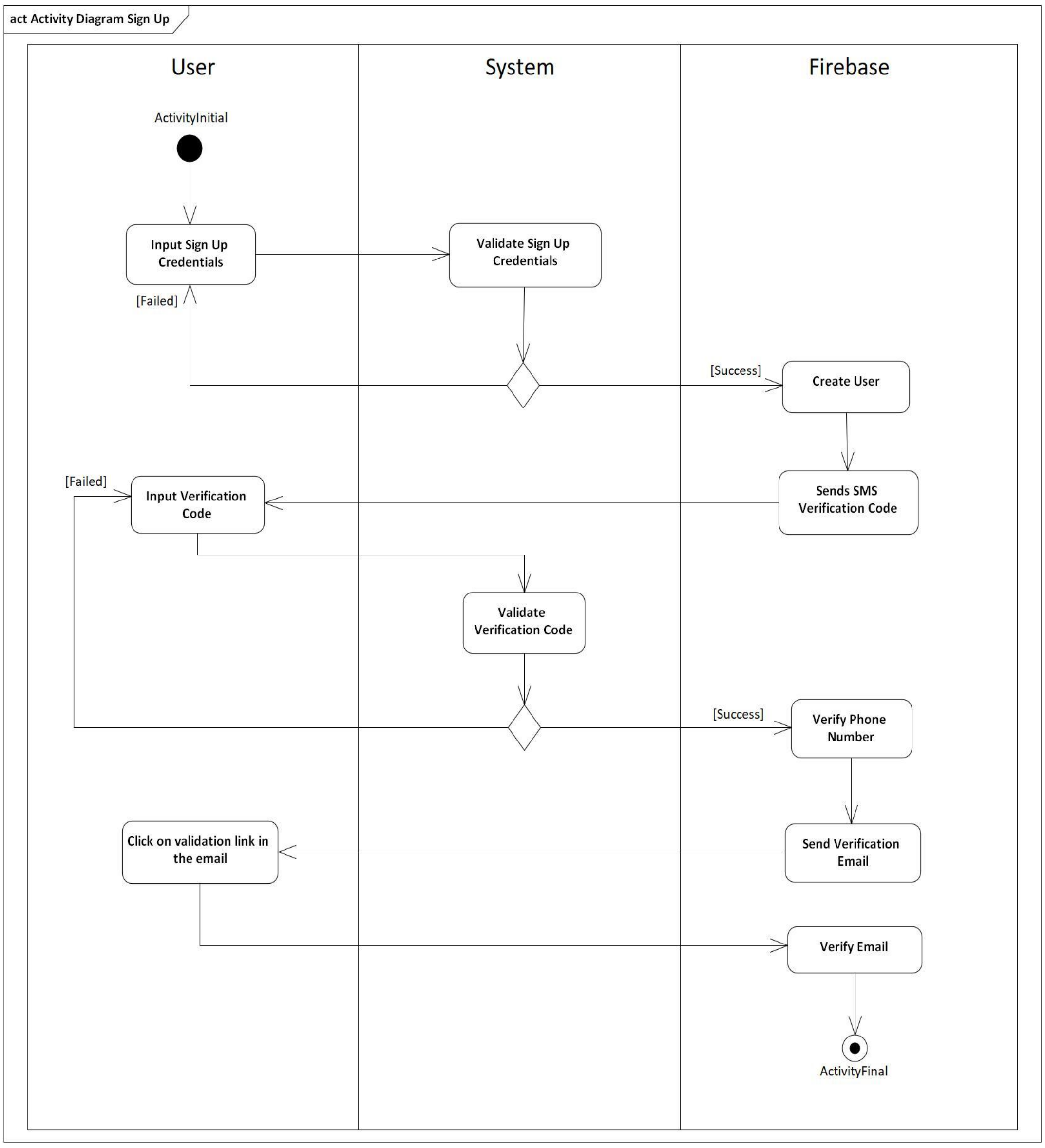

act Activity Diagram Sign Up
User
System
Firebase
ActivityInitial
Input Sign Up Credentials
Validate Sign Up Credentials
[Failed]
[Success]
Create User
Sends SMS Verification Code
[Failed]
Input Verification Code
Validate Verification Code
[Success]
Verify Phone Number
Send Verification Email
Click on validation link in the email
Verify Email
ActivityFinal

Figure 5.6: The Activity Diagram for Sign Up

5.2.3.3 View Product List

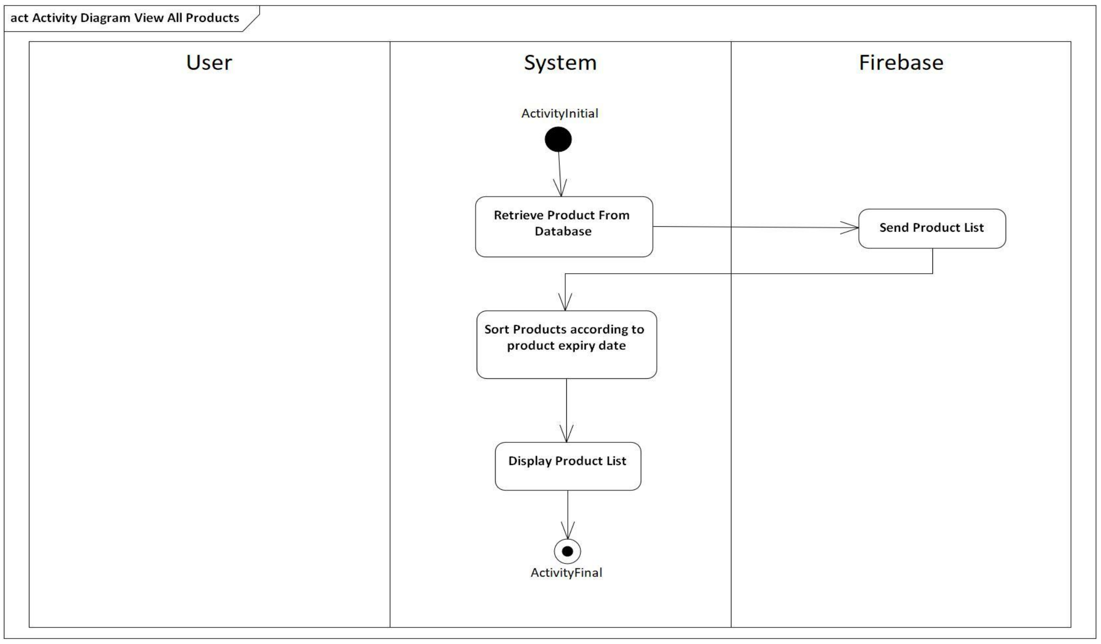

act Activity Diagram View All Products
User
System
Firebase
ActivityInitial
Retrieve Product From Database
Send Product List
Sort Products according to product expiry date
Display Product List
ActivityFinal

Figure 5.7: The Activity Diagram of View Product List

5.2.3.4 Add Product

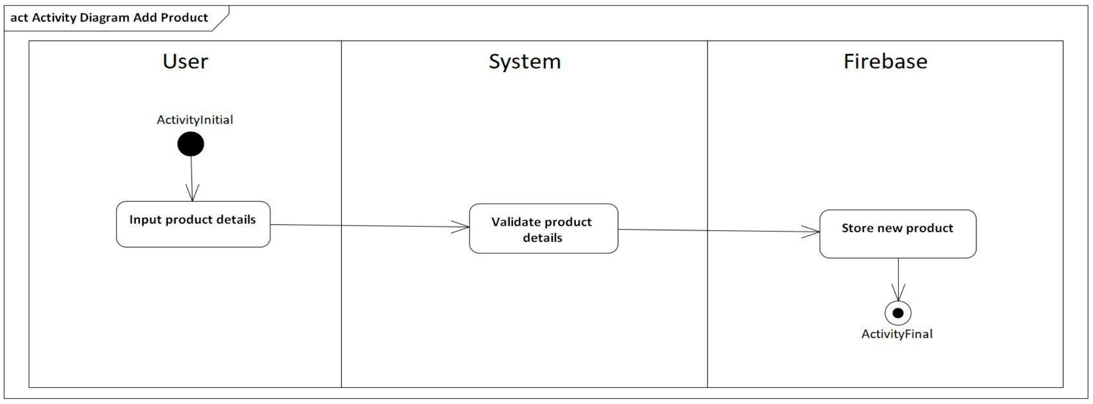


Figure 5.8: The Activity Diagram of Add Product

5.2.3.5 View Product

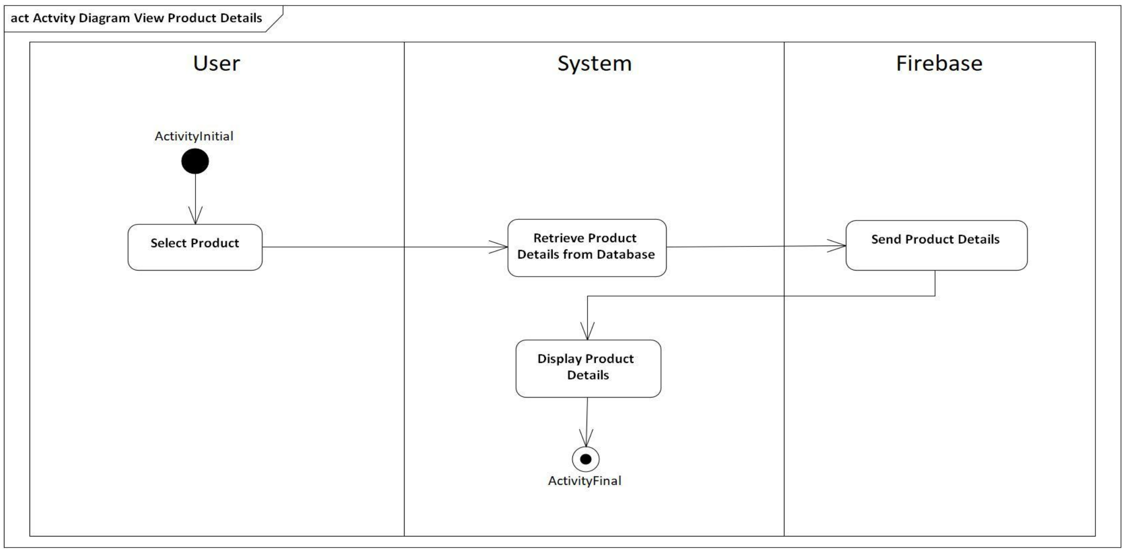


Figure 5.9: The Activity Diagram of View Product

5.2.3.6 Edit Product

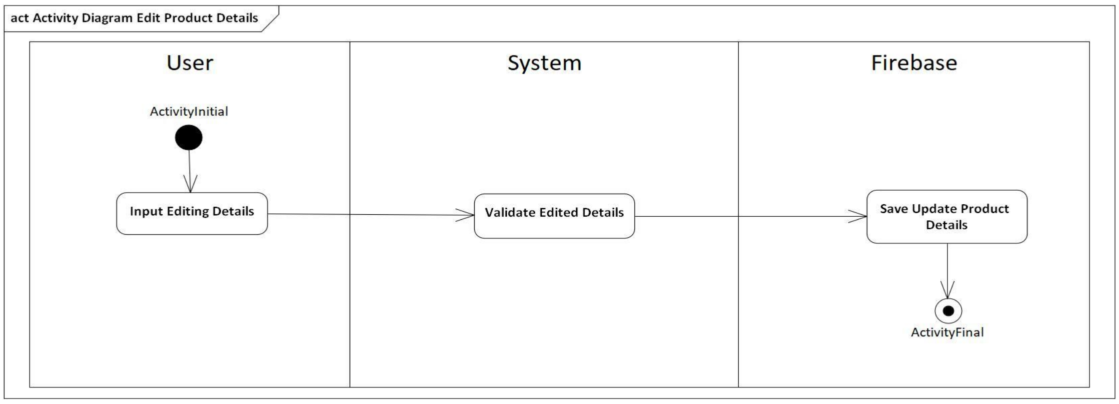


Figure 5.10: The Activity Diagram of Edit Product

5.2.3.7 Delete Product

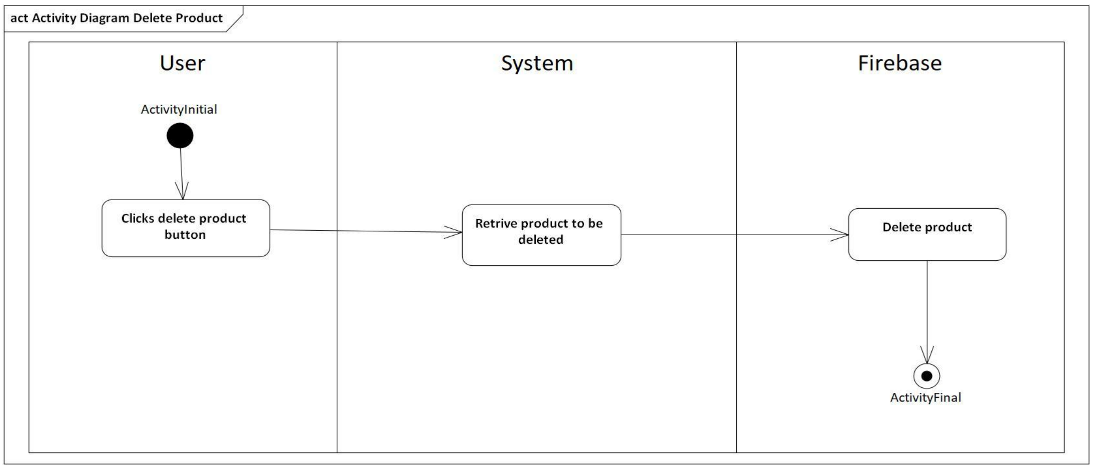

act Activity Diagram Delete Product
User
System
Firebase
ActivityInitial
Clicks delete product button
Retrive product to be deleted
Delete product
ActivityFinal

Figure 5.11: The Activity Diagram of Delete Product

5.2.3.8 Search Product

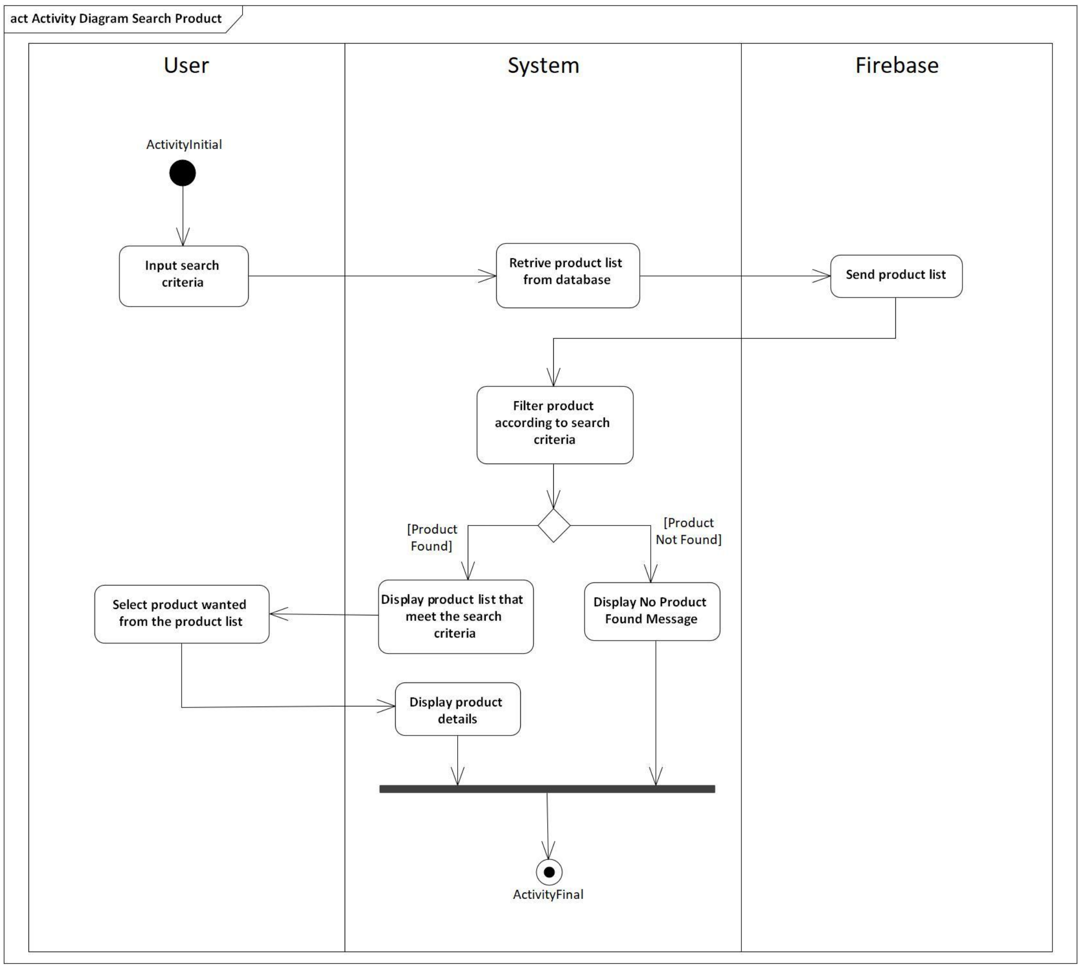

act Activity Diagram Search Product
User
System
Firebase
ActivityInitial
Input search criteria
Retrive product list from database
Send product list
Filter product according to search criteria
[Product Found]
[Product Not Found]
Select product wanted from the product list
Display product list that meet the search criteria
Display No Product Found Message
Display product details
ActivityFinal

Figure 5.12: The Activity Diagram of Search Product

5.2.3.9 Add Category

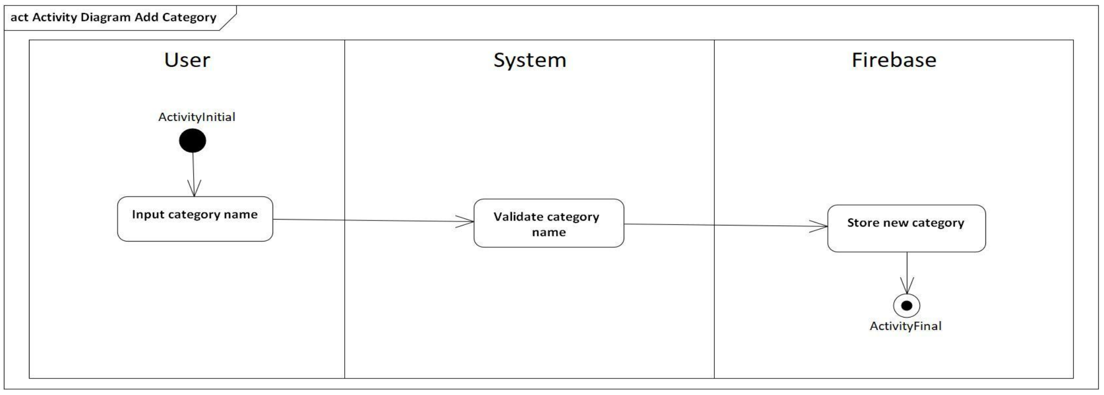


Figure 5.13: The Activity Diagram of Add Category

5.2.3.10 View Product in Category

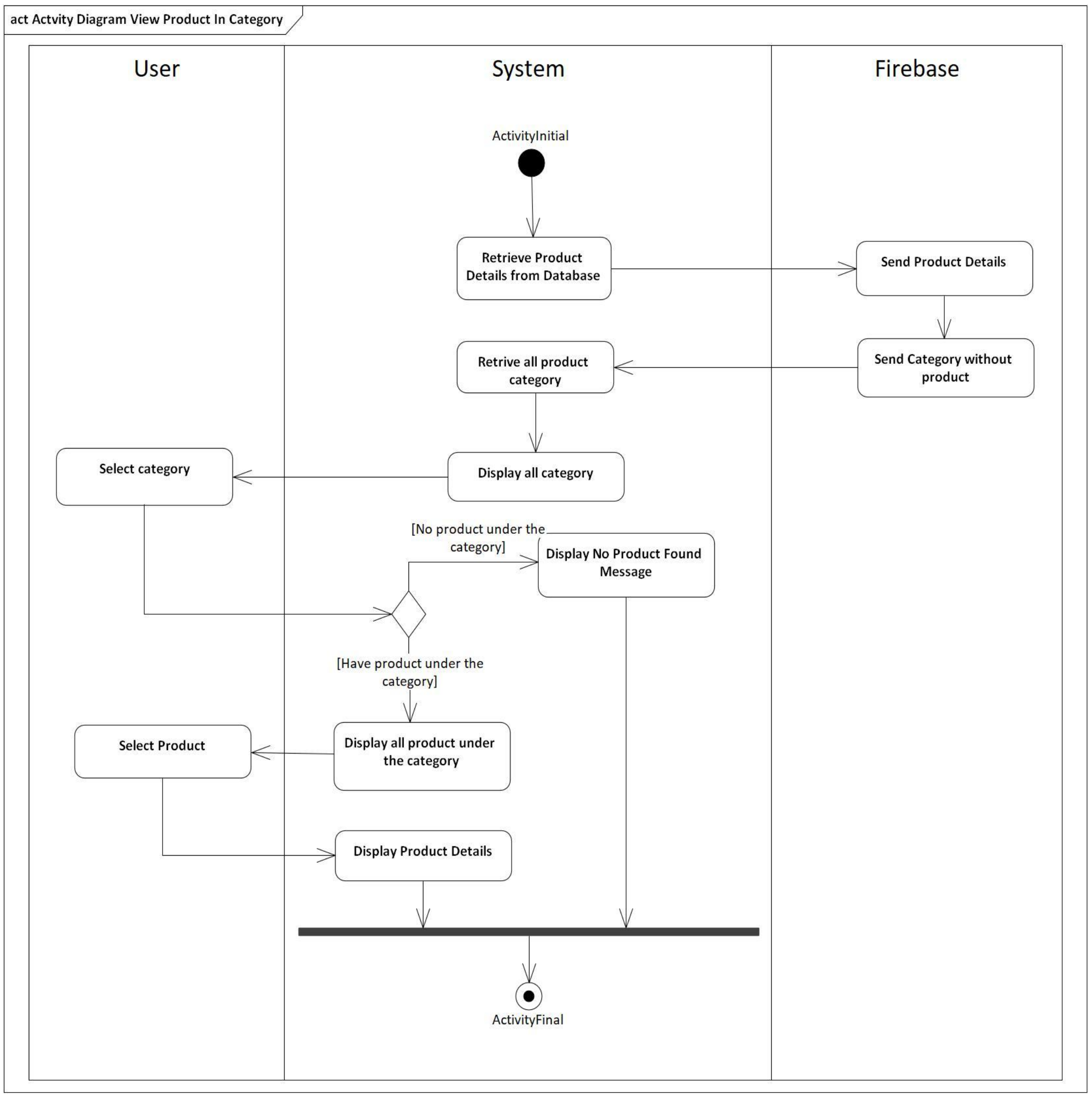

act Actvity Diagram View Product In Category
User
System
Firebase
ActivityInitial
Retrieve Product Details from Database
Send Product Details
Send Category without product
Retrive all product category
Display all category
Select category
[No product under the category]
Display No Product Found Message
[Have product under the category]
Display all product under the category
Select Product
Display Product Details
ActivityFinal

Figure 5.14: The Activity Diagram of View Product in Category

5.2.3.11 Edit Category

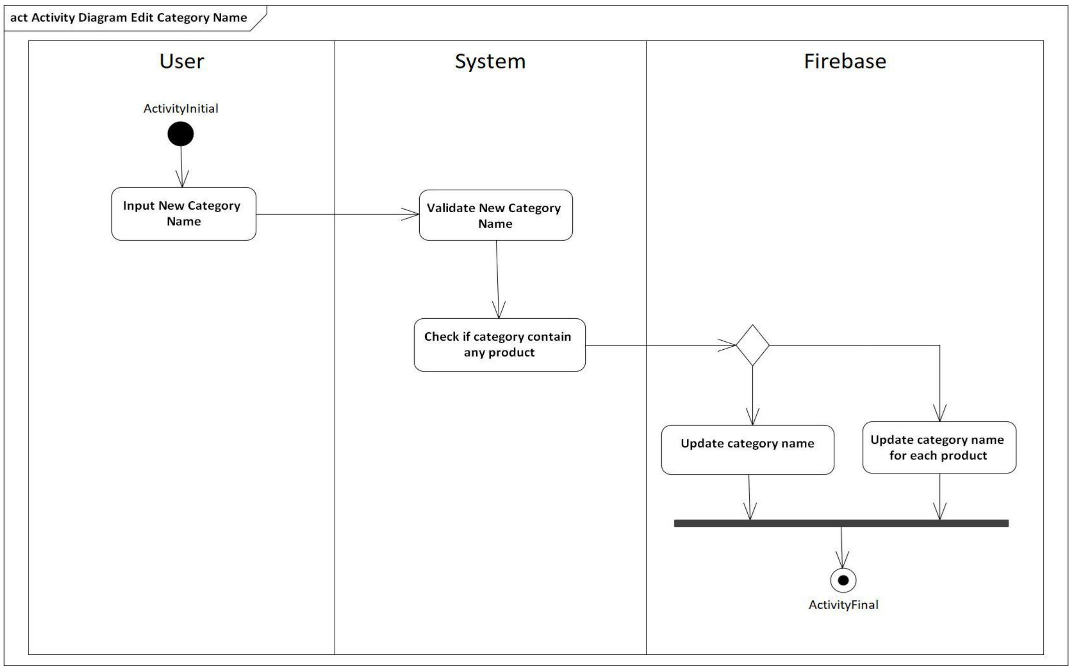

act Activity Diagram Edit Category Name
User
System
Firebase
ActivityInitial
Input New Category Name
Validate New Category Name
Check if category contain any product
Update category name
Update category name for each product
ActivityFinal

Figure 5.15: The Activity Diagram of Edit Category

5.2.3.12 Delete Category

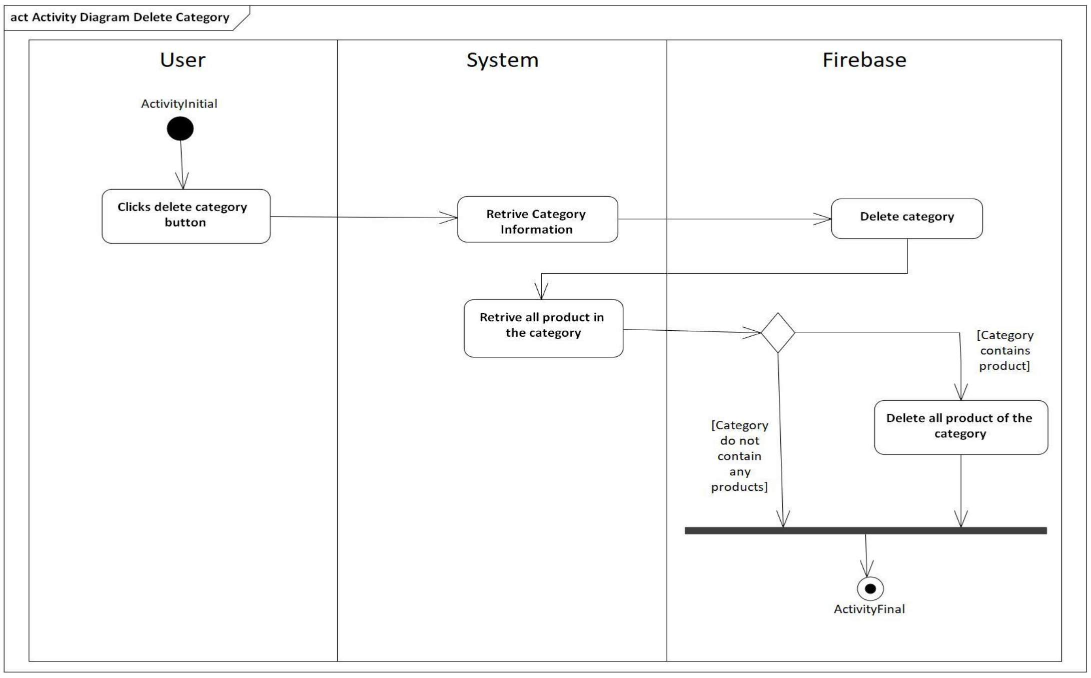

act Activity Diagram Delete Category
User
System
Firebase
ActivityInitial
Clicks delete category button
Retrive Category Information
Delete category
Retrive all product in the category
[Category contains product]
Delete all product of the category
[Category do not contain any products]
ActivityFinal

Figure 5.16: The Activity Diagram of Delete Category

5.2.3.13 Search Product with Barcode

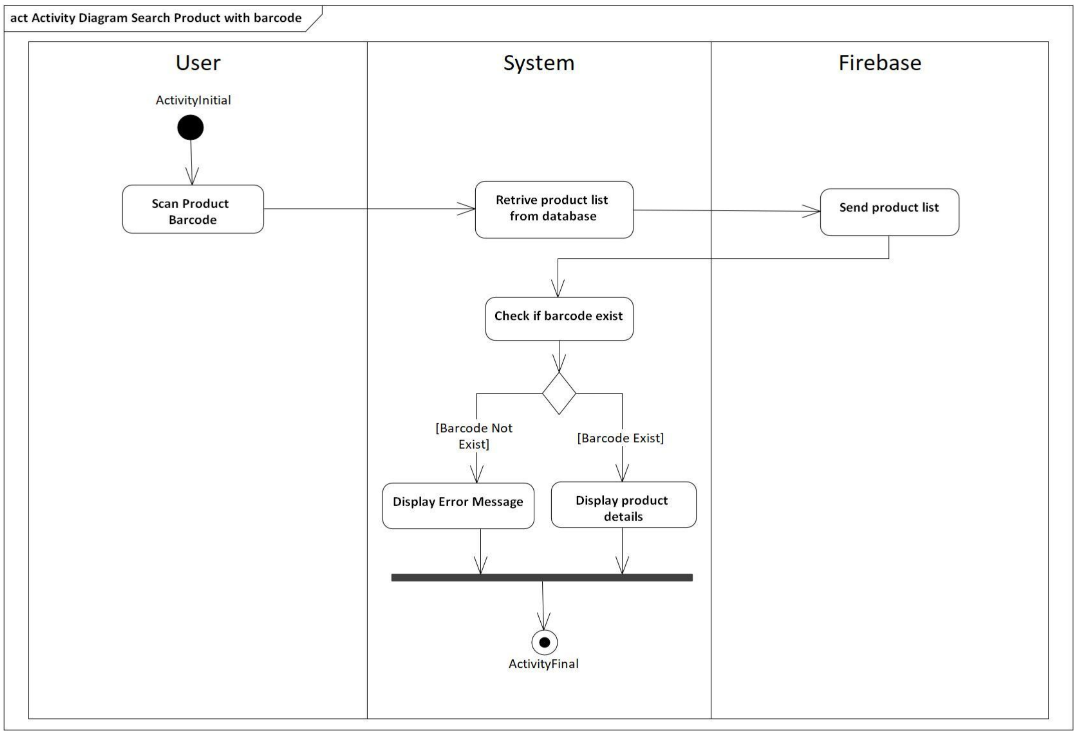


Figure 5.17: The Activity Diagram of Search Product with Barcode

5.2.3.14 View Shopping List

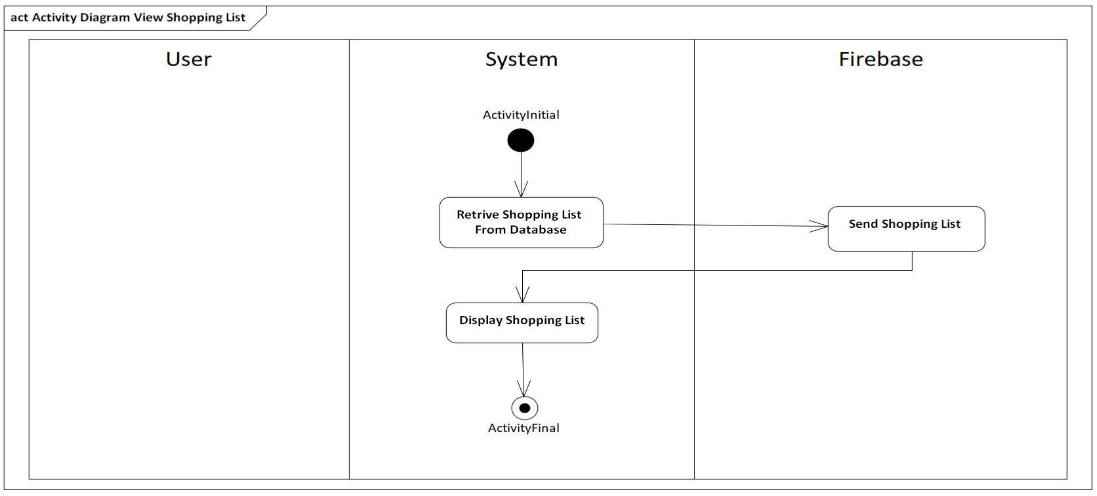


Figure 5.18: The Activity Diagram of View Shopping List

5.2.3.15 Add Shopping List Item

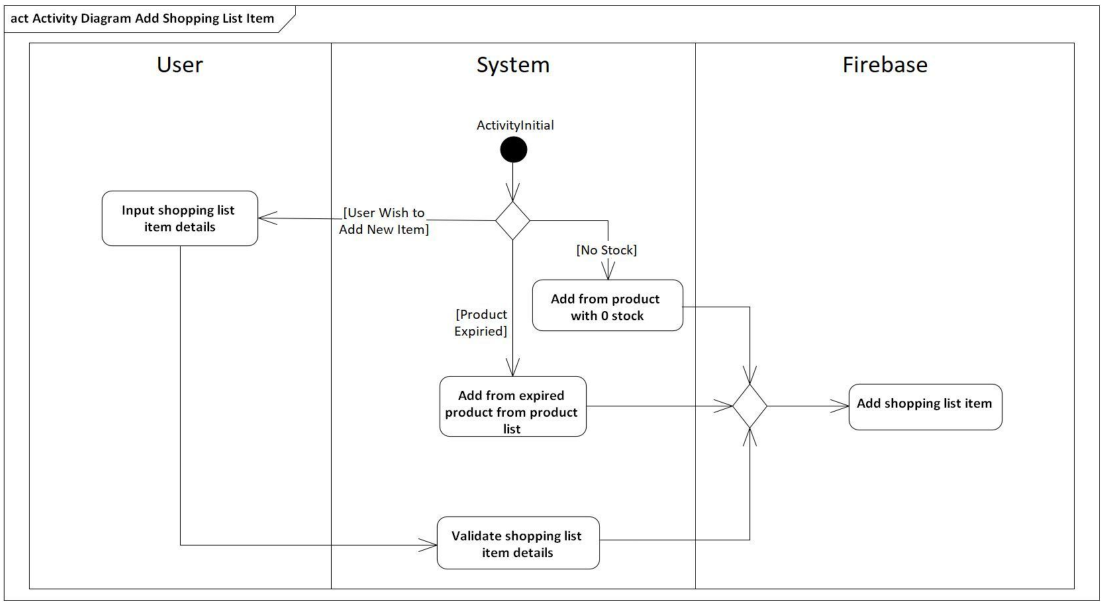

act Activity Diagram Add Shopping List Item
User
System
Firebase
ActivityInitial
Input shopping list item details
[User Wish to Add New Item]
[No Stock]
Add from product with 0 stock
[Product Expiried]
Add from expired product from product list
Add shopping list item
Validate shopping list item details

Figure 5.19: The Activity Diagram of Add Shopping List Item

5.2.3.16 Edit Shopping List Item

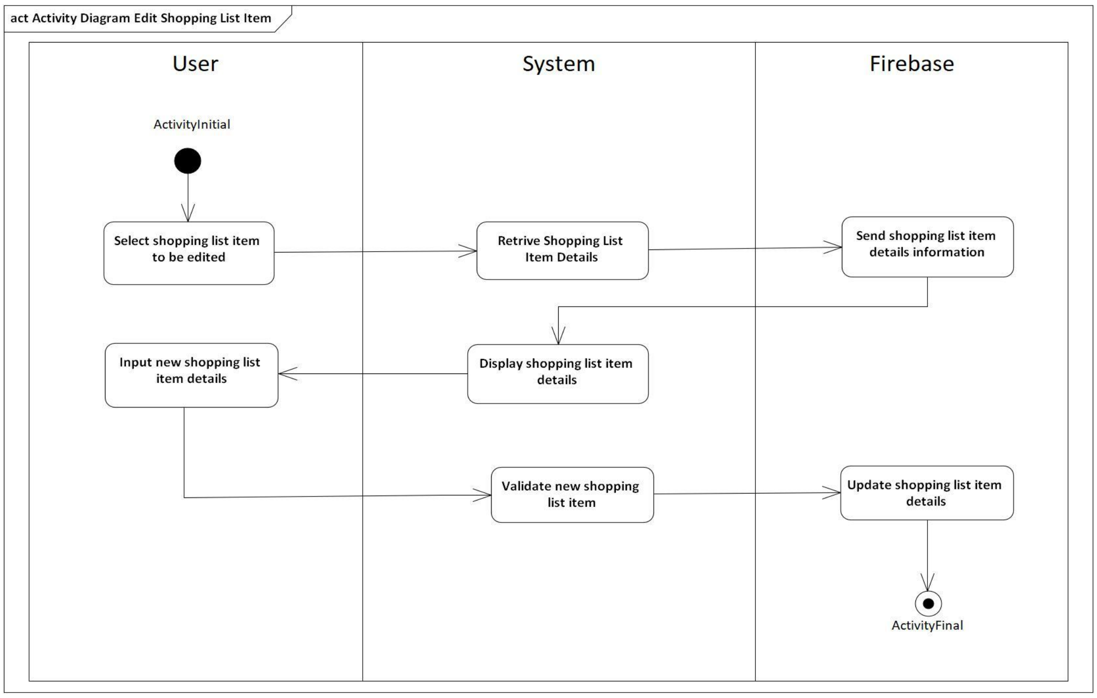

act Activity Diagram Edit Shopping List Item
User
System
Firebase
ActivityInitial
Select shopping list item to be edited
Retrive Shopping List Item Details
Send shopping list item details information
Input new shopping list item details
Display shopping list item details
Validate new shopping list item
Update shopping list item details
ActivityFinal

Figure 5.20: The Activity Diagram of Edit Shopping List Item

5.2.3.17 Delete Shopping List Item

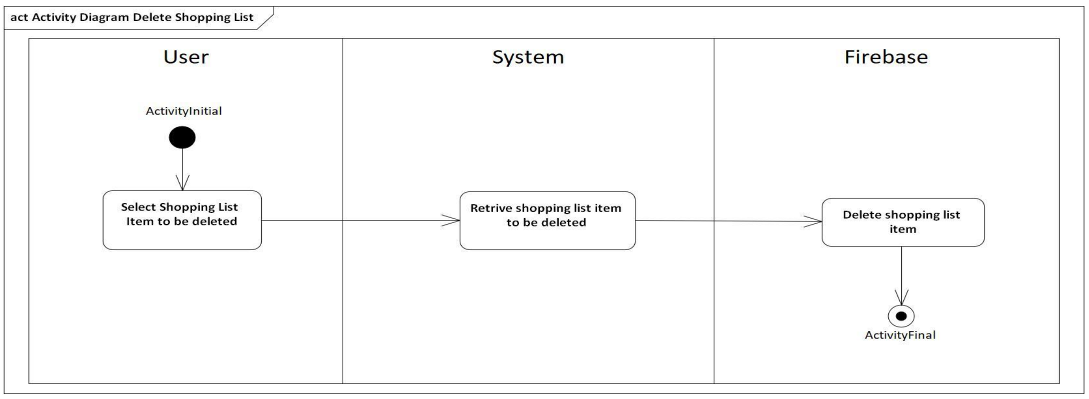

act Activity Diagram Delete Shopping List
User
System
Firebase
ActivityInitial
Select Shopping List Item to be deleted
Retrive shopping list item to be deleted
Delete shopping list item
ActivityFinal

Figure 5.21: The Activity Diagram of Delete Shopping List Item

5.2.3.18 Set Shopping List to Done

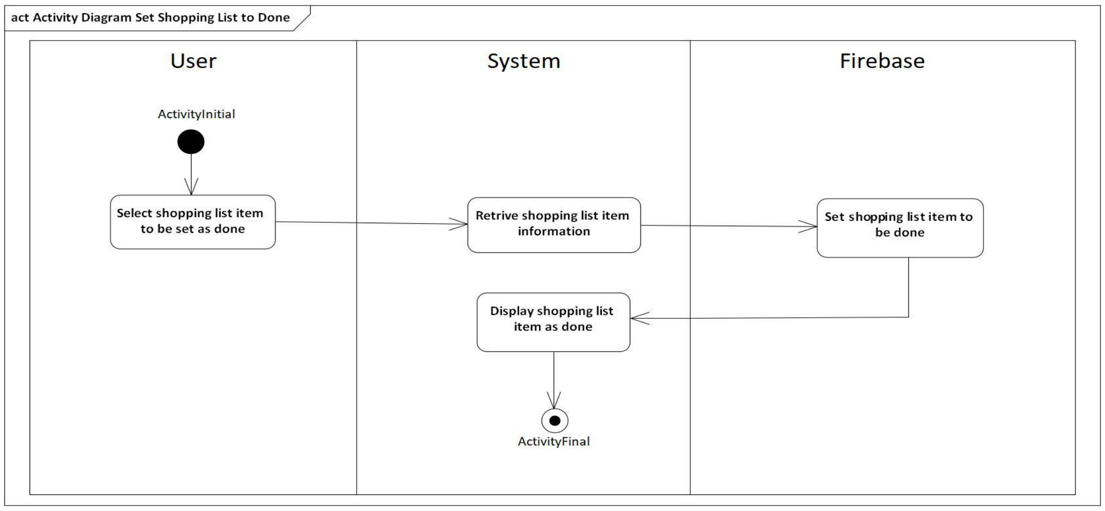

act Activity Diagram Set Shopping List to Done
User
System
Firebase
ActivityInitial
Select shopping list item to be set as done
Retrive shopping list item information
Set shopping list item to be done
Display shopping list item as done
ActivityFinal

Figure 5.22: The Activity Diagram of Set Shopping List to Done

5.2.3.19 Change Preferred Alert Time

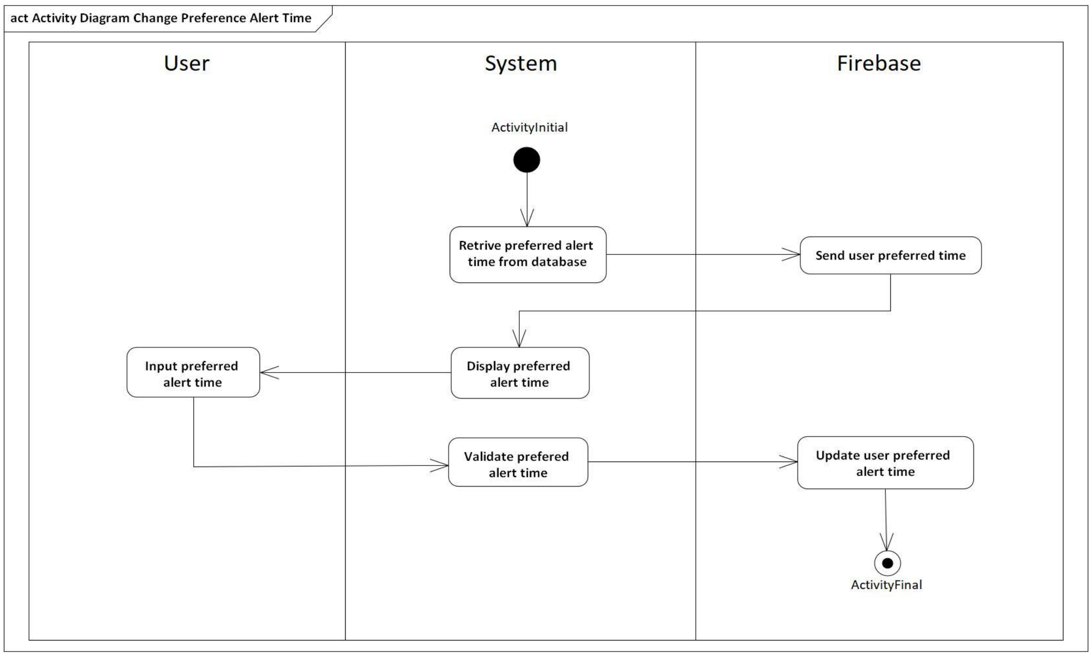

act Activity Diagram Change Preference Alert Time
User
System
Firebase
ActivityInitial
Retrive preferred alert time from database
Send user preferred time
Display preferred alert time
Input preferred alert time
Validate prefered alert time
Update user preferred alert time
ActivityFinal

Figure 5.23: The Activity Diagram of Change Preferred Alert Time

5.3 Database Design

In this section, the database design for the application is discussed.

5.3.1 Entity Relational Diagram (ERD)

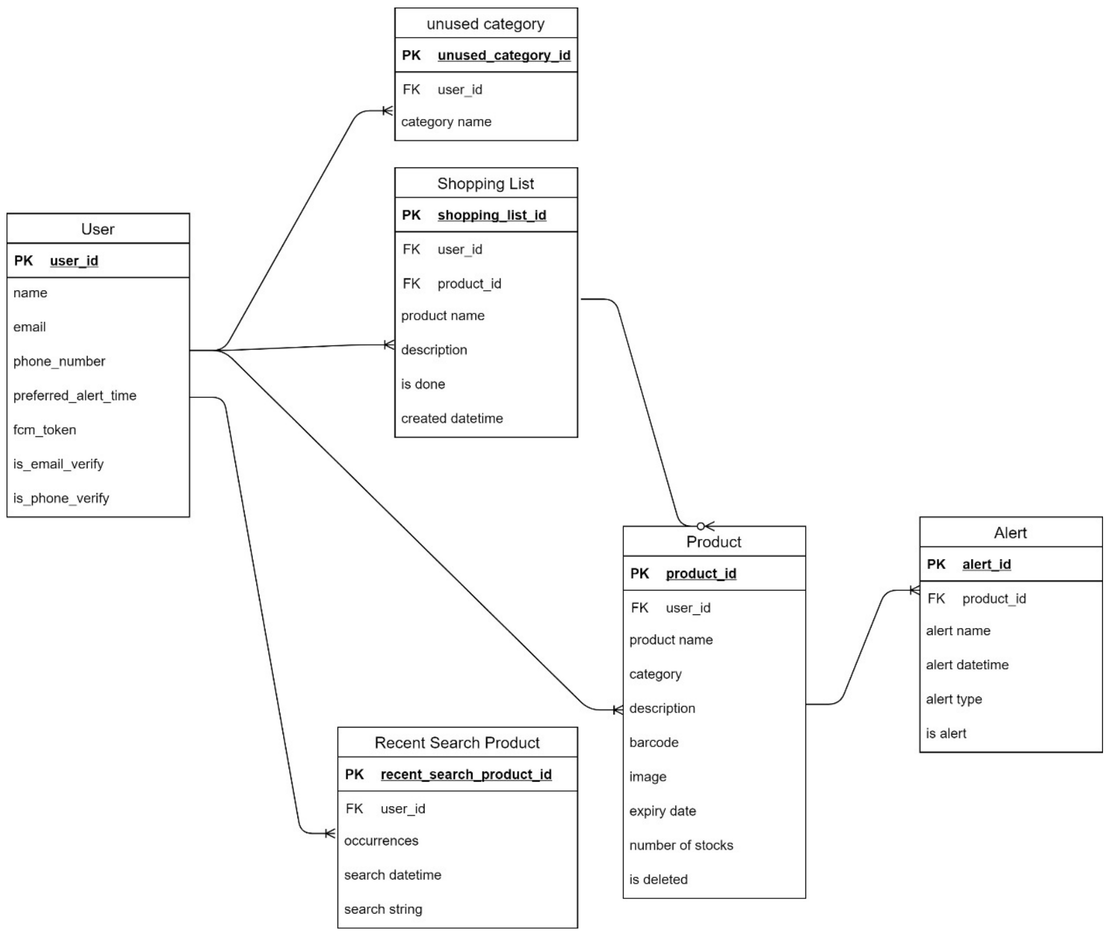


Figure 5.24: The Entity Relationship Diagram for Expiry Reminder App

5.3.2 Data Dictionary

Entity Name: User

Table 5.1: Data Dictionary for User Entity

Attribute	**Description**	**Data Type**	**PK/ FK**	**NULL**
User id	Unique identification for user	Varchar	PK	N
Name	User name	Varchar		N
Email	User email address used to receive an alert notification email when a product expired	Varchar		Y
Phone number	User phone number used to receive alert notification SMS when a product expired	Varchar		Y
Preferred alert time	User prefer time to receive notification alert when a product expired	Varchar		N
FCM token	User device token to receive push notification alert when a product expired	Varchar		N
Is email verify	Flags to indicate if user verifies his/her email address: True for email is verified, False for email is not verified	Boolean		N
Is phone verify	Flags to indicate if user verifies his/her phone number: True for phone number is verified, False for phone number is not verified	Boolean		N

Entity Name: Product

Table 5.2: Data Dictionary for Product Entity

Attribute	**Description**	**Data Type**	**PK/ FK**	**NULL**
Product id	Unique identification for product	Varchar	PK	N
User id	The user that owns this product	Varchar	FK	N
Product name	Name of the product	Varchar		N
Category	Category of the product	Varchar		N
Description	Description of the product	Varchar		Y
Barcode	Barcode of the product that can be used to search product later	Varchar		Y
Image	Image URL of the product	Varchar		Y
Expiry date	The expiry date of the product	Datetime		N
Number of stocks	Number of stocks the product left	Integer		N
Is Deleted	Flags to indicate if a product is deleted: True for a product that is deleted, False for otherwise	Boolean		N

Entity Name: Alert

Table 5.3: Data Dictionary for Alert Entity

Attribute	**Description**	**Data Type**	**PK/ FK**	**NULL**
Alert id	Unique identification for alert	Varchar	PK	N
Product Id	The product that consists of this alert	Varchar	FK	N
Alert name	Name of the alert	Varchar		N
Alert datetime	Date and time of the alert	Timestamp		N
Alert type	Array to indicates the alert type selected: 0 for Push Notification, 1 for SMS Notification, 2 for Email Notification. An alert can consist of one or more alert types.	Array <Int>		N
Is Alert	Flags to indicate whether the alert is sent to the user: True for sent, False for not yet sent.	Boolean		N

Entity Name: Shopping List

Table 5.4: Data Dictionary for Shopping List Entity

Attribute	**Description**	**Data Type**	**PK/ FK**	**NULL**
Shopping list id	Unique identification for shopping list item	Varchar	PK	N
User id	The user that owns this shopping list item	Varchar	FK	N
Product id	The product in this shopping list. Set to NULL if the shopping list is created by the user.	Varchar	FK	Y
Product name	Product Name for the shopping list item	Varchar		N
Descriptio n	Product Description for the shopping list item	Varchar		Y
Is Done	Flags to indicate whether the is completed: True for complete, False for incomplete.	Boolean		N
Created datetime	Date and time when the shopping list item is created	Datetime		N

Entity Name: unused category

Table 5.5: Data Dictionary for Unused Category Entity

Attribute	**Description**	**Data Type**	**PK/ FK**	**NULL**
Unused category id	Unique identification for unused category	Varchar	PK	N
User Id	The user that created the unused category	Varchar	FK	N
Category name	Name of the unused category	Varchar		N

Entity Name: Recent Search Product

Table 5.6: Data Dictionary for Recent Search Product Entity

Attribute	**Description**	**Data Type**	**PK/ FK**	**NULL**
Recent search product id	Unique identification for recent search product	Varchar	PK	N
User Id	The user that searched the product recently	Varchar	FK	N
Occurrence	Number of occurrences the user searched on the same product	Integer		N
Search string	The searched product's name	Varchar		N
Search Datetime	The latest date and time when the user searches for the product	Datetime		N

5.4 User Interface Design

5.4.1 Sign In Page

If a user is not signed in or a user is a first-time user, the user will be landed on the login page. From the login page, a user is allowed to enter his or her email address and password to log in. If a user prefers using his or her social media account to login, the user can log in through his or her Facebook account or Google account. If a user does not have an account and wishes to create one, a Sign-Up button is available to navigate the user to the Sign Up page. If the user forgets the password for his or her account, the user can tap on the Forget Password button to navigate to the Forgot Password page. The figure below shows the User Interface for the Sign In Page of the mobile application.

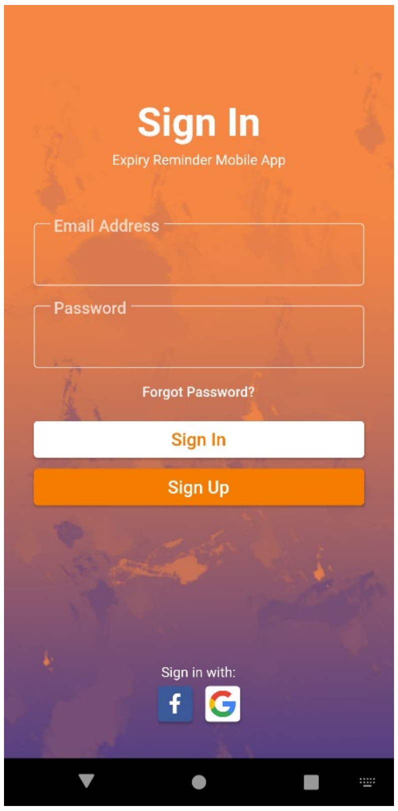


Figure 5.25: UI design for Login Page

5.4.2 Forgot Password Page

The user needs to enter his or her registered email address to get the reset password email. If the user submits an empty email, an error message will be prompt to ask the user to enter his or her email address. The figure below shows the UI of Forgot Password page.

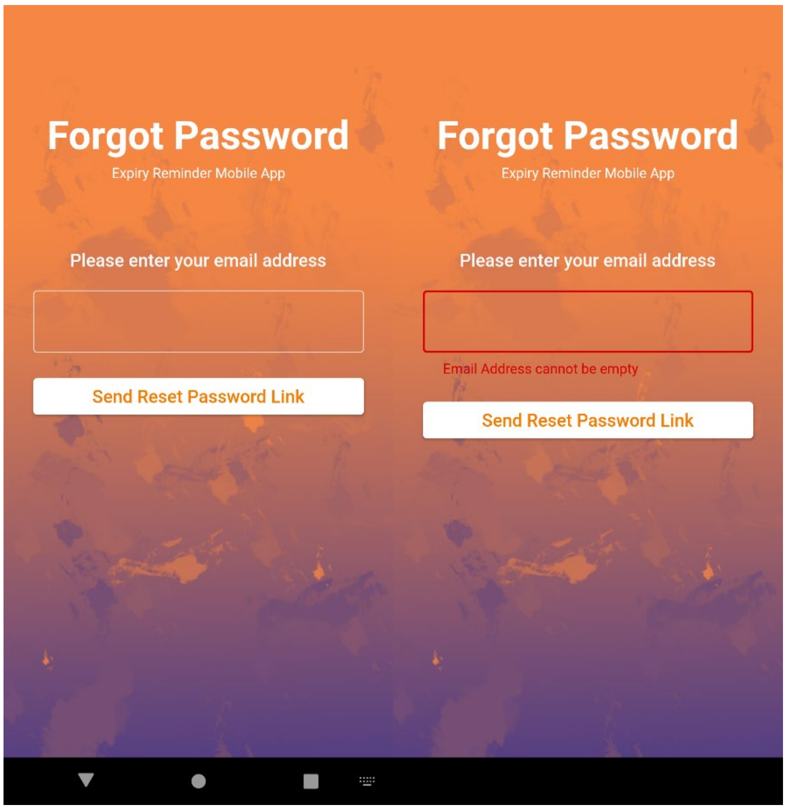
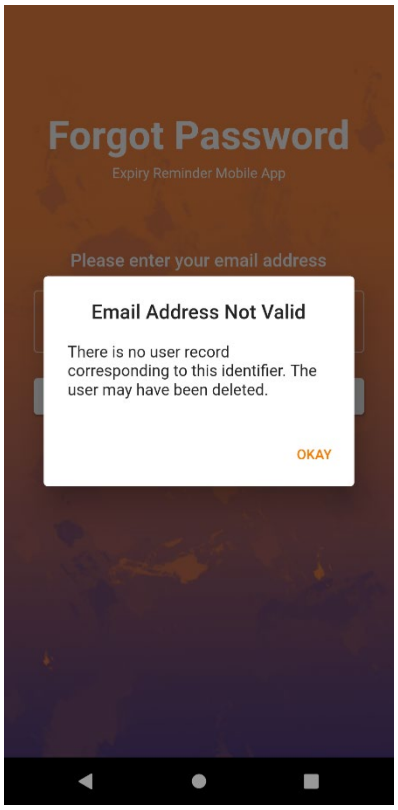

Forgot Password
Expiry Reminder Mobile App
Please enter your email address
Send Reset Password Link
Forgot Password
Expiry Reminder Mobile App
Please enter your email address
Email Address cannot be empty
Send Reset Password Link

Figure 5.26: UI design for Forgot Password page

If the email address is not registered before or do not exists, an error message will prompt. The user needs to enter the correct email address to proceed. The figure below shows the alert dialog for invalid email.

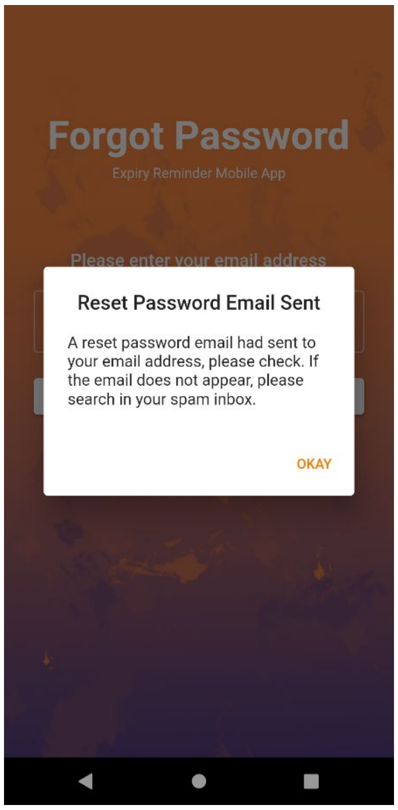


Email Address Not Valid

There is no user record corresponding to this identifier. The user may have been deleted.

Figure 5.27: Invalid Email Alert Dialog

If the email is sent successfully, a successful message will be prompt and the user will be navigated back to the Sign In page. The user needs to go to his or her inbox to tap on the reset password link provided by the email in order to reset his or her password. The figure below shows the alert dialog for the successful message.

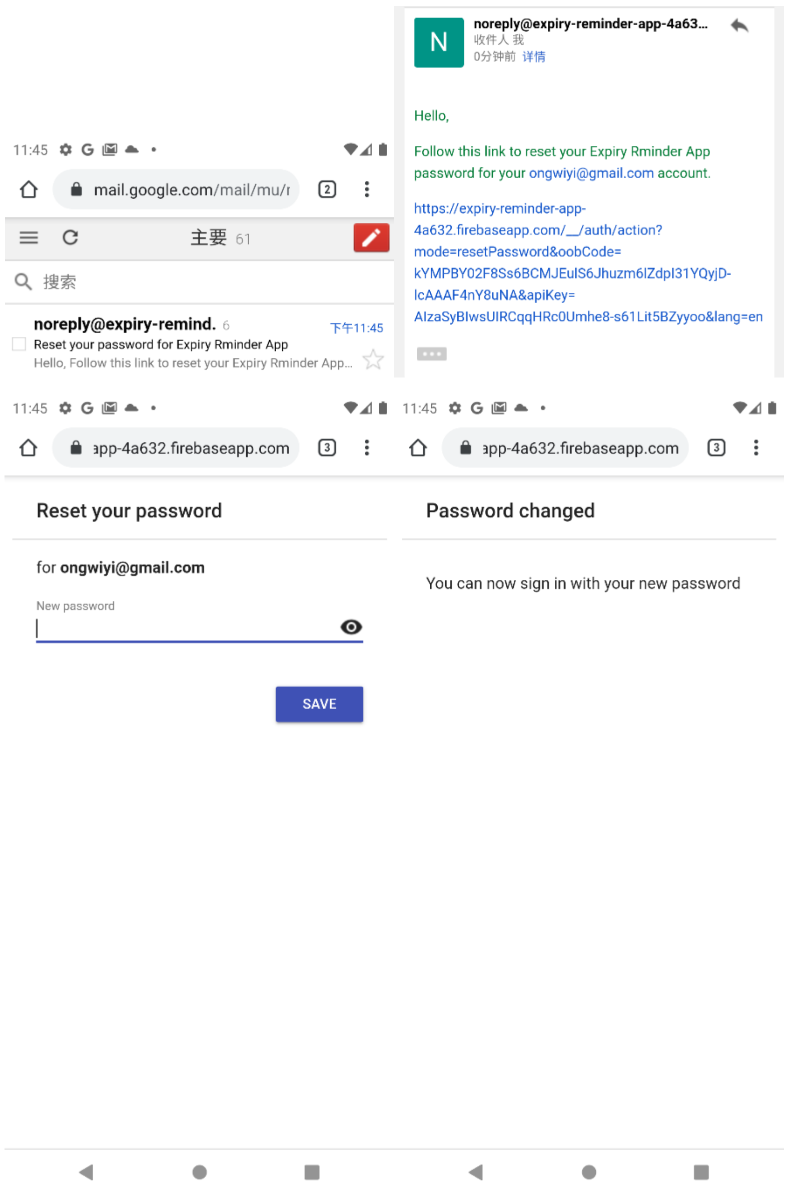


Reset Password Email Sent

A reset password email had sent to your email address, please check. If the email does not appear, please search in your spam inbox.

Figure 5.28: Successful Message Alert Dialog

The figure below shows how the password can be reset using Firebase Auth.



noreply@expiry-reminder-app-4a63...
收件人 我
0分钟前 详情

Hello,

Follow this link to reset your Expiry Rminder App password for your ongwiyi@gmail.com account.

https://expiry-reminder-app-4a632.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=kYMPBY02F8Ss6BCMJEuIS6Jhuzm6lZdpI31YQyjD-IcAAAF4nY8uNA&apiKey=AIzaSyBIwsUIRCqqHRc0Umhe8-s61Lit5BZyyoo&lang=en

11:45

mail.google.com/mail/mu/

主要 61

搜索

noreply@expiry-remind. 6 下午11:45
Reset your password for Expiry Rminder App
Hello, Follow this link to reset your Expiry Rminder App...

app-4a632.firebaseapp.com

Reset your password

for **ongwiyi@gmail.com**

New password

SAVE

app-4a632.firebaseapp.com

Password changed

You can now sign in with your new password

Figure 5.29: UI of Password Reset with Firebase Auth

5.4.3 Sign Up Page

To create an account, a user must enter every criterion in the Sign-Up form, which includes Username, Email Address, Password, Confirm Password and also Phone Number. Then, the user will be redirected back to the Sign In page to login into his or her account.

Upon successful sign-in, the user will be redirected to a page that contains instruction on how to validate his or her email address. A verification email will be sent to the email address entered by the user earlier in the Sign-Up form. This is to prevent any robot or fake account that will congest the server as if the database is hosted from the cloud.

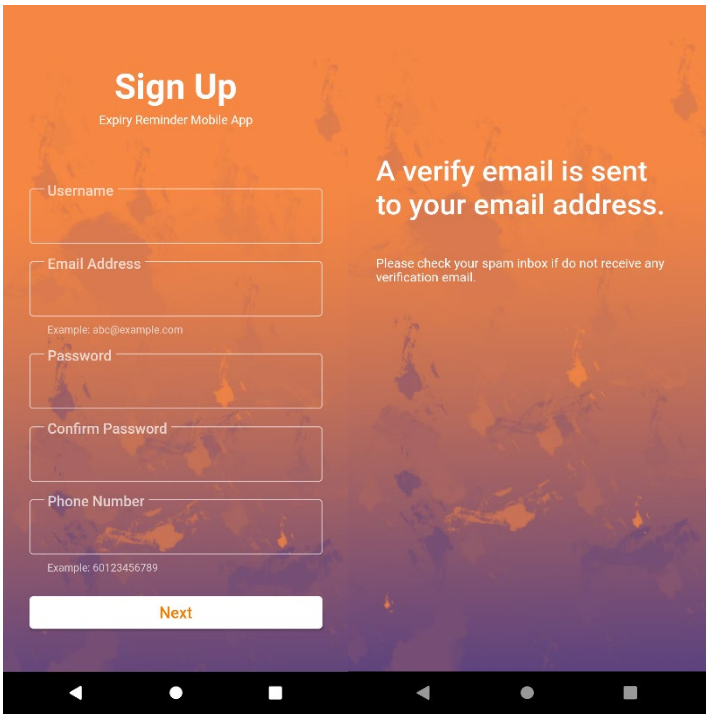

Sign Up
Expiry Reminder Mobile App
Username
Email Address
Example: abc@example.com
Password
Confirm Password
Phone Number
Example: 60123456789
Next
A verify email is sent to your email address.
Please check your spam inbox if do not receive any verification email.

Figure 5.30: UI design for Sign Up Page and Email Verification Page

A welcome page will be displayed to the first-time user when the email is verified. This is to briefly let the user know what the application does and what kind of feature is available to use in the application.

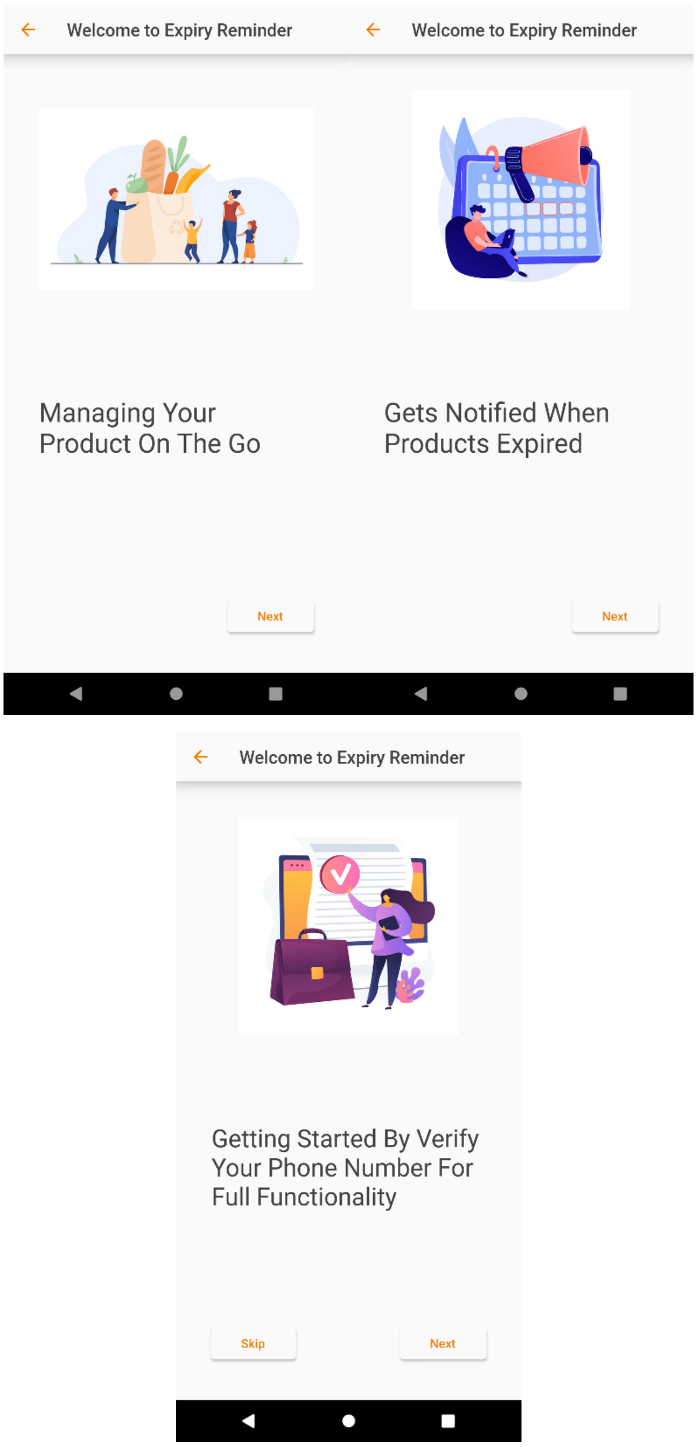

Welcome to Expiry Reminder
Managing Your Product On The Go
Next
Welcome to Expiry Reminder
Gets Notified When Products Expired
Next
Welcome to Expiry Reminder
Getting Started By Verify Your Phone Number For Full Functionality
Skip
Next

Figure 5.31: UI design for the First Time Login Instruction Page

Next, the user will have to verify his or her phone number by clicking the button to receive a verification SMS that contains 6 digits One Time Password (OTP) and enter the OTP to the textbox provided.

The verification process is to ensure the user can take advantage of the full functionality such as SMS Alert Notification and Email Alert Notification. If the user does not verify his or her phone number or email address, the user will be unable to choose SMS Alert Notification and Email Alert Notification when adding or editing an alert.

If a user accidentally quits or closes the application during the verification process, the user will have to conduct the same verification process stated above when the user reopens the application. The user can also verify his or her phone number in the Setting Page later.

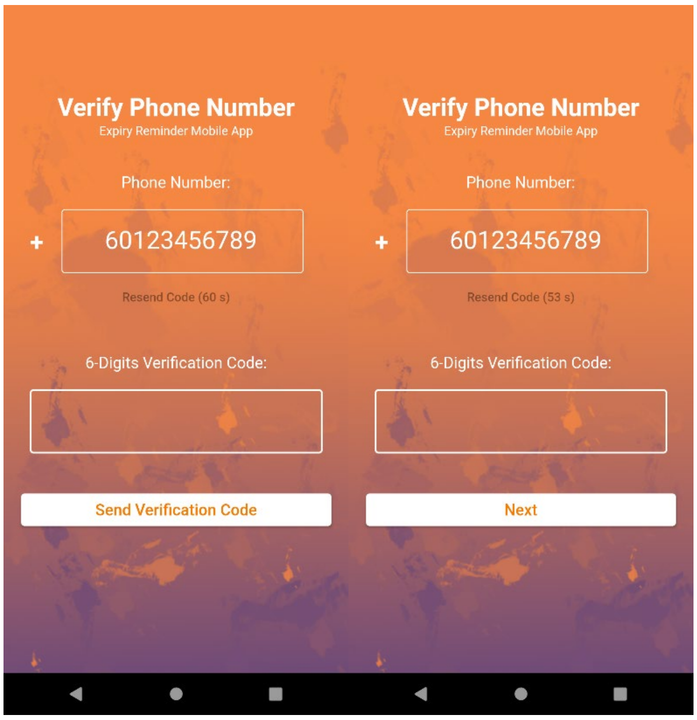
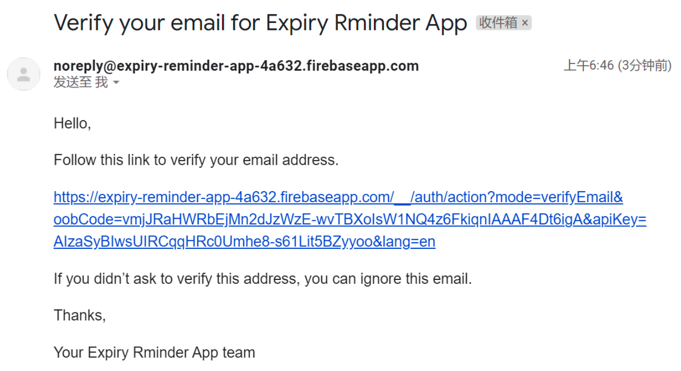


Figure 5.32: UI design for Phone Number Verification Page



Verify your email for Expiry Rminder App 收件箱 ×

noreply@expiry-reminder-app-4a632.firebaseapp.com 上午6:46 (3分钟前)
发送至 我

Hello,

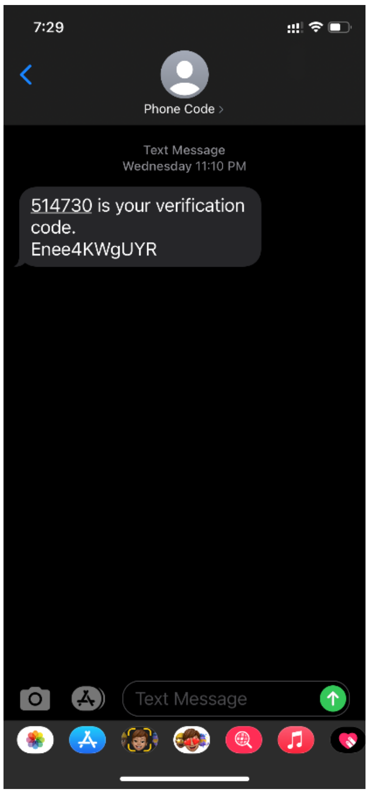
Follow this link to verify your email address.

https://expiry-reminder-app-4a632.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=vmjJRaHWRbEjMn2dJzWzE-wvTBXolsW1NQ4z6FkiqnIAAAF4Dt6igA&apiKey=AIzaSyBIwsUIRCqqHRc0Umhe8-s61Lit5BZyyoo&lang=en

If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your Expiry Rminder App team



7:29

Phone Code

Text Message
Wednesday 11:10 PM

514730 is your verification code.
Enee4KWgUYR

Figure 5.33: Sample Screenshots of Verification Email and Verification SMS

5.4.4 Home Page

On the home page, there are two buttons available at the app bar. The menu button on the left is to display a navigation drawer that navigates the user to other pages, while the add button is for users to add a new product. Below the app bar is a dashboard that displays the total number of expired products, soon to expire product and safe to use the product. Then, there is a search bar below the dashboard. The search bar is to allow users to search the product they want based on the product name. A list of products is displayed below with different colour indicators to help users to differentiate the severity level of the product status. The product closer to its expiry date will be on top of the list. When a user taps on a product, the user will be able to view product details.

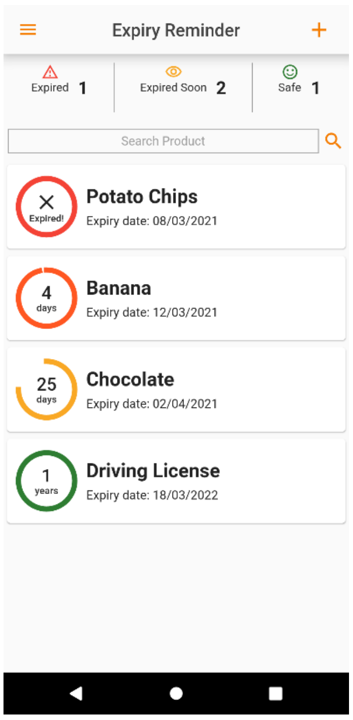


Figure 5.34: UI design for Home Page

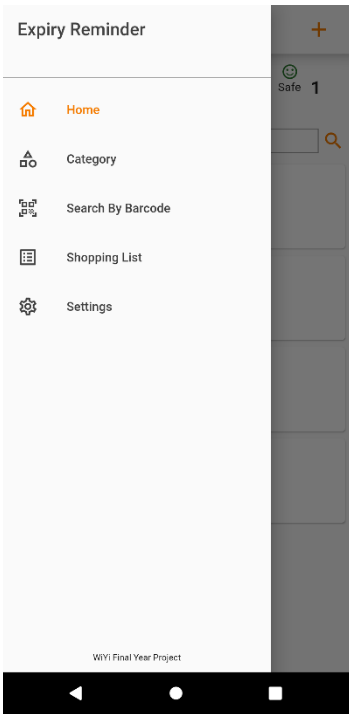


Figure 5.35: UI design for Navigation Drawer

5.4.5 Add Product Page

When the user taps on the add button at the home page, the user will be navigated to the Add Product Page. On this page, users can add an image of the product, named the product and give the product a description. Plus, users can add a new category or select a defined category for the product. Besides, users can use the Scan Date function to get the expiry date on the packaging with the Convolutional Neural Network Deep Learning Model named Inception Resnet V2. Users can also scan the product barcode to be used for search product with barcode in the future. At the bottom of the page, a button that allows users to use OCR to scan product name.

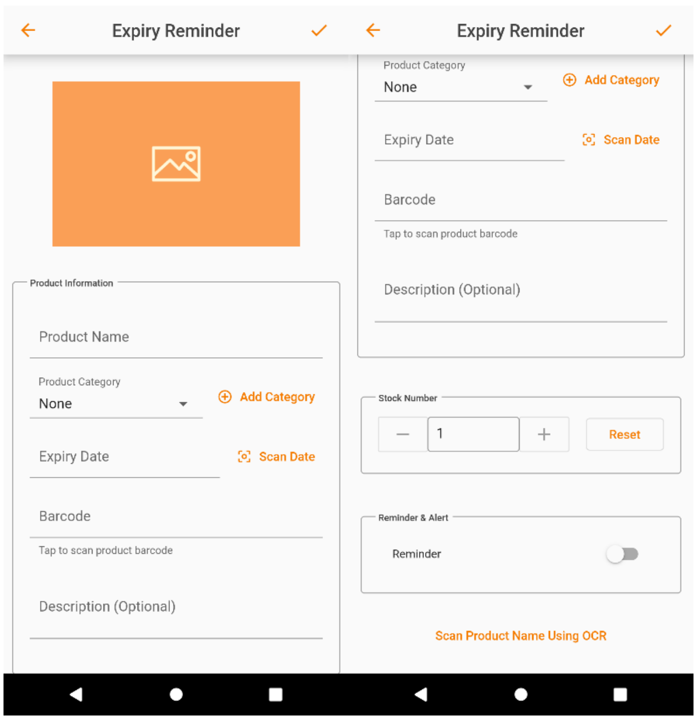
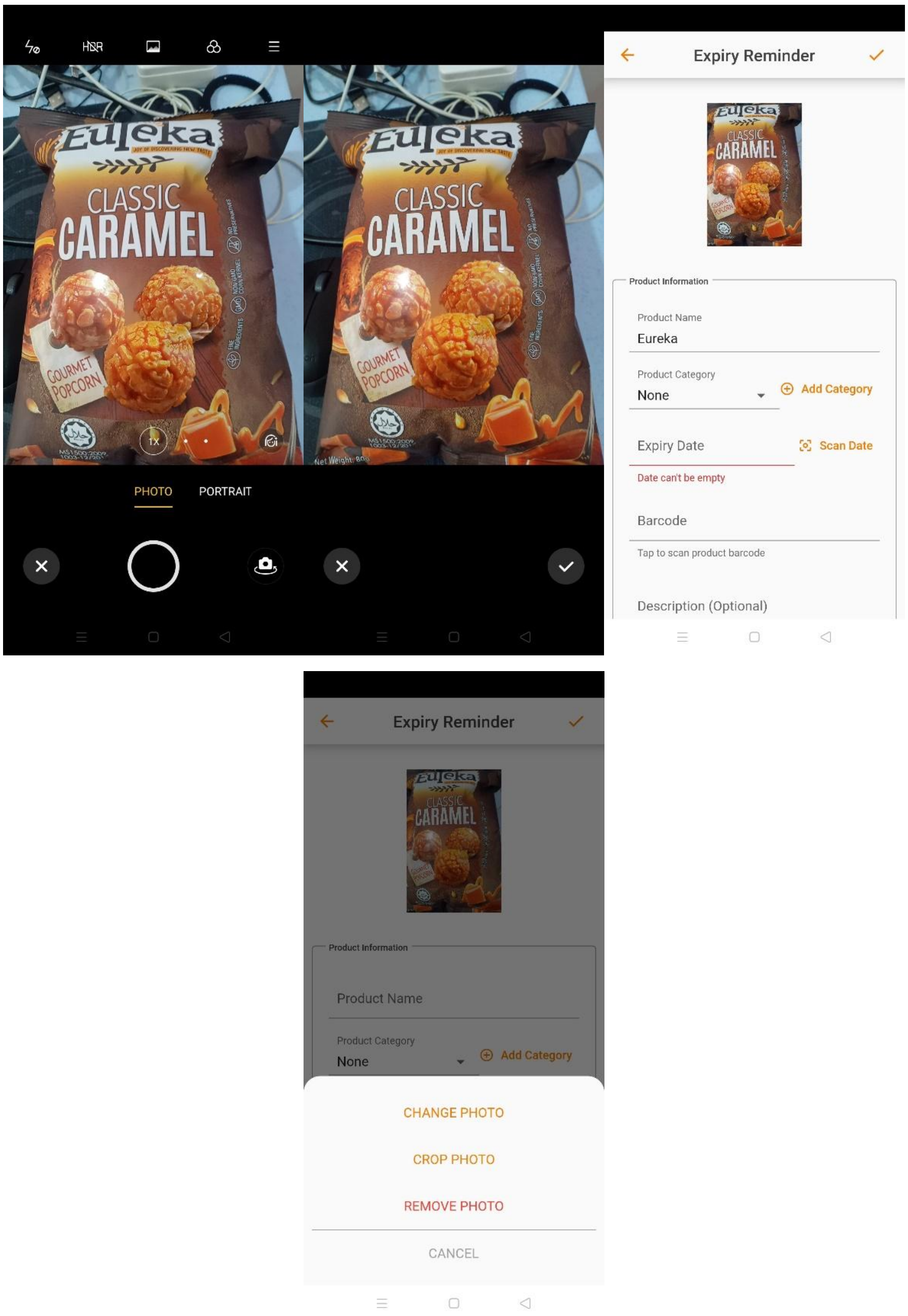


Figure 5.36: UI design for Add Product Page

5.4.5.1 Add Image

Users can add an image of the product by taking a photo of the product. When there is already an image uploaded, an alert dialog will prompt to let users choose whether to replace, edit or delete the image uploaded previously.



Expiry Reminder

Product Information

Product Name
Eureka

Product Category
None

Add Category

Expiry Date

Scan Date

Date can't be empty

Barcode

Tap to scan product barcode

Description (Optional)

Expiry Reminder

Product Information

Product Name

Product Category
None

Add Category

CHANGE PHOTO

CROP PHOTO

REMOVE PHOTO

CANCEL

Figure 5.37: UI design for Add Image Feature

5.4.5.2 Add Category

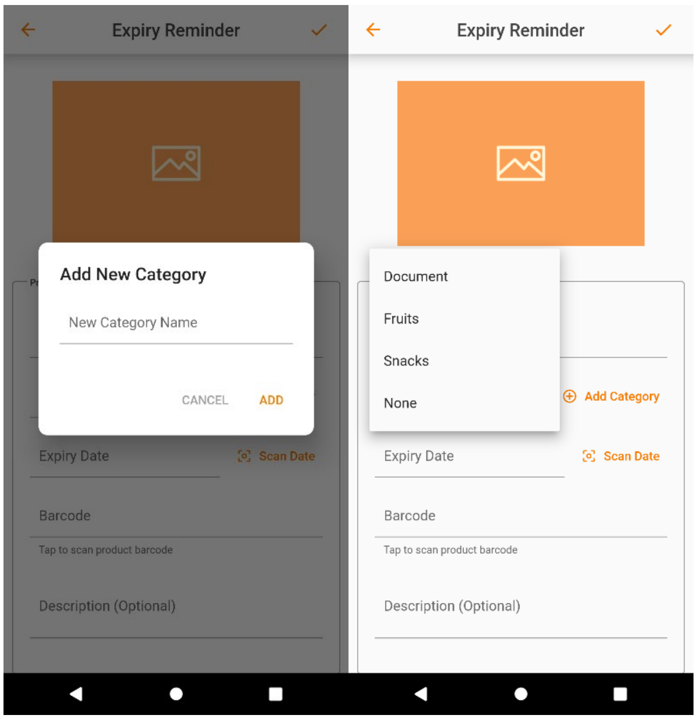
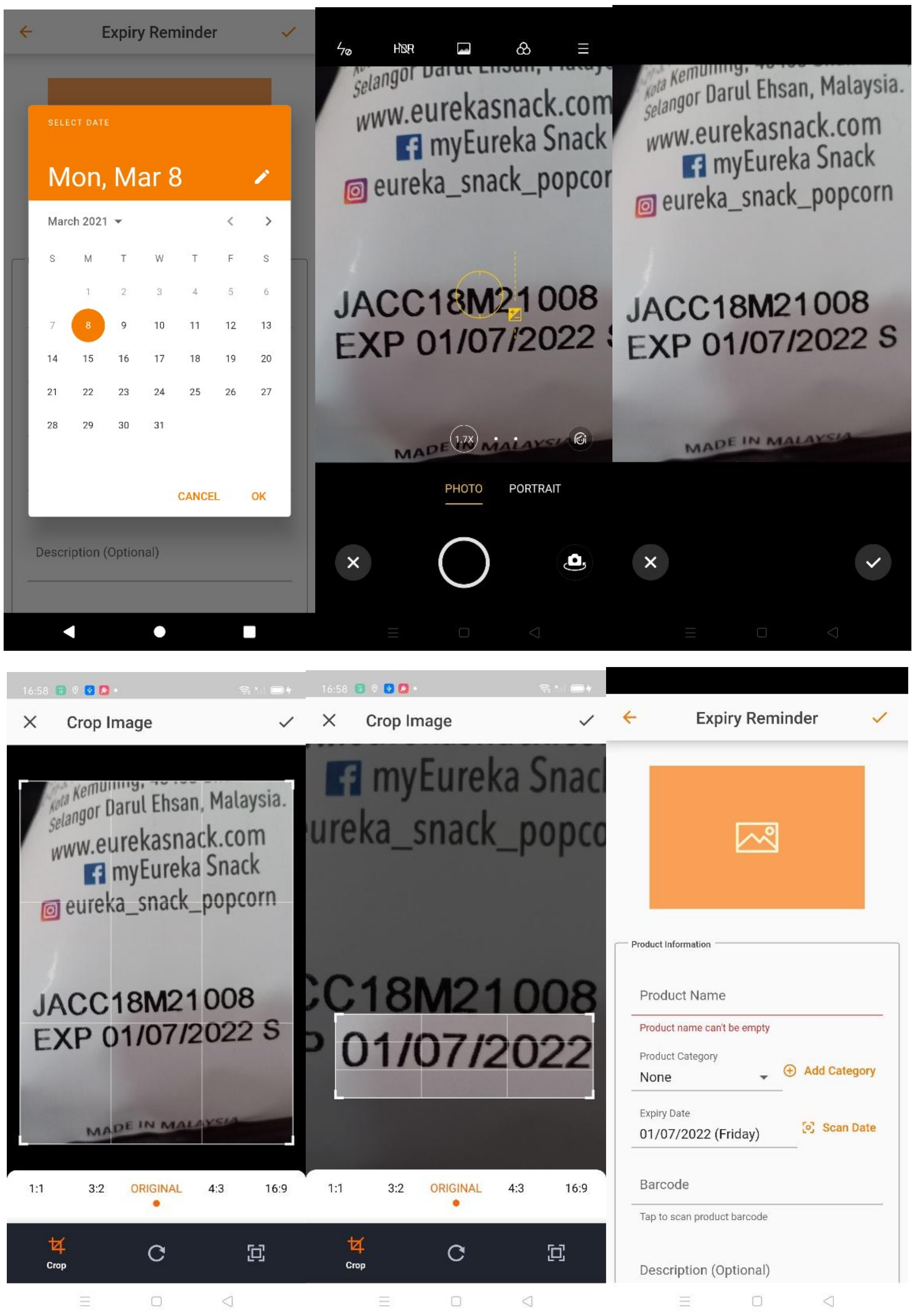


Figure 5.38: UI design for Add Category Feature

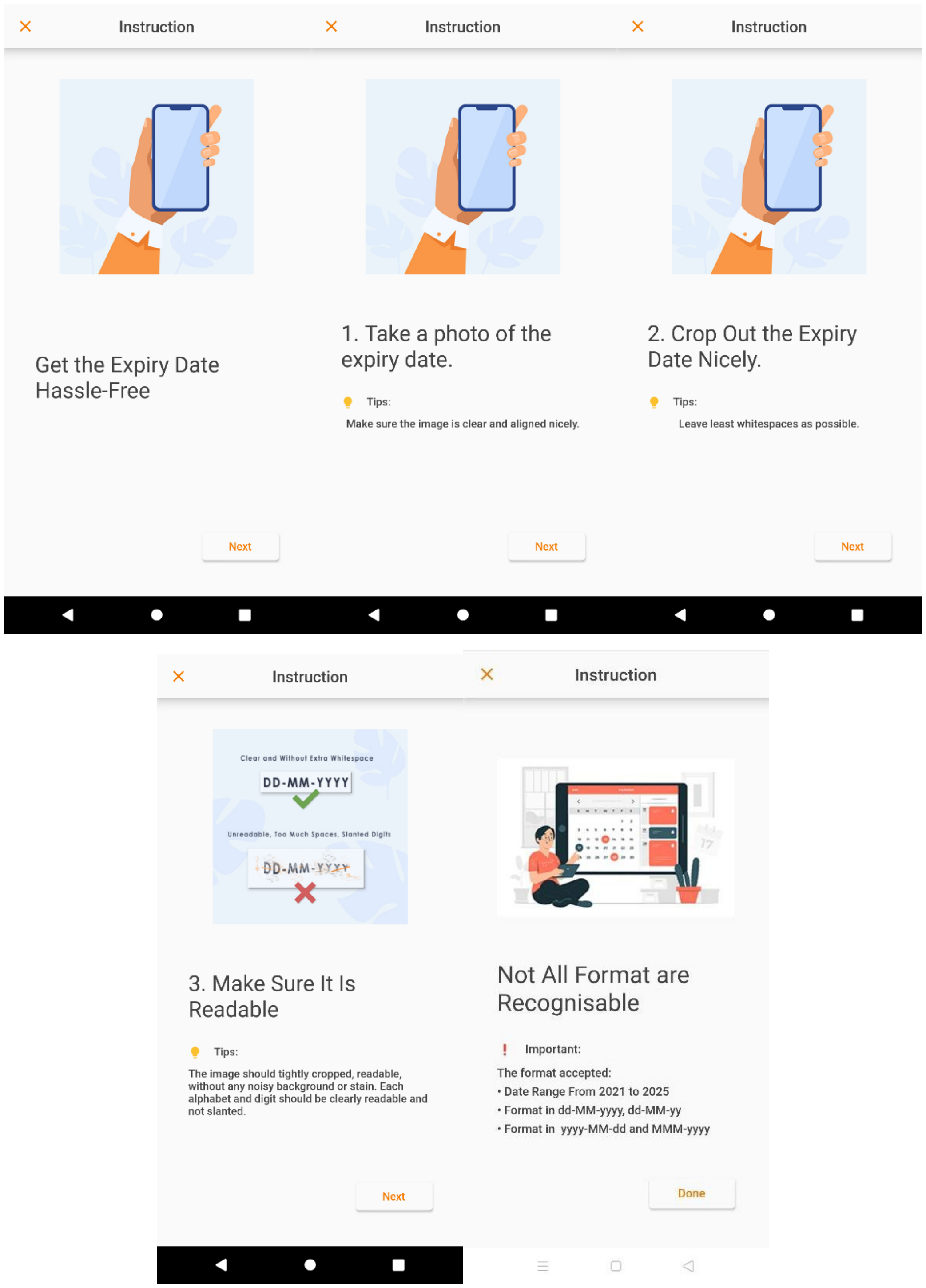
5.4.5.3 Expiry Date

Users can choose to enter the expiry date manually or use the deep learning model to recognize it. An instruction page will be displayed when it is the first time for the user to use the scan date feature.



Product Name
Product name can't be empty
Product Category: None
Expiry Date: 01/07/2022 (Friday)
Barcode
Tap to scan product barcode
Description (Optional)

Figure 5.39: UI design for Select Expiry Date Feature


Clear and Without Extra Whitespace
DD-MM-YYYY
Unreadable, Too Much Spaces, Slanted Digits
DD-MM-YYYY

Instruction

Get the Expiry Date Hassle-Free

Next

Instruction

1. Take a photo of the expiry date.

Tips:

Make sure the image is clear and aligned nicely.

Next

Instruction

2. Crop Out the Expiry Date Nicely.

Tips:

Leave least whitespaces as possible.

Next

Instruction

3. Make Sure It Is Readable

Tips:

The image should tightly cropped, readable, without any noisy background or stain. Each alphabet and digit should be clearly readable and not slanted.

Next

Instruction

Not All Format are Recognisable

Important:

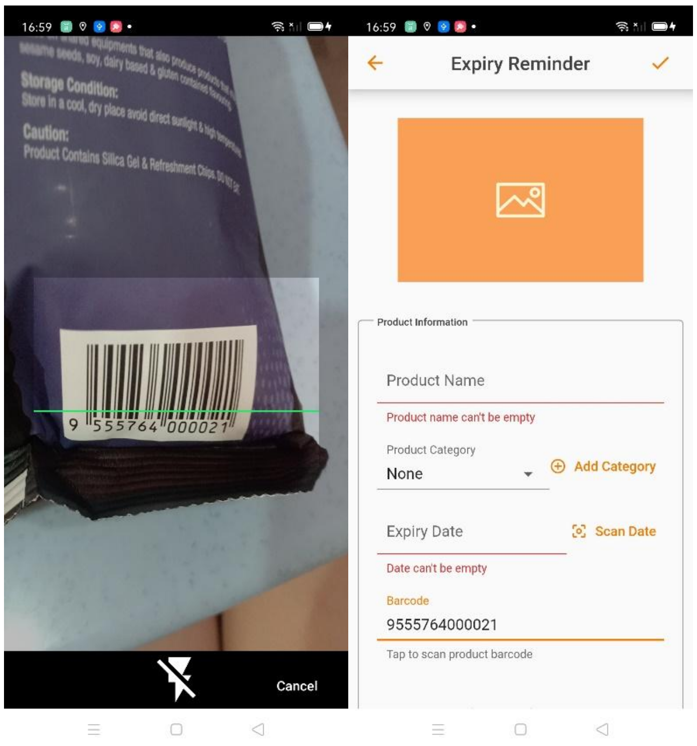
The format accepted:
- Date Range From 2021 to 2025
- Format in dd-MM-yyyy, dd-MM-yy
- Format in yyyy-MM-dd and MMM-yyyy

Done

Figure 5.40: UI design for Date Recognition Instruction Page

5.4.5.4 Barcode

Users can scan the product barcode to be used for the Search with Barcode function in the future. Users just need to scan the product barcode with their phone camera to get the barcode number.

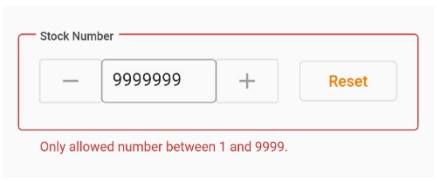


Product Name
Product name can't be empty
Product Category
None
Expiry Date
Date can't be empty
Barcode
9555764000021
Tap to scan product barcode

Figure 5.41: UI design for Scan Barcode Feature

5.4.5.5 Stock Number

Users can choose to use the add and minus button to add or subtract the number of stocks or use the keyboard to type the number of stocks. The minimum number of stocks is 1 and the maximum number of stocks is 9999. If the number of stocks is more than 9999, an error message will be prompt. There is also a reset button available for users to reset the number of stocks back to 1, which is the initial value of the number of stocks.


Stock Number
9999999

Only allowed number between 1 and 9999.

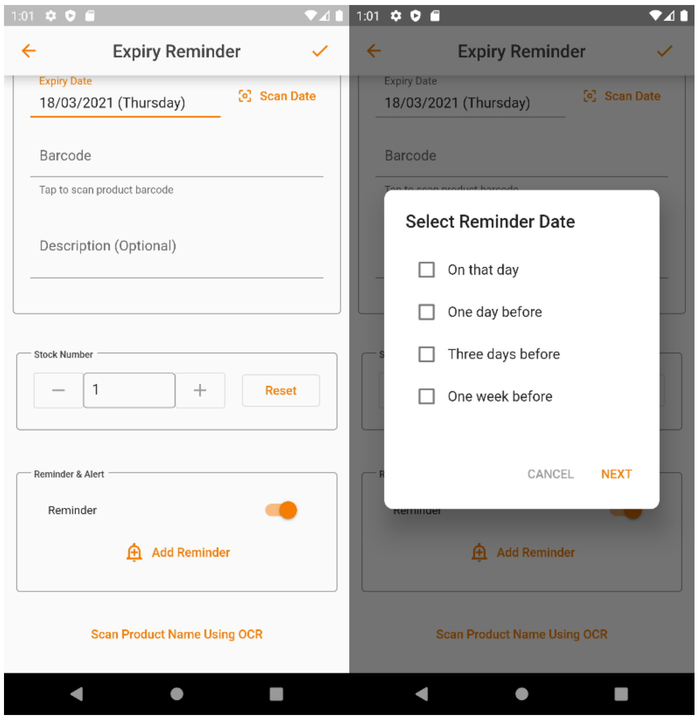
Figure 5.42: UI design for Stock Number Error Message

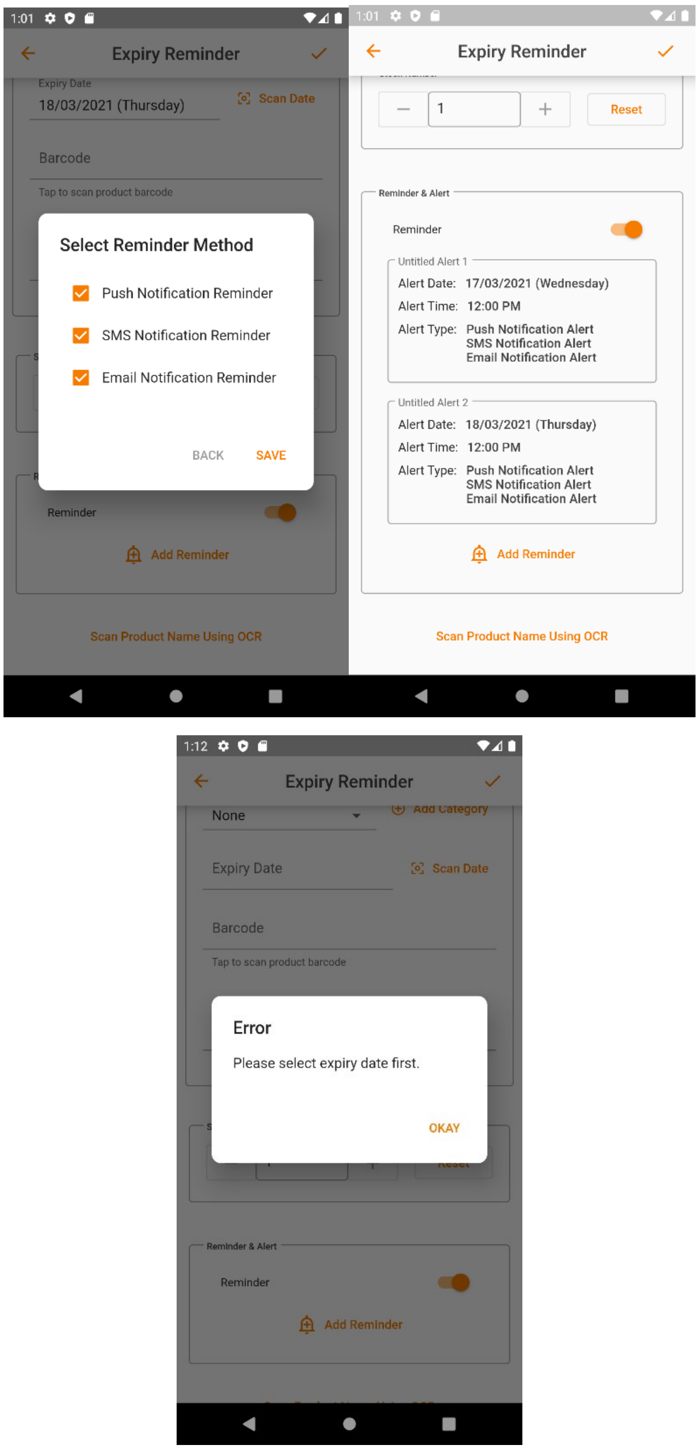
5.4.5.6 Reminder and Alert

Users can enable and disable alert reminder by just toggle the switch button in the reminder & alert section. If the alert reminder is enabled, users can add an alert reminder according to their needs. Alerts can be set on the same day, one day before, three days before, one week before, two weeks before, one month before, two months before, three months before and six months before the expiry date according to the expiry date entered above. If the expiry date is empty or having an invalid value, an alert message will be displayed.

Users can also choose their preferred alert method which includes push notification, SMS notification and Email notification according to their needs. However, if the user's phone number or email address is not validated, the corresponding alert method will not be displayed in the checkbox list.

A summary of alerts will be displayed once the alerts are added.





1:01

Expiry Reminder

Expiry Date
18/03/2021 (Thursday)
Scan Date

Barcode
Tap to scan product barcode

Select Reminder Method

✓ Push Notification Reminder
✓ SMS Notification Reminder
✓ Email Notification Reminder

BACK SAVE

Reminder
Add Reminder

Scan Product Name Using OCR

1:01

Expiry Reminder

1
Reset

Reminder & Alert

Reminder

Untitled Alert 1
Alert Date: 17/03/2021 (Wednesday)
Alert Time: 12:00 PM
Alert Type: Push Notification Alert
SMS Notification Alert
Email Notification Alert

Untitled Alert 2
Alert Date: 18/03/2021 (Thursday)
Alert Time: 12:00 PM
Alert Type: Push Notification Alert
SMS Notification Alert
Email Notification Alert

Add Reminder

Scan Product Name Using OCR

1:12

Expiry Reminder

None
Add Category

Expiry Date
Scan Date

Barcode
Tap to scan product barcode

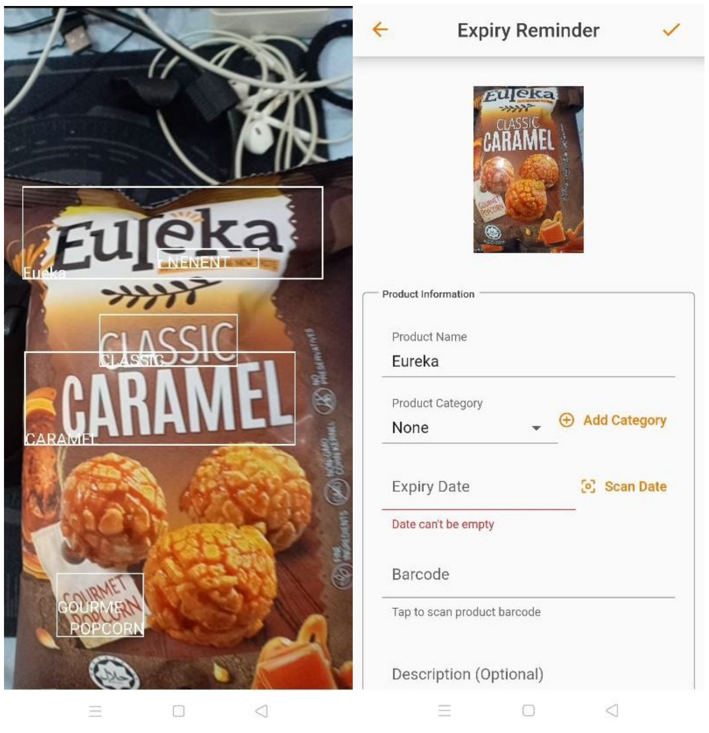
Error
Please select expiry date first.

OKAY

Reminder & Alert
Reminder
Add Reminder

Figure 5.43: UI design for Add Alert Reminder Feature

5.4.5.7 Scan Product Name Using OCR



Expiry Reminder

Product Information

Product Name
Eureka

Product Category
None

Add Category

Expiry Date

Scan Date

Date can't be empty

Barcode

Tap to scan product barcode

Description (Optional)

Figure 5.44: UI design for Scan Product Name with OCR Feature

5.4.5.8 Added Successful Message

When a product is added successfully, a successful message will be prompt and users will be redirected to the home page.

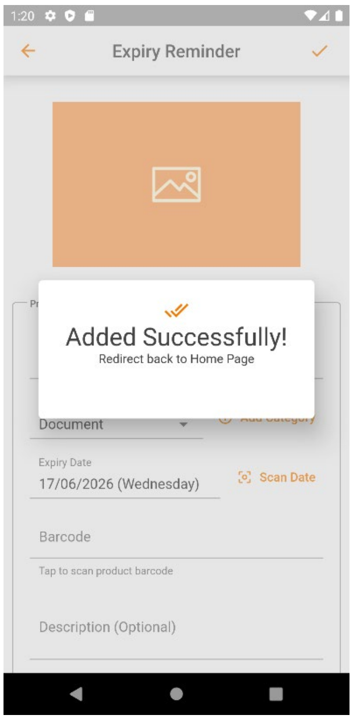
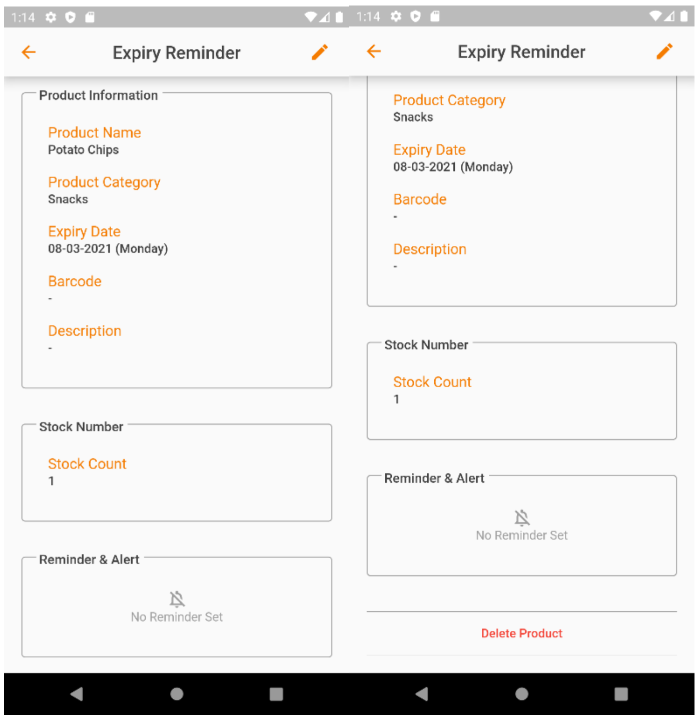


Figure 5.45: UI design for Successful Message for Add Product Feature

5.4.6 View Product Page

Users can simply tap on the product to view the product details. Users can also edit product details or delete the product from this page.



Product Name
Potato Chips

Product Category
Snacks

Expiry Date
08-03-2021 (Monday)

Barcode
-

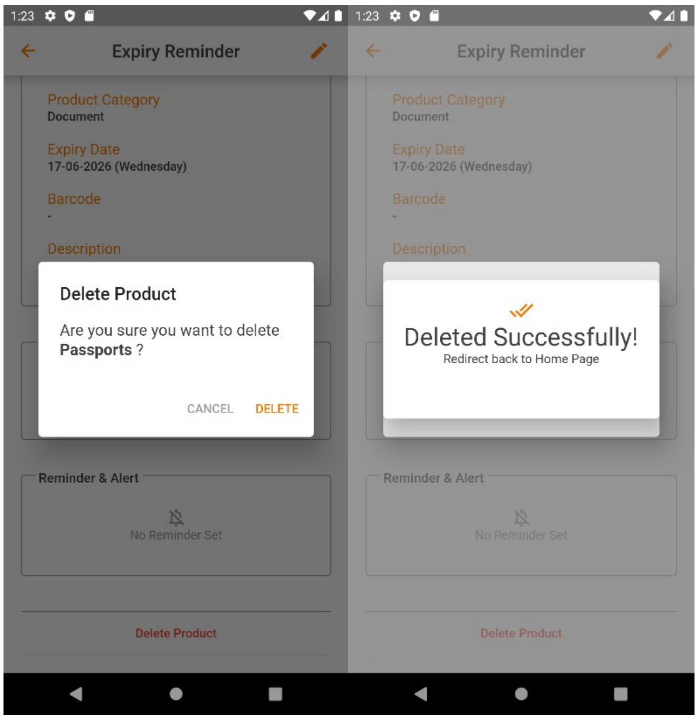
Description
-

Stock Count
1

Figure 5.46: UI design for View Product Page

5.4.6.1 Delete Confirmation Message

A confirmation message will be prompt upon deletion. If the delete button is clicked, the product will be deleted and a successful message will be prompt and users will be redirected back to the home page.



Delete Product

Are you sure you want to delete **Passports** ?

CANCEL DELETE

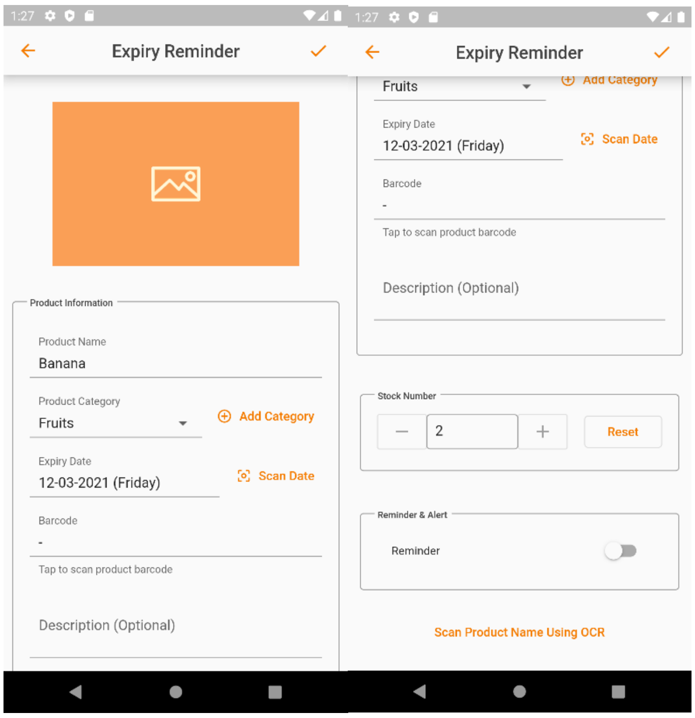
Deleted Successfully!

Redirect back to Home Page

Figure 5.47: UI design for Delete Product Feature

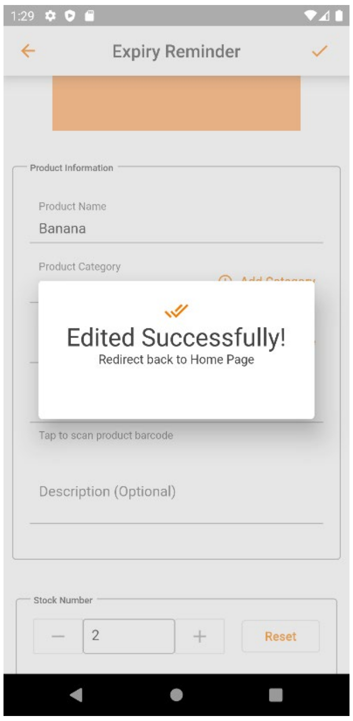
5.4.7 Edit Product

The edit product page reuses the same user interface as the add product page. A successful message will be prompted upon a successful update.



Product Information
Product Name: Banana
Product Category: Fruits
Expiry Date: 12-03-2021 (Friday)
Barcode: -
Tap to scan product barcode
Description (Optional)
Stock Number: 2
Reminder & Alert
Reminder

Figure 5.48: UI design for Edit Product Page



Product Information
Product Name: Banana
Product Category
Edited Successfully!
Redirect back to Home Page
Tap to scan product barcode
Description (Optional)
Stock Number: 2

Figure 5.49: UI design for Successful Message for Edit Product

5.4.8 Search Product

Users can search product by tapping on the search bar on the home page. If a user has searched something before, a recent search list displayed in the list.

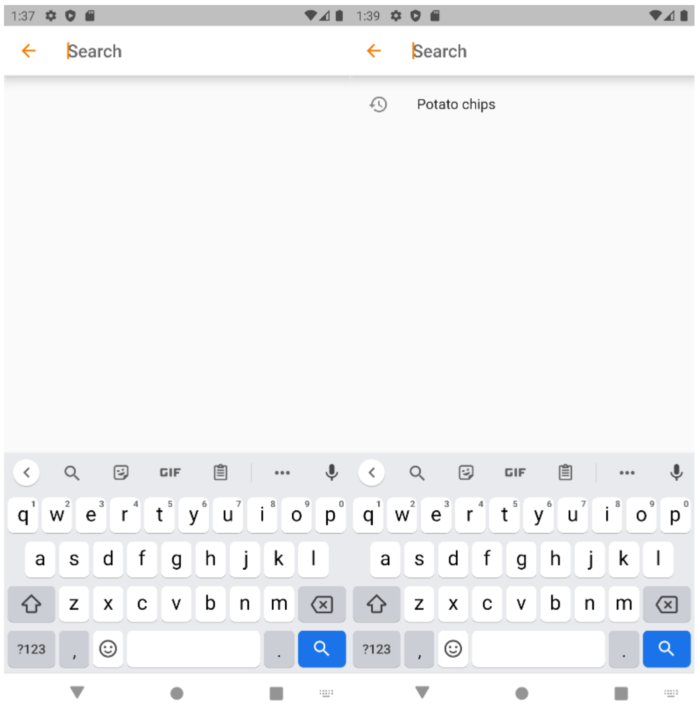


Figure 5.50: UI design for Search Product Page

When the user typed something, a suggested list will be shown if there is any.

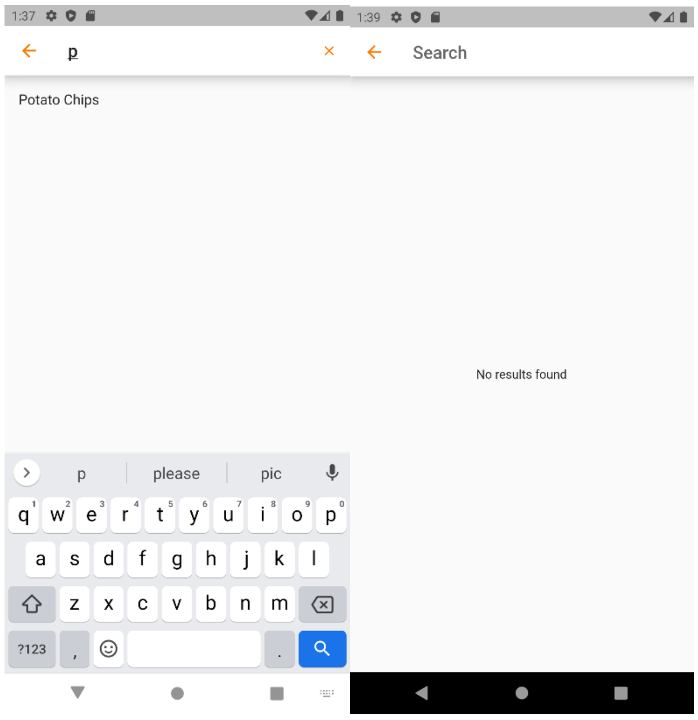
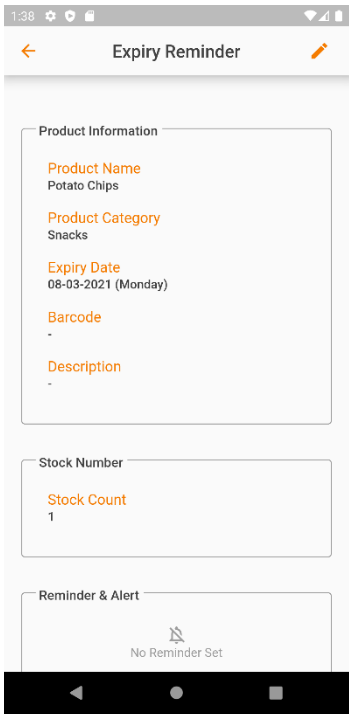


Figure 5.51: UI design for Suggestion List in Search Product Page

When the user clicks on the suggestion list, the user will be led to the product detail page.



Product Name
Potato Chips

Product Category
Snacks

Expiry Date
08-03-2021 (Monday)

Barcode
-

Description
-

Stock Count
1

Figure 5.52: UI design for View Product Detail Page

5.4.9 View Category Page

Users can view all categories on this page. Users can also add a new category, edit category and delete category on this page. Besides, users can view all products under the category by tapping on the category.

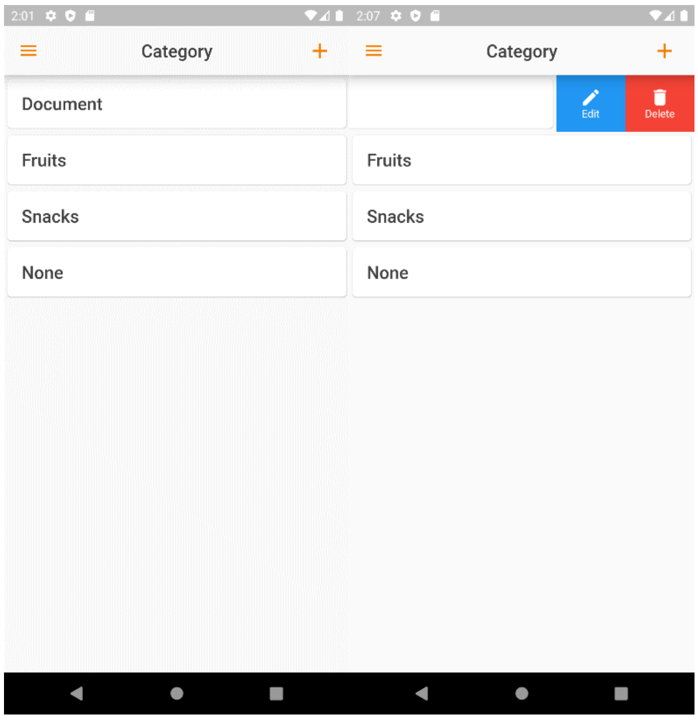


Figure 5.53: UI design for View Category Page

5.4.9.1 Add Category

Users can add a category by tapping the add button on the top right corner.

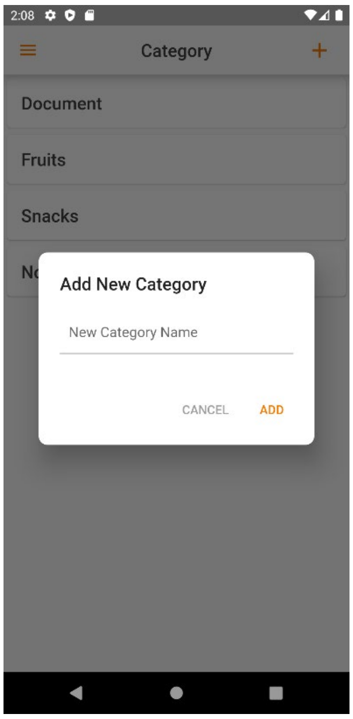


Figure 5.54: UI design for Add Category Dialog

5.4.9.2 Edit Category

Users can find the edit button by sliding the card to the left.

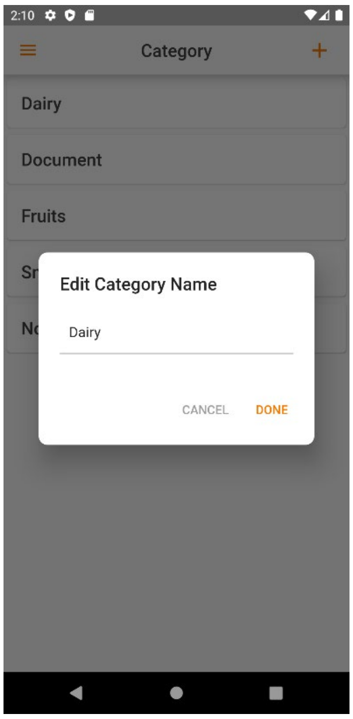
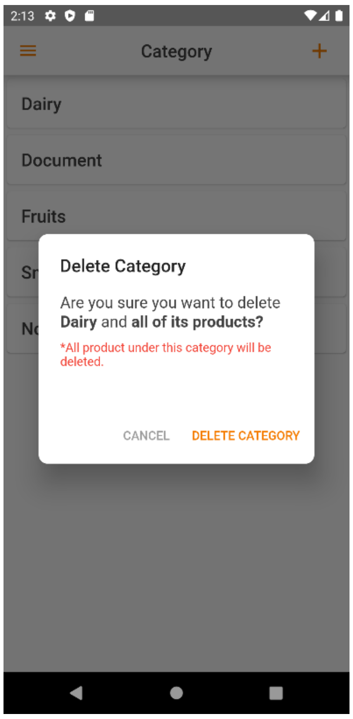


Figure 5.55: UI design for Edit Category Dialog

5.4.9.3 Delete Category

Users can find the delete button by sliding the card to the left. A confirmation message will be prompt.



Delete Category

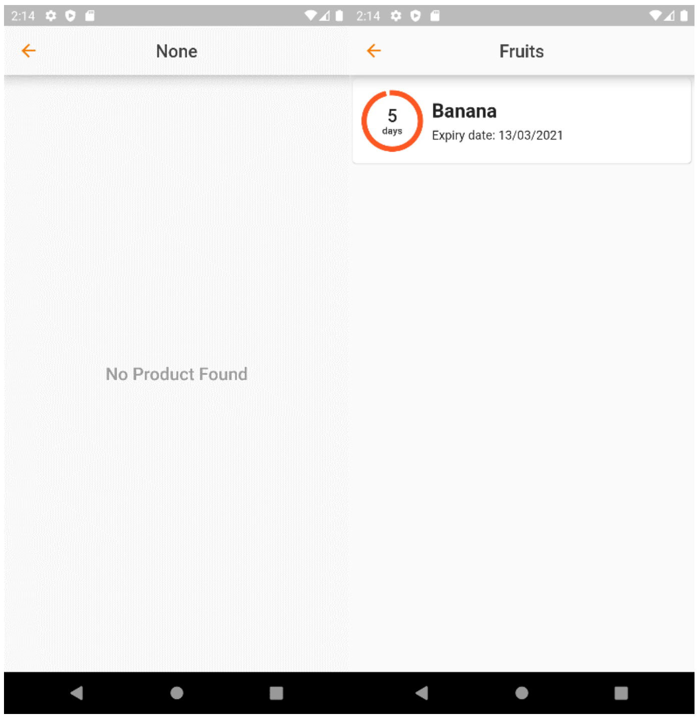
Are you sure you want to delete **Dairy** and **all of its products?**

*All product under this category will be deleted.

Figure 5.56: UI design for Delete Category Confirmation Dialog

5.4.9.4 View Product of the Category

If the category does not contain any product, a message will be shown. When users tap on the product, the product details will be displayed.



Banana
Expiry date: 13/03/2021

Figure 5.57: UI design for View Product by Category Feature

5.4.10 Search Product with Barcode

Users can search for products by scanning the barcode. The product details will be displayed upon success. If the barcode is not found, an error message will be prompt.

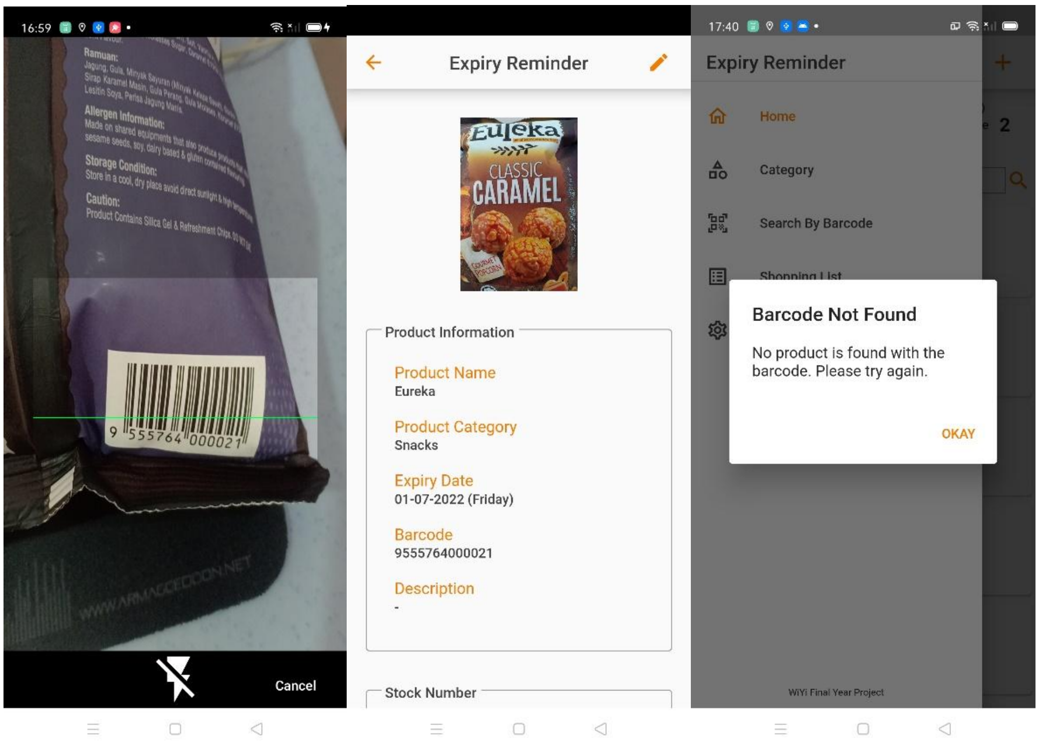


Figure 5.58: UI design for Scan Product with Barcode Page

5.4.11 Shopping List Page

On this page, users can add a shopping list item to know what to buy later. Products that are expired or products that have no stocks will be added to the shopping list automatically by the application. Users can check and uncheck the shopping list item to indicate whether the item is purchased or not.

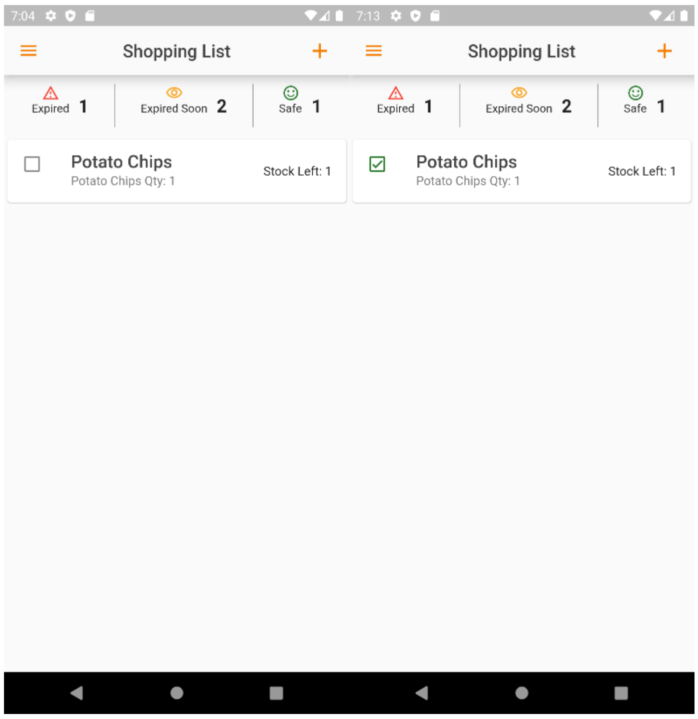


Figure 5.59: UI design for Shopping List Page

Users can also swipe to delete a shopping list item when they have purchased the item. Users can also long-press on the shopping list item to delete a shopping list item.

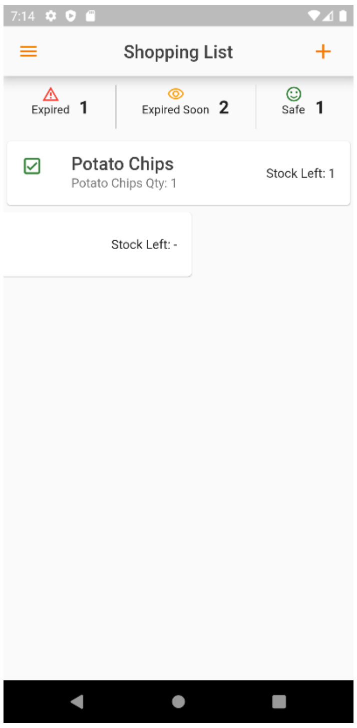


Figure 5.60: UI design for Swipe to Delete Feature in Shopping List Page

To edit the shopping list item, users have to long-press on the item they want to edit to edit the item. An alert dialog will be prompt for users to choose if they wanted to delete the item or edit the item. Upon a successful update on the shopping list item, a successful message will be prompt.

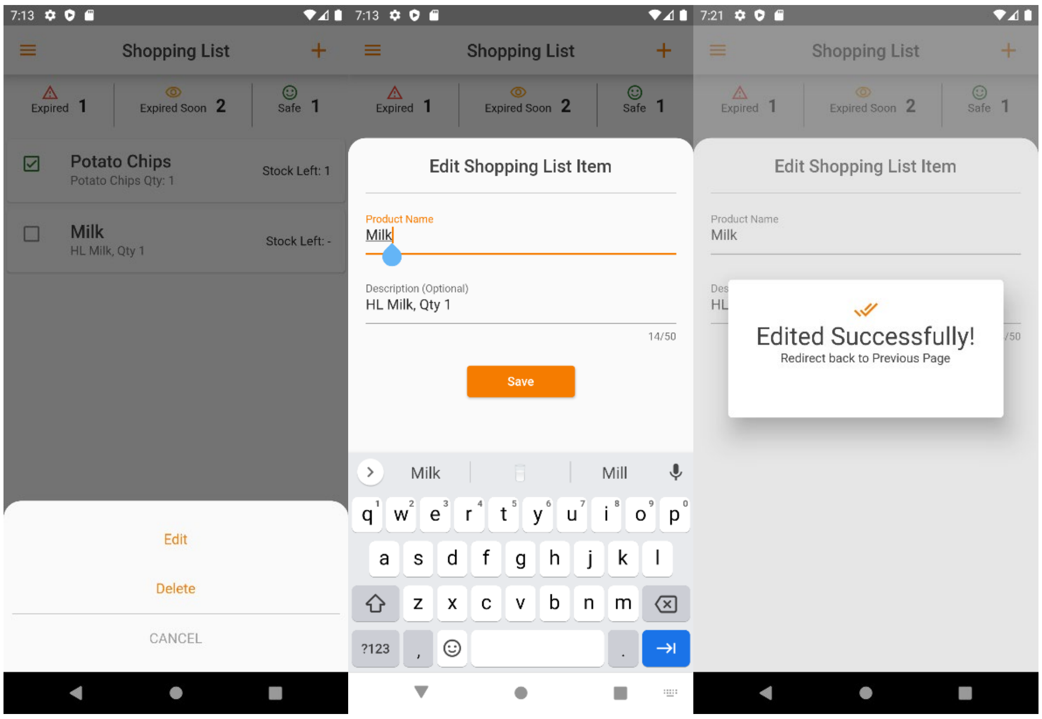


Figure 5.61: UI design for Edit Shopping List Item Feature

Besides, users can add new shopping list items manually by pressing the add button on the top right corner. A successful message will be prompted upon a successful add to the shopping list item.

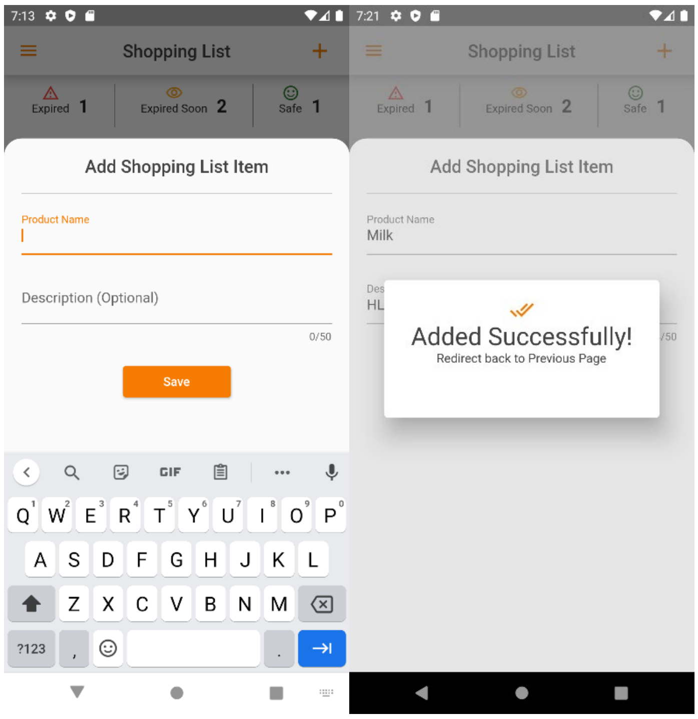
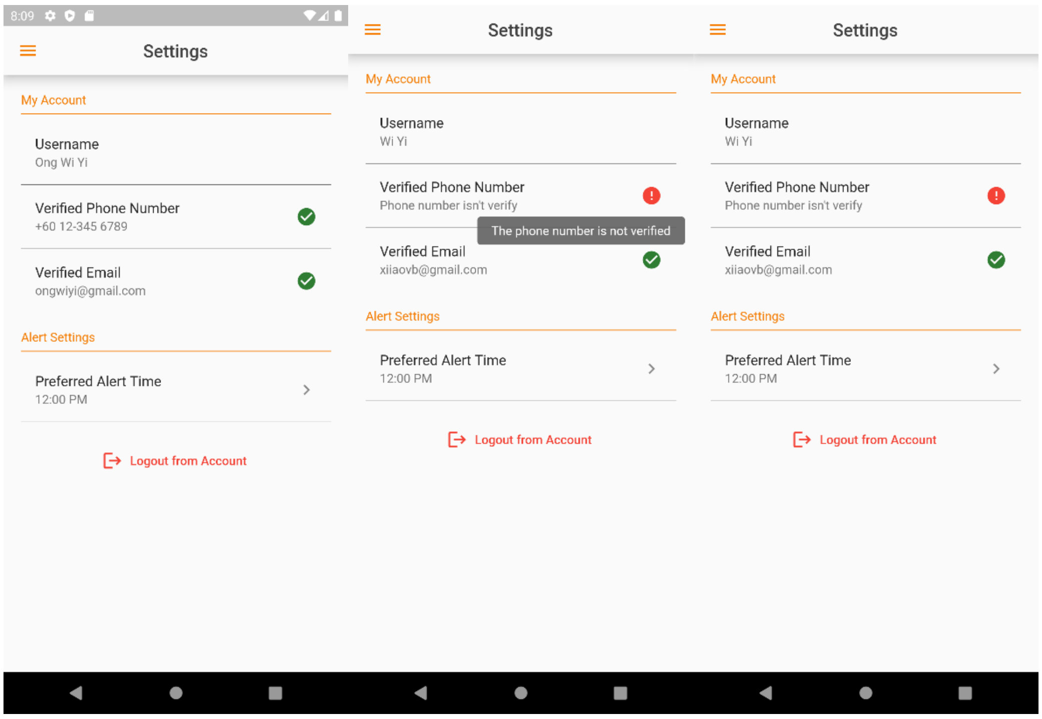


Figure 5.62: UI design for Add Shopping List Feature

5.4.12 Settings Page

On the Settings Page, users can check if their phone number and email address are verified. If any of the credentials are not verified, users can tap on the credential to enter the verification process. Besides, users can change the preferred alert time according to their preferences.


8:09

Settings

My Account

Username
Ong Wi Yi

Verified Phone Number ✓
+60 12-345 6789

Verified Email ✓
ongwiyi@gmail.com

Alert Settings

Preferred Alert Time
12:00 PM

Logout from Account

Settings

My Account

Username
Wi Yi

Verified Phone Number !
Phone number isn't verify

The phone number is not verified

Verified Email ✓
xiiaovb@gmail.com

Alert Settings

Preferred Alert Time
12:00 PM

Logout from Account

Settings

My Account

Username
Wi Yi

Verified Phone Number !
Phone number isn't verify

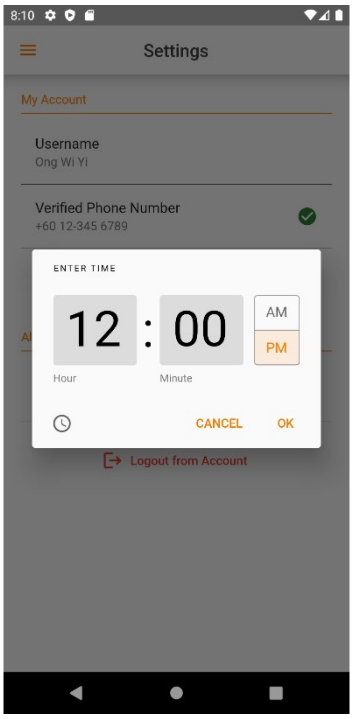
Verified Email ✓
xiiaovb@gmail.com

Alert Settings

Preferred Alert Time
12:00 PM

Logout from Account

Figure 5.63: UI design for Settings Page


8:10

Settings

My Account

Username
Ong Wi Yi

Verified Phone Number ✓
+60 12-345 6789

ENTER TIME

12 : 00 AM [✓ PM]

Hour Minute

CANCEL OK

Logout from Account

Figure 5.64: UI design for Change Preferred Alert Time Feature

CHAPTER 6

IMPLEMENTATION

6.1 Implementation of Mobile Application

The mobile application is using Flutter as the framework and Dart as the programming language. This is because Flutter provides different widgets and libraries that help in developing an iterative UI fast and efficient.

There are three main modules for the mobile application, User Module, Personal Product Module and Shopping List Module. Each module consists of respective models, services and views. Views call functions from services and use widgets to display output and accepts input. Services will access the database through models to perform various actions requested from the view. A detailed code implementation for the modules stated will be discussed in this section.

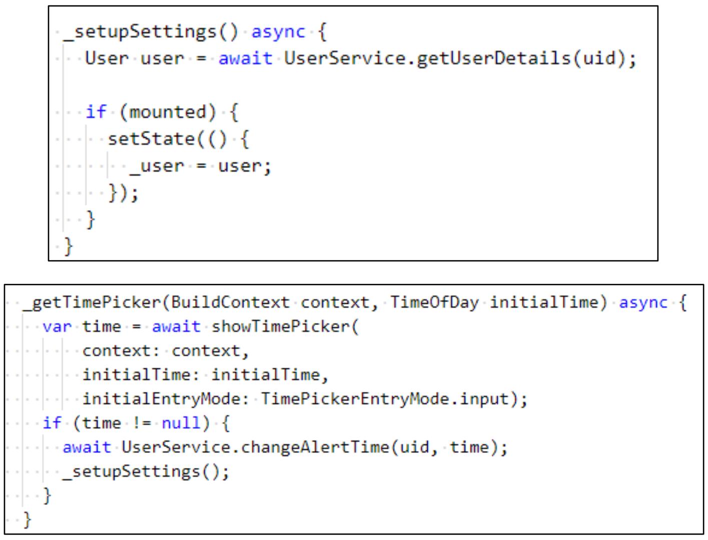
6.1.1 User Module

Table 6.1: Model, Services and View in User Module

Model	User Model
Services	User Service
Views	Settings Product Action Verify Phone Home

6.1.1.1 Settings View

On the Settings page, the user's information is retrieved to be displayed on the interface. Then, the user is allowed to change the preferred alert time and navigate to verify the phone number page and verify the email address page only if the criteria are not verified. Hence, two functions from the services will be called, one is *getUserDetails* and the other one is *changeAlertTime*. The code segment is shown in the figure below.



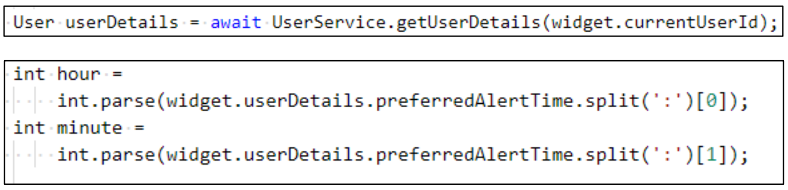
```
_setupSettings() async {
  User user = await UserService.getUserDetails(uid);

  if (mounted) {
    setState(() {
      _user = user;
    });
  }
}
```

```
_getTimePicker(BuildContext context, TimeOfDay initialTime) async {
  var time = await showTimePicker(
        context: context,
        initialTime: initialTime,
        initialEntryMode: TimePickerEntryMode.input);
  if (time != null) {
    await UserService.changeAlertTime(uid, time);
    _setupSettings();
  }
}
```

Figure 6.1: Code Segments of User Services Called on Settings View

6.1.1.2 Product Action View

On the Product Action page, the user's preferred alert time is needed when add or edit the product reminder. Hence, *getUserDetails* function will be called to retrieve the user's preferred alert time. The code segment is shown in the figure below.



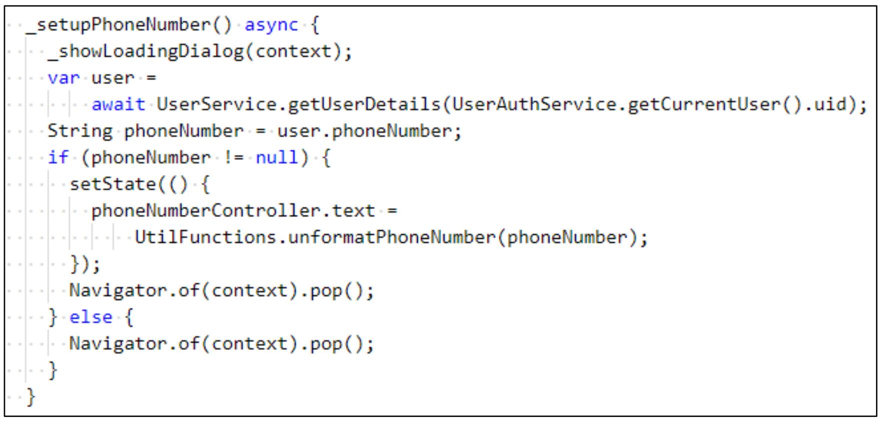
```
User userDetails = await UserService.getUserDetails(widget.currentUserId);
```

```
int hour =
    int.parse(widget.userDetails.preferredAlertTime.split(':')[0]);
int minute =
    int.parse(widget.userDetails.preferredAlertTime.split(':')[1]);
```

Figure 6.2: Code Segments of User Services Called on Product Action View

6.1.1.3 Verify Phone View

To ease user in verifying their phone number, the system will retrieve the user's phone number via *getUserDetails* function. If the user's phone number exists, the phone number will be filled automatically into the text field. If the phone number is null, the text field will be empty, and the user needs to enter the phone number manually.

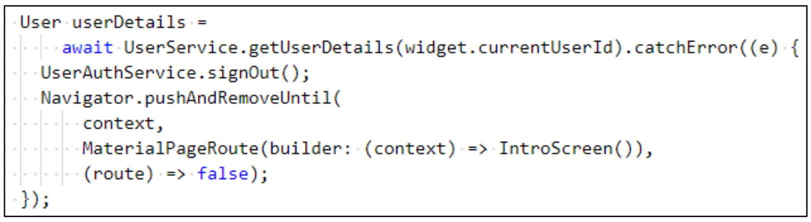


```
  _setupPhoneNumber() async {
    _showLoadingDialog(context);
    var user =
        await UserService.getUserDetails(UserAuthService.getCurrentUser().uid);
    String phoneNumber = user.phoneNumber;
    if (phoneNumber != null) {
      setState(() {
        phoneNumberController.text =
            UtilFunctions.unformatPhoneNumber(phoneNumber);
      });
      Navigator.of(context).pop();
    } else {
      Navigator.of(context).pop();
    }
  }
```

Figure 6.3: Code Segments of User Services Called on Verify Phone View

6.1.1.4 Home View

The home view uses the user module to prevent any unexpected access to the home page without any authentication. It will retrieve the user's id on load and navigate the user to the sign-in page when the user id is not found. The code segment is shown in the figure below.

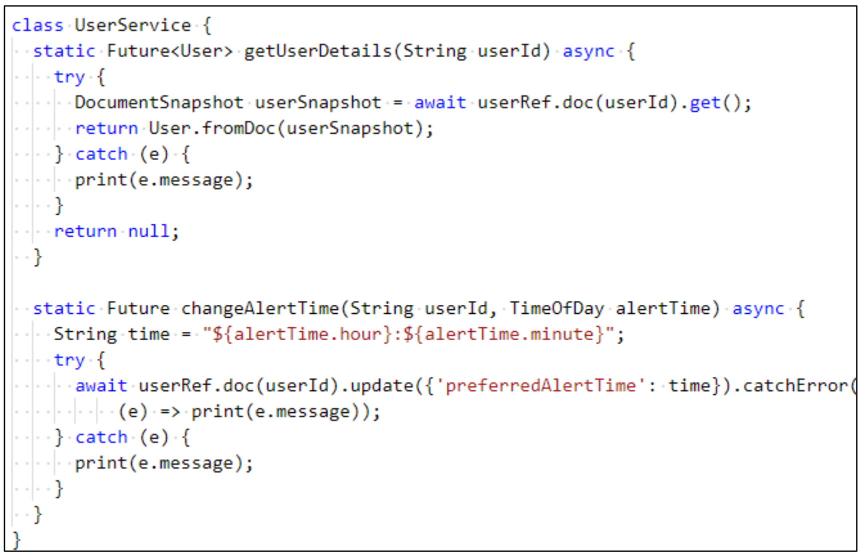


```
User userDetails =
    await UserService.getUserDetails(widget.currentUserId).catchError((e) {
  UserAuthService.signOut();
  Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(builder: (context) => IntroScreen()),
      (route) => false);
});
```

Figure 6.4: Code Segments of User Services Called on Home View

6.1.1.5 User Services

The figure below shows the code segments of each user services functions.

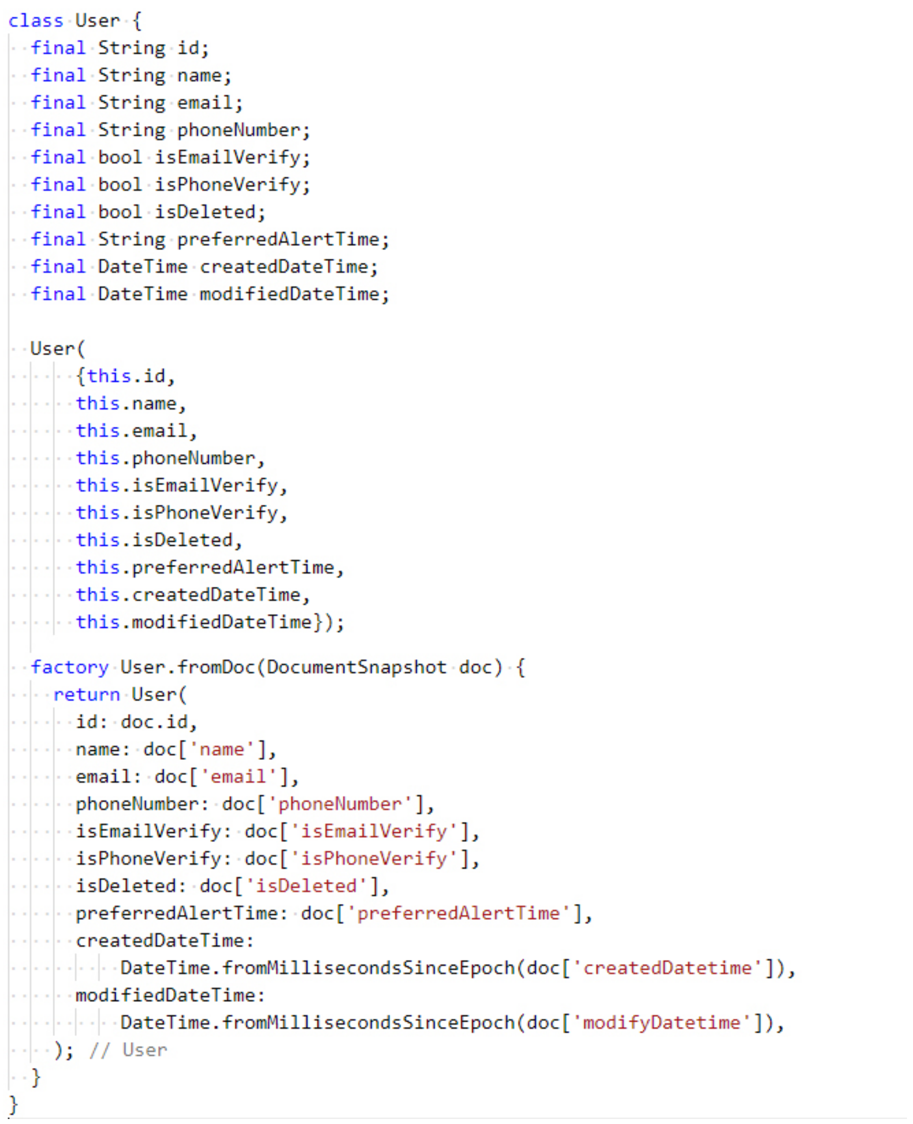


```
class UserService {
  static Future<User> getUserDetails(String userId) async {
    try {
      DocumentSnapshot userSnapshot = await userRef.doc(userId).get();
      return User.fromDoc(userSnapshot);
    } catch (e) {
      print(e.message);
    }
    return null;
  }

  static Future changeAlertTime(String userId, TimeOfDay alertTime) async {
    String time = "${alertTime.hour}:${alertTime.minute}";
    try {
      await userRef.doc(userId).update({'preferredAlertTime': time}).catchError(
          (e) => print(e.message));
    } catch (e) {
      print(e.message);
    }
  }
}
```

Figure 6.5: Code Segments for User Services

6.1.1.6 User Model

A factory method design pattern is used to quickly create a user object. The figure below shows the code segments for User Model.



```
class User {
  final String id;
  final String name;
  final String email;
  final String phoneNumber;
  final bool isEmailVerify;
  final bool isPhoneVerify;
  final bool isDeleted;
  final String preferredAlertTime;
  final DateTime createdDateTime;
  final DateTime modifiedDateTime;

  User(
      {this.id,
      this.name,
      this.email,
      this.phoneNumber,
      this.isEmailVerify,
      this.isPhoneVerify,
      this.isDeleted,
      this.preferredAlertTime,
      this.createdDateTime,
      this.modifiedDateTime});

  factory User.fromDoc(DocumentSnapshot doc) {
    return User(
      id: doc.id,
      name: doc['name'],
      email: doc['email'],
      phoneNumber: doc['phoneNumber'],
      isEmailVerify: doc['isEmailVerify'],
      isPhoneVerify: doc['isPhoneVerify'],
      isDeleted: doc['isDeleted'],
      preferredAlertTime: doc['preferredAlertTime'],
      createdDateTime:
          DateTime.fromMillisecondsSinceEpoch(doc['createdDatetime']),
      modifiedDateTime:
          DateTime.fromMillisecondsSinceEpoch(doc['modifyDatetime']),
    ); // User
  }
}
```

Figure 6.6: Code Segments for User Model

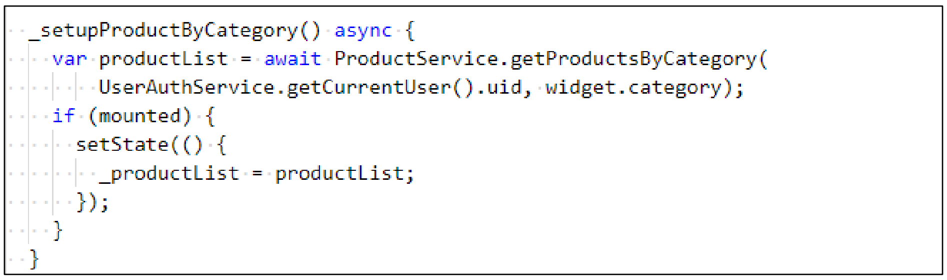
6.1.2 Personal Product Module

Table 6.2: Model, Services and View for Personal Product Module

Model	Personal Product Model
	Alert Model
Services	Personal Product Service
Views	Product by Category
	Search by Category
	Search by Name
	Search by Barcode
	Product Action
	View Product
	Home

6.1.2.1 Product by Category View

On product by category view, products of the selected category are retrieved via the *getProductsByCategory* function in the Personal Product Services. The code segments are shown in the figure below.

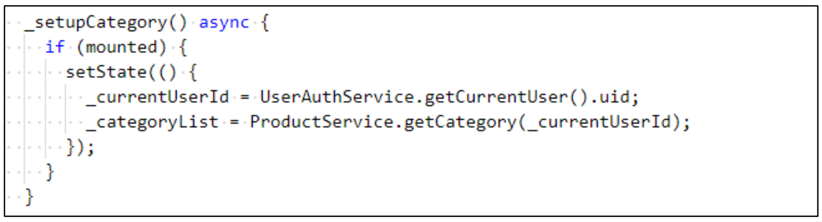


```
  _setupProductByCategory() async {
    var productList = await ProductService.getProductsByCategory(
        UserAuthService.getCurrentUser().uid, widget.category);
    if (mounted) {
      setState(() {
        _productList = productList;
      });
    }
  }
```

Figure 6.7: Code Segments of Personal Product Services Called on Product by Category View

6.1.2.2 Search by Category View

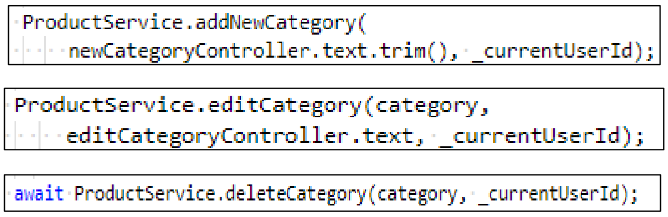
On search by category view, all category created will be retrieved via *getCategory* function in the Personal Product Services. In the database, there are two types of category, one is the unused category and the other one is the used category. The used category will exist in the personal product collection until all products in the category are removed. This is done to prevent users from deleting the category after removing the product, allowing users to handle the category more efficiently. When the user adds a product that comes under the category, the unused category is removed from the collection. This is done to avoid complex interconnections between categories and products, as well as to simplify the database structure to make queries easier when identifying whether the category contains a product or not. The code segments are shown in the figure below.



```
_setupCategory() async {
  if (mounted) {
    setState(() {
      _currentUserId = UserAuthService.getCurrentUser().uid;
      _categoryList = ProductService.getCategory(_currentUserId);
    });
  }
}
```

Figure 6.8: Code Segments of Personal Product Services Called in Setup Category View Function on Search by Category View

Furthermore, users can add a new category, delete a category and edit a category in the Search by Category View. The code segments are shown in the figure below.



```
ProductService.addNewCategory(
    newCategoryController.text.trim(), _currentUserId);
```

```
ProductService.editCategory(category,
    editCategoryController.text, _currentUserId);
```

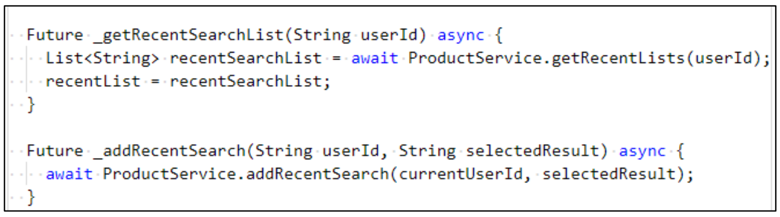
```
await ProductService.deleteCategory(category, _currentUserId);
```

Figure 6.9: Code Segments of Personal Product Services Called in Add, Edit and Delete Category Function on Search by Category View

6.1.2.3 Search by Name View

Users will search for products by name throughout the Search by Name view. To save resources on reading the same information from the server again, the entire product list will be passed from the Home page to the Search by Name View. The search delegates in the Flutter framework will use the product list to populate a search list.

Furthermore, to view recent searches using the Flutter framework's search delegates, a recent search list must be obtained from the database using the *getRecentLists* feature and updated from time to time using the *addRecentSearch* function in the Personal Product Services.

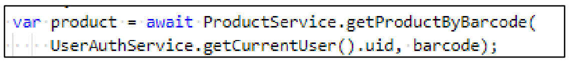


```
Future _getRecentSearchList(String userId) async {
  List<String> recentSearchList = await ProductService.getRecentLists(userId);
  recentList = recentSearchList;
}

Future _addRecentSearch(String userId, String selectedResult) async {
  await ProductService.addRecentSearch(currentUserId, selectedResult);
}
```

Figure 6.10: Code Segments of Personal Product Services Called on Search by Name View

6.1.2.4 Search by Barcode

The user can scan the product's barcode to retrieve information about the product. Hence, the *getProductByBarcode* function is used to retrieve product information via barcode.

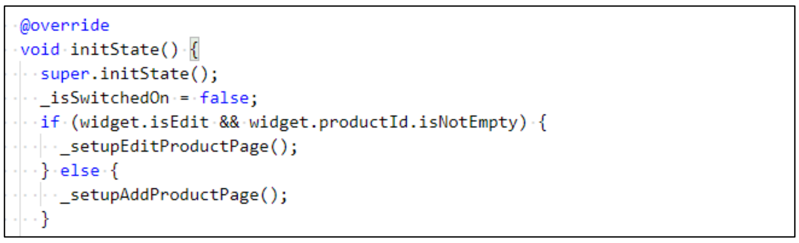


```
var product = await ProductService.getProductByBarcode(
    UserAuthService.getCurrentUser().uid, barcode);
```

Figure 6.11: Code Segments of Personal Product Services Called on Search by Barcode View

6.1.2.5 Product Action View

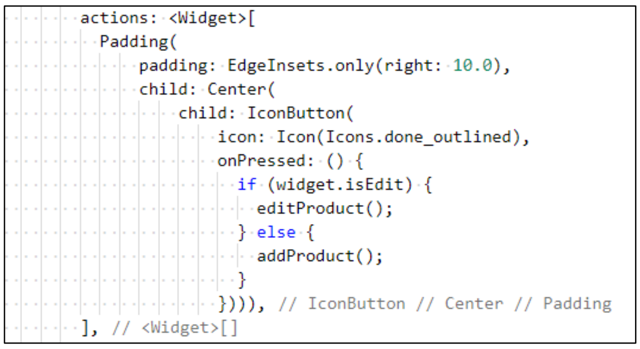
On the Product Action view, users can add or edit the product. The reason for putting add and edit in the same view is to reuse the view to prevent code repetition. There is an *isEdit* Boolean to indicate if it is for editing a product or add a product.



```
@override
void initState() {
  super.initState();
  _isSwitchedOn = false;
  if (widget.isEdit && widget.productId.isNotEmpty) {
    _setupEditProductPage();
  } else {
    _setupAddProductPage();
  }
```

Figure 6.12: Code Segments of Setup Product Action View Based on the Situation

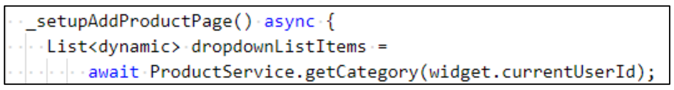
The same idea is also used for the submit function on the Product Action page.



```
        actions: <Widget>[
          Padding(
              padding: EdgeInsets.only(right: 10.0),
              child: Center(
                  child: IconButton(
                      icon: Icon(Icons.done_outlined),
                      onPressed: () {
                        if (widget.isEdit) {
                          editProduct();
                        } else {
                          addProduct();
                        }
                      }))), // IconButton // Center // Padding
        ], // <Widget>[]
```

Figure 6.13: Code Segments of Submit Product Action Forms on the Product Action Page

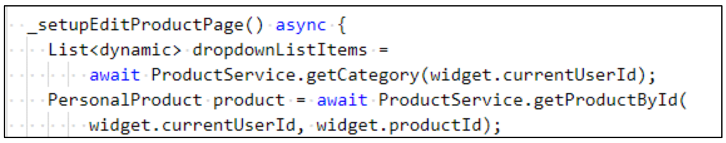
When setting up Add Product Forms, the category is retrieved via the *getCategory* functions in the Personal Product Services. The figure below shows the code segments of *getCategory* function in the setup Add Product form function.



```
  _setupAddProductPage() async {
    List<dynamic> dropdownListItems =
        await ProductService.getCategory(widget.currentUserId);
```

Figure 6.14: Code Segments of Personal Product Services Called in Setup Add Product Function on Product Action View

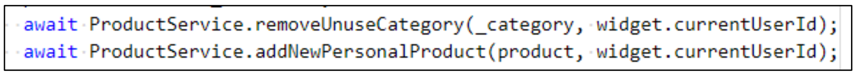
When setting up Edit Product Forms, the category is retrieved via the *getCategory* functions and the product information is retrieved via the *getProductById* functions in the Personal Product Services. The figure below shows the code segments of *getCategory* function and *getProductById* function in the setup Add Product form function.



```
  _setupEditProductPage() async {
    List<dynamic> dropdownListItems =
        await ProductService.getCategory(widget.currentUserId);
    PersonalProduct product = await ProductService.getProductById(
        widget.currentUserId, widget.productId);
```

Figure 6.15: Code Segments of Personal Product Services Called in Setup Edit Product Function on Product Action View

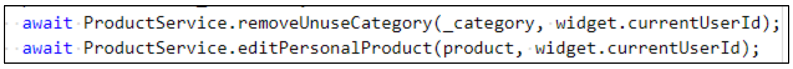
When submitting Add Product Forms, the product details will be added into the database via the *addNewPersonalProduct* function in the Personal Product Services. The category will be removed from the unused category if it is newly added or belongs to an unused category via the *removeUnuseCategory* function in the Personal Product Services. The figure below shows the code segments of *addNewPersonalProduct* function and *removeUnuseCategory* function in the Submit Add Product Form.



```
 await ProductService.removeUnuseCategory(_category, widget.currentUserId);
 await ProductService.addNewPersonalProduct(product, widget.currentUserId);
```

Figure 6.16: Code Segments of Personal Product Services Called in Submit Add Product Form Function on Product Action View

When submitting Edit Product Forms, the product details will be updated in the database via the *editPersonalProduct* function in the Personal Product Services. The category will be removed from the unused category if it is newly added or belongs to an unused category via the *removeUnuseCategory* function in the Personal Product Services. The figure below shows the code segments of *editPersonalProduct* function and *removeUnuseCategory* function in the Submit Edit Product Form.

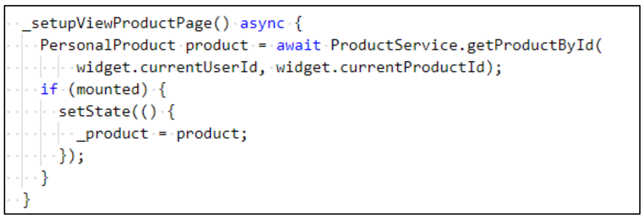


```
await ProductService.removeUnuseCategory(_category, widget.currentUserId);
await ProductService.editPersonalProduct(product, widget.currentUserId);
```

Figure 6.17: Code Segments of Personal Product Services Called in Submit Edit Product Form Function on Product Action View

6.1.2.6 View Product View

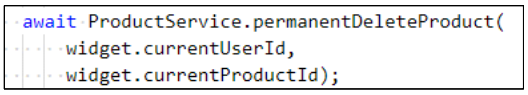
On View Product view, selected product details will be retrieved via the *getProductById* function in the Personal Product Services. The code segments are shown in the figure below.



```
_setupViewProductPage() async {
  PersonalProduct product = await ProductService.getProductById(
      widget.currentUserId, widget.currentProductId);
  if (mounted) {
    setState(() {
      _product = product;
    });
  }
}
```

Figure 6.18: Code Segments of Personal Product Services Called in Setup Product Details Function on View Product View

Besides, the user can also delete the product via the *permenentDeleteProduct* function in the Personal Product Services. The code segments are shown in the figure below.

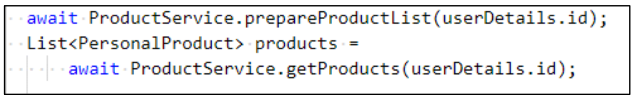


```
await ProductService.permanentDeleteProduct(
    widget.currentUserId,
    widget.currentProductId);
```

Figure 6.19: Code Segments of Personal Product Services Called in Delete Product Function on View Product View

6.1.2.7 Home View

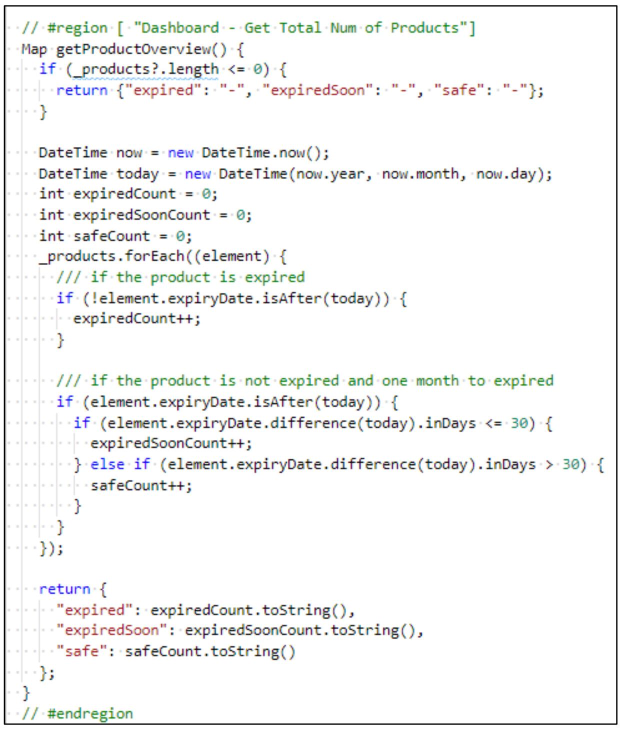
Flutter framework uses widgets to present components in a view. On Home view, all products are retrieved to be used in different parts of the Home view. Besides, the *prepareProductList* function is also called upon setting up the Home view to remove products that are used up from the full product list. The code segments are shown in the figure below.



```
await ProductService.prepareProductList(userDetails.id);
List<PersonalProduct> products =
    await ProductService.getProducts(userDetails.id);
```

Figure 6.20: Code Segments of Personal Product Services Called on Home View

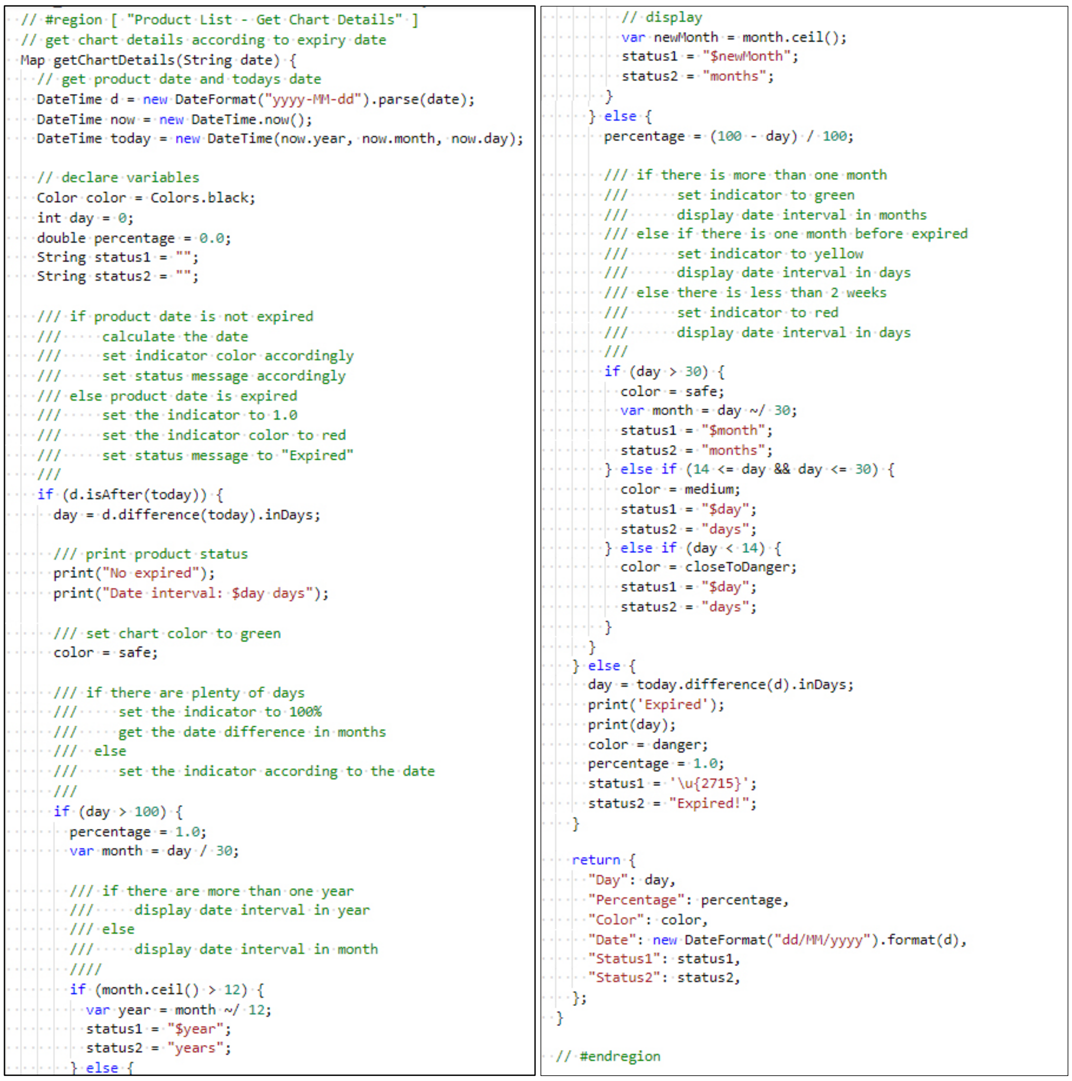
In the dashboard widget, the product list is used to calculate the total number of expired products, soon to expire products and safe to use products. The code segments of calculating the total number of expired products, soon to expire products and safe to use products are shown in the figure below.



```
// #region [ "Dashboard - Get Total Num of Products"]
Map getProductOverview() {
  if (_products?.length <= 0) {
    return {"expired": "-", "expiredSoon": "-", "safe": "-"};
  }

  DateTime now = new DateTime.now();
  DateTime today = new DateTime(now.year, now.month, now.day);
  int expiredCount = 0;
  int expiredSoonCount = 0;
  int safeCount = 0;
  _products.forEach((element) {
    /// if the product is expired
    if (!element.expiryDate.isAfter(today)) {
      expiredCount++;
    }

    /// if the product is not expired and one month to expired
    if (element.expiryDate.isAfter(today)) {
      if (element.expiryDate.difference(today).inDays <= 30) {
        expiredSoonCount++;
      } else if (element.expiryDate.difference(today).inDays > 30) {
        safeCount++;
      }
    }
  });

  return {
    "expired": expiredCount.toString(),
    "expiredSoon": expiredSoonCount.toString(),
    "safe": safeCount.toString()
  };
}
// #endregion
```

Figure 6.21: Code Segments of Getting Product Overview Function on Dashboard Widget

In the product list widget, the product list is displayed in a list and each expiry date of the products are used to calculate the appropriate percentage for the circular indicator to represent the level of severity of the product. The code segments of calculating the percentage of the circular indicator are shown in the figure below.



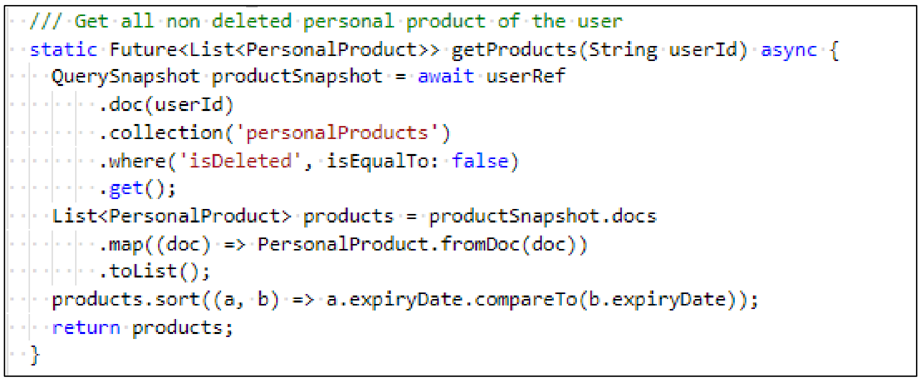
```
// #region [ "Product List - Get Chart Details" ]
// get chart details according to expiry date
Map getChartDetails(String date) {
  // get product date and todays date
  DateTime d = new DateFormat("yyyy-MM-dd").parse(date);
  DateTime now = new DateTime.now();
  DateTime today = new DateTime(now.year, now.month, now.day);

  // declare variables
  Color color = Colors.black;
  int day = 0;
  double percentage = 0.0;
  String status1 = "";
  String status2 = "";

  /// if product date is not expired
  ///     calculate the date
  ///     set indicator color accordingly
  ///     set status message accordingly
  /// else product date is expired
  ///     set the indicator to 1.0
  ///     set the indicator color to red
  ///     set status message to "Expired"
  ///
  if (d.isAfter(today)) {
    day = d.difference(today).inDays;

    /// print product status
    print("No expired");
    print("Date interval: $day days");

    /// set chart color to green
    color = safe;

    /// if there are plenty of days
    ///     set the indicator to 100%
    ///     get the date difference in months
    ///  else
    ///     set the indicator according to the date
    ///
    if (day > 100) {
      percentage = 1.0;
      var month = day / 30;

      /// if there are more than one year
      ///     display date interval in year
      /// else
      ///     display date interval in month
      ////
      if (month.ceil() > 12) {
        var year = month ~/ 12;
        status1 = "$year";
        status2 = "years";
      } else {
        // display
        var newMonth = month.ceil();
        status1 = "$newMonth";
        status2 = "months";
      }
    } else {
      percentage = (100 - day) / 100;

      /// if there is more than one month
      ///      set indicator to green
      ///      display date interval in months
      /// else if there is one month before expired
      ///      set indicator to yellow
      ///      display date interval in days
      /// else there is less than 2 weeks
      ///      set indicator to red
      ///      display date interval in days
      ///
      if (day > 30) {
        color = safe;
        var month = day ~/ 30;
        status1 = "$month";
        status2 = "months";
      } else if (14 <= day && day <= 30) {
        color = medium;
        status1 = "$day";
        status2 = "days";
      } else if (day < 14) {
        color = closeToDanger;
        status1 = "$day";
        status2 = "days";
      }
    }
  } else {
    day = today.difference(d).inDays;
    print('Expired');
    print(day);
    color = danger;
    percentage = 1.0;
    status1 = '\u{2715}';
    status2 = "Expired!";
  }

  return {
    "Day": day,
    "Percentage": percentage,
    "Color": color,
    "Date": new DateFormat("dd/MM/yyyy").format(d),
    "Status1": status1,
    "Status2": status2,
  };
}

// #endregion
```

Figure 6.22: Code Segments of Getting Chart Details Function on Product List Widget

6.1.2.8 Personal Product Services

1. Get Products function

   This function retrieves all of the current user's products that are still in stock. This feature is used for display purposes in the Home view.

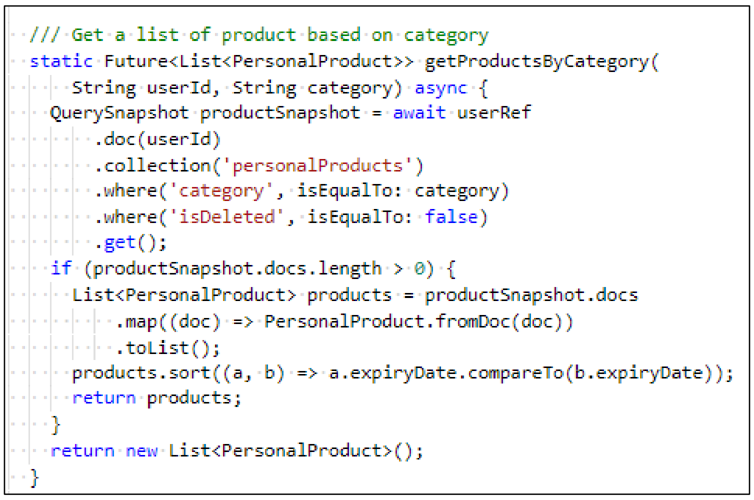


```dart
/// Get all non deleted personal product of the user
static Future<List<PersonalProduct>> getProducts(String userId) async {
  QuerySnapshot productSnapshot = await userRef
      .doc(userId)
      .collection('personalProducts')
      .where('isDeleted', isEqualTo: false)
      .get();
  List<PersonalProduct> products = productSnapshot.docs
      .map((doc) => PersonalProduct.fromDoc(doc))
      .toList();
  products.sort((a, b) => a.expiryDate.compareTo(b.expiryDate));
  return products;
}
```

Figure 6.23: Code Segments of Get Products Function in Personal Product Services

2. Get Products by Category

   This function retrieves all products by a selected category. This function is used in Product by Category View to display all products in the selected category.

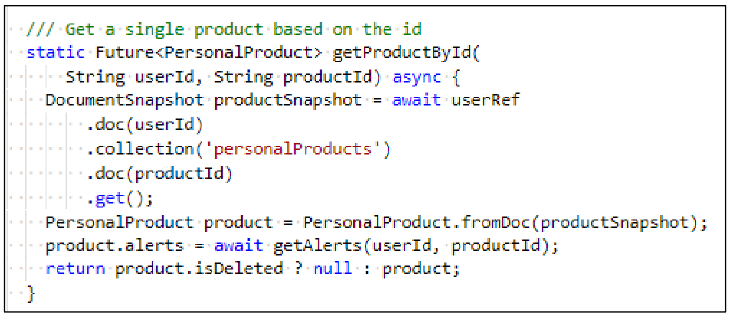


```dart
/// Get a list of product based on category
static Future<List<PersonalProduct>> getProductsByCategory(
    String userId, String category) async {
  QuerySnapshot productSnapshot = await userRef
      .doc(userId)
      .collection('personalProducts')
      .where('category', isEqualTo: category)
      .where('isDeleted', isEqualTo: false)
      .get();
  if (productSnapshot.docs.length > 0) {
    List<PersonalProduct> products = productSnapshot.docs
        .map((doc) => PersonalProduct.fromDoc(doc))
        .toList();
    products.sort((a, b) => a.expiryDate.compareTo(b.expiryDate));
    return products;
  }
  return new List<PersonalProduct>();
}
```

Figure 6.24: Code Segments of Get Products by Category Function in Personal Product Services

3. Get Product by Id

   This function retrieves a single product with its details according to the product id. This function is used in the View Product View and Product Action View to display product details.

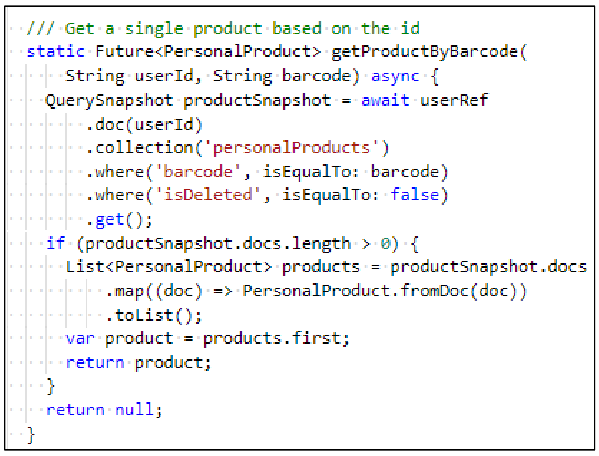


```
/// Get a single product based on the id
static Future<PersonalProduct> getProductById(
    String userId, String productId) async {
  DocumentSnapshot productSnapshot = await userRef
      .doc(userId)
      .collection('personalProducts')
      .doc(productId)
      .get();
  PersonalProduct product = PersonalProduct.fromDoc(productSnapshot);
  product.alerts = await getAlerts(userId, productId);
  return product.isDeleted ? null : product;
}
```

Figure 6.25: Code Segments of Get Product by ID Function in Personal Product Services

4. Get Product by Barcode

   This function retrieves details of a single product when the user scans on a barcode. If the product is not found, the function will return a null value. This function is used in the Search by Barcode view to retrieves product's information for view, edit and delete purpose.

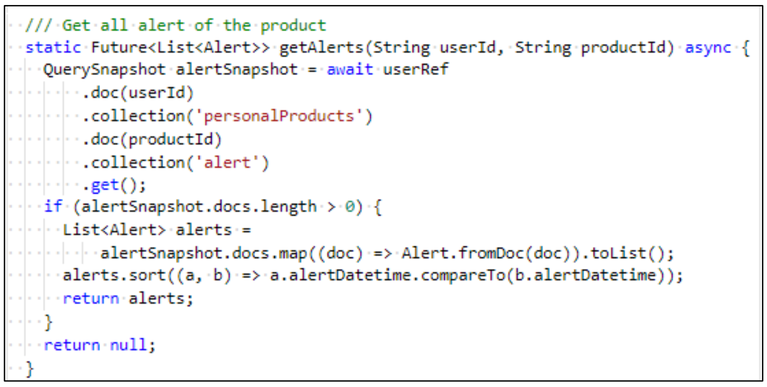


```
/// Get a single product based on the id
static Future<PersonalProduct> getProductByBarcode(
    String userId, String barcode) async {
  QuerySnapshot productSnapshot = await userRef
      .doc(userId)
      .collection('personalProducts')
      .where('barcode', isEqualTo: barcode)
      .where('isDeleted', isEqualTo: false)
      .get();
  if (productSnapshot.docs.length > 0) {
    List<PersonalProduct> products = productSnapshot.docs
        .map((doc) => PersonalProduct.fromDoc(doc))
        .toList();
    var product = products.first;
    return product;
  }
  return null;
}
```

Figure 6.26: Code Segments of Get Product by Barcode Function in Personal Product Services

5. Get Alerts

   This function retrieves all of a product's alerts by the product ID. It is used when the product information is shown or when the product's alert needs to be manipulated.

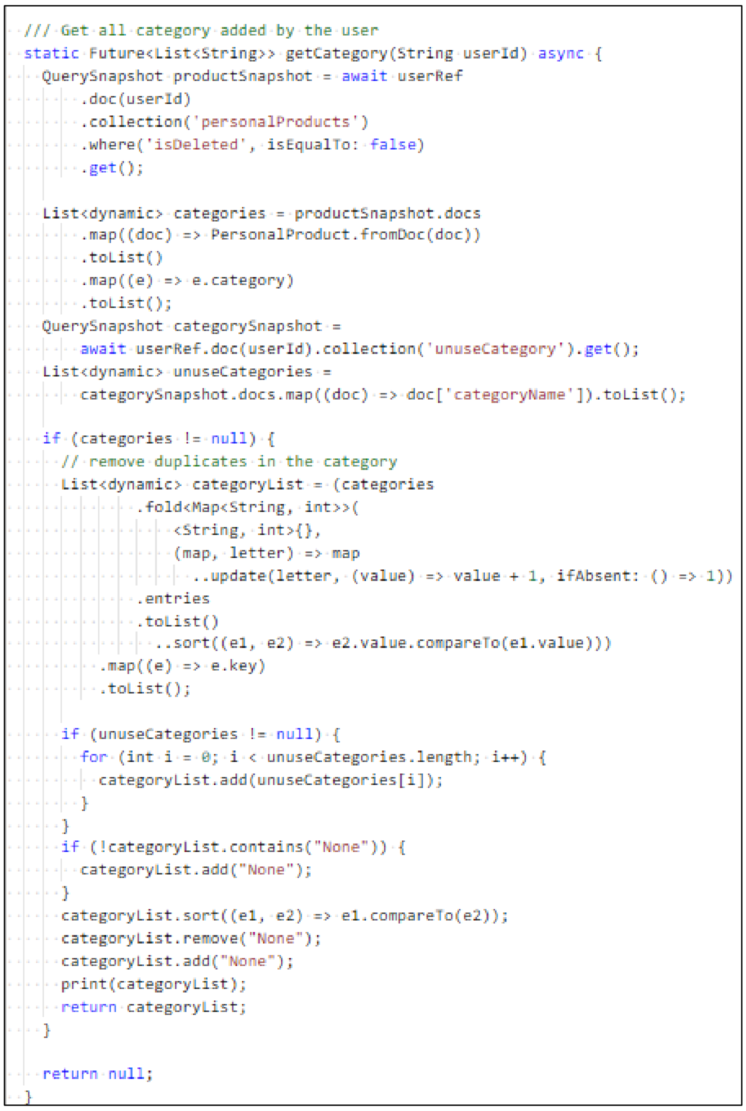


```
/// Get all alert of the product
static Future<List<Alert>> getAlerts(String userId, String productId) async {
  QuerySnapshot alertSnapshot = await userRef
      .doc(userId)
      .collection('personalProducts')
      .doc(productId)
      .collection('alert')
      .get();
  if (alertSnapshot.docs.length > 0) {
    List<Alert> alerts =
        alertSnapshot.docs.map((doc) => Alert.fromDoc(doc)).toList();
    alerts.sort((a, b) => a.alertDatetime.compareTo(b.alertDatetime));
    return alerts;
  }
  return null;
}
```

Figure 6.27: Code Segments of Get Alerts Function in Personal Product Services

6. Get Category

This function is used to obtain categories from the database, both used and unused. The retrieved categories are used to fill a dropdown list for adding and editing products, as well as to view as a list for the user to conveniently manage categories. A none category will be added on the top of the list to provide an option for the users that do not want to categorize his or her products. The function is used in the Product Action view and the Search by Category view.

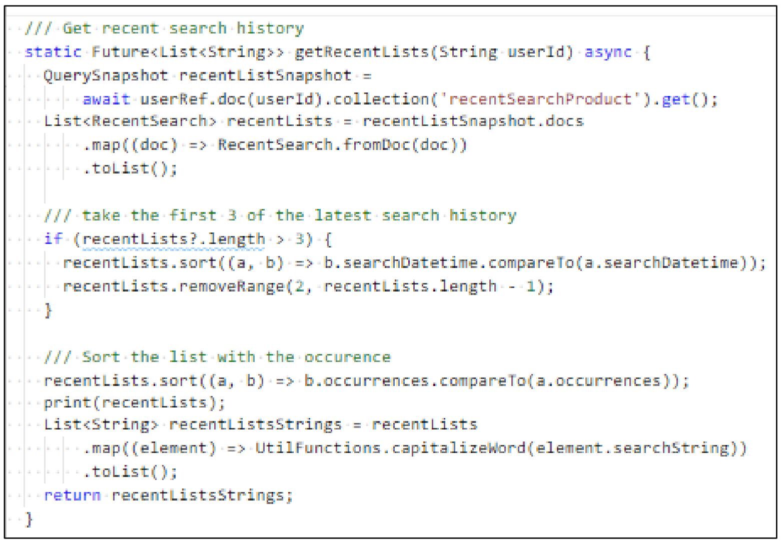


```
/// Get all category added by the user
static Future<List<String>> getCategory(String userId) async {
  QuerySnapshot productSnapshot = await userRef
      .doc(userId)
      .collection('personalProducts')
      .where('isDeleted', isEqualTo: false)
      .get();

  List<dynamic> categories = productSnapshot.docs
      .map((doc) => PersonalProduct.fromDoc(doc))
      .toList()
      .map((e) => e.category)
      .toList();
  QuerySnapshot categorySnapshot =
      await userRef.doc(userId).collection('unuseCategory').get();
  List<dynamic> unuseCategories =
      categorySnapshot.docs.map((doc) => doc['categoryName']).toList();

  if (categories != null) {
    // remove duplicates in the category
    List<dynamic> categoryList = (categories
            .fold<Map<String, int>>(
                <String, int>{},
                (map, letter) => map
                  ..update(letter, (value) => value + 1, ifAbsent: () => 1))
            .entries
            .toList()
              ..sort((e1, e2) => e2.value.compareTo(e1.value)))
        .map((e) => e.key)
        .toList();

    if (unuseCategories != null) {
      for (int i = 0; i < unuseCategories.length; i++) {
        categoryList.add(unuseCategories[i]);
      }
    }
    if (!categoryList.contains("None")) {
      categoryList.add("None");
    }
    categoryList.sort((e1, e2) => e1.compareTo(e2));
    categoryList.remove("None");
    categoryList.add("None");
    print(categoryList);
    return categoryList;
  }

  return null;
}
```

Figure 6.28: Code Segments of Get Category Function in Personal Product Services

7. Get Recent List

   This function retrieves the most recent search list to be used as the search delegate's suggested list in the Search by Name view. Only the first three most recent search strings will be obtained, with the search string of the most occurrences being at the top.

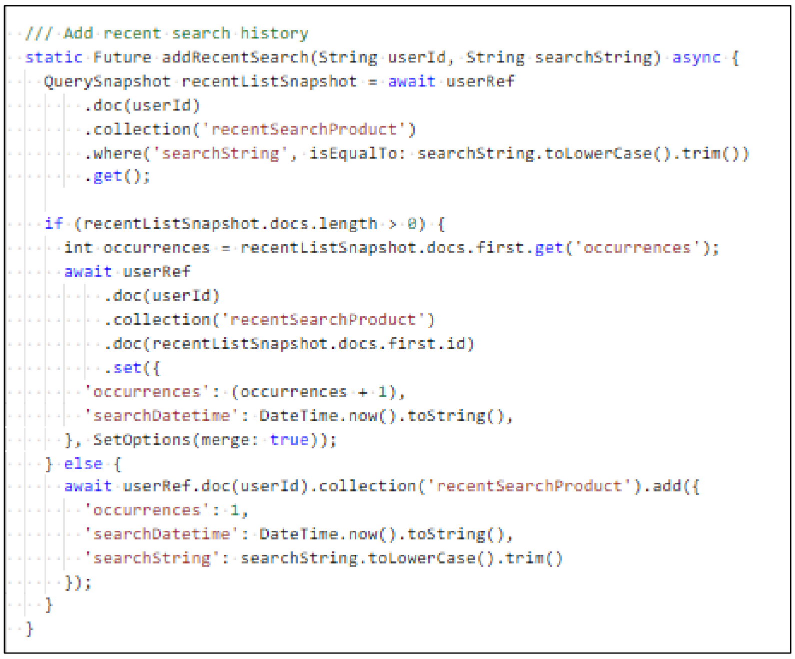


```
/// Get recent search history
static Future<List<String>> getRecentLists(String userId) async {
  QuerySnapshot recentListSnapshot =
      await userRef.doc(userId).collection('recentSearchProduct').get();
  List<RecentSearch> recentLists = recentListSnapshot.docs
      .map((doc) => RecentSearch.fromDoc(doc))
      .toList();

  /// take the first 3 of the latest search history
  if (recentLists?.length > 3) {
    recentLists.sort((a, b) => b.searchDatetime.compareTo(a.searchDatetime));
    recentLists.removeRange(2, recentLists.length - 1);
  }

  /// Sort the list with the occurence
  recentLists.sort((a, b) => b.occurrences.compareTo(a.occurrences));
  print(recentLists);
  List<String> recentListsStrings = recentLists
      .map((element) => UtilFunctions.capitalizeWord(element.searchString))
      .toList();
  return recentListsStrings;
}
```

Figure 6.29: Code Segments of Get Recent List Function in Personal Product Services

8. Add Recent Search

This function is used to update the recent search list, either by inserting a new recent search list if the search string does not exist or by adding the occurrence by one and updating the search date and time if the search string exists.

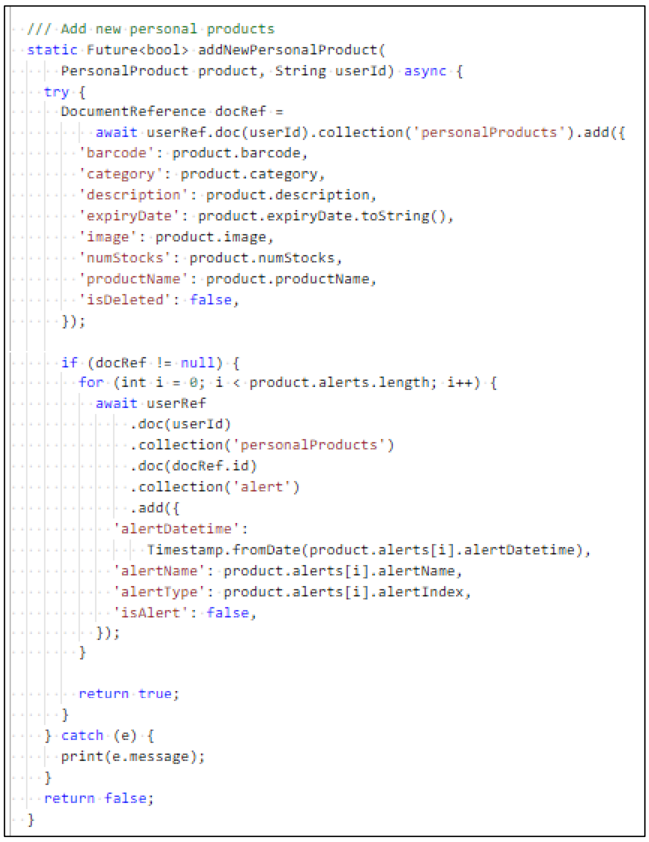


```
/// Add recent search history
static Future addRecentSearch(String userId, String searchString) async {
  QuerySnapshot recentListSnapshot = await userRef
      .doc(userId)
      .collection('recentSearchProduct')
      .where('searchString', isEqualTo: searchString.toLowerCase().trim())
      .get();

  if (recentListSnapshot.docs.length > 0) {
    int occurrences = recentListSnapshot.docs.first.get('occurrences');
    await userRef
        .doc(userId)
        .collection('recentSearchProduct')
        .doc(recentListSnapshot.docs.first.id)
        .set({
      'occurrences': (occurrences + 1),
      'searchDatetime': DateTime.now().toString(),
    }, SetOptions(merge: true));
  } else {
    await userRef.doc(userId).collection('recentSearchProduct').add({
      'occurrences': 1,
      'searchDatetime': DateTime.now().toString(),
      'searchString': searchString.toLowerCase().trim()
    });
  }
}
```

Figure 6.30: Code Segments of Add Recent Search Function in Personal Product Services

9. Add New Personal Product

This function is to add a new personal product and its alerts for the Product Action view.

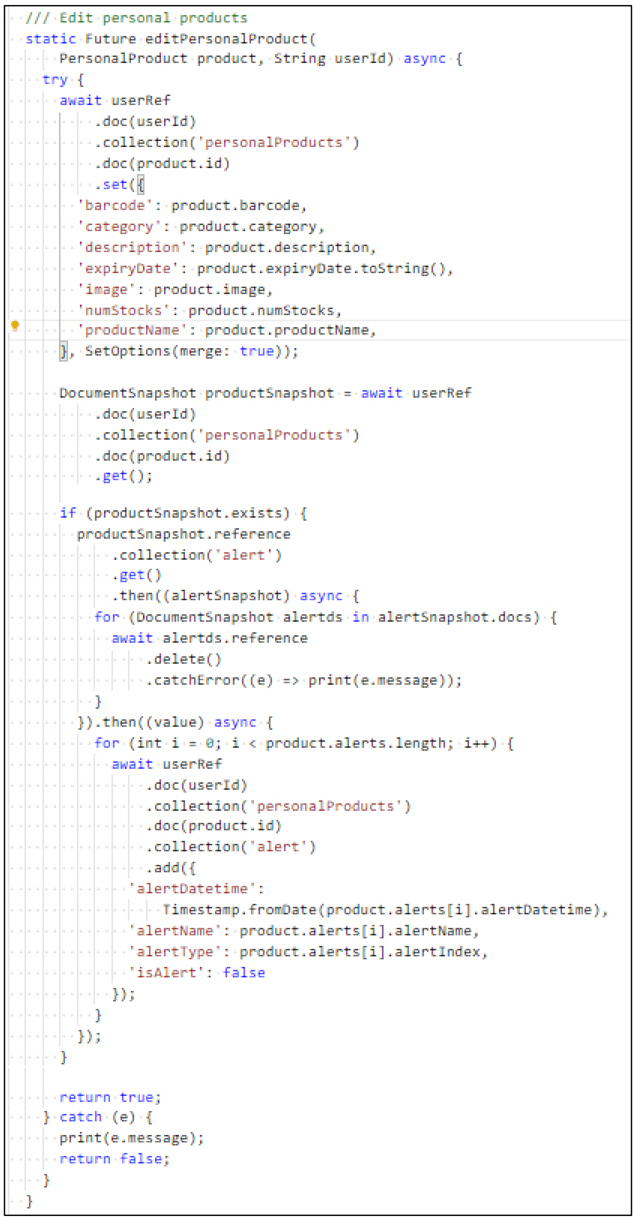


```
/// Add new personal products
static Future<bool> addNewPersonalProduct(
      PersonalProduct product, String userId) async {
  try {
    DocumentReference docRef =
        await userRef.doc(userId).collection('personalProducts').add({
      'barcode': product.barcode,
      'category': product.category,
      'description': product.description,
      'expiryDate': product.expiryDate.toString(),
      'image': product.image,
      'numStocks': product.numStocks,
      'productName': product.productName,
      'isDeleted': false,
    });

    if (docRef != null) {
      for (int i = 0; i < product.alerts.length; i++) {
        await userRef
            .doc(userId)
            .collection('personalProducts')
            .doc(docRef.id)
            .collection('alert')
            .add({
          'alertDatetime':
              Timestamp.fromDate(product.alerts[i].alertDatetime),
          'alertName': product.alerts[i].alertName,
          'alertType': product.alerts[i].alertIndex,
          'isAlert': false,
        });
      }

      return true;
    }
  } catch (e) {
    print(e.message);
  }
  return false;
}
```

Figure 6.31: Code Segments of Add New Personal Product Function in Personal Product Services

10. Edit Personal Product

This function is to edit the personal product and its alerts for the Product Action view.

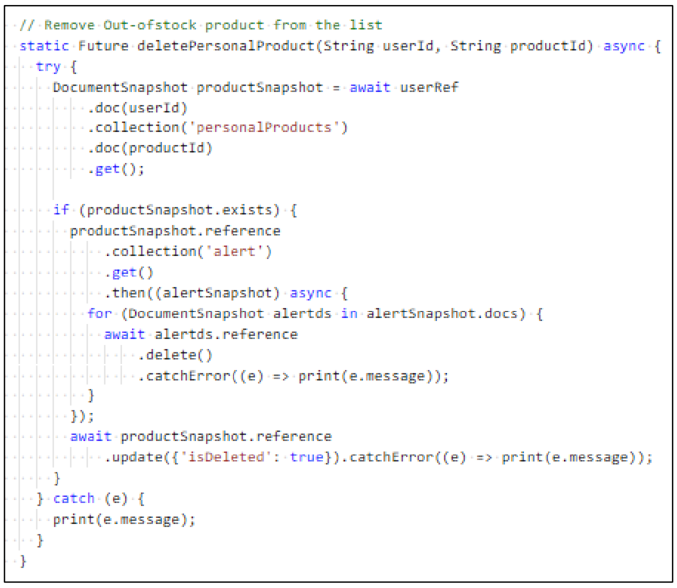


```
/// Edit personal products
static Future editPersonalProduct(
    PersonalProduct product, String userId) async {
  try {
    await userRef
        .doc(userId)
        .collection('personalProducts')
        .doc(product.id)
        .set({
      'barcode': product.barcode,
      'category': product.category,
      'description': product.description,
      'expiryDate': product.expiryDate.toString(),
      'image': product.image,
      'numStocks': product.numStocks,
      'productName': product.productName,
    }, SetOptions(merge: true));

    DocumentSnapshot productSnapshot = await userRef
        .doc(userId)
        .collection('personalProducts')
        .doc(product.id)
        .get();

    if (productSnapshot.exists) {
      productSnapshot.reference
          .collection('alert')
          .get()
          .then((alertSnapshot) async {
        for (DocumentSnapshot alertds in alertSnapshot.docs) {
          await alertds.reference
              .delete()
              .catchError((e) => print(e.message));
        }
      }).then((value) async {
        for (int i = 0; i < product.alerts.length; i++) {
          await userRef
              .doc(userId)
              .collection('personalProducts')
              .doc(product.id)
              .collection('alert')
              .add({
            'alertDatetime':
                Timestamp.fromDate(product.alerts[i].alertDatetime),
            'alertName': product.alerts[i].alertName,
            'alertType': product.alerts[i].alertIndex,
            'isAlert': false
          });
        }
      });
    }

    return true;
  } catch (e) {
    print(e.message);
    return false;
  }
}
```

Figure 6.32: Code Segments of Edit Personal Product Function in Personal Product Services

11. Delete Personal Product

This function clears the list of out-of-stock personal products. This is because after a product has been finished, there is no reason for the user to keep track of the expiry dates.

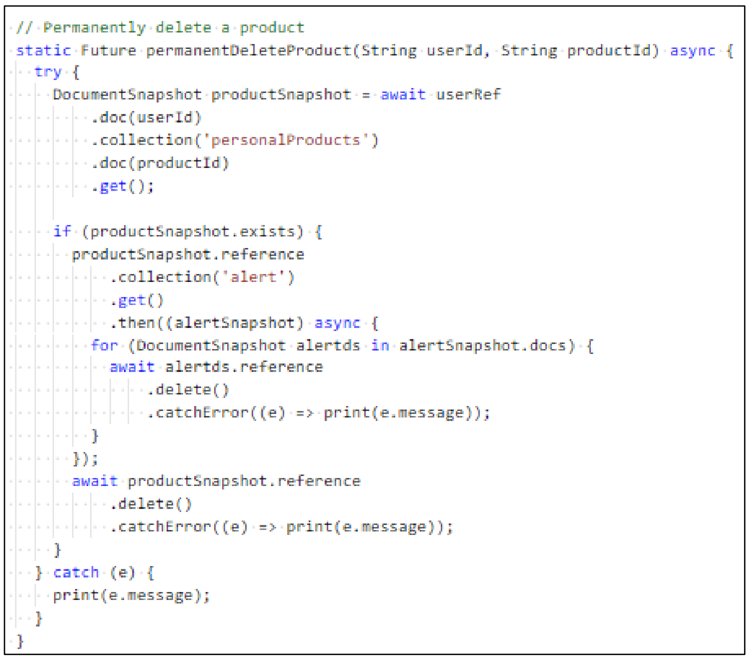


```
// Remove Out-ofstock product from the list
static Future deletePersonalProduct(String userId, String productId) async {
  try {
    DocumentSnapshot productSnapshot = await userRef
        .doc(userId)
        .collection('personalProducts')
        .doc(productId)
        .get();

    if (productSnapshot.exists) {
      productSnapshot.reference
          .collection('alert')
          .get()
          .then((alertSnapshot) async {
        for (DocumentSnapshot alertds in alertSnapshot.docs) {
          await alertds.reference
              .delete()
              .catchError((e) => print(e.message));
        }
      });
      await productSnapshot.reference
          .update({'isDeleted': true}).catchError((e) => print(e.message));
    }
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.33: Code Segments of Delete Personal Product Function in Personal Product Services

12. Permanent Delete Product

This function is to delete the personal product and its alerts permanently for the View Product view.

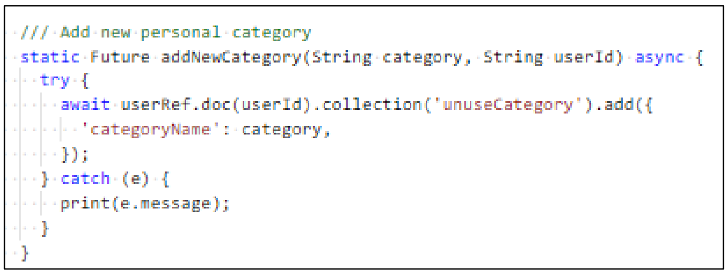


```
// Permanently delete a product
static Future permanentDeleteProduct(String userId, String productId) async {
  try {
    DocumentSnapshot productSnapshot = await userRef
        .doc(userId)
        .collection('personalProducts')
        .doc(productId)
        .get();

    if (productSnapshot.exists) {
      productSnapshot.reference
          .collection('alert')
          .get()
          .then((alertSnapshot) async {
        for (DocumentSnapshot alertds in alertSnapshot.docs) {
          await alertds.reference
              .delete()
              .catchError((e) => print(e.message));
        }
      });
      await productSnapshot.reference
          .delete()
          .catchError((e) => print(e.message));
    }
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.34: Code Segments of Delete Personal Product Function in Personal Product Services

13. Add New Category

This function is to add a new category into the unused category collection. This function is used in the Product Action view and the Search by Category view.

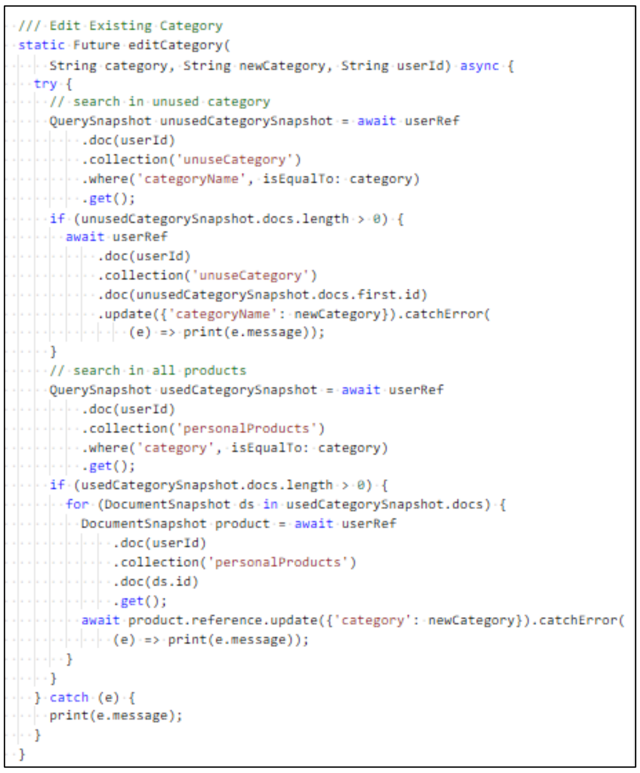


```
/// Add new personal category
static Future addNewCategory(String category, String userId) async {
  try {
    await userRef.doc(userId).collection('unuseCategory').add({
      'categoryName': category,
    });
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.35: Code Segments of Add New CategoryFunction in Personal Product Services

14. Edit Category

This function is to edit the existing category from both unused categories and used categories. This function is used in the Search by Category view.

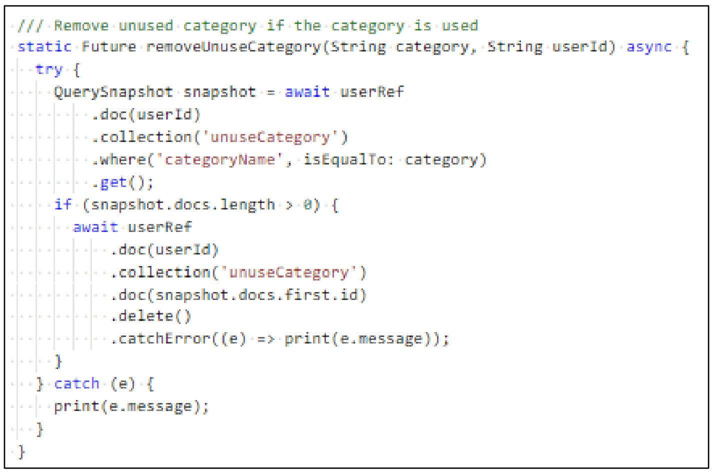


```
/// Edit Existing Category
static Future editCategory(
    String category, String newCategory, String userId) async {
  try {
    // search in unused category
    QuerySnapshot unusedCategorySnapshot = await userRef
        .doc(userId)
        .collection('unuseCategory')
        .where('categoryName', isEqualTo: category)
        .get();
    if (unusedCategorySnapshot.docs.length > 0) {
      await userRef
          .doc(userId)
          .collection('unuseCategory')
          .doc(unusedCategorySnapshot.docs.first.id)
          .update({'categoryName': newCategory}).catchError(
              (e) => print(e.message));
    }
    // search in all products
    QuerySnapshot usedCategorySnapshot = await userRef
        .doc(userId)
        .collection('personalProducts')
        .where('category', isEqualTo: category)
        .get();
    if (usedCategorySnapshot.docs.length > 0) {
      for (DocumentSnapshot ds in usedCategorySnapshot.docs) {
        DocumentSnapshot product = await userRef
            .doc(userId)
            .collection('personalProducts')
            .doc(ds.id)
            .get();
        await product.reference.update({'category': newCategory}).catchError(
            (e) => print(e.message));
      }
    }
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.36: Code Segments of Edit Personal Product Function in Personal Product Services

15. Remove Unused Category

This function is used when a product is added to an unused category. The category will be removed from the unused category collection and is called when adding a new product or editing an existing product. The function is used in the Product Action view.

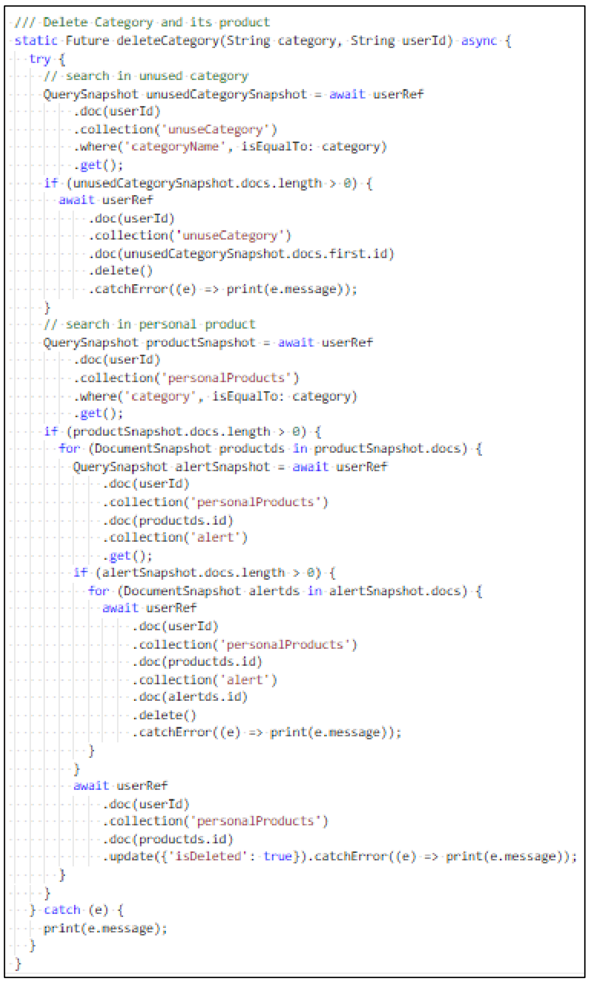


```
/// Remove unused category if the category is used
static Future removeUnuseCategory(String category, String userId) async {
  try {
    QuerySnapshot snapshot = await userRef
        .doc(userId)
        .collection('unuseCategory')
        .where('categoryName', isEqualTo: category)
        .get();
    if (snapshot.docs.length > 0) {
      await userRef
          .doc(userId)
          .collection('unuseCategory')
          .doc(snapshot.docs.first.id)
          .delete()
          .catchError((e) => print(e.message));
    }
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.37: Code Segments of Remove Unused Category Function in Personal Product Services

16. Delete Category

This function will delete the existing category permanently. This function is used in the Search by Category view.

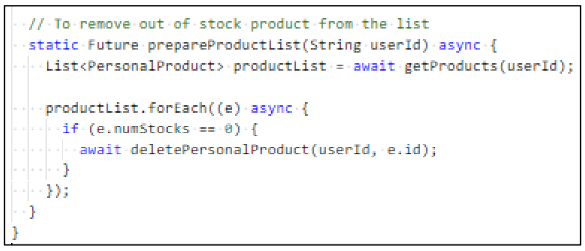


```
/// Delete Category and its product
static Future deleteCategory(String category, String userId) async {
  try {
    // search in unused category
    QuerySnapshot unusedCategorySnapshot = await userRef
        .doc(userId)
        .collection('unuseCategory')
        .where('categoryName', isEqualTo: category)
        .get();
    if (unusedCategorySnapshot.docs.length > 0) {
      await userRef
          .doc(userId)
          .collection('unuseCategory')
          .doc(unusedCategorySnapshot.docs.first.id)
          .delete()
          .catchError((e) => print(e.message));
    }
    // search in personal product
    QuerySnapshot productSnapshot = await userRef
        .doc(userId)
        .collection('personalProducts')
        .where('category', isEqualTo: category)
        .get();
    if (productSnapshot.docs.length > 0) {
      for (DocumentSnapshot productds in productSnapshot.docs) {
        QuerySnapshot alertSnapshot = await userRef
            .doc(userId)
            .collection('personalProducts')
            .doc(productds.id)
            .collection('alert')
            .get();
        if (alertSnapshot.docs.length > 0) {
          for (DocumentSnapshot alertds in alertSnapshot.docs) {
            await userRef
                .doc(userId)
                .collection('personalProducts')
                .doc(productds.id)
                .collection('alert')
                .doc(alertds.id)
                .delete()
                .catchError((e) => print(e.message));
          }
        }
        await userRef
            .doc(userId)
            .collection('personalProducts')
            .doc(productds.id)
            .update({'isDeleted': true}).catchError((e) => print(e.message));
      }
    }
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.38: Code Segments of Delete Category Function in Personal Product Services

17. Prepare Product List

This function removes out-of-stock items by calling the Delete Personal Product function. The function is used in the Home view.

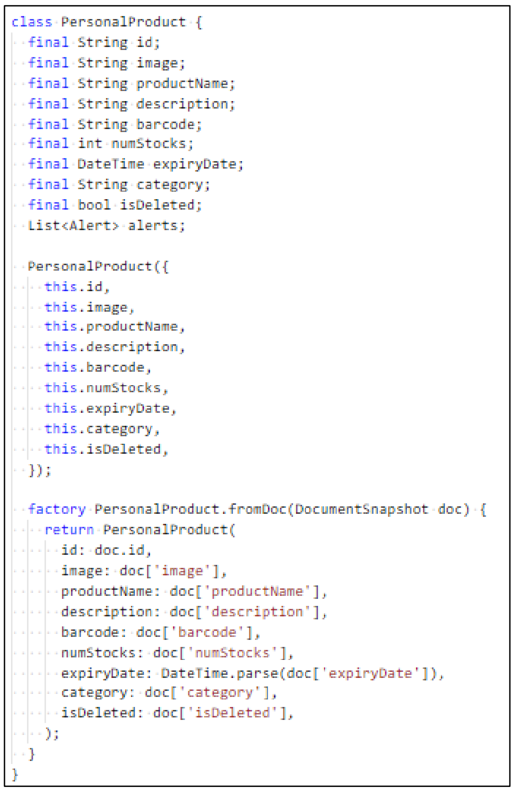


```
// To remove out of stock product from the list
static Future prepareProductList(String userId) async {
  List<PersonalProduct> productList = await getProducts(userId);

  productList.forEach((e) async {
    if (e.numStocks == 0) {
      await deletePersonalProduct(userId, e.id);
    }
  });
}
}
```

Figure 6.39: Code Segments of Prepare Product List Function in Personal Product Services

6.1.2.9 Personal Product Model

The figure below shows the code segments of the personal product model.

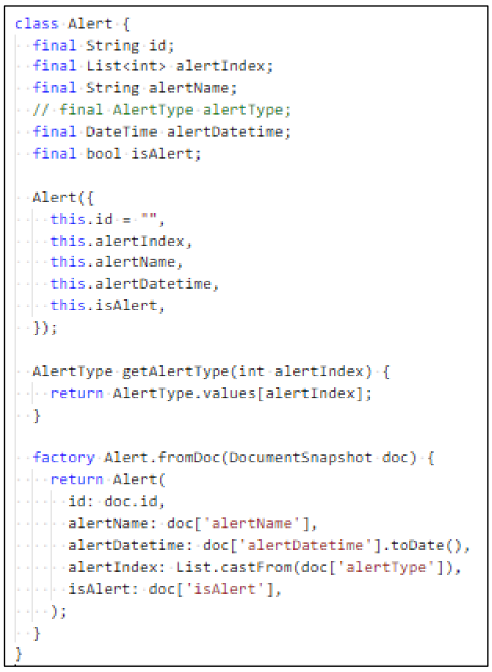


```
class PersonalProduct {
  final String id;
  final String image;
  final String productName;
  final String description;
  final String barcode;
  final int numStocks;
  final DateTime expiryDate;
  final String category;
  final bool isDeleted;
  List<Alert> alerts;

  PersonalProduct({
    this.id,
    this.image,
    this.productName,
    this.description,
    this.barcode,
    this.numStocks,
    this.expiryDate,
    this.category,
    this.isDeleted,
  });

  factory PersonalProduct.fromDoc(DocumentSnapshot doc) {
    return PersonalProduct(
      id: doc.id,
      image: doc['image'],
      productName: doc['productName'],
      description: doc['description'],
      barcode: doc['barcode'],
      numStocks: doc['numStocks'],
      expiryDate: DateTime.parse(doc['expiryDate']),
      category: doc['category'],
      isDeleted: doc['isDeleted'],
    );
  }
}
```

Figure 6.40: Code Segments of Personal Product Model

6.1.2.10 Alert Model

The figure below shows the code segments of the alert model.



```
class Alert {
  final String id;
  final List<int> alertIndex;
  final String alertName;
  // final AlertType alertType;
  final DateTime alertDatetime;
  final bool isAlert;

  Alert({
    this.id = "",
    this.alertIndex,
    this.alertName,
    this.alertDatetime,
    this.isAlert,
  });

  AlertType getAlertType(int alertIndex) {
    return AlertType.values[alertIndex];
  }

  factory Alert.fromDoc(DocumentSnapshot doc) {
    return Alert(
      id: doc.id,
      alertName: doc['alertName'],
      alertDatetime: doc['alertDatetime'].toDate(),
      alertIndex: List.castFrom(doc['alertType']),
      isAlert: doc['isAlert'],
    );
  }
}
```

Figure 6.41: Code Segments of Personal Product Model

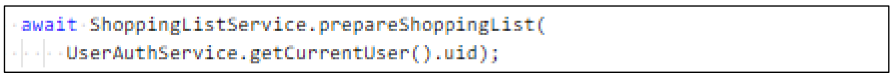
6.1.3 Shopping List Module

Table 6.3: Model, Services and View for Shopping List Module

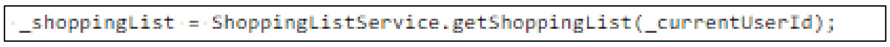
Model	Shopping List Model
Services	Shopping List Service
Views	Shopping List Shopping List Action

6.1.3.1 Shopping List View

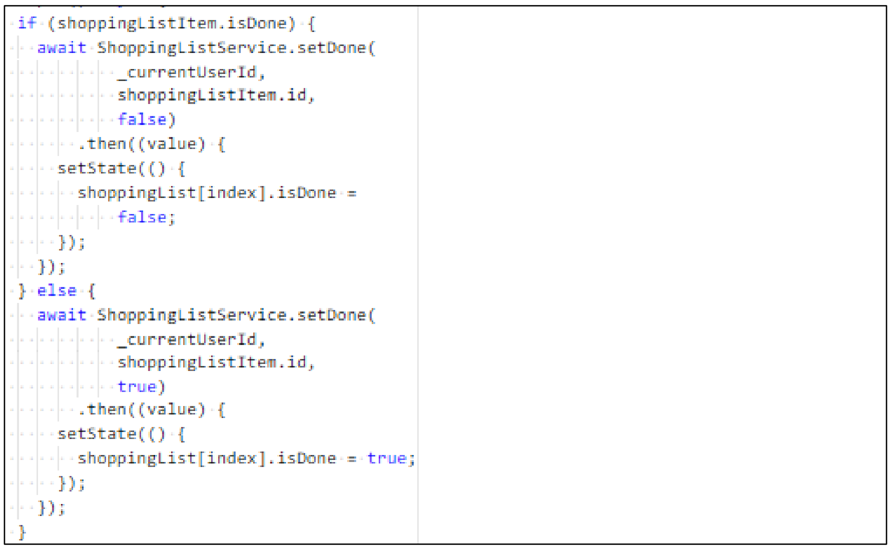
On the Shopping List view, the *getShoppingList* function is called to retrieve the shopping list while the *prepareShoppingList* function is called upon init state to move expired and out-of-stock product into the shopping list. User can either check or uncheck a shopping list item or delete a shopping list item. The *setDone* function is called to check and uncheck the shopping list item, while the *deleteShoppingList* function is called to delete the shopping list item. The code segments are shown in the figure below.



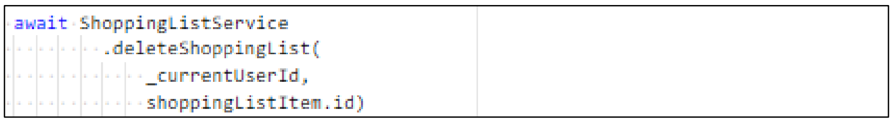
```
await ShoppingListService.prepareShoppingList(
      UserAuthService.getCurrentUser().uid);
```



```
_shoppingList = ShoppingListService.getShoppingList(_currentUserId);
```



```
if (shoppingListItem.isDone) {
  await ShoppingListService.setDone(
          _currentUserId,
          shoppingListItem.id,
          false)
      .then((value) {
    setState(() {
      shoppingList[index].isDone =
          false;
    });
  });
} else {
  await ShoppingListService.setDone(
          _currentUserId,
          shoppingListItem.id,
          true)
      .then((value) {
    setState(() {
      shoppingList[index].isDone = true;
    });
  });
}
```

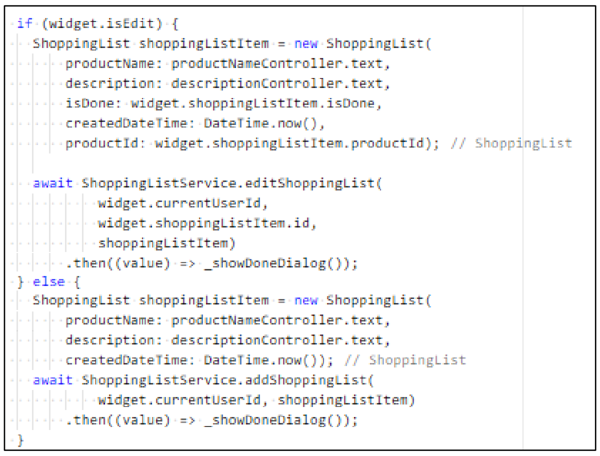


```
await ShoppingListService
        .deleteShoppingList(
            _currentUserId,
            shoppingListItem.id)
```

Figure 6.42: Code Segments of Shopping List Services Called on Shopping List View

6.1.3.2 Shopping List Action View

On the Shopping List Action view, users can perform add or edit shopping list item based on what they had selected on the Shopping List view. Same as the Product Action view, the same view is used for both add and edit functions to reduce code repetition. The code segments are shown in the figure below.



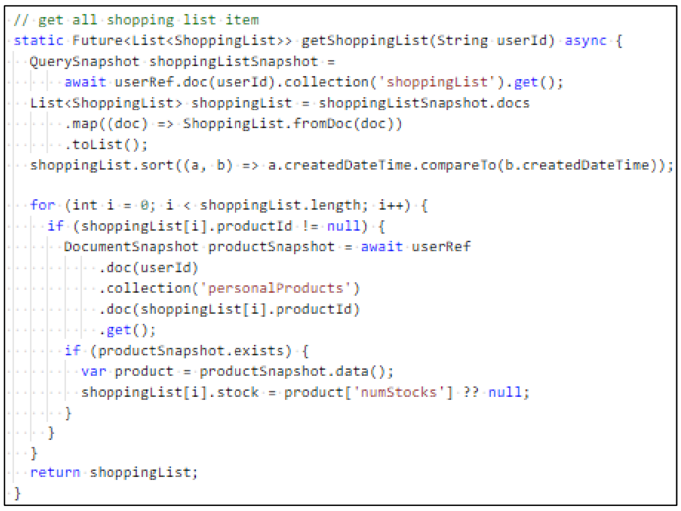
```dart
if (widget.isEdit) {
  ShoppingList shoppingListItem = new ShoppingList(
      productName: productNameController.text,
      description: descriptionController.text,
      isDone: widget.shoppingListItem.isDone,
      createdDateTime: DateTime.now(),
      productId: widget.shoppingListItem.productId); // ShoppingList

  await ShoppingListService.editShoppingList(
          widget.currentUserId,
          widget.shoppingListItem.id,
          shoppingListItem)
      .then((value) => _showDoneDialog());
} else {
  ShoppingList shoppingListItem = new ShoppingList(
      productName: productNameController.text,
      description: descriptionController.text,
      createdDateTime: DateTime.now()); // ShoppingList
  await ShoppingListService.addShoppingList(
          widget.currentUserId, shoppingListItem)
      .then((value) => _showDoneDialog());
}
```

Figure 6.43: Code Segments of Shopping List Services Called on Shopping List Action View

6.1.3.3 Shopping List Services

1. Get Shopping List

   This function retrieves the whole shopping list from the database. This function is used in the Shopping List view.

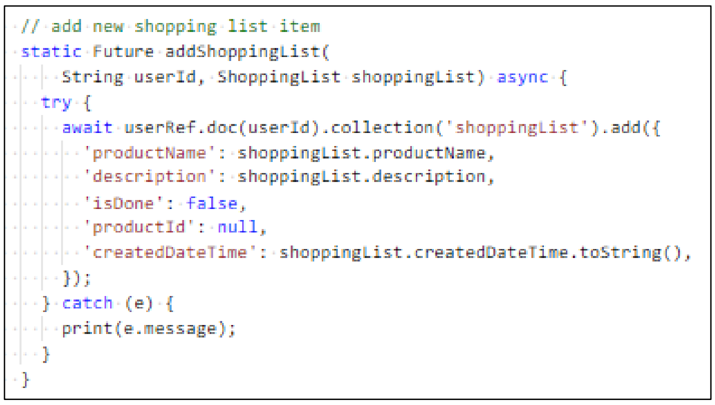


```dart
// get all shopping list item
static Future<List<ShoppingList>> getShoppingList(String userId) async {
  QuerySnapshot shoppingListSnapshot =
      await userRef.doc(userId).collection('shoppingList').get();
  List<ShoppingList> shoppingList = shoppingListSnapshot.docs
      .map((doc) => ShoppingList.fromDoc(doc))
      .toList();
  shoppingList.sort((a, b) => a.createdDateTime.compareTo(b.createdDateTime));

  for (int i = 0; i < shoppingList.length; i++) {
    if (shoppingList[i].productId != null) {
      DocumentSnapshot productSnapshot = await userRef
          .doc(userId)
          .collection('personalProducts')
          .doc(shoppingList[i].productId)
          .get();
      if (productSnapshot.exists) {
        var product = productSnapshot.data();
        shoppingList[i].stock = product['numStocks'] ?? null;
      }
    }
  }
  return shoppingList;
}
```

Figure 6.44: Code Segments of Get Shopping List Function in Shopping List Services

2. Add Shopping List

   This function adds new shopping list item into the database. This function is called in the Shopping List Action view.

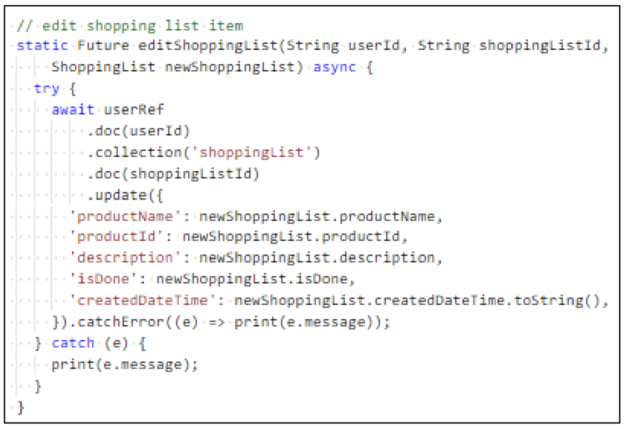


```
// add new shopping list item
static Future addShoppingList(
    String userId, ShoppingList shoppingList) async {
  try {
    await userRef.doc(userId).collection('shoppingList').add({
      'productName': shoppingList.productName,
      'description': shoppingList.description,
      'isDone': false,
      'productId': null,
      'createdDateTime': shoppingList.createdDateTime.toString(),
    });
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.45: Code Segments of Add Shopping List Function in Shopping List Services

3. Edit Shopping List

   This function edits the existing shopping list item from the database. This function is called in the Shopping List Action view.

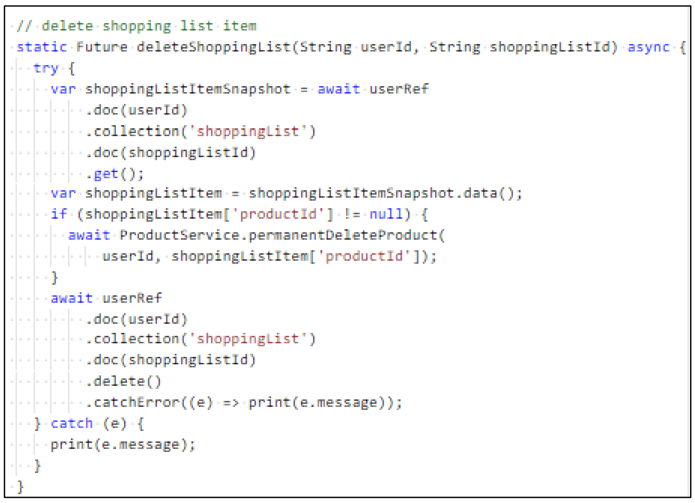


```
// edit shopping list item
static Future editShoppingList(String userId, String shoppingListId,
    ShoppingList newShoppingList) async {
  try {
    await userRef
        .doc(userId)
        .collection('shoppingList')
        .doc(shoppingListId)
        .update({
      'productName': newShoppingList.productName,
      'productId': newShoppingList.productId,
      'description': newShoppingList.description,
      'isDone': newShoppingList.isDone,
      'createdDateTime': newShoppingList.createdDateTime.toString(),
    }).catchError((e) => print(e.message));
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.46: Code Segments of Edit Shopping List Function in Shopping List Services

4. Delete Shopping List

   This function deletes the existing shopping list item from the database. This function is called in the Shopping List view.



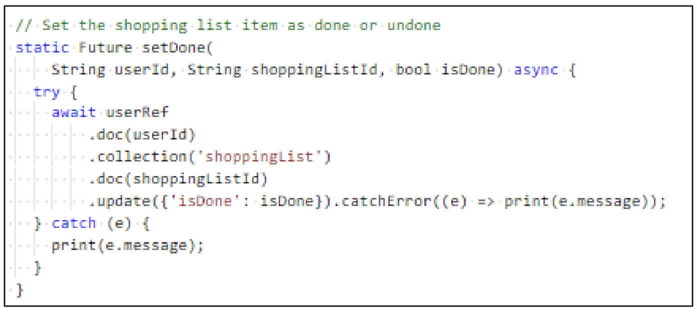
```
// delete shopping list item
static Future deleteShoppingList(String userId, String shoppingListId) async {
  try {
    var shoppingListItemSnapshot = await userRef
        .doc(userId)
        .collection('shoppingList')
        .doc(shoppingListId)
        .get();
    var shoppingListItem = shoppingListItemSnapshot.data();
    if (shoppingListItem['productId'] != null) {
      await ProductService.permanentDeleteProduct(
          userId, shoppingListItem['productId']);
    }
    await userRef
        .doc(userId)
        .collection('shoppingList')
        .doc(shoppingListId)
        .delete()
        .catchError((e) => print(e.message));
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.47: Code Segments of Delete Shopping List Function in Shopping List Services

.

5. Set Done

   This function sets the existing shopping list item from the database to done and undone. To set it as done, pass true as the isDone variable and vice versa. This function is called in the Shopping List view.

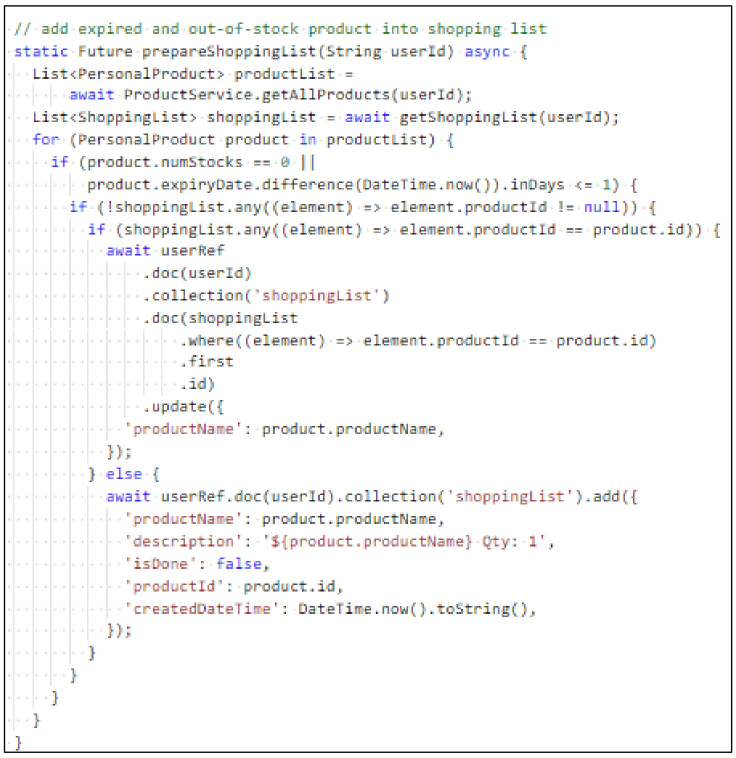


```
// Set the shopping list item as done or undone
static Future setDone(
    String userId, String shoppingListId, bool isDone) async {
  try {
    await userRef
        .doc(userId)
        .collection('shoppingList')
        .doc(shoppingListId)
        .update({'isDone': isDone}).catchError((e) => print(e.message));
  } catch (e) {
    print(e.message);
  }
}
```

Figure 6.48: Code Segments of Delete Shopping List Function in Shopping List Services

6. Prepare Shopping List

   This function is to add expired products and out-of-stock products into the shopping list. It is an automated process and will be run every time the Shopping List View is initialized.

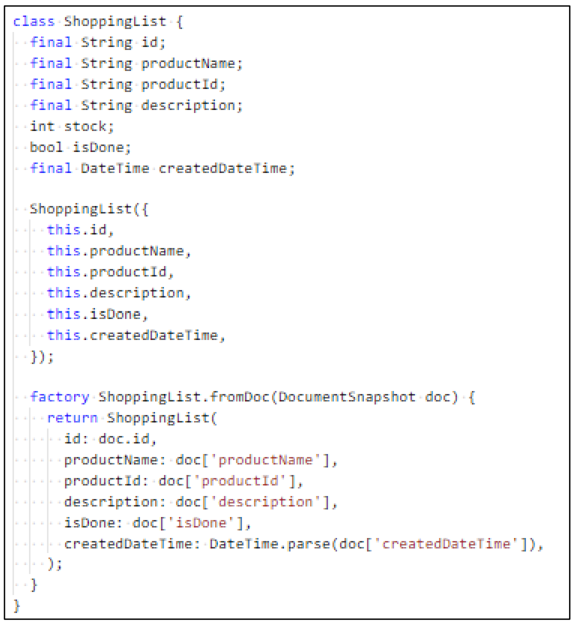


```
// add expired and out-of-stock product into shopping list
static Future prepareShoppingList(String userId) async {
  List<PersonalProduct> productList =
      await ProductService.getAllProducts(userId);
  List<ShoppingList> shoppingList = await getShoppingList(userId);
  for (PersonalProduct product in productList) {
    if (product.numStocks == 0 ||
        product.expiryDate.difference(DateTime.now()).inDays <= 1) {
      if (!shoppingList.any((element) => element.productId != null)) {
        if (shoppingList.any((element) => element.productId == product.id)) {
          await userRef
              .doc(userId)
              .collection('shoppingList')
              .doc(shoppingList
                  .where((element) => element.productId == product.id)
                  .first
                  .id)
              .update({
            'productName': product.productName,
          });
        } else {
          await userRef.doc(userId).collection('shoppingList').add({
            'productName': product.productName,
            'description': '${product.productName} Qty: 1',
            'isDone': false,
            'productId': product.id,
            'createdDateTime': DateTime.now().toString(),
          });
        }
      }
    }
  }
}
```

Figure 6.49: Code Segments of Prepare Shopping List Function in Shopping List Services

6.1.3.4 Shopping List Model

The figure below shows the code segments of the shopping list model.



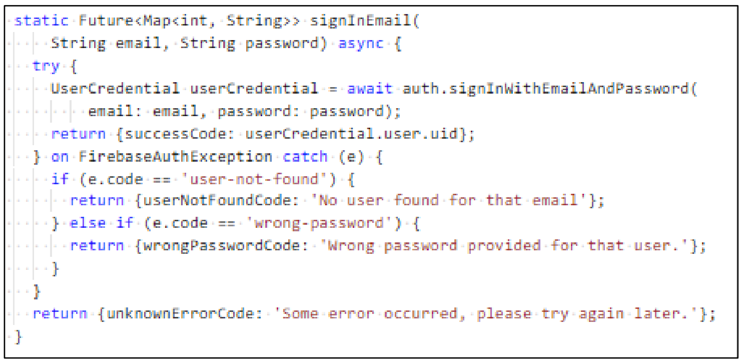
```
class ShoppingList {
  final String id;
  final String productName;
  final String productId;
  final String description;
  int stock;
  bool isDone;
  final DateTime createdDateTime;

  ShoppingList({
    this.id,
    this.productName,
    this.productId,
    this.description,
    this.isDone,
    this.createdDateTime,
  });

  factory ShoppingList.fromDoc(DocumentSnapshot doc) {
    return ShoppingList(
      id: doc.id,
      productName: doc['productName'],
      productId: doc['productId'],
      description: doc['description'],
      isDone: doc['isDone'],
      createdDateTime: DateTime.parse(doc['createdDateTime']),
    );
  }
}
```

Figure 6.50: Code Segments of Shopping List Model

6.2 Implementation of Firebase Authentication

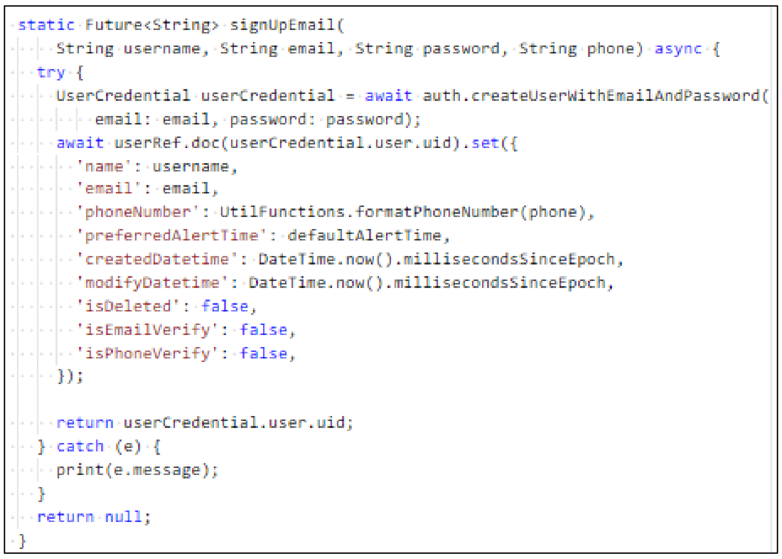
Firebase Authentication is used to authenticate users in many different ways. Besides, Firebase Authentication also provides email verification, phone number verification, and password resets functions that are ready to be used. Third-party authentication is also available to be used. Firebase Authentication is safe to use because it will generate random user id upon creating new users and automatically hash passwords before stored into the cloud storage.

All Firebase Authentication functions used for the mobile application are written in the User Auth Service so that it can be called in the views easily. The authentication methods used in the mobile application are email and password authentication, Google authentication and also Facebook authentication. The code segments are shown in the figure below.



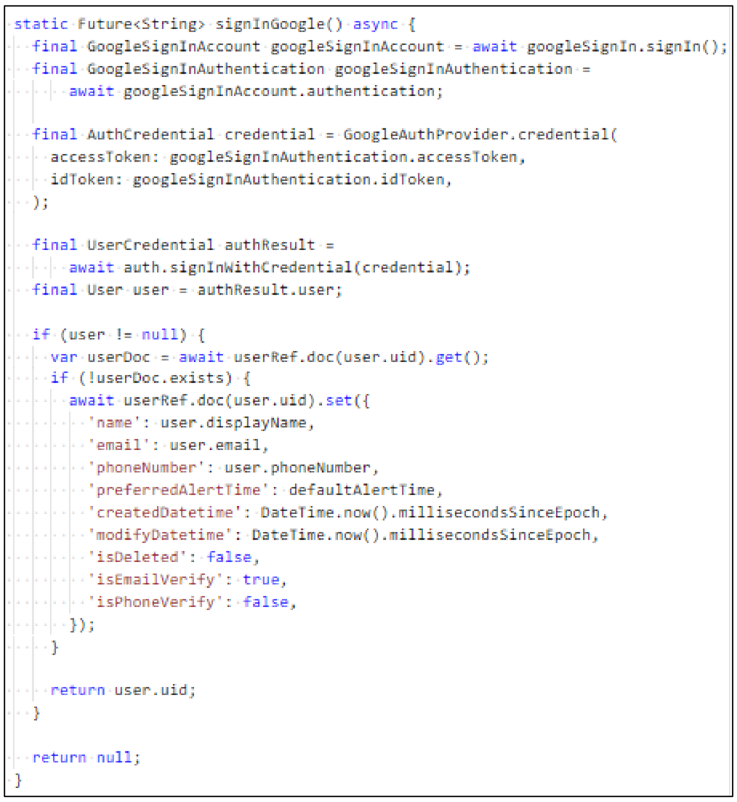
```
static Future<Map<int, String>> signInEmail(
      String email, String password) async {
  try {
    UserCredential userCredential = await auth.signInWithEmailAndPassword(
        email: email, password: password);
    return {successCode: userCredential.user.uid};
  } on FirebaseAuthException catch (e) {
    if (e.code == 'user-not-found') {
      return {userNotFoundCode: 'No user found for that email'};
    } else if (e.code == 'wrong-password') {
      return {wrongPasswordCode: 'Wrong password provided for that user.'};
    }
  }
  return {unknownErrorCode: 'Some error occurred, please try again later.'};
}
```

Figure 6.51: Code Segments of Sign In User with Email and Password Authentication



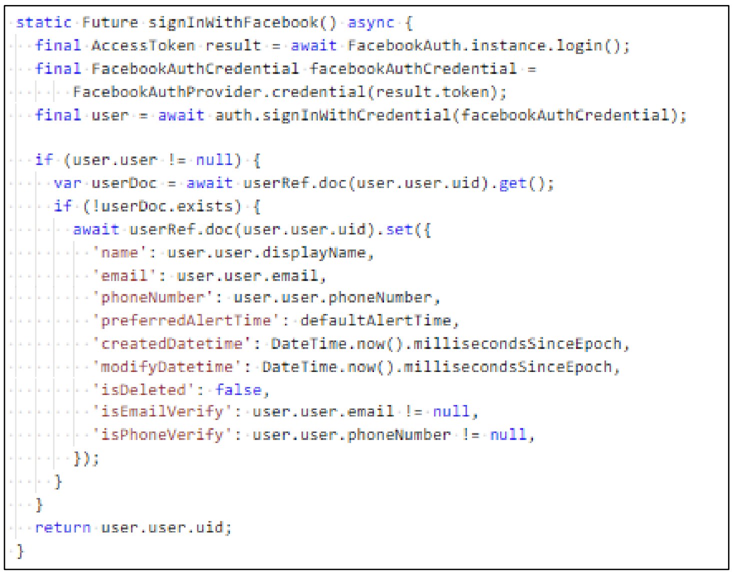
```
static Future<String> signUpEmail(
      String username, String email, String password, String phone) async {
  try {
    UserCredential userCredential = await auth.createUserWithEmailAndPassword(
        email: email, password: password);
    await userRef.doc(userCredential.user.uid).set({
      'name': username,
      'email': email,
      'phoneNumber': UtilFunctions.formatPhoneNumber(phone),
      'preferredAlertTime': defaultAlertTime,
      'createdDatetime': DateTime.now().millisecondsSinceEpoch,
      'modifyDatetime': DateTime.now().millisecondsSinceEpoch,
      'isDeleted': false,
      'isEmailVerify': false,
      'isPhoneVerify': false,
    });

    return userCredential.user.uid;
  } catch (e) {
    print(e.message);
  }
  return null;
}
```

Figure 6.52: Code Segments of Sign-Up User with Email and Password Authentication



```dart
static Future<String> signInGoogle() async {
  final GoogleSignInAccount googleSignInAccount = await googleSignIn.signIn();
  final GoogleSignInAuthentication googleSignInAuthentication =
      await googleSignInAccount.authentication;

  final AuthCredential credential = GoogleAuthProvider.credential(
    accessToken: googleSignInAuthentication.accessToken,
    idToken: googleSignInAuthentication.idToken,
  );

  final UserCredential authResult =
      await auth.signInWithCredential(credential);
  final User user = authResult.user;

  if (user != null) {
    var userDoc = await userRef.doc(user.uid).get();
    if (!userDoc.exists) {
      await userRef.doc(user.uid).set({
        'name': user.displayName,
        'email': user.email,
        'phoneNumber': user.phoneNumber,
        'preferredAlertTime': defaultAlertTime,
        'createdDatetime': DateTime.now().millisecondsSinceEpoch,
        'modifyDatetime': DateTime.now().millisecondsSinceEpoch,
        'isDeleted': false,
        'isEmailVerify': true,
        'isPhoneVerify': false,
      });
    }

    return user.uid;
  }

  return null;
}
```

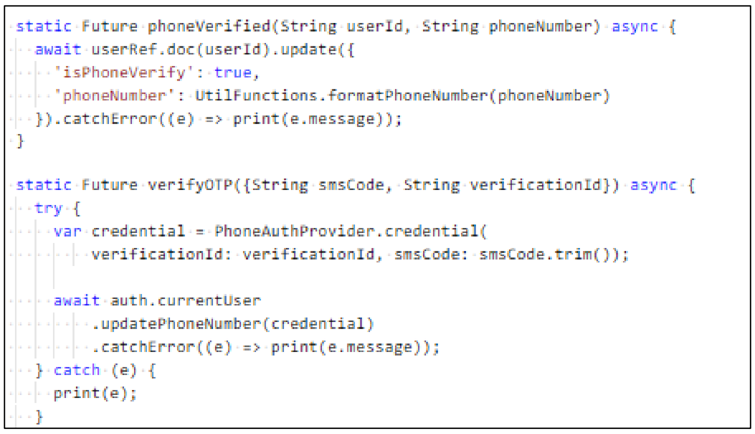
Figure 6.53: Code Segments of Google Authentication



```dart
static Future signInWithFacebook() async {
  final AccessToken result = await FacebookAuth.instance.login();
  final FacebookAuthCredential facebookAuthCredential =
      FacebookAuthProvider.credential(result.token);
  final user = await auth.signInWithCredential(facebookAuthCredential);

  if (user.user != null) {
    var userDoc = await userRef.doc(user.user.uid).get();
    if (!userDoc.exists) {
      await userRef.doc(user.user.uid).set({
        'name': user.user.displayName,
        'email': user.user.email,
        'phoneNumber': user.user.phoneNumber,
        'preferredAlertTime': defaultAlertTime,
        'createdDatetime': DateTime.now().millisecondsSinceEpoch,
        'modifyDatetime': DateTime.now().millisecondsSinceEpoch,
        'isDeleted': false,
        'isEmailVerify': user.user.email != null,
        'isPhoneVerify': user.user.phoneNumber != null,
      });
    }
  }
  return user.user.uid;
}
```

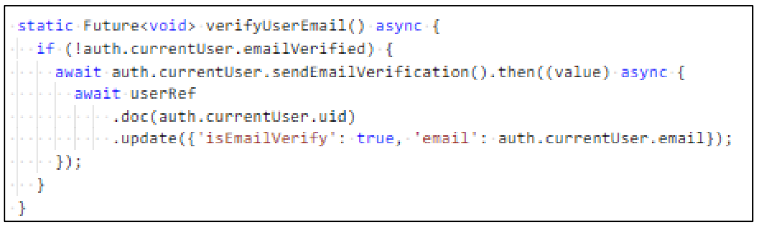
Figure 6.54: Code Segments of Facebook Authentication

Apart from that, some other functions from Firebase Authentication are also used in the mobile application. These functions include verifying phone number, verifying email address, signing out, getting current authenticated user and resetting password. The code segments are shown in the figure below.



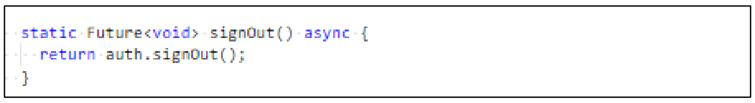
```
static Future phoneVerified(String userId, String phoneNumber) async {
  await userRef.doc(userId).update({
    'isPhoneVerify': true,
    'phoneNumber': UtilFunctions.formatPhoneNumber(phoneNumber)
  }).catchError((e) => print(e.message));
}

static Future verifyOTP({String smsCode, String verificationId}) async {
  try {
    var credential = PhoneAuthProvider.credential(
        verificationId: verificationId, smsCode: smsCode.trim());

    await auth.currentUser
        .updatePhoneNumber(credential)
        .catchError((e) => print(e.message));
  } catch (e) {
    print(e);
  }
```

Figure 6.55: Code Segments of Verify Phone Number Functions



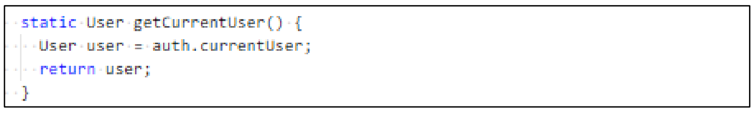
```
static Future<void> verifyUserEmail() async {
  if (!auth.currentUser.emailVerified) {
    await auth.currentUser.sendEmailVerification().then((value) async {
      await userRef
          .doc(auth.currentUser.uid)
          .update({'isEmailVerify': true, 'email': auth.currentUser.email});
    });
  }
}
```

Figure 6.56: Code Segments of Verify Email Address Function



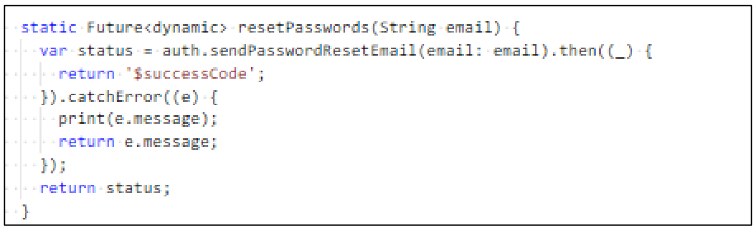
```
static Future<void> signOut() async {
  return auth.signOut();
}
```

Figure 6.57: Code Segments of Sign Out Function



```
static User getCurrentUser() {
  User user = auth.currentUser;
  return user;
}
```

Figure 6.58: Code Segments of Get Current User Function

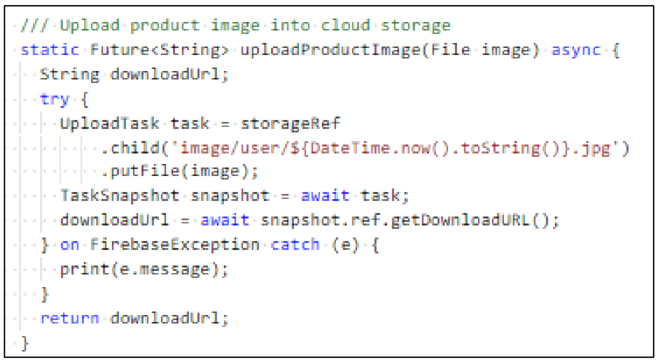


```
static Future<dynamic> resetPasswords(String email) {
  var status = auth.sendPasswordResetEmail(email: email).then((_) {
    return '$successCode';
  }).catchError((e) {
    print(e.message);
    return e.message;
  });
  return status;
}
```

Figure 6.59: Code Segments of Reset Password Function

6.3 Implementation of Firebase Storage

Due to the constraints on Firebase Firestore, the mobile application uses Firebase Storage to store product image. After storing the product image into the Firebase Storage, an image URL that points to that image will be generated. The image URL will then be stored in the Product document to be used to retrieve the product image in the future. The store image function is written in Personal Product Services. The code segments are shown in the figure below.

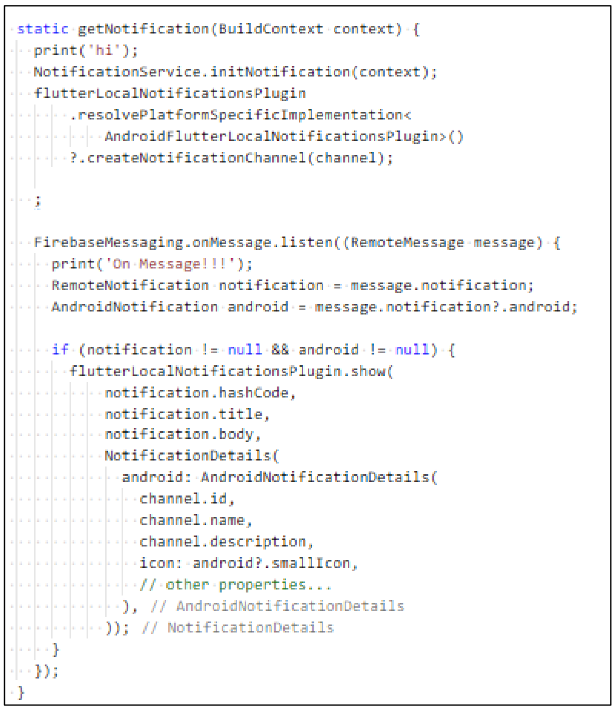


```
/// Upload product image into cloud storage
static Future<String> uploadProductImage(File image) async {
  String downloadUrl;
  try {
    UploadTask task = storageRef
        .child('image/user/${DateTime.now().toString()}.jpg')
        .putFile(image);
    TaskSnapshot snapshot = await task;
    downloadUrl = await snapshot.ref.getDownloadURL();
  } on FirebaseException catch (e) {
    print(e.message);
  }
  return downloadUrl;
}
```

Figure 6.60: Code Segments of Store Image Function

6.4 Implementation of Firebase Cloud Messaging

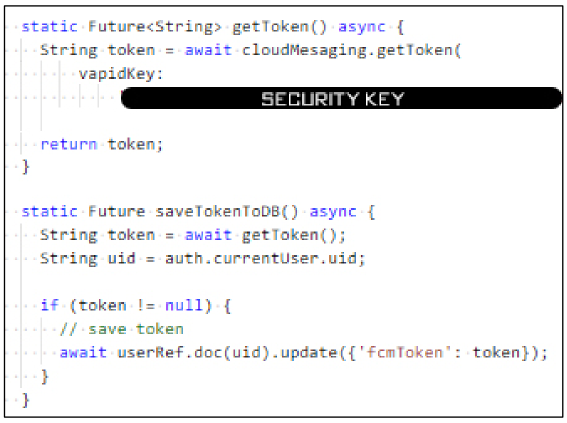
Firebase Cloud Messaging (FCM) allows the system to send notification via the cloud servers to the user's mobile phone. A Flutter library called Flutter Local Notifications is used with the FCM to display notifications to the user. A service called Cloud Messaging Service is used to listen to the Firebase Cloud Messaging API and displays notifications whenever a cloud message is sent via FCM API. The code segments to get notifications are shown in the figure below.



```
static getNotification(BuildContext context) {
  print('hi');
  NotificationService.initNotification(context);
  flutterLocalNotificationsPlugin
      .resolvePlatformSpecificImplementation<
          AndroidFlutterLocalNotificationsPlugin>()
      ?.createNotificationChannel(channel);

  ;

  FirebaseMessaging.onMessage.listen((RemoteMessage message) {
    print('On Message!!!');
    RemoteNotification notification = message.notification;
    AndroidNotification android = message.notification?.android;

    if (notification != null && android != null) {
      flutterLocalNotificationsPlugin.show(
          notification.hashCode,
          notification.title,
          notification.body,
          NotificationDetails(
            android: AndroidNotificationDetails(
              channel.id,
              channel.name,
              channel.description,
              icon: android?.smallIcon,
              // other properties...
            ), // AndroidNotificationDetails
          )); // NotificationDetails
    }
  });
}
```

Figure 6.61: Code Segments of Get Notification Functions.

To send notifications to the correct device, a device token will be generated during the first launch of the mobile application and the device token will be stored in the User Collection in the Firebase Firestore database. The code segments of Generate Device Token function are shown in the figure below.



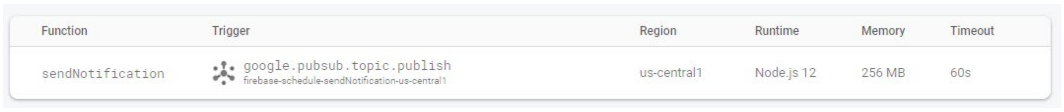
```
static Future<String> getToken() async {
  String token = await cloudMesaging.getToken(
      vapidKey:
        SECURITY KEY

  return token;
}

static Future saveTokenToDB() async {
  String token = await getToken();
  String uid = auth.currentUser.uid;

  if (token != null) {
    // save token
    await userRef.doc(uid).update({'fcmToken': token});
  }
}
```

Figure 6.62: Code Segments of Generating and Storing Device Token Functions

6.5 Implementation of Firebase Cloud Function

Firebase Cloud Function serves functions deployed on it as APIs. Besides, Firebase Cloud Function also serves as an automated backend handling to manage and connect all the backend services.

In the mobile application, a scheduler function written in JavaScript is deployed to the Firebase Cloud Function to retrieve user data and sends alerts to the user based on the alerts added by the user. The scheduler function will read user data every minute and will send a push notification alert, an SMS notification alert and an email notification alert based on what the user has chosen for the alert. The scheduler function will send the push notification alert to the user's device according to the device id stored in the Firebase Firestore through FCM. The scheduler function will send an SMS notification alert to the user according to the user's phone number stored in the Firebase Firestore through Twillio API. The scheduler function will send an email notification alert to the user according to the user's email address stored in the Firebase Firestore through Node Mailer API.



Function	Trigger	Region	Runtime	Memory	Timeout
sendNotification	google.pubsub.topic.publish firebase-schedule-sendNotification-us-central1	us-central1	Node.js 12	256 MB	60s

Figure 6.63: Functions Deployed in Firebase Cloud Function

6.6 Implementation of Deep Learning Models

In the mobile application, a Deep Learning Model is used to capture the expiry dates on the product packaging.

6.6.1 Dataset Generation

A close to reality image dataset is generated to be used to train the Deep Learning Model. The samples of the image dataset are shown in the figure below.

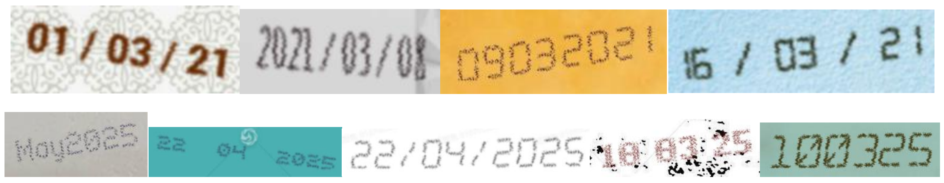
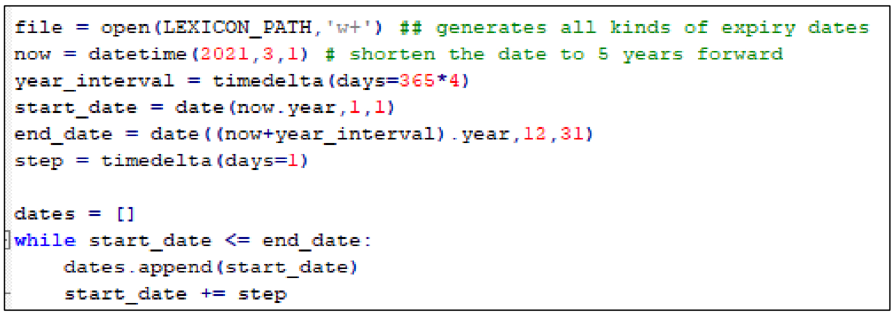


Figure 6.64: A Sample of Synthetic Date Image

A python script is written to generate the dataset automatically with various customization. The python script will generate a lexicon, which is the label of the dataset in a separate text file and a folder of date images with each class in a separate folder.

The script started by generating a list of date starting from the defined start date, to the end date. The code segments of the python script are shown in the figure below.



```
file = open(LEXICON_PATH,'w+') ## generates all kinds of expiry dates
now = datetime(2021,3,1) # shorten the date to 5 years forward
year_interval = timedelta(days=365*4)
start_date = date(now.year,1,1)
end_date = date((now+year_interval).year,12,31)
step = timedelta(days=1)

dates = []
while start_date <= end_date:
    dates.append(start_date)
    start_date += step
```

Figure 6.65: Code Segments of Generate Date List

Then, date strings with different formats are generated for each date in the date list. The date will be stored in a text file to be used by the TRDG library to generate images. A folder will be created according to the class index and the TRDG library will generate the image into the folder. There are a total of 3 classes generated for each date and 1 special class generated for each month. The table below shows the date format generated for each code segments.

Table 6.4: Date Format Generated by Each Code Segment

Code Segments	**Date Format**

`print('{0}/{1}'.format(i,len(str_date)))`
`with open('Lexicon/cmd/lbl.txt','w+') as f:`
`    dateStr = []`
`    dateStr.append(date.strftime("%d%m%y"))`
`    str_date.append(date.strftime("%d%m%y"))`
`    dateStr.append(date.strftime("%d-%m-%y"))`
`    dateStr.append(date.strftime("%d/%m/%y"))`
`    dateStr.append(date.strftime("%d.%m.%y"))`
`    dateStr.append(date.strftime("%d %m %y"))`
`    dateStr.append(date.strftime("%d - %m - %y"))`
`    dateStr.append(date.strftime("%d / %m / %y"))`
`    dateStr.append(date.strftime("%d . %m . %y"))`
`    dateStr.append(date.strftime("%d  %m  %y"))`
`    for lines in list(dateStr):`
`        f.writelines(lines+'\n')`
`os.system('trdg --output_dir "{0}" -i "{1}" -c {2} -f 80 -t 6 -e jpeg -k 10 -rk -bl 1 -rbl -b 3 -na 2 -d 3 -do 2 -tc "#000000","#666666" -m 10 -fi -fd "C:\\Users\\asus-s\\Desktop\\Untitled Folder\\latin" -id "C:\\Users\\asus-s\\Desktop\\Untitled Folder\\Background"'.format(os.path.join(IMG_PATH,str(i)), os.path.abspath('Lexicon/cmd/lbl.txt'), count))`
`i+=1`
	- ddMMyy
	- dd MM yy
	- dd MM yy
	- dd-MM-yy
	- dd - MM - yy
	- dd/MM/yy
	- dd / MM / yy
	- dd.MM.yy
	- dd . MM . yy

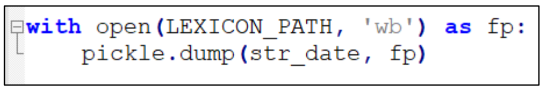
print('{0}/{1}'.format(i,len(str_date)))
with open('Lexicon/cmd/lbl.txt','w+') as f:
    dateStr = []
    dateStr.append(date.strftime("%d%m%Y"))
    str_date.append(date.strftime("%d%m%Y"))
    dateStr.append(date.strftime("%d-%m-%Y"))
    dateStr.append(date.strftime("%d/%m/%Y"))
    dateStr.append(date.strftime("%d.%m.%Y"))
    dateStr.append(date.strftime("%d %m %Y"))
    dateStr.append(date.strftime("%d - %m - %Y"))
    dateStr.append(date.strftime("%d / %m / %Y"))
    dateStr.append(date.strftime("%d . %m . %Y"))
    dateStr.append(date.strftime("%d %m %Y"))
    for lines in list(dateStr):
        f.writelines(lines+'\n')
os.system('trdg --output_dir "{0}" -i "{1}" -c {2} -f 80 -t 6 -e jpeg -k 10 -rk -bl 1 -rbl -b 3 -na 2 -d 3 -do 2 -tc "#000000","#333333" -m 10 -fi -fd "C:\\Users\\asus-s\\Desktop\\Untitled Folder\\latin" -id "C:\\Users\\asus-s\\Desktop\\Untitled Folder\\Background"'.format(os.path.join(IMG_PATH,str(i)), os.path.abspath('Lexicon/cmd/lbl.txt'),count))
i+=1
	- ddMMyyyy
	- dd MM yyyy
	- dd MM yyyy
	- dd-MM-yyyy
	- dd - MM - yyyy
	- dd/MM/yyyy
	- dd / MM / yyyy
	- dd.MM.yyyy
	- dd . MM . yyyy

print('{0}/{1}'.format(i,len(str_date)))
with open('Lexicon/cmd/lbl.txt','w+') as f:
    dateStr = []
    dateStr.append(date.strftime("%Y%m%d"))
    str_date.append(date.strftime("%Y%m%d"))
    dateStr.append(date.strftime("%Y-%m-%d"))
    dateStr.append(date.strftime("%Y/%m/%d"))
    dateStr.append(date.strftime("%Y.%m.%d"))
    dateStr.append(date.strftime("%Y %m %d"))
    dateStr.append(date.strftime("%Y - %m - %d"))
    dateStr.append(date.strftime("%Y / %m / %d"))
    dateStr.append(date.strftime("%Y . %m . %d"))
    dateStr.append(date.strftime("%Y %m %d"))
    for lines in list(dateStr):
        f.writelines(lines+'\n')
os.system('trdg --output_dir "{0}" -i "{1}" -c {2} -f 80 -t 6 -e jpeg -k 10 -rk -bl 1 -rbl -b 3 -na 2 -d 3 -do 2 -tc "#000000","#333333" -m 10 -fi -fd "C:\\Users\\asus-s\\Desktop\\Untitled Folder\\latin" -id "C:\\Users\\asus-s\\Desktop\\Untitled Folder\\Background"'.format(os.path.join(IMG_PATH,str(i)), os.path.abspath('Lexicon/cmd/lbl.txt'),count))
i+=1
	- yyyyMMdd
	- yyyy MM dd
	- yyyy MM dd
	- yyyy-MM-dd
	- yyyy - MM - dd
	- yyyy/MM/dd
	- yyyy / MM / dd
	- yyyy.MM.dd
	- yyyy . MM . dd

```python
if date.day == 1:
    print('{0}/{1}'.format(i,len(str_date)))
    with open('Lexicon/cmd/lbl.txt','w+') as f:
        dateStr = []
        dateStr.append(date.strftime("%b%Y"))
        str_date.append(date.strftime("%b%Y"))
        dateStr.append(date.strftime("%b %Y"))
        for lines in list(set(dateStr)):
            f.writelines(lines+'\n')
    os.system('trdg --output_dir "{0}" -i "{1}" -c {2} -f 80 -t 6 -e jpeg -k 10 -rk -bl 1 -rbl -b
    3 -na 2 -d 3 -do 2 -tc "#000000","#666666" -m 10 -fi -fd "C:\\Users\\asus-s\\Desktop\\Untitled
    Folder\\latin" -id "C:\\Users\\asus-s\\Desktop\\Untitled Folder\\Background"'.format(os.path.
    join(IMG_PATH,str(i)), os.path.abspath('Lexicon/cmd/lbl.txt'),count))
    i+=1
```

- MMyyyy
- MM yyyy

At the end of the script, the date string generated will be saved into a text file.

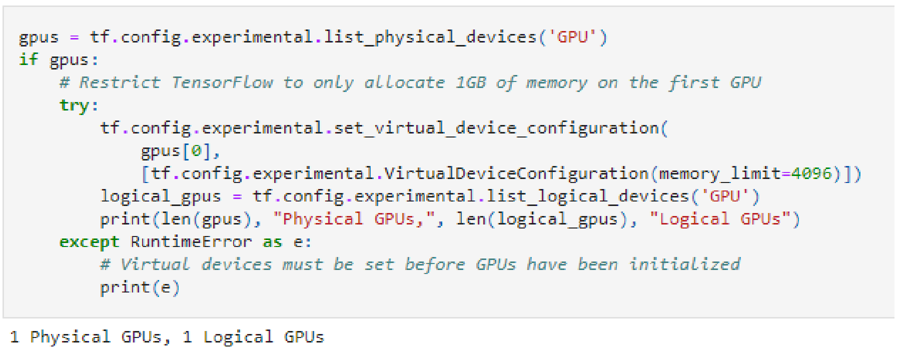


```python
with open(LEXICON_PATH, 'wb') as fp:
    pickle.dump(str_date, fp)
```

Figure 6.66: Code Segments of Saving Date String into Text File

6.6.2 Using GPU

A 4 GB GPU is used to fasten the training process. The code segments of using GPU are shown in the figure below.



```python
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    # Restrict TensorFlow to only allocate 1GB of memory on the first GPU
    try:
        tf.config.experimental.set_virtual_device_configuration(
            gpus[0],
            [tf.config.experimental.VirtualDeviceConfiguration(memory_limit=4096)])
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:
        # Virtual devices must be set before GPUs have been initialized
        print(e)
```

```
1 Physical GPUs, 1 Logical GPUs
```

Figure 6.67: Code Segments of Using GPU During the Training Process

6.6.3 Grid Search Hyperparameter Tunning

A grid search hyperparameter tunning is performed to obtain the best hyperparameter for the model to obtain a better accuracy when training the dataset. A smaller dataset with 15 images per class is used to perform hyperparameter tuning. The same training loop will be run repetitively with different combinations hyperparameter and the validation loss and validation accuracy are used to determine which hyperparameter performs the best.

Due to the constraint of the training time, each combinations hyperparameter will run only 1 epoch. The result is then converted to an excel sheet for reference during the training phase.

The result obtained from the grid search hyperparameter tunning is shown in the table below. Based on the result, training the deep learning model with 64 batch size, 0.0001 learning rate and 0.4 dropout gives a better validation accuracy and a good validation loss.

Table 6.5: Result of the Grid Search Hyperparameter Tunning

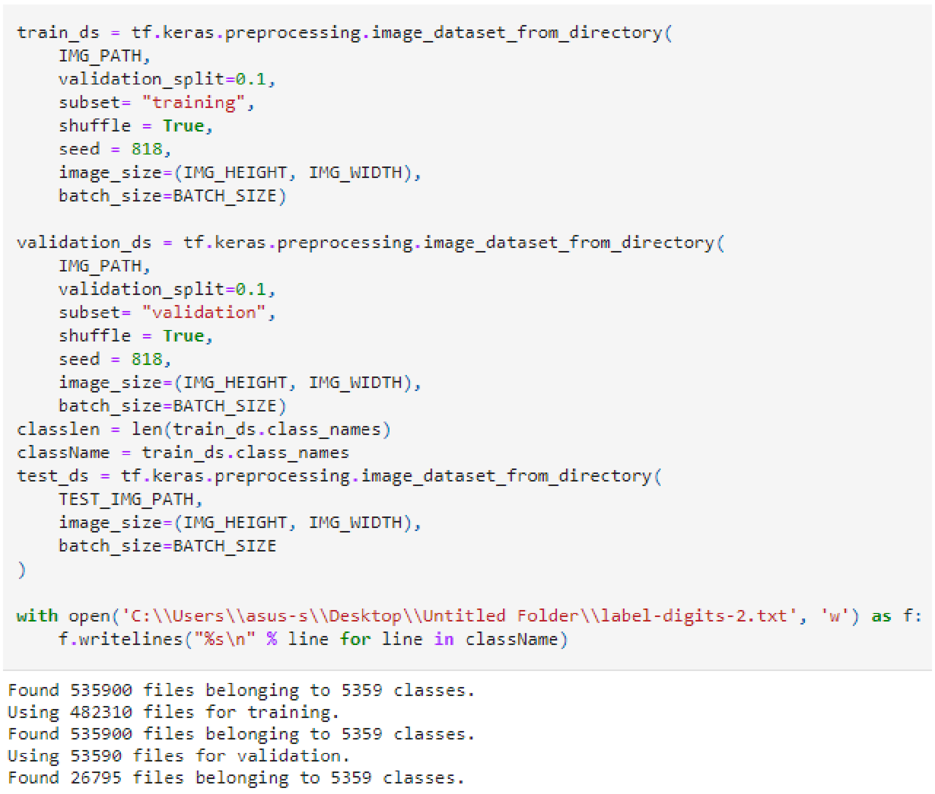
No	Epoch	Batch Size	LR	Dropout	Val Acc	Val Loss
0	1	64	0.01	0.2	0.000000	76.891197
1				0.3	0.000150	59.240292
2				0.4	0.000000	45.036331
3			0.001	0.2	0.002604	11.707423
4				0.3	0.000000	10.809799
5				0.4	0.002604	10.709365
6			0.0001	0.2	0.000000	9.620681
7				0.3	0.000000	9.950025
8				0.4	**0.002604**	**9.582140**
9		128	0.01	0.2	0.000000	78.836098
10				0.3	0.000000	67.211235
11				0.4	0.000000	49.153797
12			0.001	0.2	0.000000	11.416241
13				0.3	0.000000	11.015180
14				0.4	0.000000	10.570828
15			0.0001	0.2	0.000000	9.700305
16				0.3	0.000000	9.664173
17				0.4	0.000000	9.551310

6.6.4 Training

The whole training script is written in Python and is tested and run in Jupyter Notebook. TensorFlow, a machine learning library that is created mainly for deep learning purposes, is also used throughout the training process. Besides, the Grid Search Hyperparameter Tuning is also used to obtain the optimal hyperparameters for the deep learning model.

6.6.4.1 Load Dataset

The dataset is loaded from the directory with a batch size of 64, and each image is loaded with the size of 80 x 215. The dataset has a total of 5359 classes with 105 images in each class. It is split into 70% for the training set and 15% for the validation set and 15% for the testing set. The code segments and outputs are shown in the figure below.



```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    IMG_PATH,
    validation_split=0.1,
    subset= "training",
    shuffle = True,
    seed = 818,
    image_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE)

validation_ds = tf.keras.preprocessing.image_dataset_from_directory(
    IMG_PATH,
    validation_split=0.1,
    subset= "validation",
    shuffle = True,
    seed = 818,
    image_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE)
classlen = len(train_ds.class_names)
className = train_ds.class_names
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    TEST_IMG_PATH,
    image_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE
)

with open('C:\\Users\\asus-s\\Desktop\\Untitled Folder\\label-digits-2.txt', 'w') as f:
    f.writelines("%s\n" % line for line in className)
```

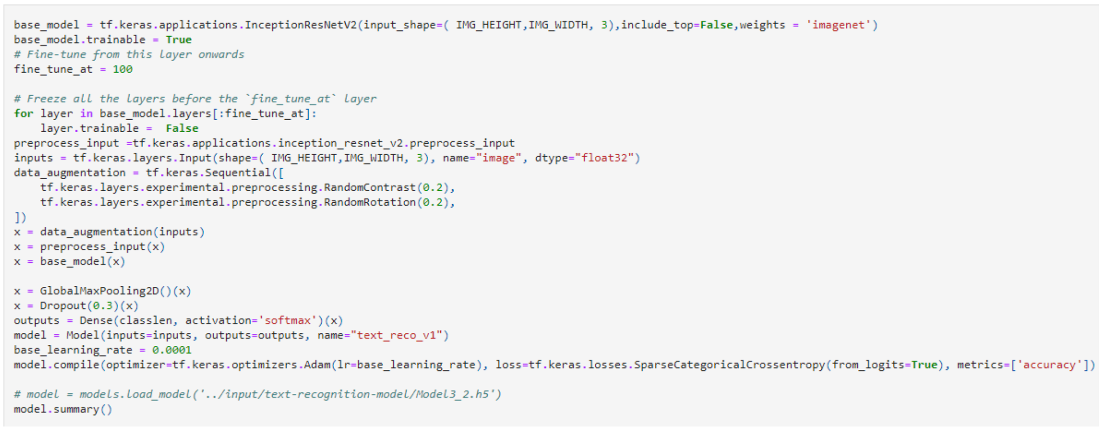
```
Found 535900 files belonging to 5359 classes.
Using 482310 files for training.
Found 535900 files belonging to 5359 classes.
Using 53590 files for validation.
Found 26795 files belonging to 5359 classes.
```

Figure 6.68: Code Segment and Output of Loading Dataset

6.6.4.2 Load Model

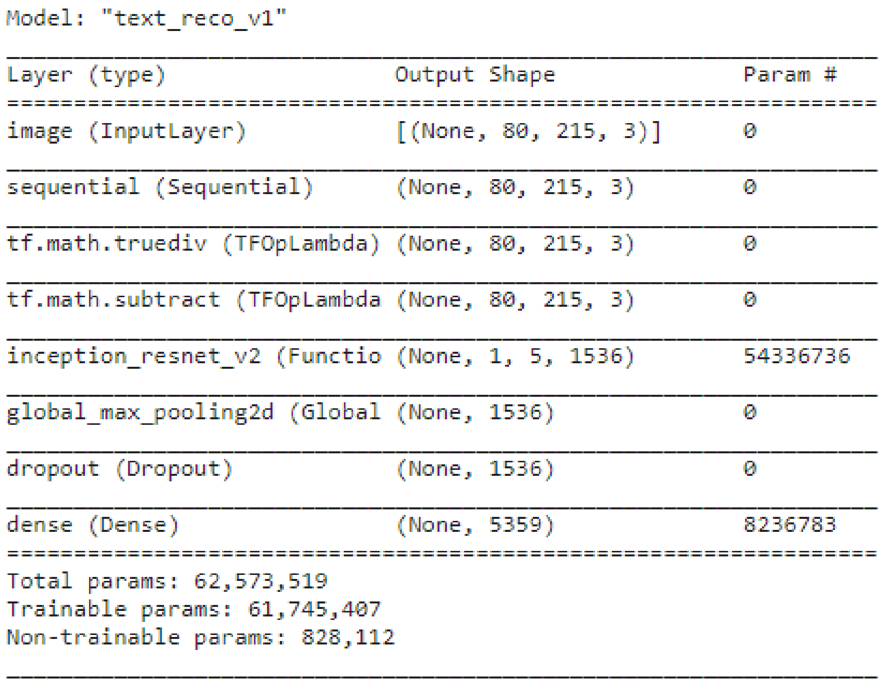
Inception Resnet V2, a CNN for image recognition, was used as the classifier. The weights trained on ImageNet, a large-scale image dataset for object recognition, are loaded into Inception Resnet V2 from the Tensorflow Keras application. The first 100 layers of Inception Resnet V2 are frozen and cannot be trained. This is because once all layers are unfrozen, the training period will become incredibly long and computationally costly. However, if too many layers are frozen, the model will fail to be trained because the dataset differs greatly from the ImageNet dataset.

A data augmentation layer is added below the input layer to generate more training instances during the training loop to prevent overfitting. A fully connected layer is then placed after the Inception Resnet V2 model, and it is connected to the output layer.



```
base_model = tf.keras.applications.InceptionResNetV2(input_shape=( IMG_HEIGHT,IMG_WIDTH, 3),include_top=False,weights = 'imagenet')
base_model.trainable = True
# Fine-tune from this layer onwards
fine_tune_at = 100

# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable =  False
preprocess_input =tf.keras.applications.inception_resnet_v2.preprocess_input
inputs = tf.keras.layers.Input(shape=( IMG_HEIGHT,IMG_WIDTH, 3), name="image", dtype="float32")
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.experimental.preprocessing.RandomContrast(0.2),
    tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),
])
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x)

x = GlobalMaxPooling2D()(x)
x = Dropout(0.3)(x)
outputs = Dense(classlen, activation='softmax')(x)
model = Model(inputs=inputs, outputs=outputs, name="text_reco_v1")
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(lr=base_learning_rate), loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

# model = models.load_model('../input/text-recognition-model/Model3_2.h5')
model.summary()
```

Figure 6.69: Code Segments of Load Model

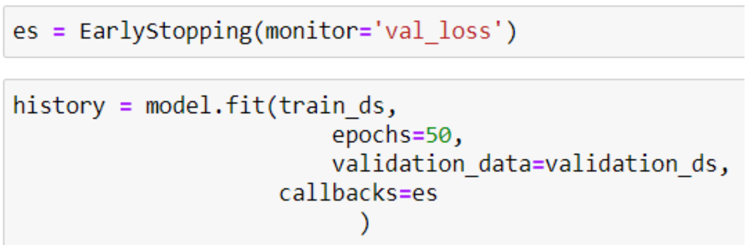


```
Model: "text_reco_v1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
image (InputLayer)           [(None, 80, 215, 3)]      0         
_________________________________________________________________
sequential (Sequential)      (None, 80, 215, 3)        0         
_________________________________________________________________
tf.math.truediv (TFOpLambda) (None, 80, 215, 3)        0         
_________________________________________________________________
tf.math.subtract (TFOpLambda (None, 80, 215, 3)        0         
_________________________________________________________________
inception_resnet_v2 (Functio (None, 1, 5, 1536)        54336736  
_________________________________________________________________
global_max_pooling2d (Global (None, 1536)              0         
_________________________________________________________________
dropout (Dropout)            (None, 1536)              0         
_________________________________________________________________
dense (Dense)                (None, 5359)              8236783   
=================================================================
Total params: 62,573,519
Trainable params: 61,745,407
Non-trainable params: 828,112
_________________________________________________________________
```

Figure 6.70: Output of Load Model

6.6.4.3 Training with Early Stopping

To avoid overfitting, the model is trained with early stopping. The training loop is terminated as the loss begins to rise, which is one of the overfitting symptoms.



```
es = EarlyStopping(monitor='val_loss')
```

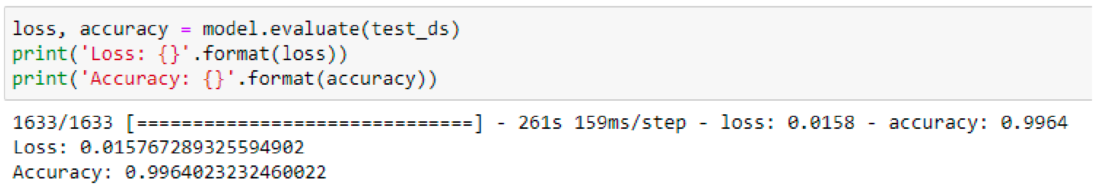
```
history = model.fit(train_ds,
                        epochs=50,
                        validation_data=validation_ds,
                    callbacks=es
                          )
```

Figure 6.71: Code Segments of Training Model

6.6.4.4 Evaluation

1. Evaluate the Testing Set

   The model is evaluated using the testing set and had achieved a good result, with a loss of 0.0158 and an accuracy of 0.9964. The code segments and output for the evaluation are shown in the figure below.



```
loss, accuracy = model.evaluate(test_ds)
print('Loss: {}'.format(loss))
print('Accuracy: {}'.format(accuracy))
```

```
1633/1633 [==============================] - 261s 159ms/step - loss: 0.0158 - accuracy: 0.9964
Loss: 0.015767289325594902
Accuracy: 0.9964023232460022
```

Figure 6.72: Code Segments and Output of Evaluation on Testing Dataset

2. Evaluate on Learning Curves

   The model is evaluated on the learning curve to see how it fits through epochs. The validation accuracy and validation loss outperformed the training accuracy and training loss on the learning curve. This may be due to differences in the scale of the training and validation examples. The model is thought to work very well after hyperparameter tuning since it also performs well when tested on unseen data. The figure below shows the learning curve for the model.

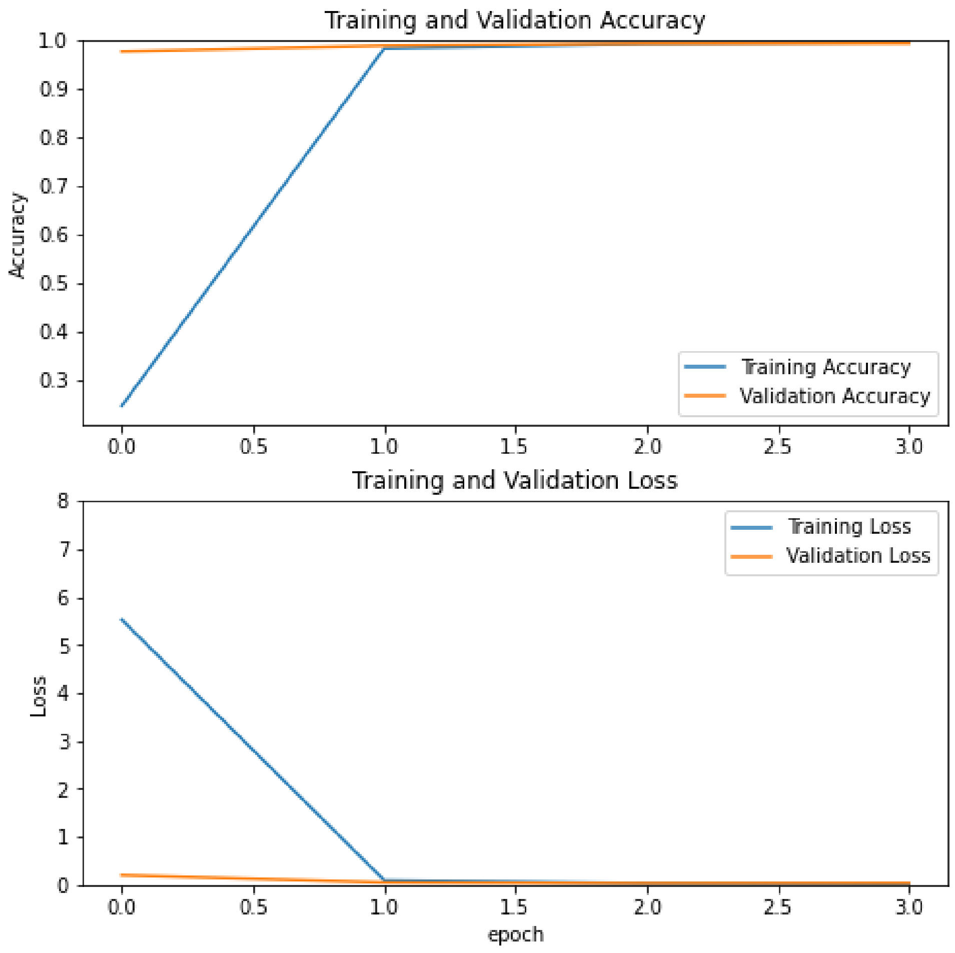


Figure 6.73: Learning Curve of Inception Resnet V2 on Date Image Dataset

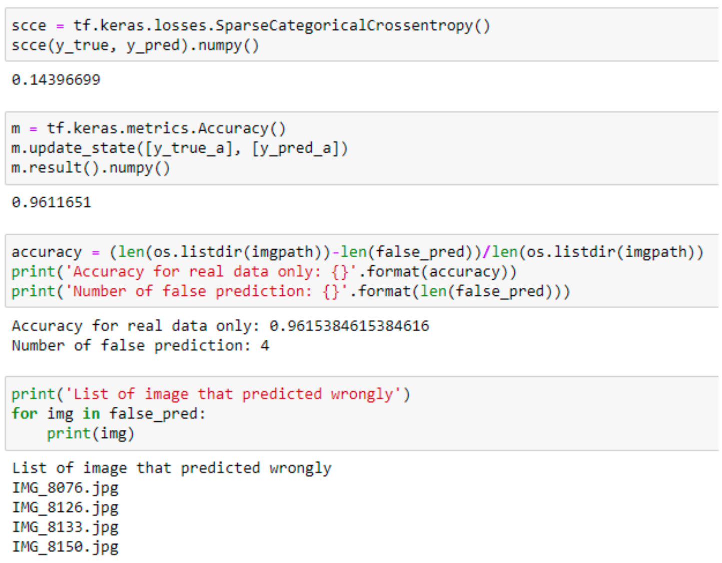
3. Evaluation on Real Image Data

In contrast to synthetic image data, actual image data can be blurry, noisy, or have an insufficient light source, which will impact the model's prediction. Furthermore, the date in actual image data can have different gaps between each character, which will impact the result.

The model is tested with 102 actual images captured from a real smartphone on a real product. Some of the images are purposely captured on a very noisy and reflective background to see how far the model is able to predict on such noisy data. Plus, some of the dates that are out of the predictable date range are purposely taken to see how the model will react to images that do not lie in any of the class. The result for false prediction of each actual image is analysed to find out the actual reason. Each finding is tabulated in Table 6.6.

Based on the finding, 0 and 1, 3 and 8 are similar to each other for the model. Besides, the model will fail on prediction when the background is noisy, reflective and is uneven with different lights and shadow.

Then, the really noisy data and out of range data are replaced with a normal image data that should be predictable by the model. There are a total number of 102 real world images are tested. From the result obtained, the model is able to predict 96.12% of the actual images correctly, which is equivalent to 98 out of 102 images. The result is shown in the code segments below.



```
scce = tf.keras.losses.SparseCategoricalCrossentropy()
scce(y_true, y_pred).numpy()
```

```
0.14396699
```

```
m = tf.keras.metrics.Accuracy()
m.update_state([y_true_a], [y_pred_a])
m.result().numpy()
```

```
0.9611651
```

```
accuracy = (len(os.listdir(imgpath))-len(false_pred))/len(os.listdir(imgpath))
print('Accuracy for real data only: {}'.format(accuracy))
print('Number of false prediction: {}'.format(len(false_pred)))
```

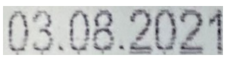
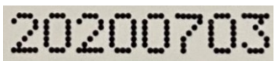
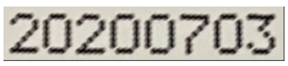
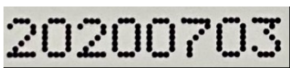
```
Accuracy for real data only: 0.9615384615384616
Number of false prediction: 4
```

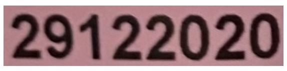
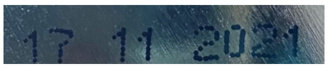
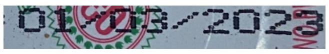
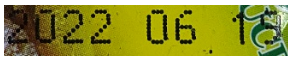
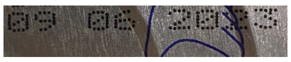
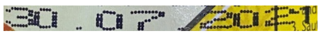
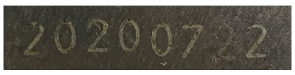
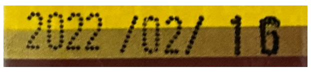
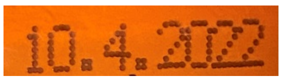
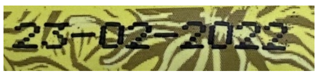
```
print('List of image that predicted wrongly')
for img in false_pred:
    print(img)
```

```
List of image that predicted wrongly
IMG_8076.jpg
IMG_8126.jpg
IMG_8133.jpg
IMG_8150.jpg
```

Figure 6.74: Code Segments of Evaluating on Real Images

Table 6.6: Finding on False Prediction for Each Actual Images

Image	**Predictions**	**Real Date**	**Possible Reason**
IMG_8076.jpg 	03032021	03082021	8 is similar to 3 for the model.
IMG_8078.jpg 	20210703	20200703	The image is out of the range for prediction. However, the model still can get the closest date possible.
IMG_8079.jpg 	20210703	20200703	Same as Image_8079.jpg
IMG_8080.jpg 	20210703	20200703	Same as Image_8079.jpg

IMG_8114.jpg 	29122024	29122020	The image is out of the range for prediction. However, the model still can get the closest date possible.
IMG_8122.jpg 	17112022	17112021	The reflective background causes the model failed to identify the darker part
IMG_8124.jpg 	04082022	01032023	The noisy background makes the model fails to predict the date. The model is able to recognize the characters that lie on a clear background
IMG_8126.jpg 	20220609	20220619	Caused by the noisy and reflective background that makes the 1 looks like 0
IMG_8127.jpg 	09062022	09062023	The model takes the dark background as part of the digits. Caused by uneven, reflective and noisy background
IMG_8128-1.jpg 	31072021	30072021	0 is similar to 1 for the model
IMG_8129.jpg 	20220220	20200722	The image is out of the range for prediction. However, the model failed to predict most of the character due to the reflective, noisy and uneven background.
IMG_8133.jpg 	20220206	20220216	0 is similar to 1 for the model
IMG_8150.jpg 	10112022	10042022	The model cannot recognize the date
IMG_8160.jpg 	25022022	23022022	Noisy background. 3 is similar to 5 for the model

4. Compiling Results Obtained

The model accuracy and loss for both synthetic test set and real images are calculated and tabulated. The out-of-range dates in the real-images are replaced with a valid date that is within the prediction range of the model to get a better understanding on the actual performance of the model.

Table 6.7: Overall Results of Inception ResNet V2 in Both Synthetic Test Set and Real-Images Dataset

Dataset	Accuracy	Loss
Synthetic Test Set	0.9964	0.0157
Real-Images Dataset	0.9612	0.1440

6.6.4.5 Convert Model into TFLite

The model is then converted into a TensorFlow Lite (tflite) file so that it can be deployed to the mobile application with a better performance. Then, the tflite model is then tested with the same test dataset to ensure it performs the same as the original model.

After that, the model is then deployed to the mobile application and a Flutter library package called tflite is used to access the TensorFlow Lite API served by the model deployed to the mobile application. The package will load the model, predict the image and retrieve the result by using Dart only.

CHAPTER 7

SYSTEM TESTING

7.1 Introduction

Various tests are conducted to ensure that the mobile application developed is within the scope and thus all requirements are met. The tests to be conduct includes unit test, integration test, cloud functional test, usability test and user acceptance testing.

7.2 Unit Test

Unit testing is performed to test all units or component separately. In this mobile application, all services of each module were tested. The overall test cases for the unit test are listed in the table below.

Table 7.1: Listing of Unit Test Cases

No	Test Case ID	Test Case Title	Status
1	User Auth Service-01	Unit Test Case for Sign Up With Email	Pass
2	User Auth Service-02	Unit Test Case for Sign In With Email	Pass
3	User Auth Service-03	Unit Test Case for Verify User Email	Pass
4	User Auth Service-04	Unit Test Case for Sign Out current user	Pass
5	User Auth Service-05	Unit Test Case for Sign In with Google Account	Pass
6	User Auth Service-06	Unit Test Case for Send Verification SMS	Pass
7	User Auth Service-07	Unit Test Case for Verify OTP	Pass
8	User Auth Service-08	Unit Test Case for Sign In with Facebook Account	Pass
9	User Auth Service-09	Unit Test Case for Reset Password	Pass
10	User Service-01	Unit Test Case for Getting Current User's details	Pass
11	User Service-02	Unit Test Case for Change User's Preferred Alert Time	Pass
12	Product Service-01	Unit Test Case for Getting All Products	Pass

13	Product Service-02	Unit Test Case for Getting Products by Category	Pass
14	Product Service-03	Unit Test Case for Getting Product by Id	Pass
15	Product Service-04	Unit Test Case for Getting Product by Barcode	Pass
16	Product Service-05	Unit Test Case for Getting All Categories	Pass
17	Product Service-06	Unit Test Case for Getting Recent Search Lists	Pass
18	Product Service-07	Unit Test Case for Add Recent Search History	Pass
19	Product Service-08	Unit Test Case for Add New Product	Pass
20	Product Service-09	Unit Test Case for Edit Product	Pass
21	Product Service-10	Unit Test Case for Delete Product	Pass
22	Product Service-11	Unit Test Case for Add New Category	Pass
23	Product Service-12	Unit Test Case for Edit Category	Pass
24	Product Service-13	Unit Test Case for Delete Category	Pass
25	Product Service-14	Unit Test Case for Scan Expiry Date	Pass
26	Product Service-15	Unit Test Case for Add Product Barcode	Pass
27	Product Service-16	Unit Test Case for Add Product Name Using OCR	Pass
28	Product Service-17	Unit Test Case for Upload Product Image	Pass
29	Product Service-18	Unit Test Case for Add Alert Reminder	Pass
30	Product Service-19	Unit Test Case for Prepare Product List	Pass
31	Shopping List Service-01	Unit Test Case for Getting All Shopping List	Pass
32	Shopping List Service-02	Unit Test Case for Add Shopping List	Pass
33	Shopping List Service-03	Unit Test Case for Edit Shopping List	Pass
34	Shopping List Service-04	Unit Test Case for Delete Shopping List	Pass
35	Shopping List Service-05	Unit Test Case for Set Shopping List Item Status	Pass

36	Shopping List Service-06	Unit Test Case for Prepare Shopping List	Pass

7.2.1 Test Results

Table 7.2: Unit Test Case for Sign Up With Email

Test Case ID	User Auth Service-01			**Test Status**	Pass
Test Title	Unit Test Case for Sign Up With Email				
Module	User				
Pre-conditions	-				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Enter valid information	1. Enter username 2. Enter email address 3. Enter password 4. Enter confirm password 5. Enter phone number 6. Click on "Next" button	Valid username, email address, password, confirm password and phone number	The user can sign up successfully with a successful alert message and is redirected to the log in page.	User sign up successfully with a successful alert message and is redirected to the login page	Pass
Enter valid information but with an empty username	1. Enter email address 2. Enter password 3. Enter confirm password 4. Enter phone number 5. Click on "Next" button	Valid email address, password, confirm password and phone number	A message "Username cannot be empty" is shown.	Error message prompted	Pass
Enter valid information but with an empty email address	1. Enter username 2. Enter password 3. Enter confirm password 4. Enter phone number 5. Click on "Next" button	Valid username, password, confirm password and phone number	A message "Email Address cannot be empty" is shown.	Error message prompted	Pass
Enter valid information but with an invalid email address	1. Enter username 2. Enter invalid email address 3. Enter password 4. Enter confirm password 5. Enter phone number 6. Click on "Next" button	Valid username, password, confirm password and phone number Invalid email address	A message "Email Address is not valid. Example: abc@example.com" is shown.	Error message prompted	Pass
Enter valid information but with an	1. Enter username 2. Enter email address 3. Enter confirm password	Valid username, email	A message "Password cannot be empty" is shown.	Error message prompted	Pass

empty password	4. Enter phone number 5. Click on “Next” button	address, confirm password, phone number			
Enter valid information but with an invalid password	1. Enter username 2. Enter email address 3. Enter invalid password 4. Enter confirm password 5. Enter phone number 6. Click on “Next” button	Valid username, email address, confirm password, phone number Invalid password	A message “Password must between 8 to 30 characters and contain one lowercase and uppercase alphabet and at least one digit” is shown.	Error message prompted	Pass
Enter valid information but with an empty confirm password	1. Enter username 2. Enter email address 3. Enter password 4. Enter phone number 5. Click on “Next” button	Valid username, email address, password, phone number	A message “Confirm Password cannot be empty” is shown.	Error message prompted	Pass
Enter valid information but with a different password in the confirm password field	1. Enter username 2. Enter email address 3. Enter password 4. Enter invalid confirm password 5. Enter phone number 6. Click on “Next” button	Valid username, email address, password, phone number Different password in confirm password field	A message “Password and Confirm Password must be the same” is shown.	Error message prompted	Pass
Enter valid information but with an empty phone number	1. Enter username 2. Enter email address 3. Enter password 4. Enter confirm password 5. Click on “Next” button	Valid username, email address, password, confirm password	A message “Phone Number cannot be empty” is shown.	Error message prompted	Pass
Enter valid information but with an invalid phone number	1. Enter username 2. Enter email address 3. Enter password 4. Enter confirm password 5. Enter an invalid phone number 6. Click on “Next” button	Valid username, email address, password, confirm password	A message “Phone Number is not valid. Example: 60123456789” is shown.	Error message prompted	Pass

		Invalid phone number			
Submit the form with empty fields	1. Click on "Next" button	-	All error messages are shown.	Error messages are prompted	Pass

Table 7.3: Unit Test Case for Sign In With Email

Test Case ID	User Auth Service-02			**Test Status**	Pass
Test Title	Unit Test Case for Sign In With Email				
Module	User				
Pre-conditions	The user had registered a valid account through the application				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Enter valid email and password	1. Enter email address 2. Enter password 3. Click on "Sign In" button	Valid email address and password	The user is redirected to the home page.	Sign in successfully and is redirected to the home page	Pass
Enter empty email and valid password	1. Enter password 2. Click on "Sign In" button	Valid password	A message "Email Address cannot be empty" is shown.	Error message prompted	Pass
Enter invalid email and valid password	1. Enter email address 2. Enter password 3. Click on "Sign In" button	Valid password Invalid email address	A message "No user found for that email" is shown.	Error message prompted	Pass
Enter valid email and empty password	1. Enter email address 2. Click on "Sign In" button	Valid email address	A message "Password cannot be empty" is shown	Error message prompted	Pass
Enter valid email and invalid password	1. Enter email address 2. Enter password 3. Click on "Sign In" button	Valid email address Invalid password	A message "Wrong password provided for that user" is shown.	Error message prompted	Pass
Enter invalid email and password	1. Enter email address 2. Enter password 3. Click on "Sign In" button	Invalid email address and password	A message "No user found for that email" is shown.	Error message prompted	Pass

Table 7.4: Unit Test Case for Verify User Email

Test Case ID	User Auth Service-03			**Test Status**	Pass
Test Title	Unit Test Case for Verify User Email				
Module	User				
Pre-conditions	The user has a valid account but the email has not verified.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
The user sign in to the application	1. Sign in to the application with an unverified email address.	Email address, password	An instruction page on how to verify the email address is shown and the verification email is sent.	The instruction email is shown and the verification email is received.	Pass
The user verifies email through the verification email	1. Click on the link in the verification email.	-	The user is redirected to the home page	Verifies successfully and is redirected to the home page	Pass

Table 7.5: Unit Test Case for Sign Out current user

Test Case ID	User Auth Service-04			**Test Status**	Pass
Test Title	Unit Test Case for Sign Out current user				
Module	User				
Pre-conditions	The user is signed in.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
The user signs out from the account	1. Navigate to the setting page through drawer navigation. 2. Click on “Logout from Account” button	-	The user is signed out and is redirected to the sign-in page	Signed out successfully and is redirected to the sign-in page.	Pass

Table 7.6: Unit Test Case for Sign In with Google Account

Test Case ID	User Auth Service-05	**Test Status**	Pass
Test Title	Unit Test Case for Sign In with Google Account		

Module	User				
Pre-conditions	The user has a valid Google Account.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
The user signs in with Google Account	1. Click on the Google icon button. 2. Log in to a valid Google account.	-	The user is signed in successfully and is redirected to the home page.	Signed in successfully and is redirected to the home page.	Pass

Table 7.7: Unit Test Case for Send Verification SMS

Test Case ID	User Auth Service-06			**Test Status**	Pass
Test Title	Unit Test Case for Send Verification SMS				
Module	User				
Pre-conditions	The user has a valid account but the phone number has not verified.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
A user with a phone number recorded in the database	1. Click on “Send Verification Code” button.	User’s phone number	An SMS with an OTP password is sent to the user’s phone number. The resend button will count from 60 to 0 and the button will be changed to the “Next” button.	An SMS with an OTP password is received, the resend button starts counting and the button changed to “Next” button.	Pass
Manually enters a valid phone number	1. Enter phone number 2. Click on “Send Verification Code” button.	Valid phone number	An SMS with an OTP password is sent to the user’s phone number. The resend button will count from 60 to 0 and the button will be changed to “Next” button.	An SMS with an OTP password is received, the resend button starts counting and the button changed to “Next” button.	Pass

Manually enters an invalid phone number	1. Enter phone number 2. Click on "Send Verification Code" button.	Invalid phone number	No SMS will be sent to the user's phone number. The resend button will not be counting down and the button remained as "Send Verification Code".	No SMS received	Pass
The user resends the code on a valid phone number	1. Click on "Resend Code" after 60 seconds of the previous attempt	Valid phone number	An SMS with an OTP password is sent to the user's phone number. The resend button will count from 60 to 0.	An SMS with an OTP password is received, the resend button starts counting	Pass
The user resends the code on an invalid phone number	1. Click on "Resend Code" after 60 seconds of the previous attempt	Invalid phone number	No SMS is sent and the "Resend Code" button will not count down again.	No SMS is received	Pass

Table 7.8: Unit Test Case for Verify OTP

Test Case ID	User Auth Service-07			**Test Status**	Pass
Test Title	Unit Test Case for Verify OTP				
Module	User				
Pre-conditions	The system already sends a verification SMS to the user.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
The OTP code entered is valid	1. Enter OTP code received 2. Click "Next" button	OTP code	Redirect to home page.	Redirect to home page	Pass
The OTP code is invalid	1. Enter OTP code received 2. Click "Next" button	Invalid OTP code	A message "The verification code is invalid. Please enter a valid 6-digits validation code." is prompt.	Error message prompted	Pass

Table 7.9: Unit Test Case for Sign In with Facebook Account

Test Case ID	User Auth Service-08			Test Status	Pass
Test Title	Unit Test Case for Sign In with Facebook Account				
Module	User				
Pre-conditions	The user has a valid Facebook account.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
The user signs in with Facebook Account	1. Click on the Facebook icon button. 2. Log in to a valid Facebook account.	-	The user is signed in successfully and is redirected to the home page.	Signed in successfully and redirected to the home page.	Pass

Table 7.10: Unit Test Case for Reset Password

Test Case ID	User Auth Service-09			Test Status	Pass
Test Title	Unit Test Case for Reset Password				
Module	User				
Pre-conditions	The user has a valid account.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
The user request to reset the password with a valid email address	1. Click on the “Forgot Password” button. 2. Enter email address 3. Click on the “Send Reset Password Link”	Email Address	The user received a reset password email.	Reset password email received	Pass
The user request to rest the password with an invalid email address	1. Click on the “Forgot Password” button. 2. Enter email address 3. Click on the “Send Reset Password Link”	Invalid email address	A message “Email Address not valid” will be prompt.	Error message prompted	Pass

Table 7.11: Unit Test Case for Getting Current User's details

Test Case ID	User Service-01	Test Status	Pass
Test Title	Unit Test Case for Getting Current User's details		
Module	User		

Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Getting user's detail with user-id	1. Open the application	User id	The user's detail retrieved.	User detail retrieved	Pass

Table 7.12: Unit Test Case for Change User's Preferred Alert Time

Test Case ID	User Service-02			**Test Status**	Pass
Test Title	Unit Test Case for Change User's Preferred Alert Time				
Module	User				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Change user's preferred alert time	1. Open setting page. 2. Click on "Preferred Alert Time" 3. Select a preferred time with the time picker	Time	The user's preferred time is change.	User preferred alert time changed	Pass

Table 7.13: Unit Test Case for Getting All Products

Test Case ID	Product Service-01			**Test Status**	Pass
Test Title	Unit Test Case for Getting All Products				
Module	Product				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Display all products	1. Open the application	-	All user's products are loaded	User's products are loaded	Pass
No products are added	1. Open the application	-	Display an empty list	Empty list displayed	Pass

Table 7.14: Unit Test Case for Getting Products by Category

Test Case ID	Product Service-02	**Test Status**	Pass
Test Title	Unit Test Case for Getting Products by Category		

Module	Product				
Pre-conditions	The user has signed in to the application. The user has added at least a product to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Display all products depend on the current logged in user and category selected	1. Select a category	-	All user's products under the category are loaded	User's products under the category are loaded	Pass

Table 7.15: Unit Test Case for Getting Product by Id

Test Case ID	Product Service-03			**Test Status**	Pass
Test Title	Unit Test Case for Getting Product by Id				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has added at least a product to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Display product details depend on the current logged in user and product selected	1. Select a product displayed in the product list	-	The correct product details are loaded	Selected product details are loaded	Pass

Table 7.16: Unit Test Case for Getting Product by Barcode

Test Case ID	Product Service-04			**Test Status**	Pass
Test Title	Unit Test Case for Getting Product by Barcode				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has added at least a product to the application. The barcode is added to the product.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Display product	1. Click on "Search By Barcode" button	-	The correct product details are loaded	The correct product	Pass

details by scanning a valid barcode	2. Scan barcode on a product			details are loaded	
Display product details by an invalid barcode	1. Click on "Search By Barcode" button 2. Scan barcode that does not exist in any of the products	-	A message "Barcode Not Found" is prompt.	Error message prompted	Pass

Table 7.17: Unit Test Case for Getting All Categories

Test Case ID	Product Service-05			**Test Status**	Pass
Test Title	Unit Test Case for Getting All Categories				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has added at least a product to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Display all category added by the user	1. Click on "Category" button in the drawer navigation.	-	All categories added by the user are shown	Categories shown	Pass

Table 7.18: Unit Test Case for Getting Recent Search Lists

Test Case ID	Product Service-06			**Test Status**	Pass
Test Title	Unit Test Case for Getting Recent Search Lists				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has added at least a product to the application. The user has searched a product before.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Display search history	1. Click on the Seach icon button	-	The top 3 latest search history will be shown.	Search history shown	Pass

Table 7.19: Unit Test Case for Add Recent Search History

Test Case ID	Product Service-07	**Test Status**	Pass
Test Title	Unit Test Case for Add Recent Search History		
Module	Product		
Pre-conditions	The user has signed in to the application. The user has added at least a product to the application.		

Test Case Summary	Test Steps	Test Data	Expected Result	Actual Result	Status
Display search history	1. Click on the Seach icon button 2. Type a product name 3. Tap on the search suggestion		The selected product will be displayed in the recent search list	The selected product is shown in the recent search list	Pass

Table 7.20: Unit Test Case for Add New Product

Test Case ID	Product Service-08			**Test Status**	Pass
Test Title	Unit Test Case for Add New Product				
Module	Product				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Enter all required information	1. Click on the Add icon button 2. Enter product name 3. Select product category 4. Enter the product expiry date 5. Click on the Save icon button	Product name, Product category, product expiry date	The product is added and is displayed in the product list.	Product added	Pass
Enter empty fields	1. Click on the Add icon button 2. Click on the Save icon button	-	All error messages for required fill is shown.	Error messages are shown.	Pass

Table 7.21: Unit Test Case for Edit Product

Test Case ID	Product Service-09			**Test Status**	Pass
Test Title	Unit Test Case for Edit Product				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has added a product.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Edit product with all required fields	1. Click on the product in the product list. 2. Click on the Edit icon button 3. Edit the product name field 4. Edit the product category 5. Edit the product expiry date 6. Click on the Save icon button	Product name, product category, product expiry date	The product details updated.	The product details updated	Pass

Edit product with empty fields	1. Click on the product in the product list 2. Click on the Edit icon button 3. Erase the product name field	-	A message "Product name cannot be empty" is showed	Error message showed	Pass

Table 7.22: Unit Test Case for Delete Product

Test Case ID	Product Service-10			**Test Status**	Pass
Test Title	Unit Test Case for Delete Product				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has added a product.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Delete a product	1. Click on the product in the product list 2. Click on "Delete Product" button	-	The product is deleted	Product deleted	Pass

Table 7.23: Unit Test Case for Add New Category

Test Case ID	Product Service-11			**Test Status**	Pass
Test Title	Unit Test Case for Add New Category				
Module	Product				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Enter a valid category name	1. Click on the Add icon button. 2. Enter a category name 3. Click on "Add" button	Category name	The category is added	Category added	Pass
Enter an empty category name	1. Click on the Add icon button. 2. Click on "Add" button	-	A message "Category cannot be empty" is shown.	Error message shown.	Pass
Enter an existing category name	1. Click on the Add icon button. 2. Enter an existing category name 3. Click on "Add" button	-	A message "Category already exist" is shown	Error message shown.	Pass

Table 7.24: Unit Test Case for Edit Category

Test Case ID	Product Service-12			**Test Status**	Pass
Test Title	Unit Test Case for Edit Category				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has added a category.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Edit a category	1. Click on the Edit icon button 2. Enter a new category name 3. Click on "Done" button	Category name	The category name of all related products are updated.	Category name updated	Pass
Edit with an empty category name	1. Click on the Edit icon button 2. Erase category name 3. Click on "Done" button	-	A message "Category cannot be empty" is shown	Error message shown	Pass
Edit an existing category name	1. Click on the Edit icon button 2. Enter an existing category name 3. Click on "Done" button	-	A message "Category already exist" is shown	Error message shown	Pass

Table 7.25: Unit Test Case for Delete Category

Test Case ID	Product Service-13			**Test Status**	Pass
Test Title	Unit Test Case for Delete Category				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has added a category.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Delete a category	1. Clicks on the Delete icon button. 2. Clicks "Delete" button in the confirmation message prompt.	-	Category and its products are deleted	Category and its products are deleted	Pass

Table 7.26: Unit Test Case for Scan Expiry Date

Test Case ID	Product Service-14			**Test Status**	Pass
Test Title	Unit Test Case for Scan Expiry Date				
Module	Product				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**

Users use scan expiry date function while adding product	1. Click on "Scan Date" button 2. Take a photo of the scan date button 3. Crop out the date properly 4. Click on Save icon button	Expiry date image	A date will be entered automatically in the expiry date fields.	Date added	Pass

Table 7.27: Unit Test Case for Add Product Barcode

Test Case ID	Product Service-15			**Test Status**	Pass
Test Title	Unit Test Case for Add Product Barcode				
Module	Product				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Users add barcode while adding product	1. Click on the barcode field 2. Align the camera to the barcode	Product barcode	The barcode will be entered automatically into the barcode field	Barcode added	Pass

Table 7.28: Unit Test Case for Add Product Name Using OCR

Test Case ID	Product Service-16			**Test Status**	Pass
Test Title	Unit Test Case for Add Product Name Using OCR				
Module	Product				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Users use Scan product name with OCR function while adding product	1. Click on "Scan Product Name Using OCR" button 2. Align the camera to the product name 3. Tap on the product name	Product name	The product name will be entered automatically in the product name fields.	Product name added	Pass

Table 7.29: Unit Test Case for Upload Product Image

Test Case ID	Product Service-17			**Test Status**	Pass
Test Title	Unit Test Case for Upload Product Image				
Module	Product				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Upload product image while adding product	1. Click on the image placeholder 2. Capture a product image with the camera 3. Crop the product image 4. Click on the Save icon button	Product image	The image is displayed	Image displayed	Pass

Table 7.30: Unit Test Case for Add Alert Reminder

Test Case ID	Product Service-18			**Test Status**	Pass
Test Title	Unit Test Case for Add Alert Reminder				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has entered an expiry date.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Add reminder alerts while adding product	1. Click on the Reminder switch to turn on the reminder 2. Click on “Add Reminder” button 3. Select reminder date 4. Select reminder method 5. Click on “Save” button	Reminder date, reminder method	The reminder is displayed in the reminder field	Reminder displayed	Pass

Table 7.31: Unit Test Case for Prepare Product List

Test Case ID	Product Service-19			**Test Status**	Pass
Test Title	Unit Test Case for Prepare Product List				
Module	Product				
Pre-conditions	The user has signed in to the application. The user has an out-of-stock item.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**

View product list	1. Open the application	-	Products with zero stock number are not shown in the product list.	Products with zero stock number are not shown in the list.	Pass

Table 7.32: Unit Test Case for Getting All Shopping List

Test Case ID	Shopping List Service-01			**Test Status**	Pass
Test Title	Unit Test Case for Getting All Shopping List				
Module	Shopping List				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
The shopping list has one or more item	1. Click on "Shopping List" in the drawer navigation	-	All user's shopping list will be displayed	User's shopping list displayed	Pass
The shopping list is empty	1. Click on "Shopping List" in the drawer navigation	-	A message "Empty Shopping List" is displayed	Message displayed	Pass

Table 7.33: Unit Test Case for Add Shopping List

Test Case ID	Shopping List Service-02			**Test Status**	Pass
Test Title	Unit Test Case for Add Shopping List				
Module	Shopping List				
Pre-conditions	The user has signed in to the application.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Enter valid information	1. Click on Add icon button 2. Enter product name 3. Enter product description 4. Click on "Save" button	Product name, product description	Shopping list item is added	Shopping list item added	Pass
Do not enter the required field	1. Click on Add icon button 2. Click on "Save" button	-	A message "Product name can't be empty" is displayed	Erro message displayed	Pass

Table 7.34: Unit Test Case for Edit Shopping List

Test Case ID	Shopping List Service-03			Test Status	Pass
Test Title	Unit Test Case for Edit Shopping List				
Module	Shopping List				
Pre-conditions	The user has signed in to the application. The user has at least one shopping list item.				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Edit with a valid information	1. Click on “Edit” button 2. Edit the product name 3. Edit the product description 4. Click on “Save” button	Product name, product description	The shopping list item is changed	Shopping list item changed	Pass
Edit with empty required fields	1. Click on “Edit” button 2. Erase the product name 3. Click on “Save” button	-	A message “Product name can’t be empty” is shown	Error message shown	Pass

Table 7.35: Unit Test Case for Delete Shopping List

Test Case ID	Shopping List Service-04			Test Status	Pass
Test Title	Unit Test Case for Delete Shopping List				
Module	Shopping List				
Pre-conditions	The user has signed in to the application. The user has at least one shopping list item				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Delete a shopping list item by swiping left	1. Swipe a shopping list item to left	-	The shopping list item is deleted.	Shopping list item deleted	Pass
Delete a shopping list item by long press	1. Long press on a shopping list item 2. Click on “Delete” button	-	The shopping list item is deleted.	Shopping list item deleted	Pass

Table 7.36: Unit Test Case for Set Shopping List Item Status

Test Case ID	Shopping List Service-05			Test Status	Pass
Test Title	Unit Test Case for Set Shopping List Item Status				
Module	Shopping List				
Pre-conditions	The user has signed in to the application. The user has at least one shopping list item				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**

Check a shopping list item	1. Click on an unchecked shopping list item	-	The shopping list item is checked	Shopping list item checked	Pass
Uncheck a shopping list item	1. Click on a checked shopping list item	-	The shopping list item is unchecked	Shopping list item unchecked	Pass

Table 7.37: Unit Test Case for Prepare Shopping List

Test Case ID	Shopping List Service-06			**Test Status**	Pass
Test Title	Unit Test Case for Prepare Shopping List				
Module	Shopping List				
Pre-conditions	The user has signed in to the application. The user has at least one expired item or one out-of-stock item				
Test Case Summary	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
Add expired product into the shopping list automatically	1. Click on “Shopping List” in the drawer navigation	-	All expired products are added to the shopping list.	Expired products are added to the list	Pass
Added out-of-stock product into the shopping list	1. Click on “Shopping List” in the drawer navigation	-	All out-of-stock products are added to the shopping list.	Out-of-stock products are added to the list	Pass

7.3 Integration Test

The purpose of integration testing is to test how each component integrates with the others as a group. The listing of integration test cases is shown in the table below.

Table 7.38: Listing of Integration Test Cases

No	Test Title	Test Step	Expected Result	Actual Result	Test Status
1.	Search Product	1. Click on the Search icon button 2. Enter product name 3. Click on the product suggested in the suggestion list	The correct product details are displayed	Correct product details are displayed	Pass
2.	Add Product and Alerts Reminder	1. Click on the Add icon button 2. Enter product details 3. Switch on Reminder switch 4. Add reminders 5. Click on the Save icon button	Product is added with the correct alert and reminder	Product with correct reminder is added	Pass
3.	Add Product with newly added category	1. Click on the Add icon button 2. Enter product details 3. Click on “Add Category” button 4. Enter a valid category 5. Click on “Add” button 6. Select the added category 7. Click on the Save icon button	Product is added with the newly added category	Product with the new category is added	Pass
4.	Add Product with Scan date function	1. Click on the Add icon button 2. Enter product details 3. Click on “Scan Date” button 4. Take a photo on the expiry date 5. Crop out the expiry date 6. Click on the Save icon button	Product is added with the detected date	Product with the detected date is added	Pass

5.	Add Product with Scan Product Name with OCR function	1. Click on the Add icon button 2. Enter product details 3. Click on “Scan Product Name Using OCR” button 4. Align the camera with the product name 5. Select the product name 6. Click on the Save icon button	Product is added with the detected name	Product with the detected name is detected	Pass
6.	Add Product with Product Image	1. Click on the Add icon button 2. Enter product details 3. Click on Image Placeholder to add an image 4. Take a photo of the product 5. Crop the product photo taken 6. Click on the Save icon button	The image will be uploaded to cloud storage and a URL to the image will be generated and is added to the product	The image URL is added to the product	Pass
7.	Edit Product Image	1. Click on the Edit icon button 2. Click on the product image 3. Click on “Change Photo” button 4. Take a photo of the product 5. Crop the product photo taken 6. Click on the Save icon button	The image is updated.	Image updated	Pass

7.4 Cloud Functional Test

Cloud functions are used to send alerts and reminder to users. Hence, all cloud functions deployed and integrated into this application should also be tested.

Table 7.39: Listing of Cloud Functional Test Cases

No	Test Title	Test Step	Expected Result	Actual Result	Test Status
1.	Send Push Notification Alert	1. Add a product with a push notification alert	Push notification alert is received on the selected notification date and preferred alert time	Push notification alert is received on the selected notification date and preferred alert time	Pass
2.	Send SMS Notification Alert	1. Add a product with an SMS notification alert	SMS notification alert is received on the selected notification date and preferred alert time	SMS notification alert is received on the selected notification date and preferred alert time	Pass
3.	Send Email Notification Alert	Add a product with an email notification alert	Email notification alert is received on the selected notification date and preferred alert time	Email notification alert is received on the selected notification date and preferred alert time	Pass

7.5 Usability Testing

The goal of usability testing is to evaluate an application by testing with actual users. According to Usability.gov (2017), usability testing allows application developers to determine whether a user can complete defined tasks successfully and how long it takes to complete them. Furthermore, developers will learn how satisfied users are with the application through usability testing.

To conduct usability testing on this mobile application, three students live in the hostel, two housewives, a working individual who lives alone, and a supermarket manager are invited to participate in the test.

At the beginning of the test, the developer will brief the participant roughly about the nature of the mobile application and its purpose. The participants are given test scenarios to perform a series of task. The participant is given 15 minutes to read all the test scenarios and perform the task according to the test scenarios after reading. All of the participants can complete all of the tasks in 20 minutes and the participants are asked to answer a System Usability Scale at the end of the usability test.

A system usability scale (SUS), developed by Brooke (1986), is used to evaluate the usability of the application. Testers will be asked to scale from 1 (Strongly Disagree) to 5 (Strongly Agree) for 10 questions and a usability score will be calculated. The table below shows the listing of the questions in the SUS.

Table 7.40: System Usability Scale Questions (adapted from System Usability Scale, Brooke (1986)

	Strongly Disagree 1	**2**	**3**	**4**	**Strongly Agree 5**
1. I think that I would like to use Expiry Reminder App frequently.					
2. I found Expiry Reminder App unnecessarily complex.					
3. I thought Expiry Reminder App was easy to use.					
4. I think that I would need the support of a technical person to be able to use Expiry Reminder App.					
5. I found the various functions in Expiry Reminder App were well integrated.					
6. I thought there was too much inconsistency in Expiry Reminder App.					
7. I would imagine that most people would learn to use Expiry Reminder App very quickly.					
8. I found Expiry Reminder App very cumbersome (awkward) to use.					
9. I felt very confident using Expiry Reminder App.					
10. I needed to learn a lot of things before I could get going with Expiry Reminder App.					

7.5.1 Test Scenarios

Table 7.41: Test Scenarios for Usability Test

No	Test Scenario Title	Test Scenario Description
1.	Create an account and Sign In	To use the application, you need to create an account. You can choose any sign-in method you like to get started.
2.	Add a product	To track the expiry date of a product, you are required to add a product first. You will wish to: 1. Add a product image 2. Use OCR to insert a product name 3. Add a new category for the product 4. Use the Scan Date function to insert the expiry dates 5. Add a barcode entry for your product 6. Add a stock count for your product 7. Add one or more alerts based on your choice How you are going to perform all of the tasks stated above?
3.	View, edit and delete the product details	Imagine you have miscounted your stock for the product added, and you wish to check your number of stock recorded just now, what will you do? Now you have to view the number of stock for your product added, how will you change it? When you do not need to keep track of the product anymore and wish to delete the product, what will you do?
4.	Update product by category	Imagine you have lots of different kind of products recorded in the application. You wish to view the products by their category, what will you do? If you wish to change the category name, what will you do?
5.	Search by Barcode	Imagine you have a long product list and you wish to search the product by just scanning the barcode, what will you do?
6.	Add a shopping list	Imagine you wish to record some product to be bought for your next grocery shopping, what will you do? After you have put the item into your shopping cart and wants to mark it down, what will you do?
7.	Delete a shopping list	After you have purchased everything and went back home, you wish to clean up your shopping list by deleting the item on it, what will you do?

7.5.2 Usability Test Result

According to Sauro (2011), SUS is measured according to the following rules:

1. Deduct 1 from the user responses for every odd number questions
2. Deduct 5 from the user responses for every even number questions
3. The scale will have values ranging from 0 to 4, with 4 representing the most positive answer
4. The scores are added up and multiply by 2.5 to get a weightage of 100 per cent

The table below shows the calculated usability test score from all seven respondents. From the results, the average SUS scores at 85.71, which is considered excellent with a weightage of 100.

Table 7.42: Result of Usability Test from All 7 Respondents

Respondent	**Usability Score for Each Question**										**Total**	**Total (100%)**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**		
Respondent 1	3	3	4	2	3	3	3	3	3	3	30	75
Respondent 2	3	3	3	3	4	3	2	3	3	3	30	75
Respondent 3	3	3	3	1	3	3	3	4	2	2	27	67.5
Respondent 4	4	4	4	4	4	4	4	4	4	4	40	100
Respondent 5	4	4	4	4	4	4	3	4	4	3	38	95
Respondent 6	4	4	4	4	4	4	4	4	4	4	40	100
Respondent 7	4	3	4	3	3	4	3	4	4	3	35	87.5
Average SUS Score											85.71429	

7.6 User Acceptance Test

A user acceptance test is performed to determine whether an end-user can carry out the tasks in a given scenario. This test includes seven participants and the participants are given test cases and the test cases include tasks that are required to be completed. The developer only assists when the participants are unable to accomplish the task.

7.6.1 Test Cases

Table 7.43: Test Cases for User Acceptance Test

Testing date			
Start Time		**End Time**	
Tester Name			
Test Module	**Test Scenario**	**Pass/ Fail**	**Comments**
Login	1. Able to create an account and sign in with one of the sign in methods		
	2. Able to verify the email address		
	3. Able to verify the phone number		
Product Management	*Add Product*		
	1. Able to add a product image to the product		
	2. Able to add a product name to the product		
	3. Able to add a category to the product		
	4. Able to add an expiry date the product		
	5. Able to add a barcode entry to the product		
	6. Able to select the number of stock for the product		

	7. Able to add alerts and reminder to the product		
	Edit Product		
	1. Able to view any product details		
	2. Able to edit any product details		
	Delete Product		
	1. Able to delete any product		
	Search Product by Name		
	1. Able to search existing product by name		
	Search Product by Barcode		
	1. Able to search product by barcode		
Category Management	*Add category*		
	1. Able to add a new category		
	View Product by Category		
	1. Able to view products by category		
	Edit Category		
	1. Able to edit the category name		
	Delete Category		
	1. Able to delete category and all products under the category		
Shopping List Category	*Add Shopping List*		
	1. Able to add a new shopping list item		
	Edit Shopping List		
	1. Able to edit shopping list item details		
	Check and Uncheck Item		
	1. Able to check an unchecked shopping list item		
	2. Able to uncheck a checked shopping list item		

	Delete Shopping List		
	1. Able to delete shopping list item		

CHAPTER 8

CONCLUSION

8.1 Conclusion

The expiry date reminder is designed to keep track of product expiry dates and notify the user before the product expires. The application uses deep learning, OCR and barcode scanner to ease the user in entering product data, which is reluctant and problematic for the user to do traditionally. In contrast to the current reminder app, the expiry reminder mobile application built in this project offers three different options: Push notification reminder, SMS reminder and Email reminder. This ensures that users are reminded every time a reminder is delivered through the application.

The automatic handling mechanism of expired products also helps users to manage their product list efficiently. Expired and used up items are immediately added to the shopping list automatically. The finished products are often immediately excluded from the product list to avoid confusion for the user when managing the product.

To allow users to have an overview of the product list, there will be a dashboard displaying the total number of products that expired, the products that soon expired and the safe to use the product in different colours. Each product in the product list will also appear in different colours such that users can distinguish the severity of the product at a glance.

8.2 Achievement of Objectives

The project objectives are achieved successfully at the end of the project. The first objective is to generate a new expiry date dataset that will be used to train a deep learning model to recognize dates in a different format. The dataset is generated successfully with the TRDG library and contains 5359 classes with dates from March 2020 to December 2025. Each class contains 130 images with dates in different formats, colours, fonts, rotation angles and orientations. Various noisy backgrounds are used to mimic the noise in a real-life situation to increase the model accuracy on real-world images.

The second objective is to train an Inception ResNet V2 to recognize expiry dates. The model is trained with the hyperparameters: Batch Size of 64, Learning Rate of 0.0001 and a dropout layer with the dropout rate of 0.4. The model has successfully achieved 99.64% of accuracy and 1.57% of loss on the synthetic test dataset generated, and 96.12% accuracy and 1.44% of loss on noisy real-world images.

The third objective is to develop an expiry date reminder application with the implementation of the deep learning model. An interactive and smart application is developed with various functionalities to help users keep track of expiry dates on different products and reminds them accordingly. The Inception ResNet V2 trained is implemented into the mobile application and is used to detect the expiry dates of a product. The model is successfully deployed and can predict most of the expiry dates in the condition of sufficient light sources, clean background and clear images.

The last objective is to carry out usability testing for the mobile application to ensure the end-users are able to make use of all of the functionalities to perform their tasks effectively with regard to the management of expiry dates. The application has achieved a usability score of 85.71, which is considered excellent.

8.3 Limitations and Future Enhancements

Even though the mobile app is ready to use, there are also shortcomings that can be further enhanced in the future. The limitations and future recommendations are listed in the table below.

Table 8.1: Table of Limitations and Recommendation for Future Work

No.	Limitation	Recommendations
1.	The date in the date image dataset is restricted in format and range	The dataset can be extended to an expanded date format and greater date range. However, it will have a vastly higher training time and compute costs to train with a 4GB Graphic Card, which is not optimal.
2.	The mobile application does not provide interaction within a household	Firebase Firestore, the cloud storage tool, is used to store the database. The application can thus use the cloud to provide multi-user synchronization and sharing products. This is because goods in a household are not only handled by one individual but can also be taken

		in by each household to improve the management process.
3.	The mobile application does not allow users to change their phone number and email address	SMS and E-mail reminders are the most essential functionalities. However, once validated, users cannot change their phone number and email address. The application should provide the features to ensure that users can update their phone number and e-mail address without any problems.
4.	The mobile application does not allow users to change their password	Although users have the forgot password features to reset their password, it is not an optimal way to change the password. The Setting page should have a change password function, which allows users to update their password once in a while.
5.	The mobile application does not allow users to link various sign-in credentials	Although multiple sign-in methods are provided, multiple user credentials should be linked on the settings page. This prevents identical users, to build many accounts with separate credentials as they have forgotten which methods of sign-in. The application can also have an icon showing the sign-in credential, so users can still track the sign-in methods used.

REFERENCES

Agarwa, R. and Umphress, D. (2008) 'Extreme programming for a single person team', *Proceedings of the 46th Annual Southeast Regional Conference on XX, ACM-SE 46*, (January 2008), pp. 82–87. doi: 10.1145/1593105.1593127.

Andrei, B.-A. *et al.* (2019) 'A Study on Using Waterfall and Agile Methods in Software Project Management', *Journal of Information Systems & Operations Management*, pp. 125–135. Available at: http://0-search.proquest.com.ditlib.dit.ie.tudublin.idm.oclc.org/docview/2237828314?accountid=10594.

Asri, S. A. *et al.* (2018) 'Web Based Information System for Job Training Activities Using Personal Extreme Programming (PXP)', *Journal of Physics: Conference Series*, 953(1). doi: 10.1088/1742-6596/953/1/012092.

Avdic, D. (2018) *React Native vs Xamarin – Mobile for industry*. Lund University.

Belval, E. (2019) 'Text Recognition Data Generator'. Available at: https://github.com/Belval/TextRecognitionDataGenerator.

Bernard, M. (2018) *10 Amazing Examples Of How Deep Learning AI Is Used In Practice?*, *Forbes*. Available at: 10 Amazing Examples Of How Deep Learning AI Is Used In Practice?

Brooke, J. (1986) *System Usability Scale*.

Casteren, W. Van (2017) *The Waterfall Model and the Agile Methodologies*. doi: 10.13140/RG.2.2.36825.72805.

Dzhurov, Y., Krasteva, I. and Ilieva, S. (2009) 'Personal Extreme Programming–An Agile Process for Autonomous Developers', *International Conference on software, services & semantic technologies*, (August 2016), pp. 252–259. Available at: https://www.researchgate.net/publication/229046039_Personal_Extreme_Programming-An_Agile_Process_for_Autonomous_Developers.

Fentaw, A. E. (2020) *Cross platform mobile application development : a comparison study of React Native Vs Flutter*. University of Jyväskylä.

Geirhos, R. *et al.* (2017) 'Comparing deep neural networks against humans: Object recognition when the signal gets weaker', *arXiv*.

Google (2017) *Barcode API Overview*. Available at: https://developers.google.com/vision/android/barcodes-overview (Accessed: 5 August 2020).

He, K. *et al.* (2016) 'Deep residual learning for image recognition', *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem, pp. 770–778. doi: 10.1109/CVPR.2016.90.

HUB, eDesk (2017) *Why Agile Methodology for Mobile App Development?*, *Medium*.

Available at: https://medium.com/@edeskhub/why-agile-methodology-for-mobile-app-development-e685b41ea462 (Accessed: 27 July 2020).

Innovative Architects (2017) *The Seven Phases of the System-Development Life Cycle*. Available at: https://www.innovativearchitects.com/KnowledgeCenter/basic-IT-systems/system-development-life-cycle.aspx (Accessed: 12 August 2020).

Jean Coutu (no date) *Avoid using expired cosmetics!* Available at: https://www.jeancoutu.com/en/beauty/beauty-tips/watch-out-for-cosmetics-that-have-expired/ (Accessed: 25 July 2020).

Kim, K. G. (2019) 'Deep learning book review', *Nature*, 29(7553), pp. 1–73.

Kranthi Kumar, K. *et al.* (2021) 'Role of convolutional neural networks for any real time image classification, recognition and analysis', *Materials Today: Proceedings*. Elsevier. doi: 10.1016/j.matpr.2021.02.186.

Lake, B. M. *et al.* (2015) 'Deep Neural Networks Predict Category Typicality Ratings for Images', *{Proceedings of the 37th Annual Conference of the Cognitive Science Society}*.

Linode (no date) *About Linode*. Available at: https://www.linode.com/company/about/ (Accessed: 5 August 2020).

Lynnandwei (2016) *Inception V4 and ResNet*, *CSDN*. Available at: http://blog.csdn.net/lynnandwei/article/details/53736235.

Malaysian Food Waste (2019) *Poverty Pollution Persecution*. Available at: http://pppp.my/malaysian-food-waste.html (Accessed: 25 July 2020).

Mallick, S. (2017) *Bias-Variance Tradeoff in Machine Learning*, *OpenCV*. Available at: https://learnopencv.com/bias-variance-tradeoff-in-machine-learning/ (Accessed: 20 March 2021).

Navamani, T. M. (2019) 'Efficient Deep Learning Approaches for Health Informatics', in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. Elsevier, pp. 123–137. doi: 10.1016/b978-0-12-816718-2.00014-2.

Nazaré, T. S. *et al.* (2018) 'Deep convolutional neural networks and noisy images', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10657 LNCS(January), pp. 416–424. doi: 10.1007/978-3-319-75193-1_50.

Pressman, R. S. (2010) *Software Engineering: A Practitioner's Approach*. 7th edn, *McGraw-Hill*. 7th edn. McGraw-Hill.

Sauro, J. (2011) *Measuring Usability with The System Usability Scale (SUS)*, *MeasuringU*. Available at: https://measuringu.com/sus/.

Shaydulin, R. and Sybrandt, J. (2017) 'To Agile, or not to Agile: A Comparison of Software Development Methodologies', pp. 1–11. Available at: http://arxiv.org/abs/1704.07469.

Sommerville, I. (2011) *Software Engineering*. 9th edn, *Addison-Wesley*. 9th edn.

Szegedy, C. *et al.* (2017) ‘Inception-v4, inception-ResNet and the impact of residual connections on learning’, *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 4278–4284.

Team, S. S. (2019) *What is SSL?* Available at: https://www.ssl.com/faqs/faq-what-is-ssl/ (Accessed: 5 August 2020).

Thomas, G. (2019) *What is Flutter and Why You Should Learn it in 2020*, *FreeCodeCamp*. Available at: https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/ (Accessed: 20 August 2020).

Uhrig, R. E. (1995) ‘Introduction to artificial neural networks’, in *Proceedings of IECON ’95 - 21st Annual Conference on IEEE Industrial Electronics*, pp. 33–37 vol.1. doi: 10.1109/IECON.1995.483329.

Usability.gov (2017) *Usability Testing*. Available at: https://www.usability.gov/how-to-and-tools/methods/usability-testing.html (Accessed: 10 March 2021).

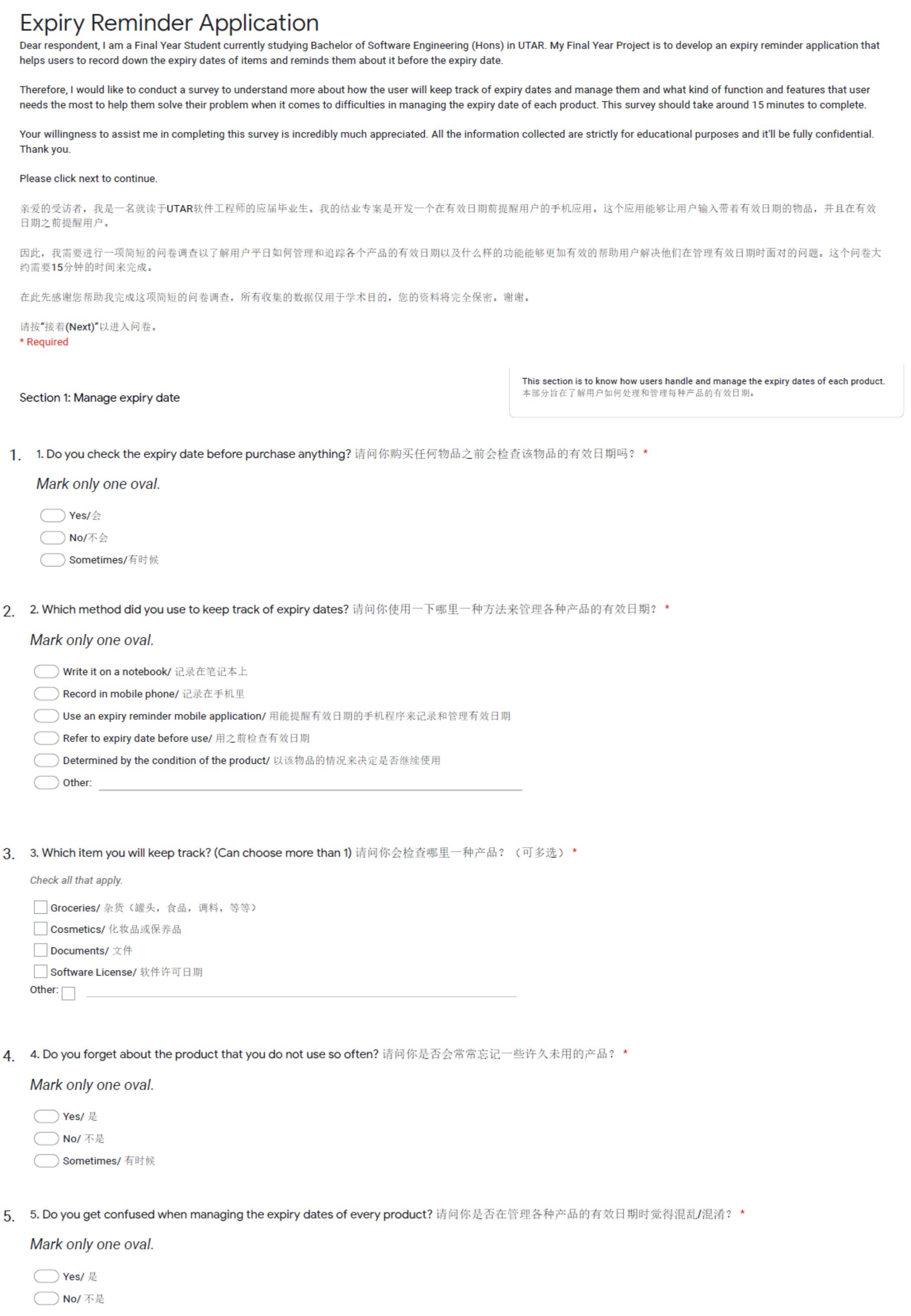
Wells, D. (1999) *When should Extreme Programming be Used?*, *ExtremeProgramming.org*. Available at: http://www.extremeprogramming.org/when.html.

Yu, H. *et al.* (2021) ‘Convolutional Neural Networks for Medical Image Analysis: State-of-the-art, Comparisons, Improvement and Perspectives’, *Neurocomputing*. Elsevier BV. doi: 10.1016/j.neucom.2020.04.157.

Yu, Y. *et al.* (2019) ‘A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures’, *Neural Computation*, 31, pp. 1235–1270.

APPENDICES

APPENDIX A : Questionnaire Survey



Expiry Reminder Application

Dear respondent, I am a Final Year Student currently studying Bachelor of Software Engineering (Hons) in UTAR. My Final Year Project is to develop an expiry reminder application that helps users to record down the expiry dates of items and reminds them about it before the expiry date.

Therefore, I would like to conduct a survey to understand more about how the user will keep track of expiry dates and manage them and what kind of function and features that user needs the most to help them solve their problem when it comes to difficulties in managing the expiry date of each product. This survey should take around 15 minutes to complete.

Your willingness to assist me in completing this survey is incredibly much appreciated. All the information collected are strictly for educational purposes and it'll be fully confidential. Thank you.

Please click next to continue.

亲爱的受访者，我是一名就读于UTAR软件工程师的应届毕业生。我的结业专案是开发一个在有效日期前提醒用户的手机应用。这个应用能够让用户输入带着有效日期的物品，并且在有效日期之前提醒用户。

因此，我需要进行一项简短的问卷调查以了解用户平日如何管理和追踪各个产品的有效日期以及什么样的功能能够更加有效的帮助用户解决他们在管理有效日期时面对的问题。这个问卷大约需要15分钟的时间来完成。

在此先感谢您帮助我完成这项简短的问卷调查，所有收集的数据仅用于学术目的，您的资料将完全保密。谢谢。

请按"接着(Next)"以进入问卷。
* Required

Section 1: Manage expiry date

This section is to know how users handle and manage the expiry dates of each product.
本部分旨在了解用户如何处理和管理每种产品的有效日期。

1. 1. Do you check the expiry date before purchase anything? 请问你购买任何物品之前会检查该物品的有效日期吗？ *

Mark only one oval.

- Yes/会
- No/不会
- Sometimes/有时候

2. 2. Which method did you use to keep track of expiry dates? 请问你使用一下哪里一种方法来管理各种产品的有效日期？ *

Mark only one oval.

- Write it on a notebook/ 记录在笔记本上
- Record in mobile phone/ 记录在手机里
- Use an expiry reminder mobile application/ 用能提醒有效日期的手机程序来记录和管理有效日期
- Refer to expiry date before use/ 用之前检查有效日期
- Determined by the condition of the product/ 以该物品的情况来决定是否继续使用
- Other: ________________

3. 3. Which item you will keep track? (Can choose more than 1) 请问你会检查哪里一种产品？ （可多选） *

Check all that apply.

- Groceries/ 杂货（罐头，食品，调料，等等）
- Cosmetics/ 化妆品或保养品
- Documents/ 文件
- Software License/ 软件许可日期
- Other: ________________

4. 4. Do you forget about the product that you do not use so often? 请问你是否会常常忘记一些许久未用的产品？ *

Mark only one oval.

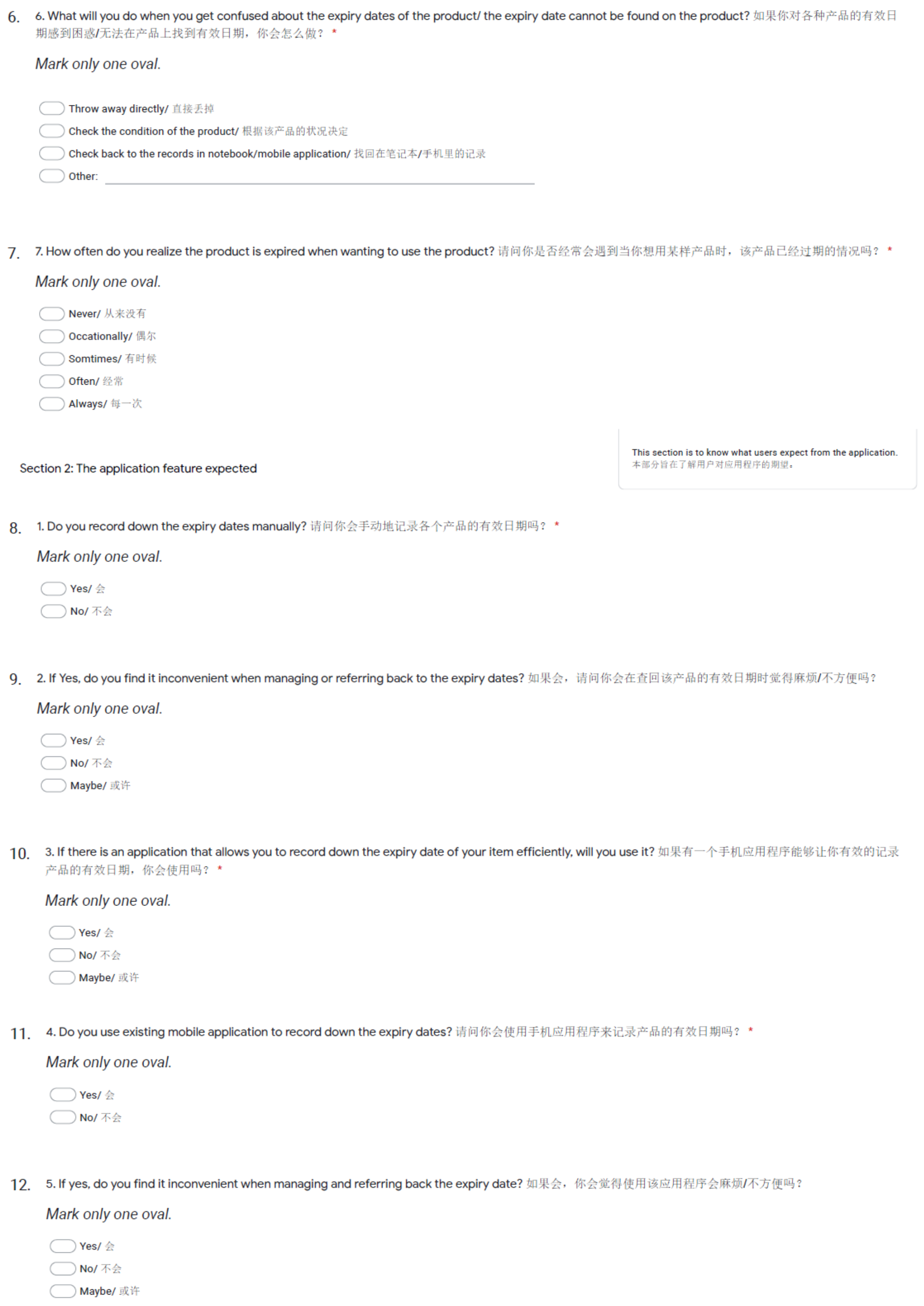
- Yes/ 是
- No/ 不是
- Sometimes/ 有时候

5. 5. Do you get confused when managing the expiry dates of every product? 请问你是否在管理各种产品的有效日期时觉得混乱/混淆？ *

Mark only one oval.

- Yes/ 是
- No/ 不是

Figure A-1: Questionnaire Survey of Expiry Reminder Mobile App (1)



6. 6. What will you do when you get confused about the expiry dates of the product/ the expiry date cannot be found on the product? 如果你对各种产品的有效日期感到困惑/无法在产品上找到有效日期，你会怎么做？ *

Mark only one oval.

- Throw away directly/ 直接丢掉
- Check the condition of the product/ 根据该产品的状况决定
- Check back to the records in notebook/mobile application/ 找回在笔记本/手机里的记录
- Other: ______________________________

7. 7. How often do you realize the product is expired when wanting to use the product? 请问你是否经常会遇到当你想用某样产品时，该产品已经过期的情况吗？ *

Mark only one oval.

- Never/ 从来没有
- Occationally/ 偶尔
- Somtimes/ 有时候
- Often/ 经常
- Always/ 每一次

Section 2: The application feature expected

This section is to know what users expect from the application.
本部分旨在了解用户对应用程序的期望。

8. 1. Do you record down the expiry dates manually? 请问你会手动地记录各个产品的有效日期吗？ *

Mark only one oval.

- Yes/ 会
- No/ 不会

9. 2. If Yes, do you find it inconvenient when managing or referring back to the expiry dates? 如果会，请问你会在查回该产品的有效日期时觉得麻烦/不方便吗？

Mark only one oval.

- Yes/ 会
- No/ 不会
- Maybe/ 或许

10. 3. If there is an application that allows you to record down the expiry date of your item efficiently, will you use it? 如果有一个手机应用程序能够让你有效的记录产品的有效日期，你会使用吗？ *

Mark only one oval.

- Yes/ 会
- No/ 不会
- Maybe/ 或许

11. 4. Do you use existing mobile application to record down the expiry dates? 请问你会使用手机应用程序来记录产品的有效日期吗？ *

Mark only one oval.

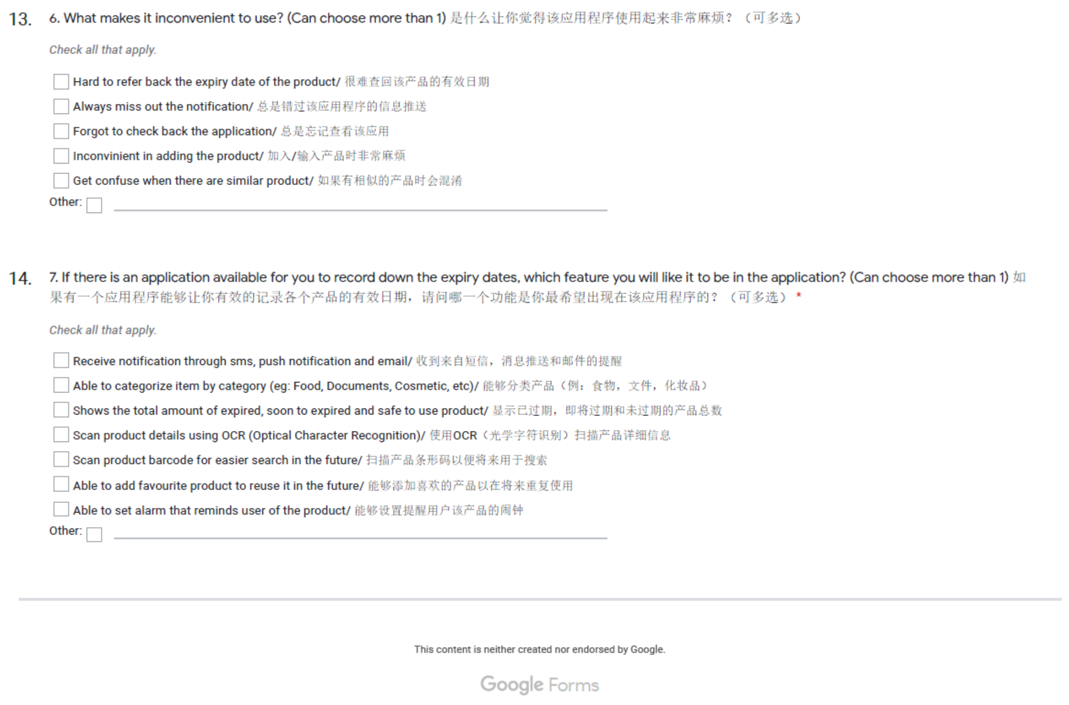
- Yes/ 会
- No/ 不会

12. 5. If yes, do you find it inconvenient when managing and referring back the expiry date? 如果会，你会觉得使用该应用程序会麻烦/不方便吗？

Mark only one oval.

- Yes/ 会
- No/ 不会
- Maybe/ 或许

Figure A-2: Questionnaire Survey of Expiry Reminder Mobile App (2)



13. 6. What makes it inconvenient to use? (Can choose more than 1) 是什么让你觉得该应用程序使用起来非常麻烦？（可多选）

Check all that apply.

- ☐ Hard to refer back the expiry date of the product/ 很难查回该产品的有效日期
- ☐ Always miss out the notification/ 总是错过该应用程序的信息推送
- ☐ Forgot to check back the application/ 总是忘记查看该应用
- ☐ Inconvinient in adding the product/ 加入/输入产品时非常麻烦
- ☐ Get confuse when there are similar product/ 如果有相似的产品时会混淆

Other: ☐ ____________________

14. 7. If there is an application available for you to record down the expiry dates, which feature you will like it to be in the application? (Can choose more than 1) 如果有一个应用程序能够让你有效的记录各个产品的有效日期，请问哪一个功能是你最希望出现在该应用程序的？（可多选） *

Check all that apply.

- ☐ Receive notification through sms, push notification and email/ 收到来自短信，消息推送和邮件的提醒
- ☐ Able to categorize item by category (eg: Food, Documents, Cosmetic, etc)/ 能够分类产品（例：食物，文件，化妆品）
- ☐ Shows the total amount of expired, soon to expired and safe to use product/ 显示已过期，即将过期和未过期的产品总数
- ☐ Scan product details using OCR (Optical Character Recognition)/ 使用OCR（光学字符识别）扫描产品详细信息
- ☐ Scan product barcode for easier search in the future/ 扫描产品条形码以便将来用于搜索
- ☐ Able to add favourite product to reuse it in the future/ 能够添加喜欢的产品以在将来重复使用
- ☐ Able to set alarm that reminds user of the product/ 能够设置提醒用户该产品的闹钟

Other: ☐ ____________________

This content is neither created nor endorsed by Google.

Google Forms

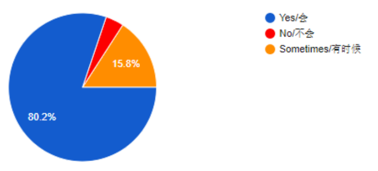
Figure A-3: Questionnaire Survey of Expiry Reminder Mobile App (3)

APPENDIX B : Summary of Results from the Questionnaire Survey

Section 1: Manage expiry date

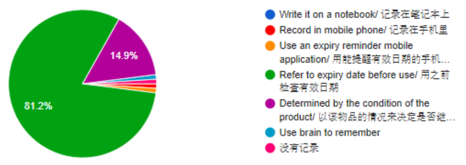
1. Do you check the expiry date before purchase anything? 请问你购买任何物品之前会检查该物品的有效日期吗？

101 responses



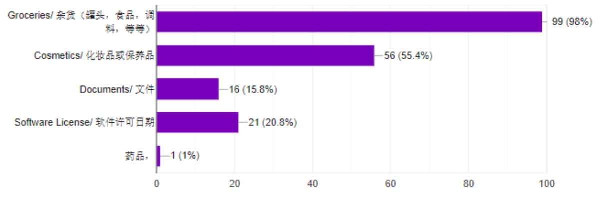
2. Which method did you use to keep track of expiry dates? 请问你使用一下哪里一种方法来管理各种产品的有效日期？

101 responses



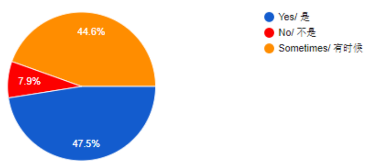
3. Which item you will keep track? (Can choose more than 1) 请问你会检查哪里一种产品？（可多选）

101 responses



4. Do you forget about the product that you do not use so often? 请问你是否会常常忘记一些许久未用的产品？

101 responses



5. Do you get confused when managing the expiry dates of every product? 请问你是否在管理各种产品的有效日期时觉得混乱/混淆？

101 responses

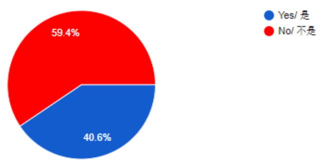
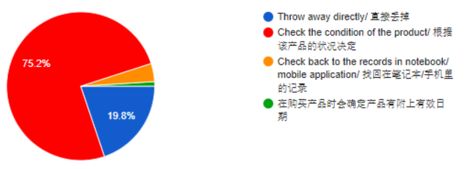


Figure B-1: Summary Results from Questionnaire of Expiry Reminder Mobile App (1)

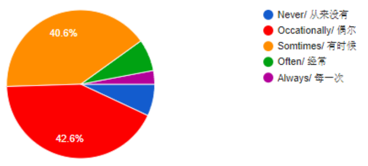
6. What will you do when you get confused about the expiry dates of the product/ the expiry date cannot be found on the product? 如果你对各种产品的有效日期感到困惑/无法在产品上找到有效日期，你会怎么做？

101 responses



7. How often do you realize the product is expired when wanting to use the product? 请问你是否经常会遇到当你想用某样产品时，该产品已经过期的情况吗？

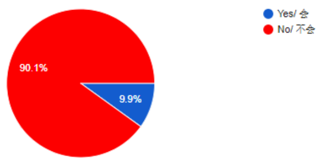
101 responses



Section 2: The application feature expected

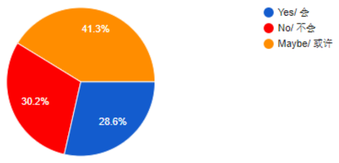
1. Do you record down the expiry dates manually? 请问你会手动地记录各个产品的有效日期吗？

101 responses



2. If Yes, do you find it inconvenient when managing or referring back to the expiry dates? 如果会，请问你会在查回该产品的有效日期时觉得麻烦/不方便吗？

63 responses



3. If there is an application that allows you to record down the expiry date of your item efficiently, will you use it? 如果有一个手机应用程序能够让你有效的记录产品的有效日期，你会使用吗？

101 responses

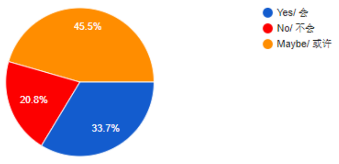
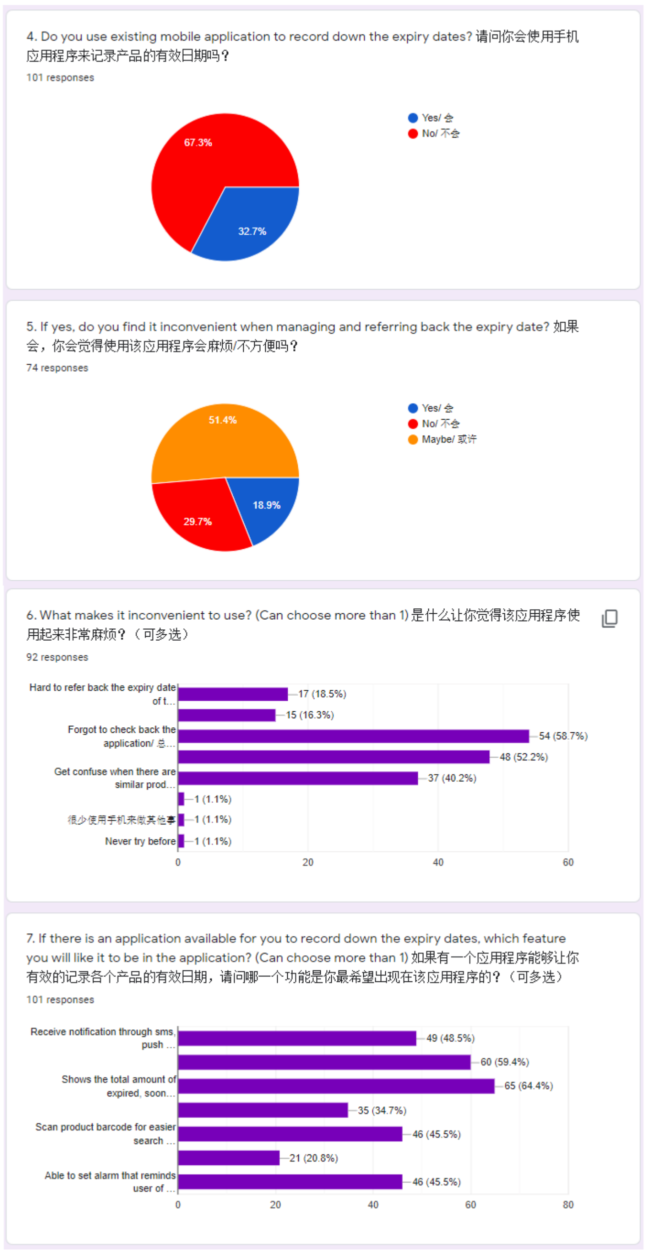


Figure B-2: Summary Results from Questionnaire of Expiry Reminder Mobile App (2)


4. Do you use existing mobile application to record down the expiry dates? 请问你会使用手机应用程序来记录产品的有效日期吗？

101 responses

5. If yes, do you find it inconvenient when managing and referring back the expiry date? 如果会，你会觉得使用该应用程序会麻烦/不方便吗？

74 responses

6. What makes it inconvenient to use? (Can choose more than 1) 是什么让你觉得该应用程序使用起来非常麻烦？（可多选）

92 responses

Hard to refer back the expiry date of t...
17 (18.5%)
15 (16.3%)
Forgot to check back the application/ 总...
54 (58.7%)
48 (52.2%)
Get confuse when there are similar prod...
37 (40.2%)
1 (1.1%)
很少使用手机来做其他事
1 (1.1%)
Never try before
1 (1.1%)
0
20
40
60

7. If there is an application available for you to record down the expiry dates, which feature you will like it to be in the application? (Can choose more than 1) 如果有一个应用程序能够让你有效的记录各个产品的有效日期，请问哪一个功能是你最希望出现在该应用程序的？（可多选）

101 responses

Receive notification through sms, push ...
49 (48.5%)
60 (59.4%)
Shows the total amount of expired, soon...
65 (64.4%)
35 (34.7%)
Scan product barcode for easier search ...
46 (45.5%)
21 (20.8%)
Able to set alarm that reminds user of ...
46 (45.5%)
0
20
40
60
80

Figure B-3: Summary Results from Questionnaire of Expiry Reminder Mobile App (3)

APPENDIX C : Result of Sample Usability Test

Participant: #1   Occupation: Students

Section 1

	Strongly Disagree 1	**2**	**3**	**4**	**Strongly Agree 5**
1. I think that I would like to use Expiry Reminder App frequently.				✓	
2. I found Expiry Reminder App unnecessarily complex.		✓			
3. I thought Expiry Reminder App was easy to use.					✓
4. I think that I would need the support of a technical person to be able to use Expiry Reminder App.			✓		
5. I found the various functions in Expiry Reminder App were well integrated.				✓	
6. I thought there was too much inconsistency in Expiry Reminder App.		✓			
7. I would imagine that most people would learn to use Expiry Reminder App very quickly.				✓	
8. I found Expiry Reminder App very cumbersome (awkward) to use.		✓			
9. I felt very confident using Expiry Reminder App.				✓	
10. I needed to learn a lot of things before I could get going with Expiry Reminder App.		✓			

Section 2

1. Do you have anything to tell the developers?

Well design, and user friendly, good good

Participant: #2 Occupation: Students

Section 1

	Strongly Disagree 1	**2**	**3**	**4**	**Strongly Agree 5**
1. I think that I would like to use Expiry Reminder App frequently.				✓	
2. I found Expiry Reminder App unnecessarily complex.		✓			
3. I thought Expiry Reminder App was easy to use.				✓	
4. I think that I would need the support of a technical person to be able to use Expiry Reminder App.		✓			
5. I found the various functions in Expiry Reminder App were well integrated.					✓
6. I thought there was too much inconsistency in Expiry Reminder App.		✓			
7. I would imagine that most people would learn to use Expiry Reminder App very quickly.			✓		
8. I found Expiry Reminder App very cumbersome (awkward) to use.		✓			
9. I felt very confident using Expiry Reminder App.				✓	
10. I needed to learn a lot of things before I could get going with Expiry Reminder App.		✓			

Section 2

1. Do you have anything to tell the developers?

____

Participant: #3 Occupation: Supermarket Manager

Section 1

	Strongly Disagree 1	**2**	**3**	**4**	**Strongly Agree 5**
1. I think that I would like to use Expiry Reminder App frequently.				✓	
2. I found Expiry Reminder App unnecessarily complex.		✓			
3. I thought Expiry Reminder App was easy to use.				✓	
4. I think that I would need the support of a technical person to be able to use Expiry Reminder App.				✓	
5. I found the various functions in Expiry Reminder App were well integrated.				✓	
6. I thought there was too much inconsistency in Expiry Reminder App.		✓			
7. I would imagine that most people would learn to use Expiry Reminder App very quickly.				✓	
8. I found Expiry Reminder App very cumbersome (awkward) to use.	✓				
9. I felt very confident using Expiry Reminder App.			✓		
10. I needed to learn a lot of things before I could get going with Expiry Reminder App.			✓		

Section 2

1. Do you have anything to tell the developers?

我觉得 scan photo description 比较难读取产品的名字

Participant: #4 Occupation: Housewife

Section 1

	Strongly Disagree 1	**2**	**3**	**4**	**Strongly Agree 5**
1. I think that I would like to use Expiry Reminder App frequently.					✓
2. I found Expiry Reminder App unnecessarily complex.	✓				
3. I thought Expiry Reminder App was easy to use.					✓
4. I think that I would need the support of a technical person to be able to use Expiry Reminder App.	✓				
5. I found the various functions in Expiry Reminder App were well integrated.					✓
6. I thought there was too much inconsistency in Expiry Reminder App.	✓				
7. I would imagine that most people would learn to use Expiry Reminder App very quickly.					✓
8. I found Expiry Reminder App very cumbersome (awkward) to use.	✓				
9. I felt very confident using Expiry Reminder App.					✓
10. I needed to learn a lot of things before I could get going with Expiry Reminder App.	✓				

Section 2

1. Do you have anything to tell the developers?

____

Participant: #5 Occupation: Housewife

Section 1

	Strongly Disagree 1	**2**	**3**	**4**	**Strongly Agree 5**
1. I think that I would like to use Expiry Reminder App frequently.					✓
2. I found Expiry Reminder App unnecessarily complex.	✓				
3. I thought Expiry Reminder App was easy to use.					✓
4. I think that I would need the support of a technical person to be able to use Expiry Reminder App.	✓				
5. I found the various functions in Expiry Reminder App were well integrated.					✓
6. I thought there was too much inconsistency in Expiry Reminder App.	✓				
7. I would imagine that most people would learn to use Expiry Reminder App very quickly.				✓	
8. I found Expiry Reminder App very cumbersome (awkward) to use.	✓				
9. I felt very confident using Expiry Reminder App.					✓
10. I needed to learn a lot of things before I could get going with Expiry Reminder App.		✓			

Section 2

1. Do you have anything to tell the developers?

I found that this is a quite useful apps especially for the busy housewife.

Participant: #6 Occupation: Teacher

Section 1

	Strongly Disagree 1	**2**	**3**	**4**	**Strongly Agree 5**
1. I think that I would like to use Expiry Reminder App frequently.					✓
2. I found Expiry Reminder App unnecessarily complex.	✓				
3. I thought Expiry Reminder App was easy to use.					✓
4. I think that I would need the support of a technical person to be able to use Expiry Reminder App.	✓				
5. I found the various functions in Expiry Reminder App were well integrated.					✓
6. I thought there was too much inconsistency in Expiry Reminder App.	✓				
7. I would imagine that most people would learn to use Expiry Reminder App very quickly.					✓
8. I found Expiry Reminder App very cumbersome (awkward) to use.	✓				
9. I felt very confident using Expiry Reminder App.					✓
10. I needed to learn a lot of things before I could get going with Expiry Reminder App.	✓				

Section 2

1. Do you have anything to tell the developers?

____

Participant: #7   Occupation: Student

Section 1

	Strongly Disagree 1	**2**	**3**	**4**	**Strongly Agree 5**
1. I think that I would like to use Expiry Reminder App frequently.					✓
2. I found Expiry Reminder App unnecessarily complex.		✓			
3. I thought Expiry Reminder App was easy to use.					✓
4. I think that I would need the support of a technical person to be able to use Expiry Reminder App.		✓			
5. I found the various functions in Expiry Reminder App were well integrated.				✓	
6. I thought there was too much inconsistency in Expiry Reminder App.	✓				
7. I would imagine that most people would learn to use Expiry Reminder App very quickly.				✓	
8. I found Expiry Reminder App very cumbersome (awkward) to use.	✓				
9. I felt very confident using Expiry Reminder App.					✓
10. I needed to learn a lot of things before I could get going with Expiry Reminder App.		✓			

Section 2

1. Do you have anything to tell the developers?

very nice app

APPENDIX D: Result of User Acceptance Test

Testing date	18/3/2021			
Start Time	1:18 PM	**End Time**		1:40 PM
Tester Name	Chen Rui Chee			
Test Module	**Test Scenario**		**Pass/ Fail**	**Comments**
Login	1. Able to create an account and sign in with one of the sign in methods		Pass	-
	2. Able to verify the email address		Pass	-
	3. Able to verify the phone number		Pass	-
Product Management	*Add Product*			
	1. Able to add a product image to the product		Pass	-
	2. Able to add a product name to the product		Pass	-
	3. Able to add a category to the product		Pass	-
	4. Able to add an expiry date the product		Pass	-
	5. Able to add a barcode entry to the product		Pass	-
	6. Able to select the number of stock for the product		Pass	-
	7. Able to add alerts and reminder to the product		Pass	-
	Edit Product			
	1. Able to view any product details		Pass	-
	2. Able to edit any product details		Pass	-
	Delete Product			
	1. Able to delete any product		Pass	-
	Search Product by Name			
	1. Able to search existing product by name		Pass	-
	Search Product by Barcode			

	1. Able to search product by barcode	Pass	-
Category Management	*Add category*		
	1. Able to add a new category	Pass	-
	View Product by Category		
	1. Able to view products by category	Pass	-
	Edit Category		
	1. Able to edit the category name	Pass	-
	Delete Category		
	1. Able to delete category and all products under the category	Pass	-
Shopping List Category	*Add Shopping List*		
	1. Able to add a new shopping list item	Pass	-
	Edit Shopping List		
	1. Able to edit shopping list item details	Pass	-
	Check and Uncheck Item		
	1. Able to check an unchecked shopping list item	Pass	-
	2. Able to uncheck a checked shopping list item	Pass	-
	Delete Shopping List		
	1. Able to delete shopping list item	Pass	-

Testing date	18/3/2021		
Start Time	1:18 PM	**End Time**	1:48 PM
Tester Name	Foong Cai Ling		
Test Module	**Test Scenario**	**Pass/ Fail**	**Comments**
Login	1. Able to create an account and sign in with one of the sign in methods	Pass	-
	2. Able to verify the email address	Pass	-
	3. Able to verify the phone number	Pass	-
Product Management	*Add Product*		
	1. Able to add a product image to the product	Pass	-
	2. Able to add a product name to the product	Pass	-
	3. Able to add a category to the product	Pass	-
	4. Able to add an expiry date the product	Pass	-
	5. Able to add a barcode entry to the product	Pass	-
	6. Able to select the number of stock for the product	Pass	-
	7. Able to add alerts and reminder to the product	Pass	-
	Edit Product		
	1. Able to view any product details	Pass	-
	2. Able to edit any product details	Pass	-
	Delete Product		
	1. Able to delete any product	Pass	-
	Search Product by Name		
	1. Able to search existing product by name	Pass	-
	Search Product by Barcode		

	1. Able to search product by barcode	Pass	-
Category Management	*Add category*		
	1. Able to add a new category	Pass	-
	View Product by Category		
	1. Able to view products by category	Pass	-
	Edit Category		
	1. Able to edit the category name	Pass	-
	Delete Category		
	1. Able to delete category and all products under the category	Pass	-
Shopping List Category	*Add Shopping List*		
	1. Able to add a new shopping list item	Pass	-
	Edit Shopping List		
	1. Able to edit shopping list item details	Pass	-
	Check and Uncheck Item		
	1. Able to check an unchecked shopping list item	Pass	-
	2. Able to uncheck a checked shopping list item	Pass	-
	Delete Shopping List		
	1. Able to delete shopping list item	Pass	-

Testing date	19/3/2021		
Start Time	7:59 PM	**End Time**	8:20 PM
Tester Name	Toh Chai Lian		
Test Module	**Test Scenario**	**Pass/ Fail**	**Comments**
Login	1. Able to create an account and sign in with one of the sign in methods	Pass	-
	2. Able to verify the email address	Pass	Takes longer time to find the email
	3. Able to verify the phone number	Pass	-
Product Management	*Add Product*		
	1. Able to add a product image to the product	Pass	-
	2. Able to add a product name to the product	Pass	OCR is hard to use
	3. Able to add a category to the product	Pass	-
	4. Able to add an expiry date the product	Pass	-
	5. Able to add a barcode entry to the product	Pass	-
	6. Able to select the number of stock for the product	Pass	-
	7. Able to add alerts and reminder to the product	Pass	-
	Edit Product		
	1. Able to view any product details	Pass	-
	2. Able to edit any product details	Pass	-
	Delete Product		
	1. Able to delete any product	Pass	-
	Search Product by Name		
	1. Able to search existing product by name	Pass	-
	Search Product by Barcode		

	1. Able to search product by barcode	Pass	-
Category Management	*Add category*		
	1. Able to add a new category	Pass	-
	View Product by Category		
	1. Able to view products by category	Pass	-
	Edit Category		
	1. Able to edit the category name	Pass	-
	Delete Category		
	1. Able to delete category and all products under the category	Pass	-
Shopping List Category	*Add Shopping List*		
	1. Able to add a new shopping list item	Pass	-
	Edit Shopping List		
	1. Able to edit shopping list item details	Pass	-
	Check and Uncheck Item		
	1. Able to check an unchecked shopping list item	Pass	-
	2. Able to uncheck a checked shopping list item	Pass	-
	Delete Shopping List		
	1. Able to delete shopping list item	Pass	-

Testing date	20/3/2021		
Start Time	2:30 PM	**End Time**	2:51 PM
Tester Name	Ong Sok Hwee		
Test Module	**Test Scenario**	**Pass/ Fail**	**Comments**
Login	1. Able to create an account and sign in with one of the sign in methods	Pass	-
	2. Able to verify the email address	Pass	-
	3. Able to verify the phone number	Pass	-
Product Management	*Add Product*		
	1. Able to add a product image to the product	Pass	-
	2. Able to add a product name to the product	Pass	-
	3. Able to add a category to the product	Pass	-
	4. Able to add an expiry date the product	Pass	-
	5. Able to add a barcode entry to the product	Pass	-
	6. Able to select the number of stock for the product	Pass	-
	7. Able to add alerts and reminder to the product	Pass	-
	Edit Product		
	1. Able to view any product details	Pass	-
	2. Able to edit any product details	Pass	-
	Delete Product		
	1. Able to delete any product	Pass	-
	Search Product by Name		
	1. Able to search existing product by name	Pass	-
	Search Product by Barcode		

	1. Able to search product by barcode	Pass	-
Category Management	*Add category*		
	1. Able to add a new category	Pass	-
	View Product by Category		
	1. Able to view products by category	Pass	-
	Edit Category		
	1. Able to edit the category name	Pass	-
	Delete Category		
	1. Able to delete category and all products under the category	Pass	-
Shopping List Category	*Add Shopping List*		
	1. Able to add a new shopping list item	Pass	-
	Edit Shopping List		
	1. Able to edit shopping list item details	Pass	-
	Check and Uncheck Item		
	1. Able to check an unchecked shopping list item	Pass	-
	2. Able to uncheck a checked shopping list item	Pass	-
	Delete Shopping List		
	1. Able to delete shopping list item	Pass	-

Testing date	20/3/2021		
Start Time	3:20 PM	**End Time**	3:51 PM
Tester Name	Tay Mei Hoon		
Test Module	**Test Scenario**	**Pass/ Fail**	**Comments**
Login	1. Able to create an account and sign in with one of the sign in methods	Pass	-
	2. Able to verify the email address	Pass	-
	3. Able to verify the phone number	Pass	-
Product Management	*Add Product*		
	1. Able to add a product image to the product	Pass	-
	2. Able to add a product name to the product	Pass	-
	3. Able to add a category to the product	Pass	-
	4. Able to add an expiry date the product	Pass	-
	5. Able to add a barcode entry to the product	Pass	-
	6. Able to select the number of stock for the product	Pass	-
	7. Able to add alerts and reminder to the product	Pass	-
	Edit Product		
	1. Able to view any product details	Pass	-
	2. Able to edit any product details	Pass	-
	Delete Product		
	1. Able to delete any product	Pass	-
	Search Product by Name		
	1. Able to search existing product by name	Pass	-
	Search Product by Barcode		

	1. Able to search product by barcode	Pass	-
Category Management	*Add category*		
	1. Able to add a new category	Pass	-
	View Product by Category		
	1. Able to view products by category	Pass	-
	Edit Category		
	1. Able to edit the category name	Pass	-
	Delete Category		
	1. Able to delete category and all products under the category	Pass	-
Shopping List Category	*Add Shopping List*		
	1. Able to add a new shopping list item	Pass	-
	Edit Shopping List		
	1. Able to edit shopping list item details	Pass	-
	Check and Uncheck Item		
	1. Able to check an unchecked shopping list item	Pass	-
	2. Able to uncheck a checked shopping list item	Pass	-
	Delete Shopping List		
	1. Able to delete shopping list item	Pass	-

Testing date	20/3/2021		
Start Time	6:06 PM	**End Time**	6:25 PM
Tester Name	Ong Sook Yeng		
Test Module	**Test Scenario**	**Pass/ Fail**	**Comments**
Login	1. Able to create an account and sign in with one of the sign in methods	Pass	-
	2. Able to verify the email address	Pass	-
	3. Able to verify the phone number	Pass	-
Product Management	*Add Product*		
	1. Able to add a product image to the product	Pass	-
	2. Able to add a product name to the product	Pass	-
	3. Able to add a category to the product	Pass	-
	4. Able to add an expiry date the product	Pass	-
	5. Able to add a barcode entry to the product	Pass	-
	6. Able to select the number of stock for the product	Pass	-
	7. Able to add alerts and reminder to the product	Pass	-
	Edit Product		
	1. Able to view any product details	Pass	-
	2. Able to edit any product details	Pass	-
	Delete Product		
	1. Able to delete any product	Pass	-
	Search Product by Name		
	1. Able to search existing product by name	Pass	-
	Search Product by Barcode		

	1. Able to search product by barcode	Pass	-
Category Management	*Add category*		
	1. Able to add a new category	Pass	-
	View Product by Category		
	1. Able to view products by category	Pass	-
	Edit Category		
	1. Able to edit the category name	Pass	-
	Delete Category		
	1. Able to delete category and all products under the category	Pass	-
Shopping List Category	*Add Shopping List*		
	1. Able to add a new shopping list item	Pass	-
	Edit Shopping List		
	1. Able to edit shopping list item details	Pass	-
	Check and Uncheck Item		
	1. Able to check an unchecked shopping list item	Pass	-
	2. Able to uncheck a checked shopping list item	Pass	-
	Delete Shopping List		
	1. Able to delete shopping list item	Pass	-

Testing date	21/3/2021		
Start Time	9:18 PM	**End Time**	9:45 PM
Tester Name	Er Sin Ming		
Test Module	**Test Scenario**	**Pass/ Fail**	**Comments**
Login	1. Able to create an account and sign in with one of the sign in methods	Pass	-
	2. Able to verify the email address	Pass	-
	3. Able to verify the phone number	Pass	-
Product Management	*Add Product*		
	1. Able to add a product image to the product	Pass	-
	2. Able to add a product name to the product	Pass	-
	3. Able to add a category to the product	Pass	-
	4. Able to add an expiry date the product	Pass	-
	5. Able to add a barcode entry to the product	Pass	-
	6. Able to select the number of stock for the product	Pass	-
	7. Able to add alerts and reminder to the product	Pass	-
	Edit Product		
	1. Able to view any product details	Pass	-
	2. Able to edit any product details	Pass	-
	Delete Product		
	1. Able to delete any product	Pass	-
	Search Product by Name		
	1. Able to search existing product by name	Pass	-
	Search Product by Barcode		

	1. Able to search product by barcode	Pass	-
Category Management	*Add category*		
	1. Able to add a new category	Pass	-
	View Product by Category		
	1. Able to view products by category	Pass	-
	Edit Category		
	1. Able to edit the category name	Pass	-
	Delete Category		
	1. Able to delete category and all products under the category	Pass	-
Shopping List Category	*Add Shopping List*		
	1. Able to add a new shopping list item	Pass	-
	Edit Shopping List		
	1. Able to edit shopping list item details	Pass	-
	Check and Uncheck Item		
	1. Able to check an unchecked shopping list item	Pass	-
	2. Able to uncheck a checked shopping list item	Pass	-
	Delete Shopping List		
	1. Able to delete shopping list item	Pass	-