# Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking

By

Tan Su Hua

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS) BUSINESS INFORMATION SYSTEMS

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2023

**UNIVERSITI TUNKU ABDUL RAHMAN**

# REPORT STATUS DECLARATION FORM

**Title**: Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking

**Academic Session**: JUNE2023

I TAN SU HUA
**(CAPITAL LETTER)**

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

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

Verified by,

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

**Address**:
No.20, Jalan Kota Harmoni 27/97,
Seksyen 27, 40400, Shah Alam,
Selangor.

Su Lee Seng
Supervisor's name

**Date**: 11/9/2023 **Date**: 11/9/2023

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

**UNIVERSITI TUNKU ABDUL RAHMAN**

Date:

**SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS**

It is hereby certified that ***Tan Su Hua*** (ID No: ***21ACB00596*** ) has completed this final year project/ dissertation/ thesis* entitled "Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking" under the supervision of Mr. Su Lee Seng (Supervisor) from the Department of Digital Economy Technology, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

Suhua

_____

Tan Su Hua

# DECLARATION OF ORIGINALITY

I declare that this report entitled "**Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : Suhua

Name : Tan Su Hua

Date : 11/9/2023

# ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Mr Su Lee Seng who has given me this bright opportunity to engage in this development of financial mobile application. A million thanks to him who has constantly given me suggestions, ideas, motivation, encouragement and support. It is worth mentioning that he put lots of effort into helping his student such as sending relevant articles related to my final year project topic. This has motivated me to progress my project in a better way.

Besides that, I would also like to thank my precious family and friends who show unconditional support to me. Throughout the whole period of Final Year Project 2, I faced lots of ups and downs. I would like to say thank you very much to those who had consistently been there for me and supported me physically and mentally.

# ABSTRACT

Managing receipts has always been considered as a hassle and troublesome task, especially when dealing with a large number of receipts. However, keeping track of receipts is crucial for various purposes, including tax audits, as evidence of purchases, and facilitating returns or refunds. This project aims to simplify the process of managing finances and receipts by integrating Optical Character Recognition (OCR) technology into a financial planning application.

OCR is the short form for Optical Character Recognition. It is a technology that was powered by Machine Learning algorithms like Convolutional Neural Networks (CNN) whereby it allows extraction of text from images or documents in a blink of eye. With the implementation of OCR technology, information from receipts, such as merchant names, dates, and amounts, it can be effortlessly retrieved, making receipt management significantly more accessible and time efficient. Then, the extracted data is displayed to the user and concurrently stored in cloud storage for future reference, such as tax audits and proof of purchase or refunds.

Besides receipt management, the application offers an array of functions to assist individuals in effectively planning and managing their finances and have better vision on their financial status in real time. Users can track their daily financial transactions, establish budget plans, and more. These functions encourage financial responsibility, a critical factor in achieving one's financial goals. Furthermore, user inputs are transformed into insightful graphical representations such as bar chart and pie chart, providing users with a comprehensive view of their financial status. Moreover, users can also generate personalized financial reports according to their preferences. For instance, producing an expense report for a specific period within few steps and these reports can be serve as a vital decision-making tool for users in managing their finances.

In summary, this project addresses the challenges of receipt management and other issues by leveraging OCR technology within a comprehensive financial planning application. It aims to empower users to efficiently handle their finances, fostering better financial habits and enabling more informed financial decisions.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| *OCR* | Optical Character Recognition |
| *API* | Application Programming Interface |
| *SME* | Small Medium Enterprise |
| *JSON* | JavaScript Object Notation |
| *SVMs* | Support Vector Machines |
| *HMMs* | Hidden Markov Models |
| *CNNs* | Convolutional Neural Networks |
| *GPS* | Global Positioning System |

# Chapter 1: Introduction

Financial planning is a continuous process of creating a flexible yet thorough strategy to manage your finances. This is to ensures that you are moving towards achieving your long or short-term financial goals and prevent yourself from encountering any unwanted financial burden [1]. Based on an article by Deloitte [2], financial planning consists of five major steps: defining your financial goal, gathering and analyzing your financial information, developing and reviewing of the financial plan. However, it is an undeniable fact that the process of financial planning is dull and boring, and some people often do not know where and how to start. Financial planning has become exceptionally important whenever people enter adulthood. As things has become more complicated, people have to start bearing with lots of responsibilities such as bill paying, feeding family, paying taxes, investment, etc [3]. In the end, all these responsibilities are highly correlated with one particular thing which is money. Not only that, managing receipts is also equally important as managing finance, as each and every receipt will provide us the necessary information such as the amount, merchant name, date spent, and act as evidence of goods exchange. Most importantly, they are used during the process of tax auditing. Thus, practicing a good habit of financial planning is highly suggested or recommended for everyone.

In this modern digitalized era, many things have been digitally transformed such as the way people communicate, make payments, planning their finances, and many more. All of these functions can be accessed through the use of mobile applications on the individual's smart phone. Imagine planning and managing your finance and receipts manually and without any help from technology and gadgets; it would be very tedious and inefficient. Therefore, this final year project aims to develop a personal financial application using technology such as Optical Character Recognition (OCR) that allows every user to have a better and seamless experience in managing their finances and receipts. It is noticeable that there are plenty of personal finance management or planning mobile applications existing in the market. However, most of them lack a significant feature which is the documenting, tracking, analyzing receipts and making use of the collected receipts to generate a personalized insightful report. This feature may be helpful for the process of auditing income tax and aiding in achieving financial goal.

## 1.1 Problem Statement and Motivation

**1) Most individual facing problem on keeping track of their finances.**

The first problem statement is most people having problem keeping track of their finances such as their spending and earning. According to a news article by The Edge Malaysia [4], one of the main reasons Malaysians are unable to repay debts and loans is due to the lack of financial planning. Financial planning consists of several steps, and one of the very first steps is to acknowledge the daily cashflow of an individual. This helps understand the latest financial situation of an individual. However, many of us simply do not know where and how to keep track of our finances effectively, especially those who are weak in mathematics and statistics. This is totally understandable, as there are no mandatory financial planning courses offered in our high school, college, and university [5]. Hence, this would result in having a financial burden in the future. On the other hand, there are people who excel in managing and planning their finances due to their education or personal background. However, most of them are busy and do not have sufficient time to finish their everyday tasks. Hence, these people may often neglect or forget to keep track of their day-to-day finances. As a result, without a proper personal financial management system, it may increase their risk of getting any unnecessary financial burden and problems such as overspending, late payments, increasing loans and debts, etc. So, they may require a longer time to achieve their financial goals, such as early retirement, starting a business, funding for emergency situations, and others. Furthermore, an article has proven that [3], having a proper financial planning has positive correlation to your health. The reason behind it was because people who have proper tracking and planning of their money do not have to worry about having insufficient money to spend in the future. As a result, these group of people will have a lesser stress and anxiety tendency, and a positive mental well-being. Whereas those who do not have a proper financial planning and tracking will often lead to overspending and unable to repay their debts. Hence, causing them to have a high stress level and in worst scenario people may get depression or even suicide to end the suffering.

As mentioned previously, keeping track of our daily finances is extremely vital, not only because it helps us to achieve our financial goals, but it also gives us a bigger picture of a person's financial status and stability. Therefore, my primary motivation to accomplish this final year project is to aid people who struggle a lot in keeping track, planning and managing their day-to-day cash flow in fast and convenient manner. So, by creating this final year project it should be able to help more people to keep track of their finances effectively and efficiently.

**2) Individual often lack efficient and effective management of receipts for tax audit purposes.**

Moving on to the second problem statement which is people often lack the habits of managing their receipts for tax audit purposes effectively and efficiently. Most people have neglected the process of managing receipt because some of them believes that the presence of receipts is very redundant, cumbersome, and non-environmentally friendly. However, the truth is that managing financial documents such as receipts is equally important as managing your finance. This is because the receipts can be used as an evidence or proof of goods purchased whenever refunding of the purchased goods, claiming insurance, aid in budgeting, etc [6]. Not only that, but keeping receipts is important especially during the process of tax audit purpose. In order to experience a seamless tax audit process, we have to provide necessary supporting financial documents such as receipts to prove that you have claimed that specific amount of tax to reduce your declared income tax annually [7]. However, there is a downside of keeping physical receipt which is the ink on the receipts will fade away as the time goes by. Hence, this would make the receipts not visible and usable after several weeks or even days. Without having the receipts presented in the complete form it would also affect the process of tax auditing or good refunding.

As a result, this led to my second motivation to accomplish this final year project which is to help people to manage their paper receipts. This can be done by scanning the receipt into the application and then the receipts will extract necessary information and store it to the cloud. So, that the user can easily retrieve the receipts whenever they need it. Moreover, the application will also be able to help the user to categorize their receipts, so that it ensures a smoother and more effective financial planning.

**3) Most individual often lack real time data visualization on their current financial status.**

The final problem statement is that people often absence of real time data visualization ability on their current financial status. Data visualization can be difficult sometimes especially when there is lots of data and you need to visualize it in real time. However, the needs of data visualization are very essential as people, as 90% of information are effectively absorbed by human brain are in visual format such as images, graph, diagram, etc [8]. Whereas the remaining 10% of information that are written in non-graphical or visual format are often forgotten and neglected. This is because our human brain was not made for absorbing information that are written in textual format [9]. Another research [10] by University of Minnesota found out that, human brain process visual sixty thousand faster than textual information. Hence, by visualizing our financial data in graphical format helps us to gain insights which may be beneficial whenever we are making a financial decision.

Therefore, this led to my final motivation of developing this application, which is to help people to display their current financial status with the use of graphical representation such as pie charts, bar charts, or even line charts. This is to let them to have a bigger picture and better acknowledgement on their current financial status. So, that they can know whether they are overspending or on the right track towards attaining their financial goals.

## 1.2 Project Objectives

The following are the three major project objectives which aligns to the problem statement stated above respectively.

**1)To enable individuals to effectively and conveniently plan their financial matters**

The first objective is to help users plan their financial matters effectively and conveniently. Financial planning is very essential and can be beneficial in lots of ways, such as helping individuals to achieve their financial goals, improve their living standards, and prepare for emergency situations [11]. In order to achieve this objective, the proposed application should allow users to keep track of their finances and set up a budget plan.

By recording and tracking their daily financial transactions, it helps them to acknowledge their current spending and earnings of a day, week, month, and year. This information is important and should be retrieved easily because it helps the users to have a bigger view of how money has been spent and earned during a specific frame of time. So, it would help the user in making better financial decisions and lead to financial goal achievement. For instance, by acknowledging the total spending and earning, the individual is able to know how much they have spent and left or whether they have overspent or underspent their money. If they have overspent, they can acknowledge these issues at an early stage and seek ways to prevent this issue from getting even worse.

Not only that but having a budget planning function in the application is also very important when it comes to planning a person's finance effectively. As budget planning helps individuals set a limit on how much he or she can spend during a period of time. Hence, this can prevent and control the individual from overspending and then ensure that they can achieve their financial goals accordingly.

**2)To conveniently and effectively manage receipts with the implementation of OCR**

Furthermore, the second objective, which is to conveniently and effectively manage receipts with the integration and usage of OCR technology. The process of managing receipts is boring and troublesome. People are often frustrated when there are lots of receipts that need to be managed and they are always scatter around here and there. What's worse is that the ink of the receipts may even fade away over time and make it unusable after a certain period of time.

Hence, to achieve this second objective, the application should be able to allow the user to manage their receipts in simple steps.

Having said that, the user can scan their receipts and able to convert their receipts into digital format using the application. Then, the application will start to extract necessary information such as the merchant's name, amount, and date. Afterward, the application will categorize the receipts based on the user's preferences. For instance, after scanning the receipts, the users have to enter the category type of the receipts for the first time. Then, for the subsequent receipts of the same merchant, it will be categorized automatically. By having this function, it helps to shorten the time the user used to categorize their receipts and at the same time, it also helps the user to record their spending transaction.

**3)To provide users with insightful graphical representation and documentation**

Finance is indivisible with data visualization [12]. This led to the third objective of this project which is to provide user with graphical representation and documents that are insightful and meaningful. In order to attain this objective, the application make use of charts and diagram in places like the main user interface where the user can have an overview of his or her current spending and earnings right after entering into the application.

Not only in the main user interface, but the personalized report should also consist of charts such as pie charts showing the income or expenses spend of each category. This allows the user to know how much the user have spent on each category such as the food, grocery, entertainment, beauty, social, etc. By having this function, it helps the users to have a better understanding and bigger picture on his or her current financial situation.

## 1.3 Project Scope and Direction

The output of this final year project is an Android-based mobile application that aims to enable users to plan and manage their financial activities effectively and efficiently to achieve their financial goals. The mobile application consists of several modules and features. Basic transaction tracking, adding transactions by scanning receipts, generating personalized financial reports, setting up budget goals or plans, changing application language and currency, providing customer support, and more are some of the major functions and features of this application.

One of the most fundamental and essential functions of this application is to enable users to record and track their daily income and expenses. Users can manually categorize their transactions into multiple categories such as salary, food, entertainment, gifts, exercise, and more. Additionally, users have the option to add their expense transactions by scanning their receipts. This is accomplished through the utilisation of the Asprise receipt OCR API technology, which detects text from receipt images and identifies patterns to determine specific values. Subsequently, the total income and expenditure of the selected month and year will be displayed on the main user interface in the format of pie and bar charts. These charts will be updated each time a transaction is added or edited. This functionality grants users a better understanding and visualization of their total spending and earnings for a specific month and year. For example, daily income and expenses are represented in a bar chart, while the total income and expenses of a particular month are presented in a pie chart format. Through these graphical representations, users can easily compare and differentiate their total income and expenses.

The mobile application consists of several modules, including basic login and sign-up, changing passwords, deleting accounts, adding transactions via scanning receipts, manually adding transactions, budget planning, generating personalized financial reports about spending and earnings, customer support, changing language and currency, a search function for transactions and receipts, and customizable alarm settings for reminders to add transactions.

**Log in**

1. Existing users need to input their valid email address and password.
2. The system will subsequently verify whether the provided email has been registered and cross-check if the entered password matches the associated email.

**Sign up**

1. Users are required to sign up using their personal information, including first name, last name, email, phone number, and to choose a desired username and password.
2. The application will validate the user's entered email and password to ensure they are valid.

3. A valid email should have an ending such as @gmail.com, @yahoo.com, @outlook.com, and so on.
4. A valid password should comprise a minimum of 6 characters.
5. Additionally, users need to enter their chosen password twice during registration to ensure its accuracy.

**Adding transaction via receipt scanning using OCR**

1. The application will prompt the user about whether they want to scan the receipts using the camera or select from the gallery.
2. The application will extract necessary information such as the merchant's name, date, and amount from the receipt, and display it on the next screen.
3. The following screen will showcase the receipt's image, the extracted information, and the receipt's category.
4. Users are allowed to edit the extracted information in case the OCR technology misinterpreted the information.
5. For the initial instance, users need to categorize the receipt themselves. Subsequent receipts with the same merchant's name will be automatically categorized by the application.
6. Following that, the receipts and their information will be uploaded to real-time Firebase and cloud storage, making it easier to retrieve the receipts in the future.
7. The receipts will be accessible on another screen named "View Receipts," enabling users to reference the receipts whenever they require.

**Generate personalized financial report.**

1. The application will generate the financial report based on the user’s choice of type, category, and date range.
2. The user can also choose how they want to view the report, such as transactions only, receipts only, or a combination of both.
3. After the user enters all the necessary input, it will lead them to another screen that will display the financial report.
4. The financial report will consist of:
    - The title.
    - The selected transactions, receipts, or both.

- A pie chart – showing the total for each category.
- The total amount of the displayed transactions.
- A print button that allows the user to convert the report into PDF format.

**Budget Planner:**

1. The application shall prompt the user to input the budget name, amount, type, recurrence, and start date of the budget.
2. Users can select the budget's recurrence from various options such as daily, weekly, monthly, and yearly.
3. The budget type corresponds to available expense categories like Entertainment, Food, Grocery, Exercise, and more.
4. A progress bar will display the remaining amount compared to the initial budget.
5. Whenever the user adds a transaction that fulfills the budget criteria, it will increase or decrease the budget's progress.
6. The application will also send notifications, accordingly, based on the progress of each budget.

**Manually Add Transactions:**

1. The application enables users to manually add new transactions based on:
    - the category (e.g., income or expenses)
    - the type of transaction (such as salary, food, grocery, dividend, ...)
    - the date of the transaction
    - the transaction amounts.
    - Additional notes for the transaction
    - the person involved in the transaction.
2. These transactions will be recorded in real-time on Firebase, allowing for easy retrieval and editing.
3. Users can also edit transactions afterward, and any updates they make will be automatically reflected in the real-time Firebase database.

## 1.4 Contributions

This personal finance management application namely “Financie”, it will make a big difference in society because it lets individuals and businesses plan their money matters in an easy and effective way. This application will help users to reach their financial goals and keep an eye on how they spend. This is extremely important since many countries are having a tough time and fluctuation with their economies. Lots of countries are dealing with high prices, and people are still trying to recover from bad times due to Covid, which have led to much unemployment.

The existence of this application will help users plan and monitor their spending and money plans in a better way. Besides that, it is helpful during the process of individual tax audit. As it helps users to keep track of and save their receipts so they can show proof for what they have declared in a fast and appropriate way. Not only that, but the user can also help others to track their transactions as there is an attribute that allow users to input the owner of the transaction. Young adults can also get lots of benefits from this application. This is because it helps them to see what they are spending every day, week, month, or year so they know where their money goes. It also lets them make a budget plan, so they can put money aside for things they need. For instance, they could make a plan for unexpected things like accidents or hospital visits, so they are ready when they needed.

Moreover, Financie is not just good for the individual, but it is also good for businesses like small startup and Small Medium Enterprise (SME). This business can use this application to do their work better and more efficiently. For example, by using the major function of this application - scanning receipts, it saves time when looking at how money moves in a business and allow a smoother and seamless audits process. As most of the process are highly automated and all they have to do is just scanning the receipts. Likewise, tracking money and making graphs can be part of a business's important insights in their financial report.

Lastly, this application can also help a country to grow in a holistic way. By using the application, each person can learn about managing and planning their finance like setting up budgets. When people have their finance planned properly, they are less worried about their future. Hence, they tend to have a more positive mindset and attitudes towards their life. As a result, fewer people have finance problems, it can even make the country safer because there

will be lesser crime. The individuals can have a more stable and happy life in a way it also boosts the quality of life of a nation.

## 1.5 Report Organization

In chapter 1 of this report, it covers the introduction, 3 major issue and objectives, scope and direction and contribution. In chapter 2, some of the similar popular finance applications are listed and compared. Then, in chapter 3, it shows how the system works, including diagrams that explain things like how people will use it and what the system will do. Chapter 4 talks about what's been done so far, like how the application looks and how it's used. Finally, the last chapter, which is chapter 5, wraps up everything from all the chapters and provide suggestion for improvement within this application.

# Chapter 2 Literature Review

## 2.1 Review of Technology

### 2.1.1 Optical Character Recognition

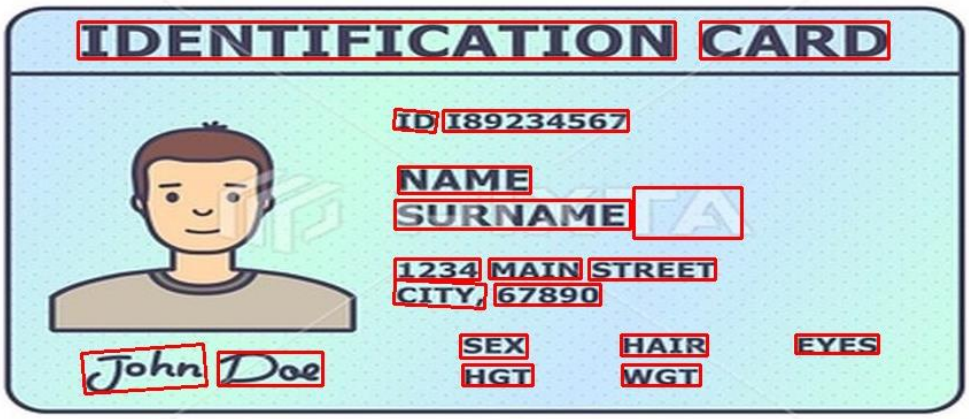



**Figure 2.1.1 Identification card with bounding box**

**(Image source: https://printjourney.com/product/id-cards/)**

Optical Character Recognition also known as character recognition or shortened as OCR has been existed and applied in various fields for many years [13]. So, what is OCR? In general, it is a computer vision technique that recognizes and convert texts from images into a text machine readable format [14] On the other hand, in a more technical terms, OCR is the process of capturing, recognizing and retrieving characters from each bounding boxes into a string or JavaScript Object Notation (JSON) format. As shown in figure 2.1.1, there are lots of bounding boxes surrounding each word. It is an imaginary box that surrounds each text objects in an image and it also state the position of the object [15]. This process is mandatory and essential in the image processing phase, as it defines the X and Y position of each object detected which makes the application of Machine Learning (ML) even simpler. One of the most common applications of OCR in our daily life nowadays can be found when using our smartphones. For instance, individual can select the text from the images by simply by holding the text for a few seconds and then the user can choose to copy the text and paste it wherever they feel like it. This can be done in less than 5 seconds depending on the length of the size. Hence, it helps us to save time and be more productive. An example can be selecting and extracting the bank account number from the screenshots into a text format input for the bank application. However, OCR often uses several machine learning algorithms such as Support Vector Machines (SVMs), Hidden Markov models (HMMs), Convolutional Neural Networks (CNNs) and many more to capture texts from images [16].

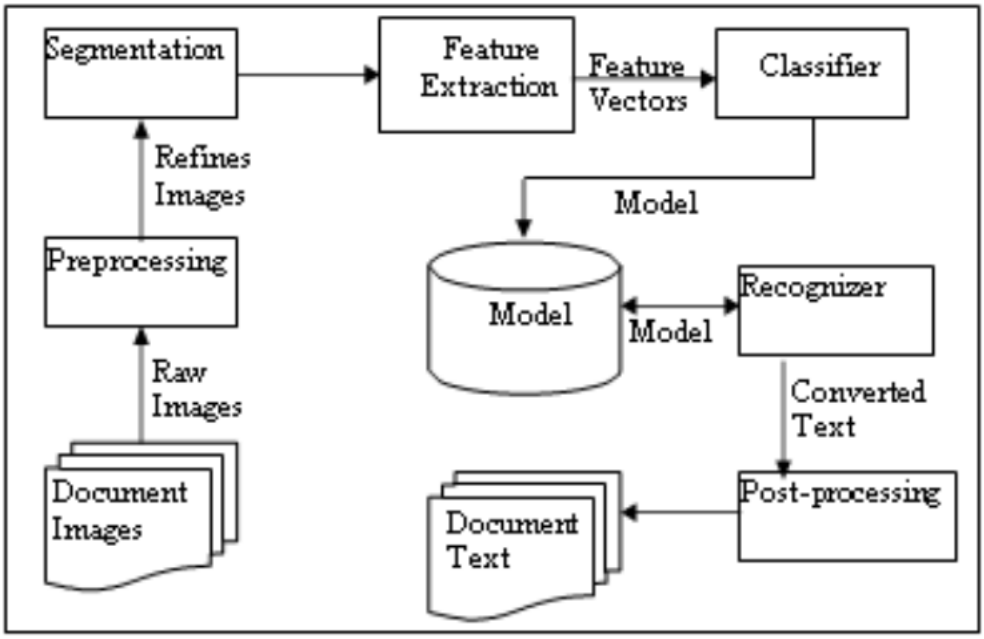



**Figure 2.1.2 – Overview of how OCR technology works**

OCR not only can detect unstructured printed text characters but also handwritten characters and convert it to structured machine readable format [17]. Figure 2.1.2 highlighted an overview of how OCR technology works published by [13]. After inputting documents and images, several actions were taken in the image preprocessing process such as de-skew, transforming to gray scale, filter and reducing noises of the characters from the image. Moving on, there are three important steps in segmentation process such as line segmentation, word segmentation and character segmentation. This segmentation process is essential as it would help to determine the next crucial process of this system which is feature extraction. In feature extraction the system will identify important patterns from the identified characters. The selected feature should be unique and can easily distinguish one character from another. Then, the process of classification will use the selected feature from the segmented characters rather than the original characters from images. Next, the system will start to apply statistical technique or machine learning algorithm.

Finally, one last step to require before delivering the output results. This steps often involve in error detection, and correction or enhancement. However, it is knowledgeable that it is impossible to reach 100% accuracy of identified characters and there will always be minor errors here and there. All in all, the quality of OCR output result highly depends on the quality of the image inputted [18]. For instance, the result may be less accurate when trying to apply OCR on old and broken printed documents, unless there is sufficient and huge amount of training set that enable to train the models effectively.

## 2.2 Review on the existing application

### 2.2.1 Spendee [19]

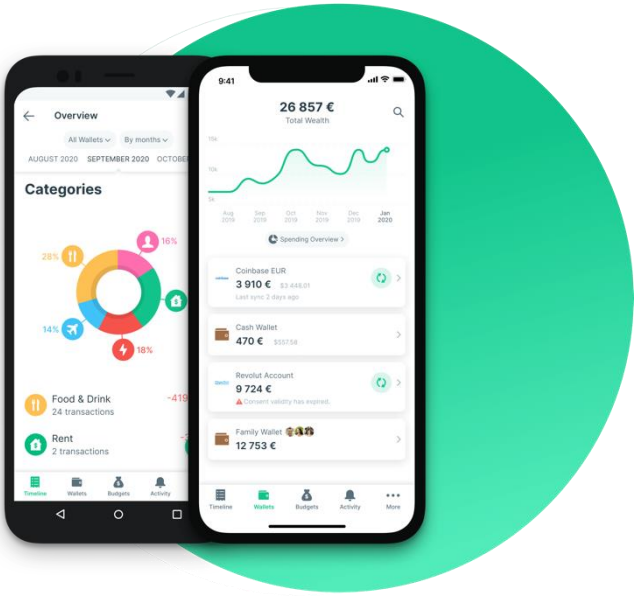



**Figure 2.2.1 Spendee's User Interface**

Spendee is a well-known budget and money tracker application that can be found and used on both mobile application and websites. This application is invented and founded by a group of adults in year 2017. About 2 years later, the application has started to become popular in the financial technology or also known as the FinTech industry. Ever since then, Spendee has received lots of prestigious awards and achievements such as being featured in the Apple App Store and Google Play, winning the Mobile UX 2017 award, etc. The reason why this application able to establish a success and a stable position in the industry is because it has lots of unique functions and features that competitors do not have. For instance, one of the unique functions is that it allows their user to track and manage their crypto wallet. This function is especially beneficial for people who owns cryptocurrency. Not only that, but the application also strongly emphasizes the "Track-Analyze-Budget" rules [20] and this can be done by tracking their finance by connecting to their incumbent's account. By connecting to the user's bank account, the user can easily analyze his or her spending behaviors and automatically categorize the transaction based on the reduction from the banks. Then, the user is able to gain insights and take action accordingly. For example, if the user acknowledges that he or she has been spending too much money on beauty products. Then he or she can set up a budget to limit his or her spending. Another amazing feature about Spendee is that it will give a comprehensive recommendation on the maximum amount a user can spend each day during the budget period. This feature gives the user a real time insight about how much he or she can spend and left. Therefore, it aids the user to attain his or her budget goals effortlessly and effectively.

**Spendee basic functions**

1. **Monitor cash flow** – keep an eye on how money goes in and out on a daily, weekly, monthly and yearly basis.
2. **Personalized budget planner** – able to set up a budget plan based on user preference such as the category user want to budget for, budget name, budget plan icon, etc. However, the user can only create a budget plan on a basic account.
3. **Scheduled transaction** – this allows the user to record their transaction on a predefined date and time.
4. **Import and export of data** – enable user to download and restore data to and from the application.

**Spendee additional features**

1. **Display the overview of all wallets** – showing the amount left in each wallet with the use various graphical representation and percentages.
2. **Provide real time financial guidance and advice** – for every budget created the user will be notified on how much he or she can spend each day in order to achieve the budget goal. Each spending that is related to the budget plan will affect the money that can be spend each day by the user.
3. **Manage e-wallet and crypto wallet** – whereby the user can connect their e-wallet or crypto wallet to the application to track and record any transactions made using these 2 wallets.
4. **Combine of categories from income or expenses** – the users are allowed to merge 2 or more expenses or income categories together.
5. **Bulk editing** – enable user to edit categories, date, and notes of multiple transaction altogether with minimal efforts.

**Strengths of Spendee**

1. Support multiple various type of electronic devices as it is available in mobile and web application.
2. Allow user to connect with their existing bank account for recording and tracking of the transactions made.
3. Provides various customizations that may be useful for some users who love personalization experiences such as multiple currencies available, creating a new category or subcategory for expense or income, schedule transaction, etc.

4. Strong and enhanced security system like using biometric authentication to enter into the application and utilize an encrypted bank connection and secured storage cloud.
5. Able to connect and manage an e-wallet or crypto wallet.
6. Tutorials and guidance are provided when the user first launch the application, to ensure that the users are less confused when using the application.

**Weaknesses of Spendee**

1. More restriction for the free version in term of feature and functionality offered.
2. Limited choice of bank for connection.

## 2.2.2 Money manager Expense & Budget [21]

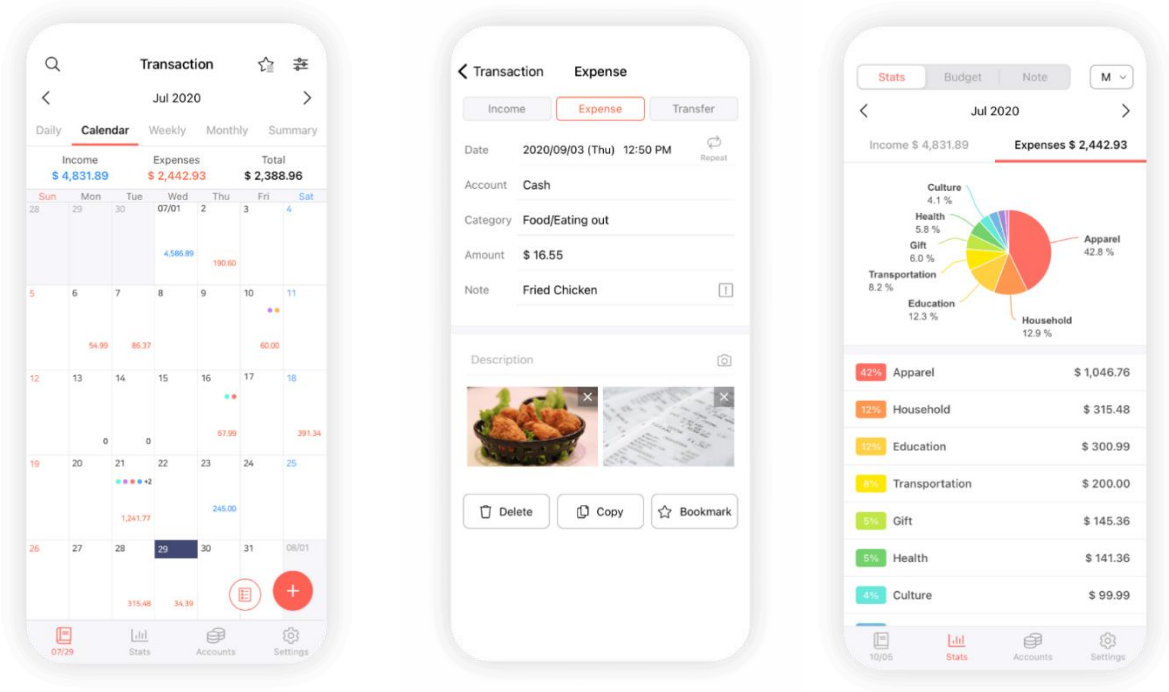



**Figure 2.2.2 Money Manager's User Interface**

Another popular money management software that is available on both Google Play and App Store is Money Manager Expense and Budget. Realbyte Inc is the firm behind this application, and it is based in South Korea. It is an excellent and easy to use application with a rating of 4.7 out of 5.0 because it is simple to use. In addition, it includes functions and features like reporting, budgeting, and the recording of transactional data. By all means, the major objective of this application is to make difficult things, such household costs, simpler. This is to help the novice with little or no literacy in finance or technology to feel less overwhelmed or confused when using the application. One of the highlights of this application is the use of unique graphical representation such as calendar and double entry form. The calendar view will allow individual to view their transaction recorded in a calendar form. Whereas the double entry form will display the user with the debit and credit amount in each transaction.

**Basic Functions of Money Manager**

1. **Budget planning** – allow user to create a personalized budget plan.
2. **Record their spending and earnings** – the transactions will also be categorized.
3. **Generate financial report** – The daily, weekly, monthly and yearly spending and earnings reports will be generated.

4. **Import and export data** – enable user to download and restore the data to and from a .csv excel format.

**Additional Features of Money Manager**

1. **Calendar visual** — display all the incoming and outgoing transactions in calendar format.
2. **In app calculator functions** – to let the users to calculate simple calculations within the application.
3. **Create new subcategories** – allow the user to customize a new subcategory from a main category.
4. **Double entry bookkeeping** – to have a better overview and outline on the money spend and earn.
5. **Sub currency feature** – multiple currency is offered to the users and the users can choose one of it as a sub currency.

**Strengths of Money Manager**

1. Give users the option to customize things like the app icon, app mode (dark or light mode), the ability to attach media to transaction data, the start weekday, etc.
2. Make use of a variety of interactive graphs, such as line charts, pie charts, calendar views, etc.
3. Support many languages and currencies.
4. Make data import and export possible.

**Money Manager's weaknesses**

1. Offer a limited choice of banks for bank sync.
2. The PC version is inefficient since it takes a long time to sync data with the user's PC through Wi-Fi.
3. Limited features compared to other existing applications in the market.
4. Although the application is easy to use, however it doesn't give new users any instructions and guidance. This can cause some of the new users to be confused.

### 2.2.3 Expensify [22]

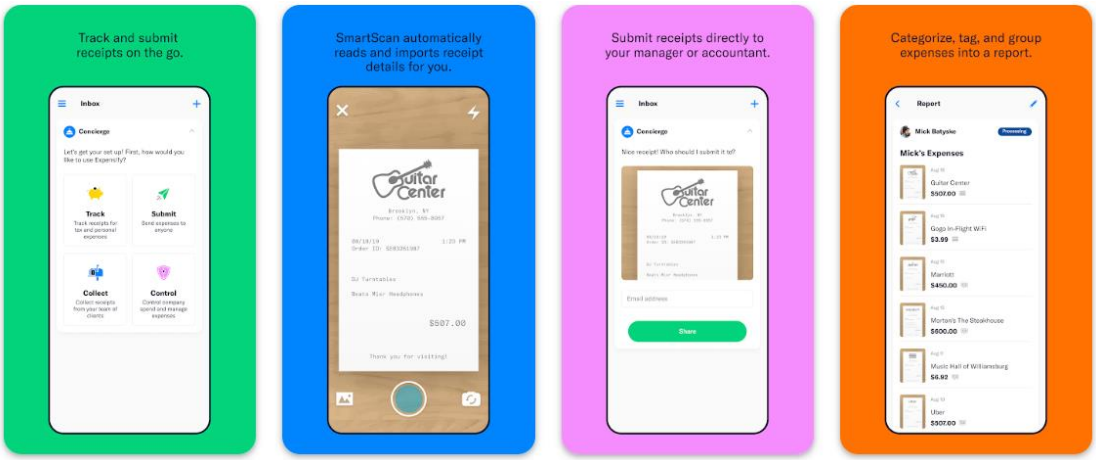



**Figure 2.2.3 Expensify's User Interface**

A travel and spending management application called Expensify focuses mostly on scanning receipts, tracking company expenses, paying bills, and managing personal and travel expenses. This is an application that was invented by a group of students from the University of Michigan started it in 2008, and ever since then it has won several awards including App Partner of the Year, Innovation Partner of the Year, Top rated expenditure management software, and many more. This application is useful for people who frequently misplace their paper receipts since it allows them to save and organize their receipts. Furthermore, it serves as a platform to monitor and guarantee the validity of a group of people's financial records, such as receipts, making it well suited for large corporations to manage their employees' spending and reimbursement.

**Basic Functions of Expensify**

1. **Track spending or expenses** – monitor the daily input and output of cash or transactions made. So, users have a better control on the company spending and management on expenses.
2. **Scan and manage receipts** – it allows the users to scan and manage their personal or company receipts. Moreover, it acts as a platform for people to share and collect of receipts. This allows a better management of receipts as the receipts can be retrieve from cloud with the use of filter easily.
3. **Real time alerts** – a notification will be sent to the user to update him or her of any changes has been made.

**Additional features of Expensify**

1. Automated bill payment - set up an automatic payment for invoices and amount that you specify previously. Expensify will automatically pay and shut off any bills that satisfy the criteria you specify.
2. Invoice Generation - automated generation and processing of invoices, payment collection, and much more
3. Track mileage – with the use of Global Positioning System (GPS), it helps the users to calculate the mileage expenses easily and effortlessly.
4. Offline mode – able to use the application without accessing the Wi-Fi.

**Strengths of Expensify**

1. Enable more effective and efficient management of group costs and employee reimbursement.
2. Having both a mobile and online application makes it very accessible and available.
3. 24/7 customer care provided by an AI chatbot.
4. Work best for the business since the majority of the functions are simple to automate and improve on a regular basis.

**Weaknesses of Expensify**

1. The Smart Scan feature provide an inaccurate result inconsistently.
2. No tutorials or guidance are provided. Users must learn how to use this application by themselves.
3. Small selection of banks and less than 20 banks have partnered with it.
4. Unable to import and export of data.

### 2.2.4 Money Lover [23]

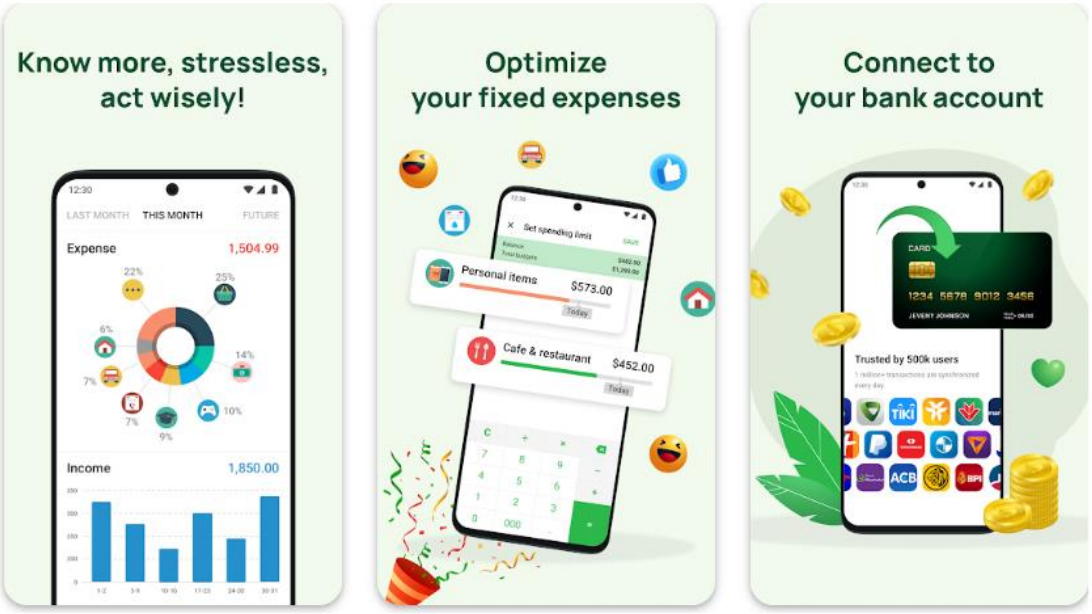



**Figure 2.2.4 Money Lover's User Interface**

For individuals who dislike traditional budgeting, the money management software Money Lover is an excellent substitute. It was created in 2011 by a Vietnamese company called Finsify. This application combines several tasks, including tracking, budgeting, reporting, and planning, into a single application. Due of the uniqueness of user interface and availability of functions and features in this application, people love it as it has received a 4.9 out of 5.0 rating. In addition, Money Lover also received variety of achievements and honors, including App of the Day and 100K+ 5-star ratings. One of the unique features of this application is the travel mode whereby it helps the user to record the total amount spent on that specific trip. Hence, the users do not have to calculate the expenses later on and it also give a bigger picture on the user's spending in a real time form.

**Basic functions of Money Lover**

1. **Keep track and categorize the transaction** – it helps the user to manage their finance and give them an overview of their spending in real time.
2. **Budget plans** – like most of the other similar applications, it also has a budget plan that allow users to set a constraint on their spending in order to avoid overspending or saving for a rainy day.
3. **Bill reminders** – notification will be sent to the users to remind them to pay for their bills.

4. **Scheduled bill payment for recurring transactions** – once the user has set up the scheduled bill payment, the application will automatically help the users to pay for their bills by deducting from their connected incumbents.

**Additional features of Money Lover**

1. **Able to hide the total balance of the user with hash symbols** – this can improve the customer experience of the user whenever using the application, as their balance is not shown to anyone unless they click it to view it.
2. **Travel mode** – this can easily help the users to calculate their total spending on their trip.

**Strengths of Money Lover**

1. Offer their services in smartphone app, a web app, and even an apple watch app. Thus, making it highly accessible
2. Instructions and advice were given before the new users can enter into the application.
3. Utilize Secure Sockets Layer (SSL) encryption for high security.
4. Accessible in a variety of languages and currencies

**Weaknesses of Money Lover**

1. Limited choice of bank for connection
2. Not allowed to import or export of data.

## 2.3 Comparison between existing and proposed application

| Features | Spendee | Money Manager | Expensify | Money Lover | Proposed application |
|---|---|---|---|---|---|
| Real time tracking income and expenditure | ✓ | ✓ | ✗ | ✓ | ✓ |
| Budget planning | ✓ | ✗ | ✗ | ✓ | ✓ |
| Able to scan receipt | ✗ | ✓ | ✓ | ✗ (Paid version only) | ✓ |
| Support multiple currency | ✓ | ✓ | ✓ | ✓ | ✓ |
| Support multiple languages | ✓ | ✓ | ✗ | ✓ | ✓ |
| Able generate customizable financial report | ✗ | ✗ | ✓ | ✓ | ✓ |
| Search function | ✓ | ✓ | ✗ | ✓ | ✓ |
| Use graphical representation | ✓ | ✓ | ✗ | ✓ | ✓ |
| Customer support | ✓ | ✓ | ✓ | ✓ | ✓ |
| Notification for overspending, record transactions | ✓ | ✓ | ✓ | ✓ | ✓ |

| | | | | | |
|---|---|---|---|---|---|
| Customizable notification to remind user to record transaction | ✗ | ✓ | ✗ | ✓ | ✓ |
| Require sign up or log in | ✓ | ✗ | ✓ | ✓ | ✓ |
| Cost | Free/Paid | Free | Free/Paid | Free/Paid | Free |

**Table 2.3 Comparison between the existing and proposed application**

According to table 2.3, it shows a comparison of the properties and features of the current and proposed application. 4 popular financial management application such as Spendee, Money Manager, Expensify and Money Lover are chosen to be compared with the proposed application. All of the existing applications can be found in the Google Play Store and Apple App Store. The functions and features such as recording expenses and income, scanning receipts, budgeting, generate personalized report and others listed on the table are what the proposed application able to achieve.

# Chapter 3: System Methodology/Approach

## 3.1 System Methodology

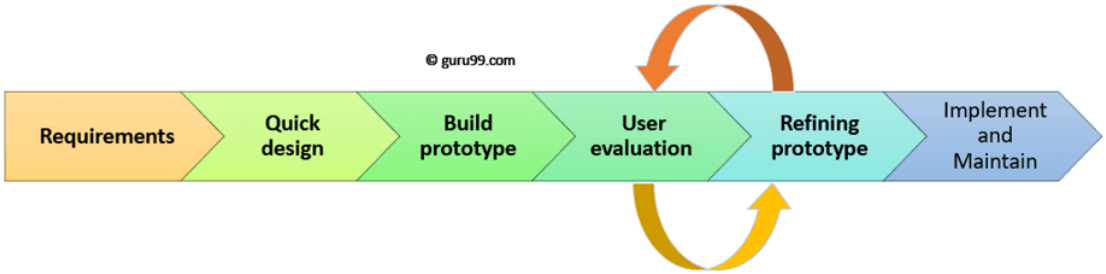



**Figure 3.1 – Prototyping methodology (source: guru99.com)**

The application is designed according to a well-known methodology, namely the prototyping methodology. It has six major phases, which include requirements gathering, quick design, building a prototype, user evaluation, prototype refinement, and deployment, also known as implementation and maintenance.

In the first phase, which is requirements gathering, user requirements are collected. This can be achieved by conducting surveys or interviews with target markets to gain insights into the target audience's preferences. This requirement gathering phase is essential as it helps us establish our goals and direction for achieving the project's objectives.

Moving on to the second phase, which involves creating a quick sketch of the application's design. The sketch should be simple and easy for most of the target audience to understand, and it is expected to evolve based on feedback from the target audience and personal research. Once the design aligns with the current requirements and needs, the next phase is to build a prototype based on the previously sketched design.

After this phase, it is crucial to gather feedback and evaluations from potential users to understand their thoughts and impressions. Once we've collected necessary and important feedback from customers, the prototype should be refined and enhanced once again. The fourth and fifth steps are iterative and continuous until the system successfully meets user requirements.

Following a continuous process of prototype testing and enhancement, it is time for implementation. The prototype should be transformed into a usable and complete application that satisfies all necessary user requirements. Additionally, system testing should be conducted on the system regularly. This ensures that the application's performance remains consistent and can be enhanced to accommodate new user requirements.

## 3.2 System Design Diagram

### 3.2.1 System Architecture Diagram

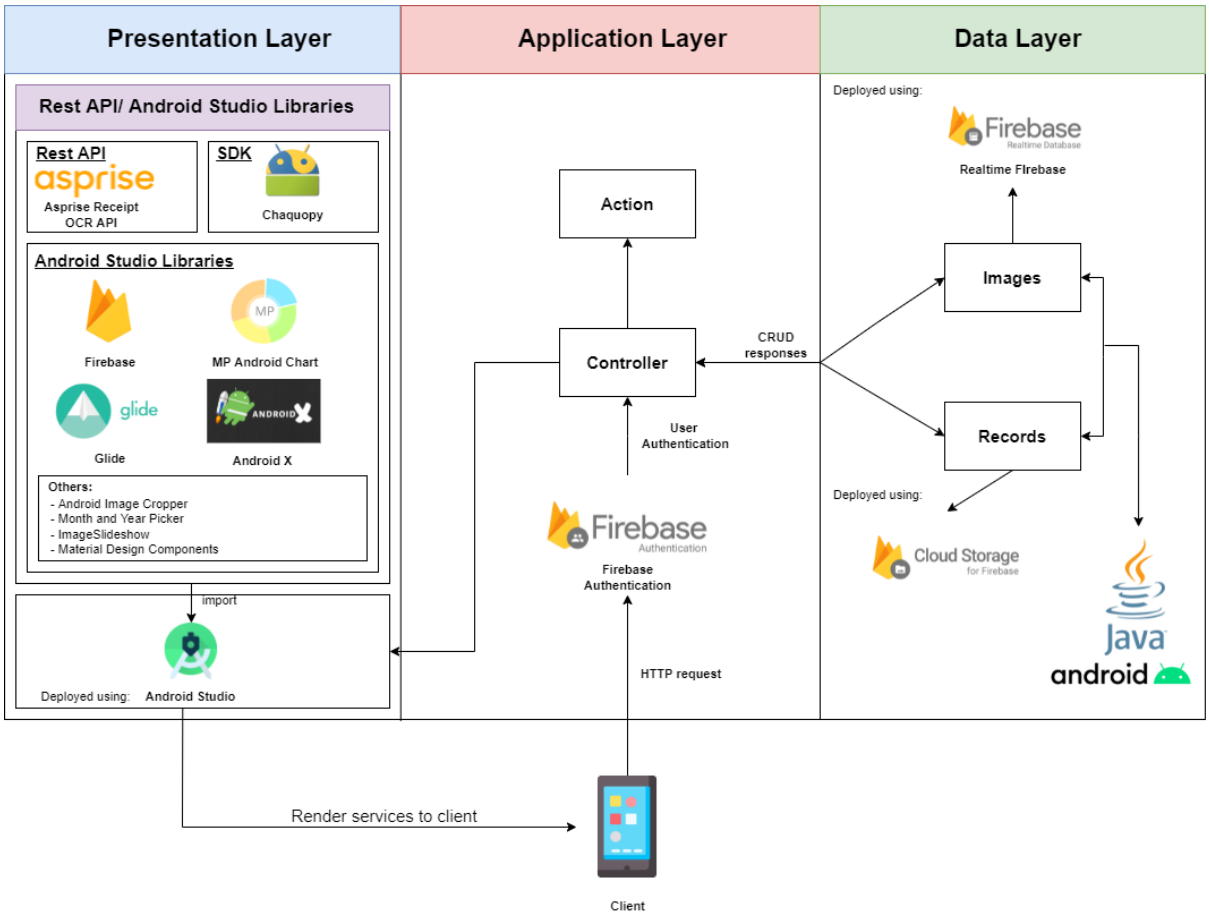



**Figure 3.2.1 – System Architecture Diagram**

The diagram above illustrates the system architecture of the proposed mobile application, which consists of three main layers: the presentation layer, application layer, and data layer whereby the client/user represent the mobile devices. In the presentation layer, it manages all user interactions with the application, making it the most user-accessible layer that defines the application's overall appearance and usability. The diagram also displays the APIs, SDK and Android Studio libraries used in developing the application, including Firebase, Glide, AndroidX, MP Android Chart, and more, all deployed using Android Studio. Moving to the second layer, the application layer, this is where the core functions are implemented and coded. The client sends HTTP requests to Firebase Authentication for user authentication. After successful verification, the user can fully utilize the application, and the controller handles CRUD (Create, Read, Update, Delete) responses for records or images stored in Realtime Firebase and Firebase Cloud Storage. An example of how this system architecture works: Suppose the user is already verified with Firebase Authentication. When the user wants to view

or add an expense transaction in the presentation layer's user interface, the controller in the application layer sends a CRUD request to the data layer. The data layer then performs the requested action (in this case, create and read operations only) and sends the CRUD responses back to the application layer. The application layer processes these responses and displays the results to the user in the presentation layer.

### 3.2.2 Use Case Diagram and Description

**Figure 3.2.2 Use Case diagram of the application**

The relationship between the actors and the application is depicted in the figure above. In this system, there are four actors: the user, Firebase Authentication, Firebase Storage, and Real-Time Firebase. Each actor has various use cases, which will be discussed in detail below.

| **Use case** | Login | **ID** | 1 |
|---|---|---|---|
| **Aim** | To allow user to enter to the application | **Importance Level** | Moderate |
| **Actor** | User and Firebase Authentication | | |
| **Trigger** | When the user clicks on the enter button on the login screen | | |
| **Pre-condition** | Connected to WIFI and owns an account | | |
| **Main flow** | 1. Prompt the user to enter email and password.<br>2. Check whether the email and password are correct and exist in the Firebase or not.<br>3. Direct to the main screen<br>4. | | |
| **Alternative flow** | Invalid email or password<br>1. Alert the user that the password or email is incorrect.<br>2. Remain on the login screen. | | |

**Table 3.2.2.1 – Use Case Description of Login Module**

| **Use case** | Register Account | **ID** | 2 |
|---|---|---|---|
| **Aim** | To allow the new user to register an account and check whether the user has an account already | **Importance Level** | Moderate |
| **Actor** | User and Firebase Authentication | | |
| **Trigger** | When the user clicks on the enter button on the register page | | |
| **Pre-condition** | Connected to WIFI | | |
| **Main flow** | 1. Prompt the user to enter his or her details such as full name, contact number, email, password, etc.<br>2. Check the validity of the email and the correctness of the password (after prompting the user to enter the password twice).<br>3. Store the user details in Firebase.<br>4. Direct to the main screen | | |
| **Alternative flow** | Invalid email<br>1. Alert the user that the email is unavailable or invalid (already registered or in a wrong format)<br>2. Remain in the register screen.<br><br>Invalid password | | |

| | |
|---|---|
| | 1. Alert user that the password entered does not fulfil the requirement (at least 6 characters)<br>2. Remain on the register screen. |

**Table 3.2.2.2 – Use Case Description of Register Account Module**

| | | | |
|---|---|---|---|
| **Use case** | Add Transaction | **ID** | 3 |
| **Aim** | To allow users to decide whether to add transaction manually or by scanning receipts | **Importance Level** | High |
| **Actor** | User and Real-Time Firebase | | |
| **Trigger** | When the user clicks on the add button on the main page | | |
| **Pre-condition** | The user must be logged in and have WIFI connected | | |
| **Main flow** | **Add Transaction by scanning receipt.**<br>1. Press the scan receipt button.<br>2. Users select the option of selecting picture.<br>2a. from gallery<br>2ai. Select receipt image.<br>2b. from camera<br>2bi. Take receipt image from default camera.<br>3. Resize the receipt image (optional).<br>4. System displays the captured receipt details.<br>5. Check the captured receipt details and image of receipt.<br>6. Edit the details if necessary.<br>7. System ensures that every attribute is not empty except for notes.<br>8. Categorize the receipt (optional)<br>9. Enter the submit button.<br>10. System uploads the data to Real-Time Firebase.<br>11. Return back to Main Module | | |
| | **Add Transaction manually.**<br>1. Press the manual add button.<br>2. Selects the type of the transaction such as Food, Grocery, Entertainment, Salary, Borrow, etc.<br>3. Enter the transaction details such as amount, date, etc.<br>4. System ensures that all attributes is not empty.<br>5. Press the submit button.<br>6. System uploads the data to Real-Time Firebase.<br>7. Return back to Main module. | | |

**Table 3.2.2.3 – Use Case Description of Add Transaction Module**

| **Use case** | Edit Transaction | **ID** | 4 |
|---|---|---|---|
| **Aim** | To allow the user to edit their transaction made previously. | **Importance Level** | Moderate |
| **Actor** | User and Real Time Firebase | | |
| **Trigger** | When the user clicks on the top right corner edit button on the transaction details page. | | |
| **Pre-condition** | The user must be logged in, has WIFI connected and has at least one transaction recorded. | | |
| **Main flow** | 1. System pops out dialog box showing that the user has enter into editing mode.<br>2. The user can start to edit the transaction details.<br>3. The user presses the top right corner icon once finish editing.<br>4. System will pop out another dialog box showing that all changes has been saved.<br>5. System will also update the data from Real-Time Firebase. | | |
| **Alternative flow** | None | | |

**Table 3.2.2.4 – Use Case Description of Edit Transaction Module**

| **Use case** | View Transaction | **ID** | 5 |
|---|---|---|---|
| **Aim** | To allow the user to view their transaction history | **Importance Level** | Low |
| **Actor** | User and Real Time Firebase | | |
| **Trigger** | When the user clicks on the transaction history on the main page | | |
| **Pre-condition** | The user must be logged in, has WIFI connected and has at least one transaction recorded. | | |
| **Main flow** | 1. System displays all the transaction in a recycler view.<br>2. User clicks on each transaction.<br>3. System displays more details about the transaction user selected. | | |
| **Alternative flow** | None | | |

**Table 3.2.2.5 – Use Case Description of View Transaction Module**

| **Use case** | Create Budget | **ID** | 6 |
|---|---|---|---|
| **Aim** | To allow the user to initiate a budget plan | **Importance Level** | High |
| **Actor** | User and Real Time Firebase | | |
| **Trigger** | When the user presses the top right corner add icon on the budget plan page | | |
| **Pre-condition** | User must be logged in and connected to WIFI | | |

| | |
|---|---|
| **Main flow** | 1. Prompt the user to enter the name and amount of the budget.<br>2. Prompt the user to enter the expenses type for budget.<br>3. Prompt the user to enter the start date and recurrence period.<br>4. The system displays the amount left to spend.<br>5. Notification enables when the user overspending.<br>6. Save the budget plan into the Real-Time Firebase. |
| **Alternative flow** | None |

**Table 3.2.2.6 – Use Case Description of Create Budget Module**

| | | | |
|---|---|---|---|
| **Use case** | View Budget | **ID** | 7 |
| **Aim** | To allow the user to view their budget plan | **Importance Level** | Low |
| **Actor** | User and Real Time Firebase | | |
| **Trigger** | When the user enters the budget plan icons on the "More" screen | | |
| **Pre-condition** | User must own an account, connected to WIFI and owns at least one budget. | | |
| **Main flow** | 1. System displays all the budgets in a recycler view.<br>2. User clicks on each budget.<br>3. System displays more details about the budget of the user selected. | | |
| **Alternative flow** | None | | |

**Table 3.2.2.7 – Use Case Description of View Budget Module**

| | | | |
|---|---|---|---|
| **Use case** | Generate report | **ID** | 8 |
| **Aim** | To allow the user to generate a personalized income or expenses report on certain type of transaction and a specific duration. | **Importance Level** | High |
| **Actor** | User and Real Time Firebase | | |
| **Trigger** | When the user clicks on the generate report at the "More" page | | |
| **Pre-condition** | The user must be logged in, has WIFI connected and has at least one transaction recorded. | | |
| **Main flow** | 1. The system asks the user to pick the kind of report they want, like total expenses, total income, or both.<br>2. Based on the first step, the system shows different options for the user to choose from, such as all expenses, food expenses, entertainment expenses, etc.<br>3. The system then asks the user to pick a start and end date for the report.<br>4. The user also needs to choose the report's content, like transactions only, receipts only, or both.<br>5. To move forward, the system makes sure all the required information is filled in.<br>6. Once the user provides all the necessary details, the system creates the report. | | |

| | |
|---|---|
| | 7. The system matches the user's choices with the real-time data from Firebase to generate the report.<br>8. If the user wants, they can print the report by pressing the print button.<br>9. The system generates the report in .PDF format and saves it in the user's phone storage. |
| **Alternative flow** | None |

**Table 3.2.2.8 – Use Case Description of Generate Report Module**

| | | | |
|---|---|---|---|
| **Use case** | View user detail | **ID** | 9 |
| **Aim** | To allow the user to see their profile | **Importance Level** | Low |
| **Actor** | User and Real Time Firebase | | |
| **Trigger** | When the user clicks on the user icon on the main screen or press on the account settings on the "Settings" page. | | |
| **Pre-condition** | The user must be logged in and connected to the WIFI. | | |
| **Main flow** | 1. The user can scroll the screen to view more information about the user information. | | |
| **Alternative flow** | None | | |

**Table 3.2.2.9 – Use Case Description of View user detail Module**

| | | | |
|---|---|---|---|
| **Use case** | Edit user details | **ID** | 10 |
| **Aim** | To allow the user to edit user details that he or she has previously entered | **Importance Level** | Low |
| **Actor** | User and Real Time Firebase | | |
| **Trigger** | When the user press on the top right corner icon on the user details' user interface | | |
| **Pre-condition** | The user must be logged in and connected to the WIFI. | | |
| **Main flow** | 1. The user press on the top right corner button to start editing mode.<br>2. System pops out a dialog box showing the user has entered into editing mode.<br>3. The user starts to edit their details such as email.<br>4. Validate the new user details entered.<br>5. Update the new user details in Firebase. | | |
| **Alternative flow** | Invalid new user details<br>1. Display error message.<br>2. The user details remain unchanged.<br>3. Remain in user details' user interface. | | |

**Table 3.2.2.10 – Use Case Description of Edit user detail Module.**

| **Use case** | Change Language | **ID** | 11 |
|---|---|---|---|
| **Aim** | To allow the user to change the language of the system | **Importance Level** | Low |
| **Actor** | User | | |
| **Trigger** | When the user press on the language settings on "Settings" module. | | |
| **Pre-condition** | User must be logged in and connected to WIFI. | | |
| **Main flow** | 1. System displays all languages offered in the system.<br>2. User selects one language by pressing on the selection from the dialog box.<br>3. System returns to the "Main" screen with new language implemented all over the system. | | |
| **Alternative flow** | None | | |

**Table 3.2.2.11 – Use Case Description of Change Language Module**

| **Use case** | Change Currency | **ID** | 12 |
|---|---|---|---|
| **Aim** | To allow the user to change the currency of the system | **Importance Level** | Low |
| **Actor** | User | | |
| **Trigger** | When the user press on the currency settings on "Settings" module. | | |
| **Pre-condition** | User must be logged in and connected to WIFI. | | |
| **Main flow** | 1. System displays all currencies offered in the system.<br>2. User selects one currency by pressing on the selection from the dialog box.<br>3. System returns to the "Main" screen with new currency implemented on the subsequent transaction. | | |
| **Alternative flow** | None | | |

**Table 3.2.2.12 – Use Case Description of Change Currency Module**

| **Use case** | Set Alarm | **ID** | 13 |
|---|---|---|---|
| **Aim** | To allow the user to set an alarm to remind user to record their transaction | **Importance Level** | Low |
| **Actor** | User | | |
| **Trigger** | When the user press on the alarm settings on "Settings" module. | | |
| **Pre-condition** | User must be logged in and connected to WIFI. | | |
| **Main flow** | 1. User selects the time from the dialog box.<br>2. System will send the notification based on the alarm set by the user previously.<br>3. System returns to the "Main" screen. | | |
| **Alternative flow** | None | | |

**Table 3.2.2.13 – Use Case Description of Alarm Settings Module**

## 3.3 Activity Diagram

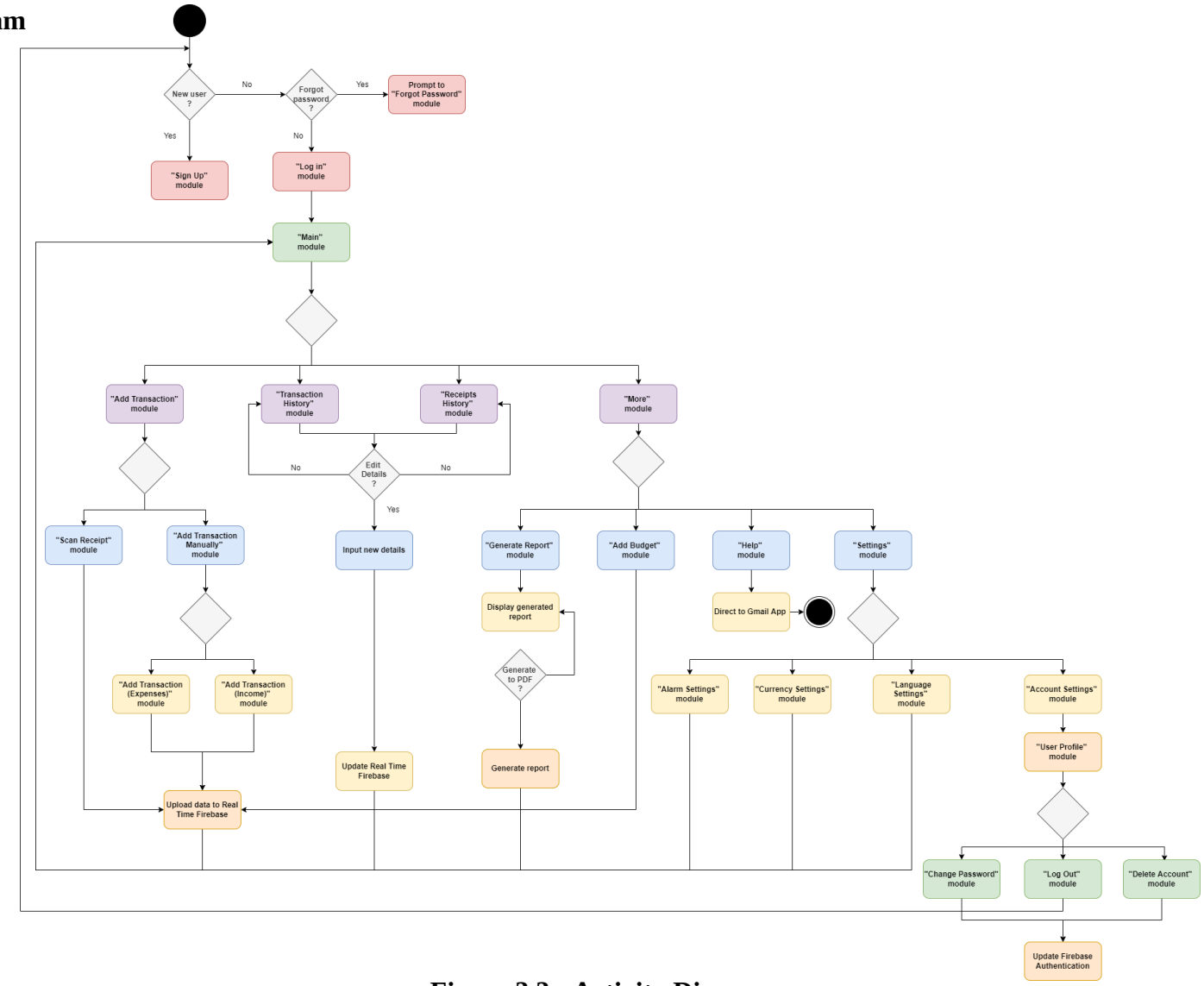



**Figure 3.3 - Activity Diagram**

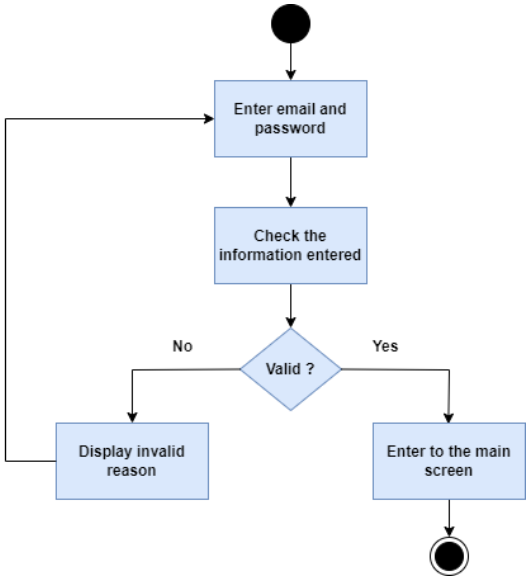



**Figure 3.1.3.1 - Activity Diagram of Login module**

Figure 3.1.3.1 displays the login module of the application. This module is intended for users who have previously registered with the system. Initially, they must input their email and password. The system will verify this information by checking it against Firebase Authentication. If the entered information is valid, the system will then direct the user to the main screen. However, if the user enters incorrect or invalid information, the system will display an error message, notifying the user of the error.

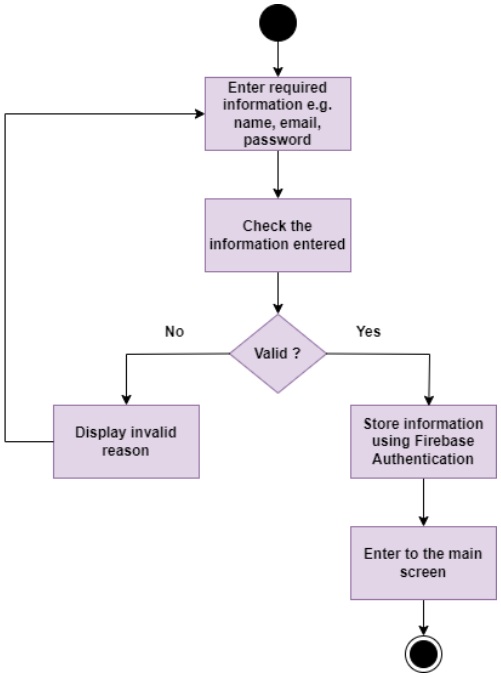



**Figure 3.1.3.2 - Activity Diagram of Register module**

Based on Figure 3.1.3.2, it depicts the activity diagram of the registration module. Users are required to input necessary information such as their full name, email, password, and a

username when registering for an account. After the user enters the information, the system must verify the details provided by the user. If the user enters valid information, an account will be registered using Firebase Authentication and will direct the user to the main interface. However, if the user enters invalid information, the system will display an error message, notifying the user that the entered information is invalid.

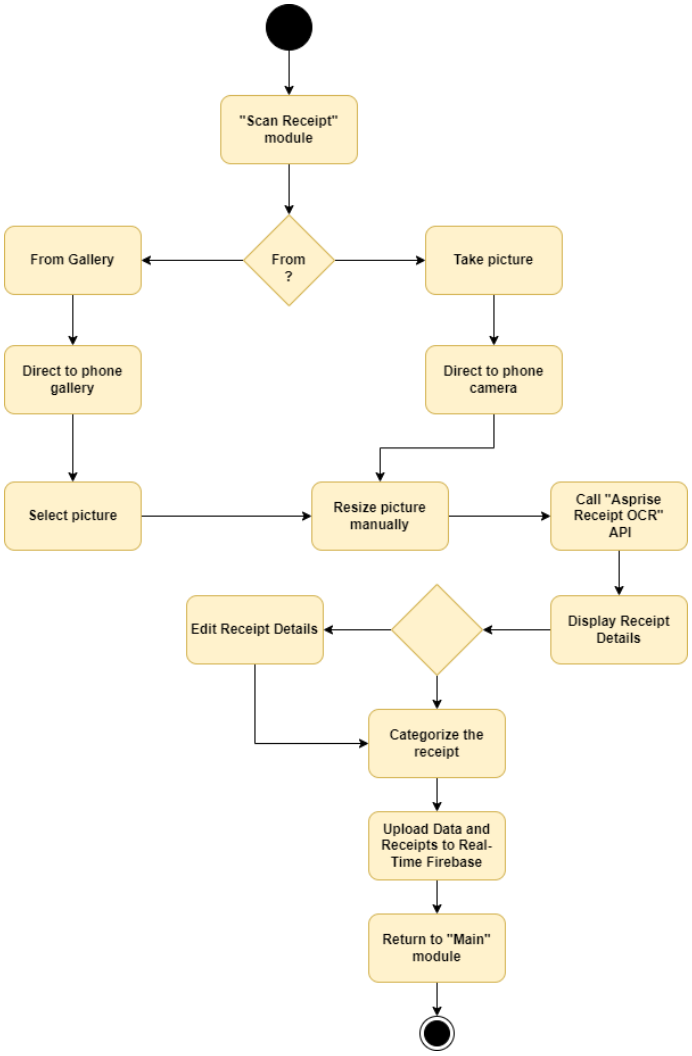



**Figure 3.1.3.3 - Activity Diagram of Scan Receipt module**

This module is the main function of the application, and it operates once the user chooses to add a transaction by scanning a receipt. According to the figure above, the user can choose to scan the receipt by selecting from their phone's gallery or by taking a picture of the receipt. If the user chooses to select a picture from their gallery, the application will direct the user to their local gallery. Then, the user has to select a picture containing the receipt. However, if the user wishes to take a picture of the receipt directly, the application will direct the user to the phone's camera application, where they can take a snapshot of the receipt. After obtaining the

image of the receipt, the user is given the option to resize the image receipt. This is to reduce any background noise from the image and allow for faster optical character recognition (OCR). However, the user can choose to ignore this step if they do not want to resize the image. After that, the application will call an Application Programming Interface (API) known as Asprise Receipt OCR. This API helps the application detect relevant keywords such as the amount, date, merchant name, etc. The application will then display all the detected details, and the user is allowed to edit these details. Since the accuracy of the API and the quality of the picture used for character recognition can greatly affect the result, the user is free to make edits. Furthermore, the user is required to categorize the receipt for the first time. Once the user submits similar receipts from the same merchant, the application will automatically help categorize the subsequent receipts. After the user has reviewed the displayed details, they can press the submit button. This action stores the data in Real-Time Firebase for easy retrieval and reference in the future. Meanwhile, the application will return to the main user interface, allowing the user to continue using the application.

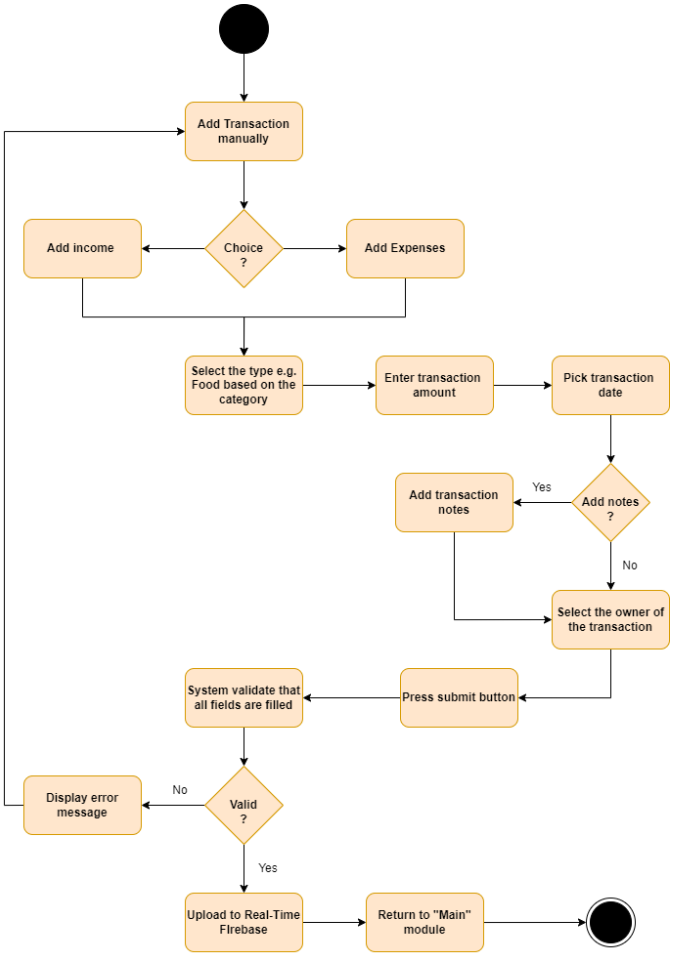



**Figure 3.1.3.4 – Activity Diagram of add transaction manually**

Figure 3.1.3.4 displays the activity diagram for manually adding transactions in the application. This module is also triggered when the user chooses to add a transaction manually. First, the user must select whether to add income or expenses. Regardless of the user's choice, they still need to specify the type of transaction, such as food, groceries, salary, borrowing, etc. Furthermore, the user is required to enter essential details, including the amount, date, notes, and specify the owner of the transaction. Once the user presses the submit button, all the data will be uploaded to Real-Time Firebase. At the same time, the user will be directed back to the main user interface.

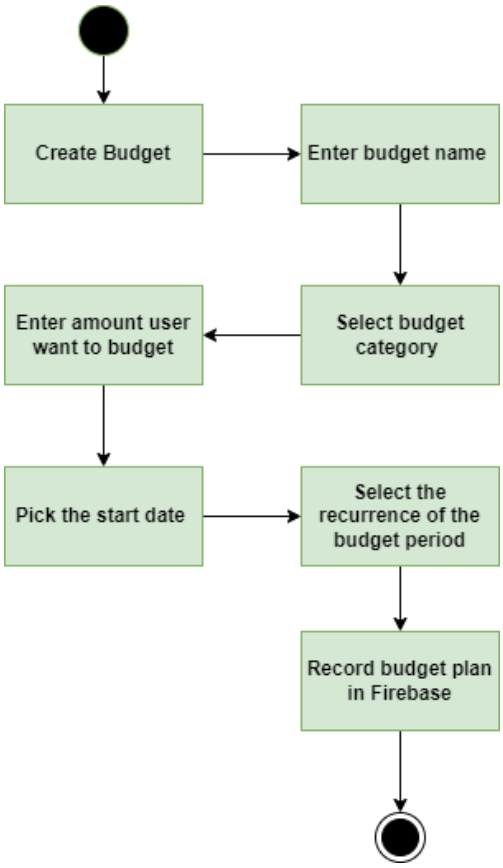



**Figure 3.1.3.5 – activity diagram of add budget**

The figure above shows the activity diagram of adding a budget. Firstly, the user has to enter the budget name based on their preference. Moving on, the user has to select the category of the budget and the user can select more than one type from this category. After that the user has to enter the budget amount, budget start date, and recurrence period. For instance, the user has chosen to budget on his or her food and entertainment expenses. Then, the application will calculate the money user spent on these 2 categories during the budget period. This to ensure that the user do not exceed his or her budget limit. However, if the user does exceed the limit a notification will be received by the user.

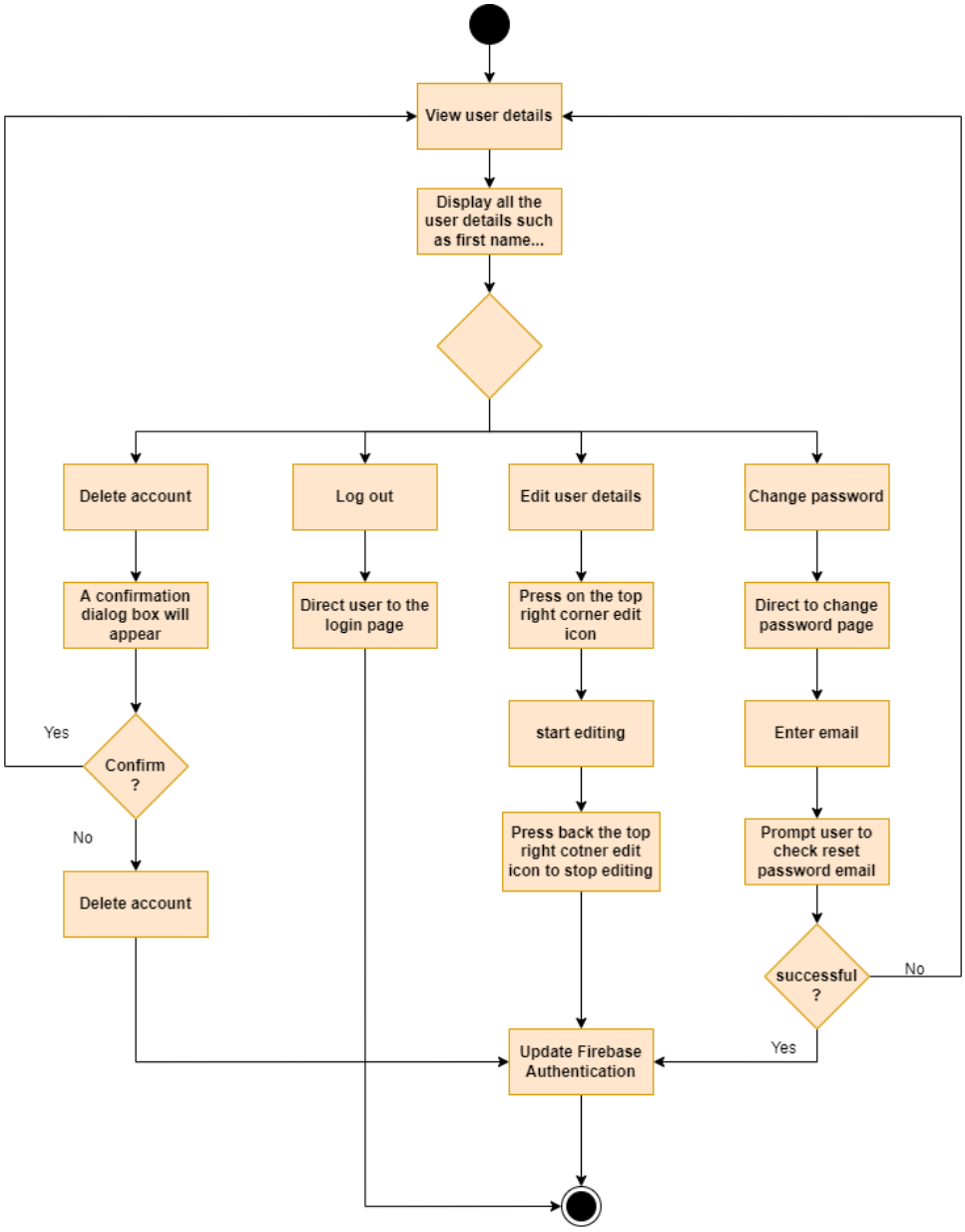



**Figure 3.1.3.6 – Activity diagram of user details**

Figure 3.1.3.6 shows the activity diagram of the user details. Firstly, when the user presses on the top left corner profile picture on the main screen or presses on the accounting settings on the settings page, it will trigger this module. Whenever the user enters this interface, it will display the user's user details such as first name, last name, email, and username. Besides viewing their personal information, the user can also delete the account, log out, edit their user details, and change the account password. If the user chooses to delete the account, a pop-up message will be displayed to prompt the user to confirm the action, which is

to delete the account. If the user chooses yes, then the account will be deleted, and they will return to the login page. However, if the user presses no, then nothing will happen. Moreover, if the user presses the log out button, the user's account will be logged out immediately and directed to the login page as well. Next, if the user chooses to edit their user details, they can press the edit icon on the top right corner of the page to start editing. Once the user finishes editing, they can press the back icon to exit from the editing mode. Lastly, if the user chooses to change their password, the application will prompt the user to enter their email address. This is to ensure that the email the user enters is valid and registered. Moving on, an email will be received by the user. If the user clicks on the email, there is a link that helps the user change their password. Once the password is changed successfully, the user is directed to the login page and can log on to the application using the new password and the same email.

# Chapter 4: System Design

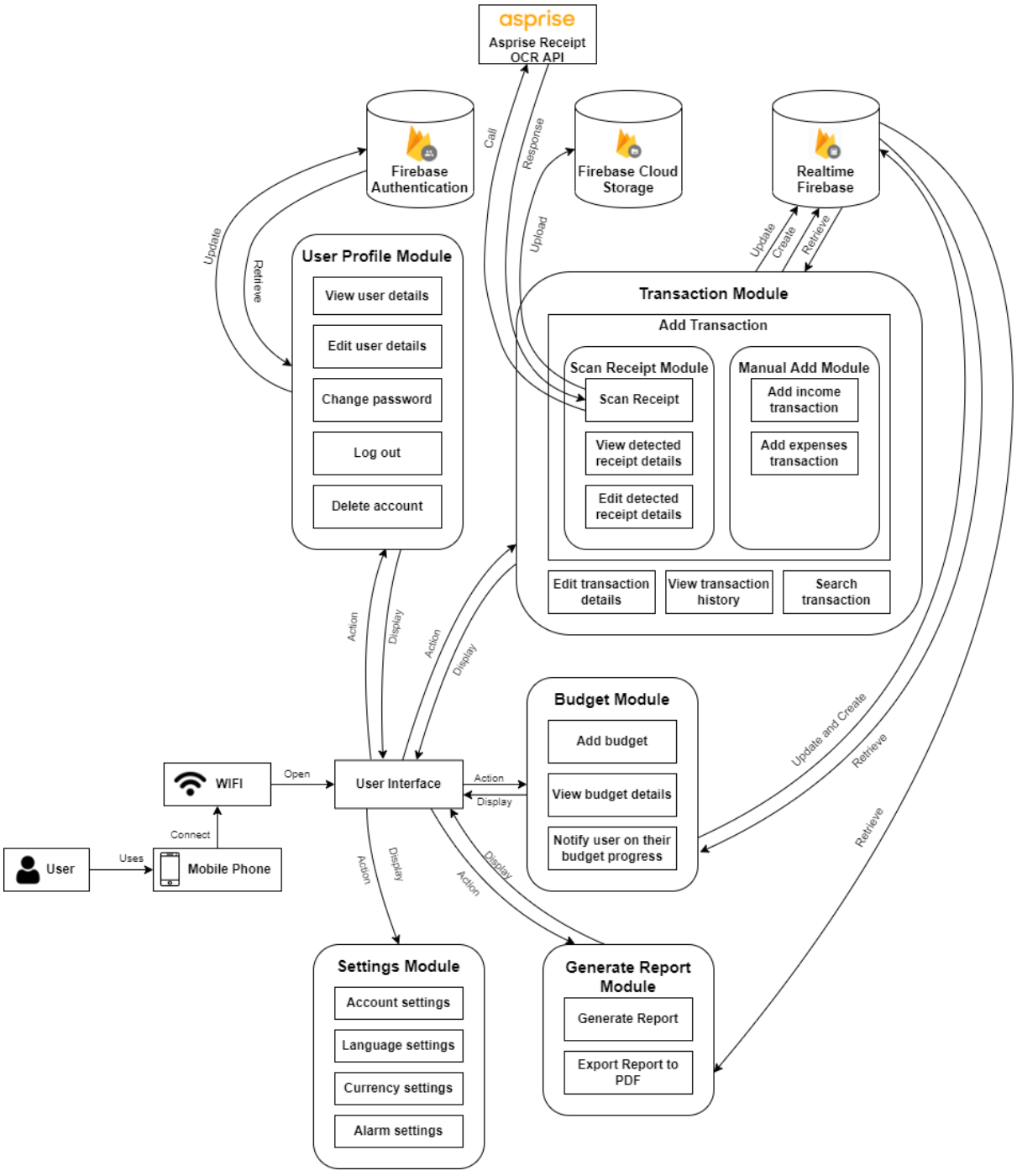



**Figure 4.1 – Block diagram of the system**

According to the figure above, it illustrates the system's block diagram. This system comprises five main modules: the transaction module, budget module, generate report module, settings module, and user details module. Users begin by opening the application on their Wi-Fi-connected phones. The application features straightforward and user-friendly interfaces.

In the User Profile module, the system retrieves the user's personal information, like their first name and username, from Firebase. Users can view or edit their personal details, change passwords, and delete their accounts. Any changes made by the user are updated and validated by Firebase.

Moving on to the transaction module, users can add transactions in two ways: by scanning receipts or adding them manually. If users choose to add a transaction by scanning a receipt, they take a photo of it or select one from their gallery. The system then uses the Asprise Receipt OCR API to extract necessary information from the captured receipt. Simultaneously, the captured receipts are stored in Firebase Cloud Storage, and the extracted information is displayed in the user interface and uploaded to Realtime Firebase. Alternatively, users can manually add income or expense transactions, which are also stored in Realtime Firebase. Additionally, this module allows users to edit transaction details, view transaction history, and search for transactions, all involving the retrieval and updating of Realtime Firebase.

In the budget module, users can create personalized budget plans by filling in necessary fields in the user interface. Once the system collects all the required budget details, it stores them in Realtime Firebase. Users can also view their budget details, including their progress, and the system sends notifications based on budget progress.

In the generate module, similar to the budget module, users provide necessary details for generating personalized reports. The system retrieves the required information from Realtime Firebase such as the total expenses from 1 January 2023 to 31 January 2023 of Alice and presents users with the option to generate reports in PDF format.

Finally, the settings module doesn't interact with Firebase. Instead, it offers functions such as account settings, language settings, currency settings, and alarm settings for users to configure.

## 4.2 System Flow Diagram

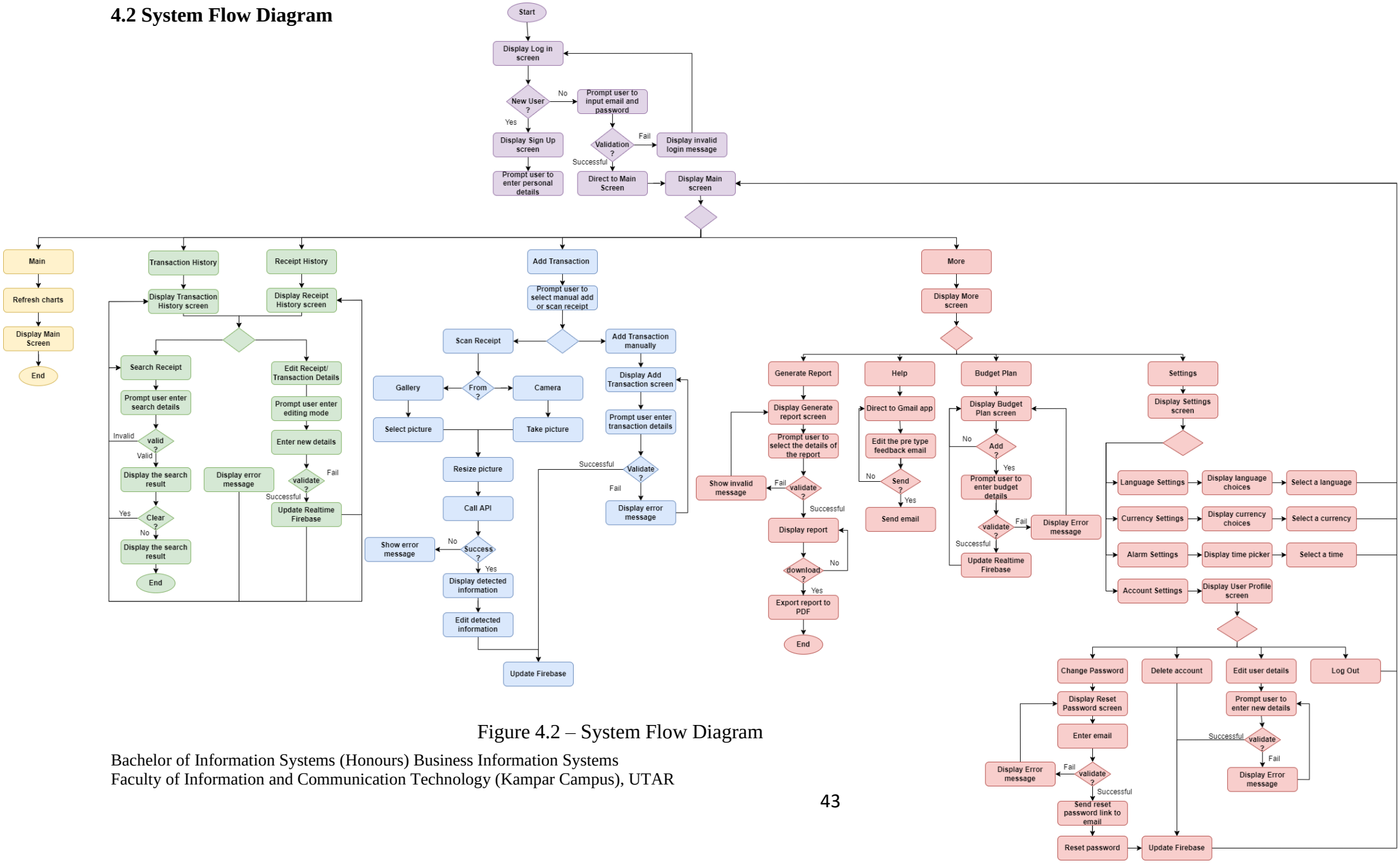



Figure 4.2 – System Flow Diagram

Figure 4.2 displays the system flow diagram for the proposed application, similar in concept to the activity diagram in the previous chapter. However, this figure provides a more detailed look at the module relationships. The main screen offers five main functions that users can trigger: the main module, transaction history module, receipt history module, add transaction module, and more module. Each module involves various actions and decisions.
In the main module, when the user taps the main icon in the bottom navigation bar on the main screen, the page refreshes and returns to the main screen. Users will notice that the pie chart and bar chart are updated.

There are two other icons in the bottom navigation bar on the main screen: viewing transactions and accessing receipt history. These two modules share common functions, allowing users to edit transaction or receipt details and search for transactions or receipts. Both modules involve retrieving and updating Realtime Firebase.

The primary function of the proposed application is to enable users to add transactions. Initially, users are prompted to select their preferred method for adding transactions, such as scanning receipts or manually entering them. The application takes different actions based on the user's choice. If the user chooses to scan a receipt, the application calls an API to display detected information like the merchant's name, amount, date, and more. On the other hand, if the user opts for manual entry, no API is called. Ultimately, all edited and detected information is stored in Realtime Firebase.

Lastly, the "more" screen leads users to different screens based on their choices. However, the primary functions on this screen are the generate report module, budget plan module, and help module. In the generate report module, users can generate personalized reports, such as the total income from 1$^{st}$ May 2023 to 31$^{st}$ May 2023, for Brian. The budget module allows users to create personalized budget plans and receive notifications for any changes made in accordance with the budget. The help module directs users to their phone's email application, where they can write feedback emails to the application support email address. This enables us to acknowledge user feedback and improve the application in future versions. Finally, if users select the settings button on the "more" screen, they are directed to another page where they can modify their account settings, system language, currency preferences, and alarm settings for reminders.

## 4.3 System Flow Description (Code)

### 4.3.1 Login.java

```
        gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN).requestEmail().build();
        gsc = GoogleSignIn.getClient( activity: this,gso);


        loginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String e = email.getText().toString();
                String p = psw.getText().toString();
                //to check email pattern
                if (!Patterns.EMAIL_ADDRESS.matcher(e).matches()) {
                    email.setError("Please enter a valid email");
                    email.requestFocus();
                    return;
                }
                //if email and password are empty
                if (e.equals("") || p.equals("")) {
                    Toast.makeText( context: Login.this,  text: "Please enter all the fields", Toast.LENGTH_SHORT).show();
                } else {
                    //sign in with email and password using the firebase authentication
                    mAuth.signInWithEmailAndPassword(e,p).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            //if login successful, send to main page
                            if (task.isSuccessful()) {
                                sendtoMain();
                            }

                            else {
                                //display error
                                String error = task.getException().getMessage();
                                Toast.makeText( context: Login.this,  text: "Error: " + error, Toast.LENGTH_SHORT).show();
                            }
                        }
                    });
                }
            }
        });


        fp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent( packageContext: Login.this, ResetActivity.class);
                startActivity(i);
            }
        });


        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                signIn();
                Intent i = new Intent( packageContext: Login.this, SignUp.class);
                startActivity(i);
            }
        });
    }

    void signIn(){
        Intent signInIntent = gsc.getSignInIntent();
        startActivityForResult(signInIntent,  requestCode: 1000);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode == 1000){
            Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);

            try{
                task.getResult(ApiException.class);
                navigateToSecondActivity();
            }
            catch(ApiException e){
                Toast.makeText(getApplicationContext(),  text: "Something went wrong", Toast.LENGTH_SHORT).show();
            }

        }

    }

    void navigateToSecondActivity(){
        finish();
        Intent intent = new Intent ( packageContext: Login.this, MainActivity.class);
        startActivity(intent);
    }

    private void sendtoMain() {
        Intent intent = new Intent( packageContext: Login.this, MainActivity.class);
        startActivity(intent);
        finish();
    }
```

**Figure 4.3.1.1 – Login.java**

The figure above shows the login interface of the application. It consists of several functions: signIn(), navigateToSecondActivity(), and sendtoMain(). The signIn() function authenticates and verifies the user's email and password input. Meanwhile, navigateToSecondActivity() and sendtoMain() direct the user to another activity using the Intent() function. Additionally, inside the onCreate() method, variables such as “loginBtn”, “fp”, and “signup” are associated with a function called setOnClickListener(). So, whenever the user presses these variables, they will perform specific actions according to the code inside the function.

### 4.3.2 SignUp.java

```
signupBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        uploadToFirebase();
        String firstname = fname.getText().toString().trim();
        String lastname = lname.getText().toString().trim();
        String username = uname.getText().toString().trim();
        String e = email.getText().toString().trim();
        String pass = psw.getText().toString().trim();
        String confirm_password = cpsw.getText().toString().trim();
        //if the patterns of the email does not match, tells theuser to provide valid email
        if (!Patterns.EMAIL_ADDRESS.matcher(e).matches()) {
            email.setError("Please provide valid email");
            email.requestFocus();
            return;
        }

        //if firstname, lastname, username, contact, email,password, confirm password are empty
        //tell the user to enter all the respective fields
        if (firstname.equals("") || lastname.equals("") ||
                username.equals("")  || email.equals("") ||
                pass.equals("") || confirm_password.equals("")) {
            Toast.makeText( context: SignUp.this,  text: "Please enter all the fields", Toast.LENGTH_SHORT).show();
        } else {

            //if the password entered second time is the same as the first time
            if (pass.equals(confirm_password)) {

                //create an account for the user
                mAuth.createUserWithEmailAndPassword(e,pass).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        //if create user successful
                        if (task.isSuccessful()) {
                            User user = new User(firstname, lastname, username, e, pass);

                            FirebaseDatabase.getInstance().getReference( path: "Users").child(FirebaseAuth.getInstance
                                    ().getCurrentUser().getUid()).child("user details").setValue(user).addOnCompleteListener(new OnCo
                                @Override
                                public void onComplete(@NonNull
                                                       Task<Void> task) {
                                    //if register successful
                                    if (task.isSuccessful()) {

                                        Toast.makeText( context: SignUp.this,  text: "User has been registered successfully!", Toast.LENGTH
                                        sendtoMain();
                                    }
                                    //fail to register
                                    else {

                                        Toast.makeText( context: SignUp.this,  text: "Failed to register! Try again.",
                                                Toast.LENGTH_LONG).show();
                                    }
                                }
                            });
                        }
                        else { //if not successful
                            String error = task.getException().getMessage();
                            Toast.makeText( context: SignUp.this,  text: "Error: " + error, Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            } else { //if password not matching
                Toast.makeText( context: SignUp.this,  text: "Password not matching", Toast.LENGTH_SHORT).show();
            }
        }
    }
});
}
```

**Figure 4.3.2.1 – SignUp.java (1/2)**

```
public Uri getImageUri(Context inContext, Bitmap inImage) {
    ByteArrayOutputStream bytes = new ByteArrayOutputStream();
    inImage.compress(Bitmap.CompressFormat.JPEG,  quality: 100, bytes);
    String path = MediaStore.Images.Media.insertImage(inContext.getContentResolver(), inImage,  title: "Title",  description: null);
    return Uri.parse(path);
}


private void sendtoMain() {
    Intent intent = new Intent( packageContext: SignUp.this, MainActivity.class);
    startActivity(intent);
    finish();
}
//to check whether the user is already logged in
@Override
protected void onStart() {
    super.onStart();
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    if(user != null) {
        sendtoMain();
    }
}


private void uploadToFirebase(){
    String img;

    SimpleDateFormat formatter = new SimpleDateFormat( pattern: "yyyy_MM_dd_HH_mm_ss", Locale.CANADA);

    img = String.valueOf(countimage);
    String fileName = img;
    storageReference = FirebaseStorage.getInstance().getReference( location: "images/"+fileName);

    storageReference.putFile(imageUri)
            .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {

                    pic.setImageURI(null);
                    // Toast.makeText(SignUp.this,"Successfully Uploaded",Toast.LENGTH_SHORT).show();


                     storageReference2.child(img)
                            .getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {
                                    root.child(img).child("img").setValue(uri.toString());

                                }
                            }).addOnFailureListener(new OnFailureListener() {
                                @Override
                                public void onFailure(@NonNull Exception exception) {
                                    // Handle any errors
                                    Toast.makeText( context: SignUp.this,  text: "Failed to download image", Toast.LENGTH_SHORT).show();
                                }
                            });
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {

                    Toast.makeText( context: SignUp.this, text: "Failed to Upload",Toast.LENGTH_SHORT).show();
                }
            });
```

**Figure 4.3.2.2 – SignUp.java (2/2)**

As the name suggests, “SignUp.java” contains the code for signing up in this application. It prompts the user to enter all the necessary field inputs, such as first name, last name, password,

and others. When the user presses the sign-up button, the system validates the user's input. If the input is successful, it stores it in Firebase, as shown in Figure 4.3.2.1.

In this class, various functions are created and called in the “onCreate()” method. These functions include “uploadToFirebase()”, “onStart()”, “getImageUri()”, “askCameraPermission()”, and many more, some of which are displayed in Figure 4.3.2.2.

### 4.3.3 MainActivity.java

```
//for the graph
fragmentManager = getSupportFragmentManager();
fragment = new piechart();
replaceFragment(fragment);

backward.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (fragment instanceof BarChart) {
            fragment = new piechart();
            replaceFragment(fragment);
        } else if (fragment instanceof piechart) {
            fragment = new BarChart();
            replaceFragment(fragment);
        }
    }
});
forward.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (fragment instanceof BarChart) {
            fragment = new piechart();
            replaceFragment(fragment);
        } else if (fragment instanceof piechart) {
            fragment = new BarChart();
            replaceFragment(fragment);
        }
    }
});

my.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        inputMonthYear(view);

    }
});

profile.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent( packageContext: MainActivity.this, UserProfile.class);
        startActivity(i);
    }
});
```

**Figure 4.3.3.1 – MainActivity.java (1/4)**

```
bnv.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()) {
            //if the user click the home icon, redirect to the MainPage
            case R.id.home:
                Intent i = new Intent( packageContext: MainActivity.this, MainActivity.class);
                startActivity(i);
                break;

            //if the user click the plus icon, redirect to the display cooking gas interface
            case R.id.viewTrans:
                Intent i3 = new Intent( packageContext: MainActivity.this, TransHistory.class);
                startActivity(i3);
                break;


            case R.id.addtrans:
                Intent i9 = new Intent( packageContext: MainActivity.this, AddTrans_ManuallyOrScan.class);
                startActivity(i9);
                break;


            //if the user click the profile icon, redirect to the profile interface
            case R.id.viewReceipt:
                Intent i0 = new Intent( packageContext: MainActivity.this, ViewReceipts.class);
                startActivity(i0);
                break;

            case R.id.more:
                Intent i7 = new Intent( packageContext: MainActivity.this, More.class);
                startActivity(i7);
                break;
        }

        return true;
    }
});


viewmore.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent( packageContext: MainActivity.this, TransHistory.class);
        startActivity(i);
    }
});
```

**Figure 4.3.3.2 – MainActivity.java (2/4)**

```
//getting RecyclerView from xml file
final RecyclerView rv = findViewById(R.id.main_rv);

//setting recyclerview size fixed for every item
rv.setHasFixedSize(true);

//setting layout manager to the recyclerview
rv.setLayoutManager(new LinearLayoutManager( context: MainActivity.this));


ref = FirebaseDatabase.getInstance().getReference().child("Users").child(FirebaseAuth.getInstance
        ().getCurrentUser().getUid()).child("user details");
ref.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        String username = snapshot.child("username").getValue().toString();
        wlctxt.setText(username);
        rt.child("expenses").setValue("0");
        rt.child("income").setValue("0");


    }

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

    }
});

root.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        details.clear();
        double income = 0.0, expenses = 0.0;
        int cnt = (int) snapshot.child("transaction").getChildrenCount();

        for (int i = cnt - 1; i > cnt - 6; i--) {
            final String getType = snapshot.child("transaction").child(String.valueOf(i)).child("type").getValue(String.class);
            final String getAmount = snapshot.child("transaction").child(String.valueOf(i)).child("amount").getValue(String.class);
            final String getDate = snapshot.child("transaction").child(String.valueOf(i)).child("date").getValue(String.class);
            final String getPic = snapshot.child("transaction").child(String.valueOf(i)).child("pic").getValue(String.class);
            final String getNotes = snapshot.child("transaction").child(String.valueOf(i)).child("notes").getValue(String.class);
            final String getCategory = snapshot.child("transaction").child(String.valueOf(i)).child("category").getValue(String.class);
            final Long getDateInMillis = snapshot.child("transaction").child(String.valueOf(i)).child("dateInMillis").getValue(Long.class
            final String getMname = snapshot.child("transaction").child(String.valueOf(i)).child("merchant name").getValue(String.class);
            final String getPerson = snapshot.child("transaction").child(String.valueOf(i)).child("for").getValue(String.class);

            Details detail = new Details(getType, getAmount, getDate, getNotes, getPic, getCategory, getMname, getDateInMillis, getPerson

            //adding this user details to list
            details.add(detail);

            //Expenses
            if (getCategory != null) {
                if (getCategory.equals("Expenses")) {
                    expenses = expenses + Double.parseDouble(getAmount.substring(5));
                }
                //Income
                else if (getCategory.equals("Income")) {
                    income = income + Double.parseDouble(getAmount.substring(5));
                }
            }
        }
        rt.child("expenses").setValue(decfor.format(expenses).toString());
        rt.child("income").setValue(decfor.format(income).toString());
        rv.setAdapter(new Adapter1(details, context: MainActivity.this));
    }

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

    }
});
```

**Figure 4.3.3.3 – MainActivity.java (3/4)**

```java
private void replaceFragment(Fragment fragment) {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.main_frame, fragment);
    fragmentTransaction.commit();
}

//the month and year picker
public void inputMonthYear(View view) {
    final Calendar today = Calendar.getInstance();
    MonthPickerDialog.Builder builder = new MonthPickerDialog.Builder( context: MainActivity.this, new MonthPickerDialog.OnDateSetListener()
        @Override
        public void onDateSet(int selectedMonth, int selectedYear) {

            if (selectedMonth == 0) {
                selectedMONTH = "January";
            } else if (selectedMonth == 1) {
                selectedMONTH = "February";
            } else if (selectedMonth == 2) {
                selectedMONTH = "March";
            } else if (selectedMonth == 3) {
                selectedMONTH = "April";
            } else if (selectedMonth == 4) {
                selectedMONTH = "May";
            } else if (selectedMonth == 5) {
                selectedMONTH = "June";
            } else if (selectedMonth == 6) {
                selectedMONTH = "July";
            } else if (selectedMonth == 7) {
                selectedMONTH = "August";
            } else if (selectedMonth == 8) {
                selectedMONTH = "September";
            } else if (selectedMonth == 9) {
                selectedMONTH = "October";
            } else if (selectedMonth == 10) {
                selectedMONTH = "November";
            } else if (selectedMonth == 11) {
                selectedMONTH = "December";
            }

            my.setText(selectedMONTH + " " + selectedYear);
            monthyear = (String) my.getText();

            Intent i = new Intent( packageContext: MainActivity.this, MainActivity.class);
            startActivity(i);

        }
    }, today.get(Calendar.YEAR), today.get(MONTH));

    builder.setActivatedMonth(Calendar.JUNE)
            .setMinYear(2020)
            .setActivatedYear(today.get(Calendar.YEAR))
            .setMaxYear(2060)
            .setMinMonth(Calendar.JANUARY)
            .setTitle("Select the month and year")
            .build().show();
```

**Figure 4.3.3.4 – MainActivity.java (4/4)**

MainActivity.java contains the main user interface functions and code. As shown in figures 4.3.3.1 and 4.3.3.2, it responds to user clicks on variables like “backward”, “profile”, “viewmore”, and others, leading to various actions such as screen transitions and function calls. Additionally, this screen features a bottom navigation bar with five main functions. Each icon press directs the user to a corresponding screen.

Figure 4.3.3.3 illustrates how the system displays data from Firebase in the MainActivity.xml's recycler view. Meanwhile, Figure 4.3.3.4 highlights two crucial functions in this class: “replaceFragment()” and “inputMonthYear()”, “replaceFragment()” replaces the main screen's fragment when the user clicks the “forward” or “backward” button. Conversely, “inputMonthYear()” displays a calendar view, allowing the user to select options for graphical representation based on their preferences.

### 4.3.4 ReceiptScanner.java

```
if (!Python.isStarted()) {
    Python.start(new AndroidPlatform( context: ReceiptScanner.this));
}

Python py = Python.getInstance();

PyObject pyObject = py.getModule( name: "OCR");

PyObject pyobj = pyObject.callAttr( key: "main",  ...args: "/data/data/com.example.myapplication/files/chaquopy/AssetFinder/app/img.jpg

String r = pyobj.toString();
try {
    JSONObject json = new JSONObject(r);
    JSONArray ja = json.getJSONArray( name: "receipts");
    for (int i = 0; i < ja.length(); i++) {
        JSONObject jsonObject = ja.getJSONObject(i);
        ttl = jsonObject.getString( name: "total");
        mn = jsonObject.getString( name: "merchant_name");
        d = jsonObject.getString( name: "date");
        DateFormat originalFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd", Locale.ENGLISH);
        DateFormat targetFormat = new SimpleDateFormat( pattern: "dd MMM yyyy");
        Date date = originalFormat.parse(d);
        formattedDate = targetFormat.format(date);  // 20120821
        c = jsonObject.getString( name: "currency");
        if (c.equals("MYR")) {
            sym = "RM ";
        } else {
            sym = " ";
        }
    }
    total.setText(sym + ttl);
    date.setText(formattedDate);
    merchantName.setText(mn);


} catch (JSONException | ParseException e) {

    total.setText("0");
    date.setText("null");
    merchantName.setText("null");

    Toast.makeText( context: ReceiptScanner.this, r.toString(), Toast.LENGTH_SHORT).show();
    Log.i( tag: "testing",r);
    e.printStackTrace();
```

**Figure 4.3.4.1 – RceiptScanner.java (1/3)**

```
AddTrans addTrans = new AddTrans();

d = date.getText().toString();

//if the day is less than 10
if(d.length()<11){
    d = "0"+date.getText().toString();
}

SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd MMM yyyy");

try {
    dt = sdf.parse(d);
} catch (ParseException e) {
    e.printStackTrace();
}
date_millis = dt.getTime();
//send the information to real time firebase
submit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        root.child("transaction").child(String.valueOf(number)).child("dateInMillis").setValue(date_millis);
        root.child("transaction").child(String.valueOf(number)).child("amount").setValue("- " + total.getEditableText().toString(
        root.child("transaction").child(String.valueOf(number)).child("date").setValue(date.getEditableText().toString());
        root.child("transaction").child(String.valueOf(number)).child("notes").setValue(notes.getEditableText().toString());
        root.child("transaction").child(String.valueOf(number)).child("category").setValue("Expenses");
        root.child("transaction").child(String.valueOf(number)).child("pic").setValue(getImgURL());
        root.child("transaction").child(String.valueOf(number)).child("type").setValue(selected);
        root.child("transaction").child(String.valueOf(number)).child("for").setValue(selectedName);
        root.child("transaction").child(String.valueOf(number)).child("receiptURI").setValue(resultUri.toString());
        root.child("transaction").child(String.valueOf(number)).child("merchant name").setValue(merchantName.getEditableText().to
        root.child("transaction").child(String.valueOf(number)).child("monthyear").setValue(addTrans.getMonthYear(d));

        root.child("receipts").child(String.valueOf(number)).child("merchant name").setValue(merchantName.getEditableText().toStr
        root.child("receipts").child(String.valueOf(number)).child("total").setValue(total.getEditableText().toString());
        root.child("receipts").child(String.valueOf(number)).child("date").setValue(date.getEditableText().toString());
        root.child("receipts").child(String.valueOf(number)).child("notes").setValue(notes.getEditableText().toString());
        root.child("receipts").child(String.valueOf(number)).child("category").setValue(selected);
        root.child("receipts").child(String.valueOf(number)).child("receiptURI").setValue(resultUri.toString());
        root.child("receipts").child(String.valueOf(number)).child("for").setValue(selectedName);

        rt.child(String.valueOf(nc)).child("category name").setValue(String.valueOf(sp.getSelectedItemPosition()));
        rt.child(String.valueOf(nc)).child("merchant name").setValue(merchantName.getEditableText().toString());
        Intent i = new Intent( packageContext: ReceiptScanner.this, MainActivity.class);
        startActivity(i);
```

**Figure 4.3.4.2 – RceiptScanner.java (2/3)**

```
rt.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        for (DataSnapshot ds : snapshot.getChildren()) {
            nc = (int) snapshot.getChildrenCount();
            String mName = ds.child("merchant name").getValue(String.class);
            String merchantname = String.valueOf(merchantName.getEditableText());
            if (merchantname.equals(mName)) {
                ct = ds.child("category name").getValue(String.class);
                sp.setSelection(Integer.parseInt(ct));
            } else {

            }

        }


    }

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

    }
});
root.child("for").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        z = (int) snapshot.getChildrenCount();
        itemsName.clear();
        for (DataSnapshot ds : snapshot.getChildren()) {
            String spinnerName = ds.child("name").getValue(String.class);


            itemsName.add(spinnerName);
            ArrayAdapter<String> myAdapter2 = new ArrayAdapter<~>( context: ReceiptScanner.this, android.R.layout.simple_list_item_1, itemsN

            myAdapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
            forsp.setAdapter(myAdapter2);
            forsp.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
                @Override
                public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
                    selectedName = adapterView.getItemAtPosition(i).toString();

                    if (selectedName.equals("Others")) {
                        AlertDialog.Builder builder = new AlertDialog.Builder( context: ReceiptScanner.this);
                        builder.setTitle("Enter a new name:");
                        View v = getLayoutInflater().inflate(R.layout.newname_dialogbox,  root: null);
                        EditText newName;
                        Button entBtn;
                        builder.setView(v);
                        AlertDialog dialog = builder.create();

                        newName = v.findViewById(R.id.DB_newName);
                        entBtn = v.findViewById(R.id.DB_newEntBtn);
                        entBtn.setOnClickListener(new View.OnClickListener() {
                            @Override
                            public void onClick(View view) {
                               // nName.setText(newName.getText().toString());
                                root.child("for").child(String.valueOf(z + 1)).child("name").setValue(newName.getText().toString());

                                dialog.dismiss();
                            }
                        });
                        dialog.show();
                    }
                }

                @Override
                public void onNothingSelected(AdapterView<?> adapterView) {

                }


            });
        }
    }

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

**Figure 4.3.4.3– RceiptScanner.java (3/3)**

This class helps the user add a transaction by scanning a receipt. Figure 4.3.4.1 shows the code that uses libraries from Chaquopy to connect Python code to Java. The OCR API is implemented in Python. After calling the API, it extracts information in JSON format, and the code displays this extracted information on a well-organized screen. As mentioned earlier, the user can edit the details of the extracted information, such as the merchant's name, amount, date, and more. When the user presses the submit button, as shown in Figure 4.3.4.2, it stores all the latest and updated information in Firebase. There are other codes in this class, however only the major functions are displayed in the figures above.

### 4.3.5 AddTrans.java

```
public class AddTrans extends AppCompatActivity {
    Button income, expenses;
    ImageView back;
    DatabaseReference database = FirebaseDatabase.getInstance().getReference( path: "Users").child(F
    static int num;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_trans);

        back = findViewById(R.id.at_back);
        back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent( packageContext: AddTrans.this, MainActivity.class);
                startActivity(i);
            }
        });
        income = findViewById(R.id.addTrans_incomeButton);
        expenses = findViewById(R.id.addTrans_expButton);
        replaceFragment(new AddTrans_Expenses());
        database.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                for(DataSnapshot dataSnapshot: snapshot.getChildren()){
                    num = (int) snapshot.getChildrenCount();
                }
            }

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

            }
        });

        income.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                income.setTextColor(Color.rgb( red: 255,  green: 255,  blue: 255));
                expenses.setTextColor(Color.rgb( red: 137,  green: 137,  blue: 137 ));
                replaceFragment(new AddTrans_Income());
            }
        });

        expenses.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                income.setTextColor(Color.rgb( red: 137,  green: 137,  blue: 137  ));
                expenses.setTextColor(Color.rgb( red: 255,  green: 255,  blue: 255));
                replaceFragment(new AddTrans_Expenses());
            }
        });
    }
```

**Figure 4.3.5.1 – AddTrans.java (1/2)**

```
private void replaceFragment(Fragment fragment){
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.frameLayout,fragment);
    fragmentTransaction.commit();
}
public String getMonthYear(String date){

    String m,y,my,mon="";

    m = date.substring(3,6);
    y = date.substring(7,11);

    if(m.equals("Jan")){ mon = "January";}
    else if(m.equals("Feb")){ mon = "February"; }
    else if(m.equals("Mar")){ mon = "March";}
    else if(m.equals("Apr")){ mon = "April";}
    else if(m.equals("May")){ mon = "May"; }
    else if(m.equals("Jun")){ mon = "June"; }
    else if(m.equals("Jul")){ mon = "July"; }
    else if(m.equals("Aug")){ mon = "August"; }
    else if(m.equals("Sep")){ mon = "September"; }
    else if(m.equals("Oct")){ mon = "October"; }
    else if(m.equals("Nov")){ mon = "November"; }
    else if(m.equals("Dec")){ mon = "December"; }

    my = mon+" "+y;


    return my;

}
```

**Figure 4.3.5.2 – AddTrans.java(2/2)**

In this class, it assists the user in manually adding transactions. This class includes a fragment frame, and the "replaceFragment()" function, as shown in Figure 4.3.5.2, is called every time the user chooses to enter an income or expense transaction. For example, if the user selects to add an income transaction, the application will execute the "AddTrans_Income.java". Similarly, for expense transactions, it will execute the "AddTrans_Expenses.java", as shown in Figure 4.3.5.1. Additionally, Figure 4.3.5.2 also illustrates other functions, such as "getMonthYear()", which receives a date as a parameter and returns a string in the "Month Year" format.

### 4.3.6 BudgetPlanner.java

```
    add.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent i = new Intent( packageContext: BudgetPlanner.this, AddBudget.class);
            startActivity(i);
        }
    });

    back.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent i = new Intent( packageContext: BudgetPlanner.this, MainActivity.class);
            startActivity(i);
        }
    });


    root.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            bdetails.clear();
            for(DataSnapshot users : snapshot.child("budget").getChildren()) {

                    final String getName = users.child("name").getValue(String.class);
                    final String getAmount = users.child("amount").getValue(String.class);
                    final String getPurpose = users.child("purpose").getValue(String.class);
                    final String getRecurrence = users.child("recurrence").getValue(String.class);
                    final String getsDate = users.child("startdate").getValue(String.class);
                    final String geteDate = users.child("enddate").getValue(String.class);
                    final String getProgress = users.child("progress").getValue(String.class);
                    //creating item with user details
                    BDetails bdetail = new BDetails(getName, getPurpose, getRecurrence,getAmount,getsDate, geteDate,getProgress);

                    bdetails.add(bdetail);
                };
            rv.setAdapter(new Adapter7(bdetails,  context: BudgetPlanner.this));
        }


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

        }
});
}
```

**Figure 4.3.6.1 – BudgetPlanner.java**

"BudgetPlanner.java" displays budget plans in a RecyclerView format. As shown in the figure above, the code assists the user in retrieving data recorded in Firebase. After successfully retrieving the data, the application places it into a predefined class called "BDetails". This allows the application to easily access the details in the future.

### 4.3.7 AddBudget,java

```
//drop down list for recurrence
String[] r = getResources().getStringArray(R.array.recurrence);
ArrayAdapter adapter2 = new ArrayAdapter( context: this, android.R.layout.simple_spinner_dropdown_item, r);
adapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
recurrence.setAdapter(adapter2);


back.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent( packageContext: AddBudget.this,BudgetPlanner.class);
        startActivity(i);
    }
});
database.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        for(DataSnapshot dataSnapshot: snapshot.getChildren()){
            num = (int) snapshot.getChildrenCount();
        }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});


budgetFor.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        for(int j=0; j<selectedCat.length;j++){
            selectedCat[j] = false;

            catList.clear();
            budgetFor.setText("select here");

        }
        categoryDropdown(catArray2);
    }
});
```

**Figure 4.3.7.1 – AddBudget.java (1/3)**

The figure above shows that whenever user want press on certain elements such as “budgetfor” and “back” arrow it will perform certain actions. Not only that, but the number of item in the firebase path are also retrieved and calculated. It is essential because it helps us to acknowledge the current total number of items. So, that it would not overlap the previous item when adding a new budget transaction to the Firebase.

```
enter.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        getname = name.getEditableText().toString();
        getamount = amount.getEditableText().toString();
        getpurpose = budgetFor.getText().toString();
        getsdate =startdate.getText().toString();
        getrecurrence = recurrence.getSelectedItem().toString();

        if(recurrence.getSelectedItem().equals("Daily")){
            edate = getsdate;
        }
        else if(recurrence.getSelectedItem().equals("Weekly")){
            String dt = getsdate;  // Start date
            SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd MM yyyy");
            Calendar c = Calendar.getInstance();
            try {
                c.setTime(sdf.parse(dt));
            } catch (ParseException e) {
                e.printStackTrace();
            }
            c.add(Calendar.DATE, i1: 7);  // number of days to add, can also use Calendar.DAY_OF_MONTH in place of Calendar.DATE
            SimpleDateFormat sdf1 = new SimpleDateFormat( pattern: "dd MM yyyy");
            String output = sdf1.format(c.getTime());
            edate = output;
        }
        else if(recurrence.getSelectedItem().equals("Biweekly")){
            String dt = getsdate;  // Start date
            SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd MM yyyy");
            Calendar c = Calendar.getInstance();
            try {
                c.setTime(sdf.parse(dt));
            } catch (ParseException e) {
                e.printStackTrace();
            }
            c.add(Calendar.DATE, i1: 14);  // number of days to add, can also use Calendar.DAY_OF_MONTH in place of Calendar.DATE
            SimpleDateFormat sdf1 = new SimpleDateFormat( pattern: "dd MM yyyy");
            String output = sdf1.format(c.getTime());
            edate = output;
        }
        else if(recurrence.getSelectedItem().equals("Monthly")){
            String dt = getsdate;  // Start date
            SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd MM yyyy");
            Calendar c = Calendar.getInstance();
            try {
                c.setTime(sdf.parse(dt));
            } catch (ParseException e) {
                e.printStackTrace();
            }
            c.add(Calendar.DATE, i1: 30);  // number of days to add, can also use Calendar.DAY_OF_MONTH in place of Calendar.DATE
            SimpleDateFormat sdf1 = new SimpleDateFormat( pattern: "dd MM yyyy");
            String output = sdf1.format(c.getTime());
            edate = output;

        }
        getedate = edate;

            root.child("budget").child(String.valueOf(num+1)).child("name").setValue(getname);
            root.child("budget").child(String.valueOf(num+1)).child("amount").setValue(getamount);
            root.child("budget").child(String.valueOf(num+1)).child("purpose").setValue(getpurpose);
            root.child("budget").child(String.valueOf(num+1)).child("startdate").setValue(getsdate);
            root.child("budget").child(String.valueOf(num+1)).child("enddate").setValue(getedate);
            root.child("budget").child(String.valueOf(num+1)).child("recurrence").setValue(getrecurrence);
            root.child("budget").child(String.valueOf(num+1)).child("progress").setValue("0");


        Intent i = new Intent( packageContext: AddBudget.this, BudgetPlanner.class);
        startActivity(i);

    }
});
```

**Figure 4.3.7.2 – AddBudget.java (2/3)**

```
public void categoryDropdown(String []array){
    AlertDialog.Builder builder = new AlertDialog.Builder( context: AddBudget.this);

    builder.setTitle("Select category");
    builder.setCancelable(false);
    builder.setMultiChoiceItems(array, selectedCat, new DialogInterface.OnMultiChoiceClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i, boolean b) {
            if(b){
                catList.add(i);
            }
            else{
                catList.remove(i);
            }
        }
    }).setPositiveButton( text: "Ok", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            StringBuilder stringBuilder = new StringBuilder();
            for(int j=0;j<catList.size();j++){
                stringBuilder.append(array[catList.get(j)]);

                if(j!= catList.size()-1){
                    stringBuilder.append(", ");
                }
                budgetFor.setText(stringBuilder.toString());
            }
        }
    }).setNegativeButton( text: "Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
        }
    }).setNeutralButton( text: "Clear all", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            for(int j=0; j<selectedCat.length;j++){
                selectedCat[j] = false;

                catList.clear();
                budgetFor.setText("select here");
            }
        }
    });
    builder.show();
}
```

**Figure 4.3.7.3 – AddBudget.java (3/3)**

Figure 4.3.7.1 to Figure 4.3.7.3 shows the code in AddBudget.java. These codes perform actions when user press on the “enter” button as shown in figure 4.3.7.2. Once the system has gathered all the user input the system will store the data into Firebase. Whereas figure 4.3.7.3 shows a screenshot of a function named categoryDropdown() whereby its main function is to perform actions to let the user select the type of expenses he or she would like to budget.

### 4.3.8 GenerateReport.java

```
generate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(typeSpinner.getSelectedItem().equals("Total expenses")){
            s="Expenses";
        }
        if(typeSpinner.getSelectedItem().equals("Total income")){
            s="Income";
        }
        if(typeSpinner.getSelectedItem().equals("Total income and expenses")){
            s="Both";
        }
        int checkedID = radioGroup.getCheckedRadioButtonId();
        radioButton = findViewById(checkedID);
        //transaction only
        if(radioButton.getText().equals("Transaction only")){
            Intent i = new Intent( packageContext: GenerateReport.this, PrintTransactionOnly.class);
            i.putExtra( name: "selectedType", s);
            i.putExtra( name: "selectedCategory",catSpinner.getText().toString());
            i.putExtra( name: "selectedPeriod", period.getText().toString());
            i.putExtra( name: "selectedPerson", forSpinner.getSelectedItem().toString());
            i.putExtra( name: "selectedReport",checkedID);
            i.putExtra( name: "startDate",formattedStartDate);
            i.putExtra( name: "endDate",formattedEndDate);
            startActivity(i);
        }
        //receipt only
        if(radioButton.getText().equals("Receipt only")&&typeSpinner.getSelectedItem().equals("Total expenses")){
            Intent i = new Intent( packageContext: GenerateReport.this, PrintReceiptOnly.class);
            i.putExtra( name: "selectedType", s);
            i.putExtra( name: "selectedCategory",catSpinner.getText().toString());
            i.putExtra( name: "selectedPeriod", period.getText().toString());
            i.putExtra( name: "selectedPerson", forSpinner.getSelectedItem().toString());
            i.putExtra( name: "selectedReport",checkedID);
            i.putExtra( name: "startDate",formattedStartDate);
            i.putExtra( name: "endDate",formattedEndDate);
            startActivity(i);
        }
        //transaction and receipts
        if(radioButton.getText().equals("Both")&&typeSpinner.getSelectedItem().equals("Total expenses")){
            Intent i = new Intent( packageContext: GenerateReport.this, PrintBoth.class);
            i.putExtra( name: "selectedType", s);
            i.putExtra( name: "selectedCategory",catSpinner.getText().toString());
            i.putExtra( name: "selectedPeriod", period.getText().toString());
            i.putExtra( name: "selectedPerson", forSpinner.getSelectedItem().toString());
            i.putExtra( name: "selectedReport",checkedID);
            i.putExtra( name: "startDate",formattedStartDate);
            i.putExtra( name: "endDate",formattedEndDate);
            startActivity(i);
        }
        if((typeSpinner.getSelectedItem().equals("Total income")||(typeSpinner.getSelectedItem().equals("Total income and expenses")
            builder.setMessage("Receipt is not applicable")
                    .setCancelable(false)
                    .setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) { finish(); }
                    });
            AlertDialog alert = builder.create();
            alert.setTitle("Error");
            alert.show();
        }

        if((typeSpinner.getSelectedItem().equals("Total income")||(typeSpinner.getSelectedItem().equals("Total income and expenses")
            builder.setMessage("Receipt is not applicable")
                    .setCancelable(false)
                    .setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) { finish(); }
                    });
            AlertDialog alert = builder.create();
            alert.setTitle("Error");
            alert.show();
        }


    }
});
}
```

**Figure 4.3.8.1 – GenerateReport.java**

The main function of “GenerateReport.java” is to generate a personalized report. As shown in the figure above, when the user clicks on the “generate button, the system performs certain conditions and passes the data to subsequent pages, namely “PrintTransactionOnly.java”, “PrintReceiptOnly”, or “PrintBoth” classes using the “Intent()” function. This allows for a better and smoother process of displaying the report.

### 4.3.9 UserDetails.java

```
        }
    });
    ref = FirebaseDatabase.getInstance().getReference().child("Users").child(FirebaseAuth.getInstance
            ().getCurrentUser().getUid()).child("user details");
    ref.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
           String username = snapshot.child("username").getValue().toString();
            String firstName = snapshot.child("firstname").getValue().toString();
           String lastName = snapshot.child("lastname").getValue().toString();
            String eMail = snapshot.child("email").getValue().toString();

            uname.setText("@"+username);
            fname.setText(firstName);
            lname.setText(lastName);
            email.setText(eMail);

        }

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

        }
    });

    logout.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            auth.signOut();
            Intent intent = new Intent( packageContext: UserProfile.this, Login.class);
            startActivity(intent);

        }
    });

    del.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String[] items = {"Yes", "No"};
            AlertDialog.Builder dialog = new AlertDialog.Builder( context: UserProfile.this);
            dialog.setTitle("Are you sure you want to delete this account?");
            dialog.setItems(items, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    if (i == 0) {
                        auth.getCurrentUser().delete();
                        Intent intent = new Intent( packageContext: UserProfile.this, Login.class);
                        startActivity(intent);

                    }
                    if (i == 1) {

                    }
                }
            });
            dialog.create().show();


        }
    });
```

**Figure 4.3.9.1 – UserDetails.java (1/2)**

```
changeP.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent( packageContext: UserProfile.this, ResetActivity.class);
        startActivity(i);
    }
});

edit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder builder = new AlertDialog.Builder( context: UserProfile.this);

        builder.setMessage("Enter edit mode");
        builder.setTitle("Alert !");
        builder.setCancelable(false);

        builder.setPositiveButton( text: "OK", (DialogInterface.OnClickListener) (dialog, which) -> {
            dialog.cancel();
        });

        AlertDialog alertDialog = builder.create();
        alertDialog.show();
        edit.setImageDrawable(null);
        edit.setBackgroundResource(R.drawable.ic_baseline_done_24);
        fname.setEnabled(true);
        lname.setEnabled(true);
        email.setEnabled(true);


        root.child("user details").child("email").setValue(email.getEditableText().toString());
        root.child("user details").child("firstname").setValue(fname.getEditableText().toString());
        root.child("user details").child("lastname").setValue(lname.getEditableText().toString());

        edit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                AlertDialog.Builder builder = new AlertDialog.Builder( context: UserProfile.this);
                edit.setImageDrawable(null);
                edit.setBackgroundResource(R.drawable.ic_baseline_edit_note_24);

                builder.setMessage("All changes saved");
                builder.setTitle("Exit editing mode");
                builder.setCancelable(false);

                builder.setPositiveButton( text: "OK", (DialogInterface.OnClickListener) (dialog, which)
                    dialog.cancel();
                });

                AlertDialog alertDialog = builder.create();
                alertDialog.show();

                fname.setEnabled(false);
                lname.setEnabled(false);
               email.setEnabled(false);
            }
        });
```

**Figure 4.3.9.2 – UserDetails.java (2/2)**

In the UserDetail.java the system will first retrieve the user details from Firebase as shown in figure 4.3.9.1 and display the details appropriately. Moreover, different actions are performed when the user click on the buttons like change password, edit icon, delete account and so on. All these actions involved in Firebase with CRUD actions.

### 4.3.10 Help.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_help);
    composeEmail();
}


private void composeEmail() {
    String[] recipients = {"financie_support777@gmail.com"}; // Specify recipient email addresses
    String subject = "Reporting App Bug Issue";
    String message = "Dear [App Support Team],\n" +
            "\n" +
            "I hope this email finds you well. I am writing to report a critical bug that I encountered while using your [App Name] on my
            "\n" +
            "Bug Description:\n" +
            "When I attempt to [describe the specific action or scenario that triggers the bug], the app crashes abruptly without any err
            "\n" +
            "Steps to Reproduce:\n" +
            "\n" +
            "Launch the [App Name].\n" +
            "Navigate to [specific section or screen where the bug occurs].\n" +
            "Perform [specific action that triggers the bug].\n" +
            "Observe the app crash.\n" +
            "Expected Behavior:\n" +
            "The app should [describe the expected behavior or outcome when performing the action].\n" +
            "\n" +
            "Actual Behavior:\n" +
            "The app crashes and closes unexpectedly, forcing me to restart it.\n" +
            "\n" +
            "This bug is particularly concerning as it disrupts my workflow and prevents me from using [specific feature or functionality
            "\n" +
            "Screenshot: [Attach the screenshot if possible]\n" +
            "\n" +
            "I kindly request that your technical team investigate and rectify this bug as soon as possible. The seamless functioning of
            "\n" +
            "If you require any additional information, logs, or details from my end to assist with the troubleshooting process, please l
            "\n" +
            "Thank you for your attention to this matter. I look forward to receiving an update on the progress of resolving this bug. Yo
            "\n" +
            "Best regards,\n" +
            "[Your Name]\n" +
            "[Your Contact Information]";

    Intent intent = new Intent(Intent.ACTION_SENDTO);
    intent.setData(Uri.parse("mailto:")); // Specify the "mailto" data scheme
    intent.putExtra(Intent.EXTRA_EMAIL, recipients);
    intent.putExtra(Intent.EXTRA_SUBJECT, subject);
    intent.putExtra(Intent.EXTRA_TEXT, message);

    // Check if there's an app available to handle the intent
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    } else {
        // Handle this case if no email client is available.
    }
}
```

**Figure 4.3.10.1 – Help.java**

This class only consists of one function namely composeEmail() whereby once the user is link to this class it will trigger the function. The function will first direct the user to the email application in the user's phone that consists of a predefined feedback email. However, the user is free to change the feedback email and it just for the user's reference.

### 4.3.11 Settings.java

```
curr.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        final String[] currency = {"RM", "$ ", "€ ", "£ ", "¥ "};
        int selectedItemIndex = -1; // Initialize to no selection
        for (int i = 0; i < currency.length; i++) {
            if (currency[i].equals(selectedCurrency)) {
                selectedItemIndex = i;
                break;
            }
        }

        AlertDialog.Builder builder = new AlertDialog.Builder( context: Settings.this);
        builder.setTitle("Choose currency");
        builder.setSingleChoiceItems(currency, selectedItemIndex, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                selectedCurrency = currency[i];
            }
        });
        builder.setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                // No need to modify selectedCurrency here
                SharedPreferences preferences = getSharedPreferences( name: "MyPrefs", MODE_PRIVATE);
                SharedPreferences.Editor editor = preferences.edit();
                editor.putString( s: "selectedCurrency", selectedCurrency);
                editor.apply();

                dialogInterface.dismiss();
                Intent k = new Intent( packageContext: Settings.this, MainActivity.class);
                startActivity(k);
            }
        });
        builder.show();
    }
```

**Figure 4.3.11.1 – Settings.java (1/5)**

```
alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);

alarm.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Calendar calendar = Calendar.getInstance();
        timePickerDialog = new TimePickerDialog(
                context: Settings.this,
                android.R.style.Theme_Holo_Light_Dialog_MinWidth,
                new TimePickerDialog.OnTimeSetListener() {
                    @Override
                    public void onTimeSet(TimePicker timePicker, int i, int i1) {

                        calendar.set(Calendar.HOUR_OF_DAY, i); // Set the alarm hour
                        calendar.set(Calendar.MINUTE, i1);     // Set the alarm minute
                        Log.i( tag: "testing", msg: String.valueOf(i)+String.valueOf(i1));


                    }
                }, hourOfDay: 12, minute: 0, is24HourView: false
        );
        timePickerDialog.setTitle("Select a time for your daily reminder to record your transactions");
        timePickerDialog.show();
        Intent intent = new Intent( packageContext: Settings.this, AlarmReceiver.class);
        pendingIntent = PendingIntent.getBroadcast( context: Settings.this,  requestCode: 0, intent,  flags: 0);

        alarmManager.setRepeating(
                AlarmManager.RTC_WAKEUP,
                calendar.getTimeInMillis(),
                AlarmManager.INTERVAL_DAY,
                pendingIntent
        );
    }

});
```

**Figure 4.3.11.2 – Settings.java (2/5)**

```
lang.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        final String[] language = {"English", "Malay","Chinese", "Korean", "Japanese"};
        int selectedItemIndex = -1; // Initialize to no selection
        for (int i = 0; i < language.length; i++) {
            if (language[i].equals(selectedLanguage)) {
                selectedItemIndex = i;
                break;
            }
        }

        AlertDialog.Builder builder = new AlertDialog.Builder( context: Settings.this);
        builder.setTitle("Select a language");
        builder.setSingleChoiceItems(language, selectedItemIndex, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                selectedLanguage = language[i];
            }
        });
        builder.setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Log.i( tag: "testing",selectedLanguage);
                if(selectedLanguage.equals("English")){
                    setLanguage("en");
                }
                else if(selectedLanguage.equals("Malay")){
                    setLanguage("ms");
                }
                else if(selectedLanguage.equals("Chinese")){
                    setLanguage("zh");
                }
                else if(selectedLanguage.equals("Korean")){
                    setLanguage("ko");
                }else if(selectedLanguage.equals("Japanese")){
                    setLanguage("ja");
                }


                startActivity(new Intent( packageContext: Settings.this,MainActivity.class));

                dialogInterface.dismiss();
            }
        });
        builder.show();
    }
});

rivate void setLanguage(String lan) {
    Resources resources = this.getResources();
    Configuration configuration = resources.getConfiguration();
    Locale locale = new Locale(lan);
    Locale.setDefault(locale);
    configuration.setLocale(locale);
    resources.updateConfiguration(configuration, resources.getDisplayMetrics());
```

**Figure 4.3.11.3 – Settings.java (3/5)**

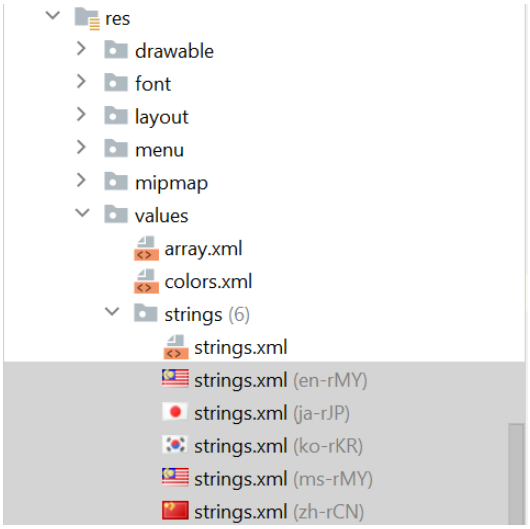



**Figure 4.3.11.4 – Settings.java (4/5)**

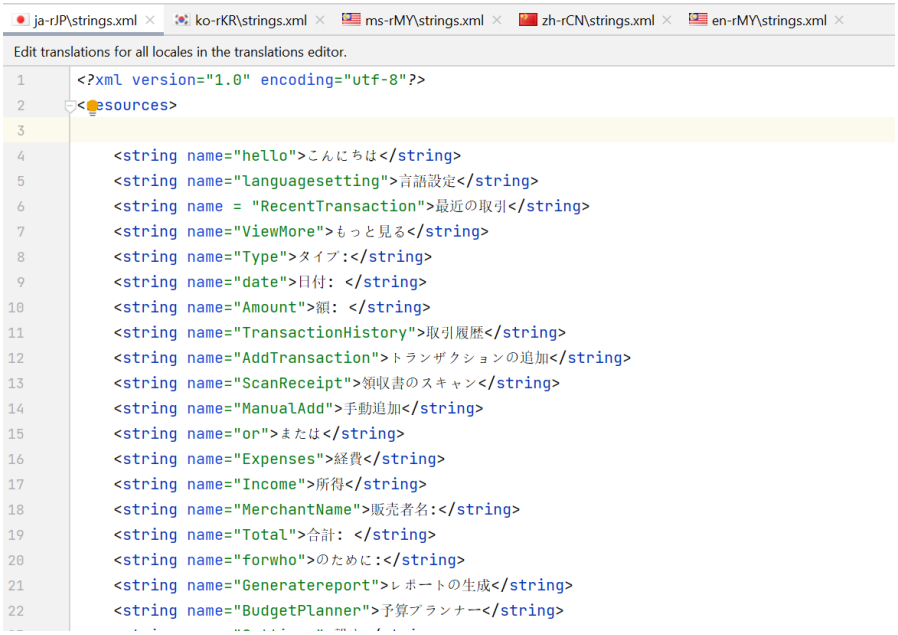



**Figure 4.3.11.5 – Settings.java (5/5)**

In the setting.java there are several actions will be performed based on the selection of the user. For instance, if the user chooses to click on the “Currency settings" it will perform certain actions as shown in figure 4.3.11.1. Whereas the “Alarm settings” it will perform actions as shown in figure 4.3.11.2. Lastly, if the user chooses to click on the “language settings” it will perform actions and execute a function called setLanguage() as shown in figure 4.3.11.3. However, the user is required to 5 strings.xml to the res folder of the application whereby each different strings.xml consists of the translation of the string and each has a unique id, so that it can be easily replaced with the existing string.

# CHAPTER 5 SYSTEM IMPLEMENTATION (FOR DEVELOPMENT- BASED PROJECT)

## 5.1 Project Timeline

**PROJECT: Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking**

Project start date: 30/1/2023

By Tan Su Hua

| Milestone description | Category | Start | Days |
|---|---|---|---|
| **Chapter 1: Introduction** | | | |
| Evalaute the Proposed Report | On Track | 6/2/2023 | 3 |
| Refine the 3 Problem Statement | On Track | 9/2/2023 | 4 |
| Refine the 3 Project Objective | On Track | 13/2/2023 | 4 |
| Refine Project Contribution | On Track | 17/2/2023 | 2 |
| Refine Project Scope and direction | On Track | 19/2/2023 | 2 |
| Accomplish Introduction | Milestone | **21/2/2023** | **1** |
| **Chapter 2: Literature Review** | | | |
| Review Technology | On Track | 21/2/2023 | 4 |
| Analyze and Summarize Related Technology | On Track | 25/2/2023 | 3 |
| Review Existing System | Low Risk | 28/2/2023 | 5 |
| Analyze and Summarize Systems | On Track | 5/3/2023 | 4 |
| Compare the exisiting system with the proposed system | On Track | 9/3/2023 | 2 |
| Finish Literature Review | Milestone | **11/3/2023** | **1** |

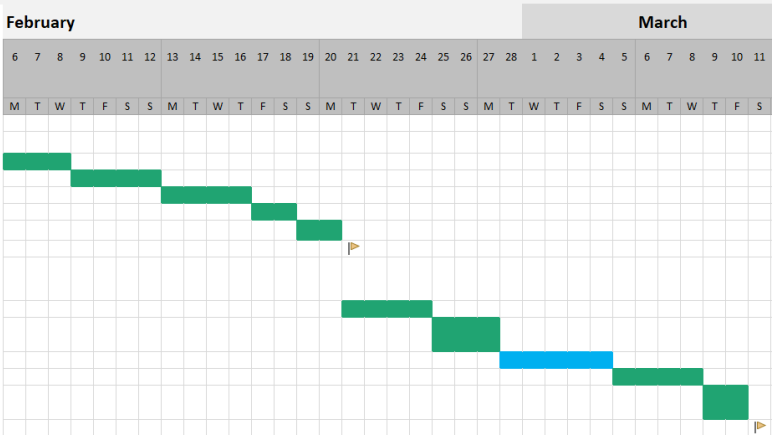



**Figure 5.1.1 – Project Timeline (1/6)**

**PROJECT: Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking**

Project start date: 30/1/2023

By Tan Su Hua

| Milestone description | Category | Start | Days |
|---|---|---|---|
| **Chapter 3: System Methodology / Approach** | | | |
| Planning | On Track | 11/3/2023 | 4 |
| System Architecture Diagram | On Track | 15/3/2023 | 4 |
| Use Case Diagram | On Track | 19/3/2023 | 3 |
| Activity Diagram | On Track | 22/3/2023 | 3 |
| Complete System Methodology | Milestone | **25/3/2023** | **1** |
| **Chapter 4: System Design** | | | |
| System Flow Diagram | On Track | 25/3/2023 | 2 |
| Database Class Diagram | On Track | 27/3/2023 | 2 |
| System Flow Description | On Track | 29/3/2023 | 3 |
| Complete System Design | Milestone | **1/4/2023** | **1** |
| **Chapter 5: System Implementation** | | | |
| Hardware Setup | On Track | 1/4/2023 | 1 |
| Software Setup | On Track | 2/4/2023 | 1 |
| Setting Configuration | On Track | 3/4/2023 | 3 |
| Software Development Life Cycle (SDLC) | High Risk | 6/4/2023 | 136 |
| Complete System Implentation | Milestone | **20/8/2023** | **1** |

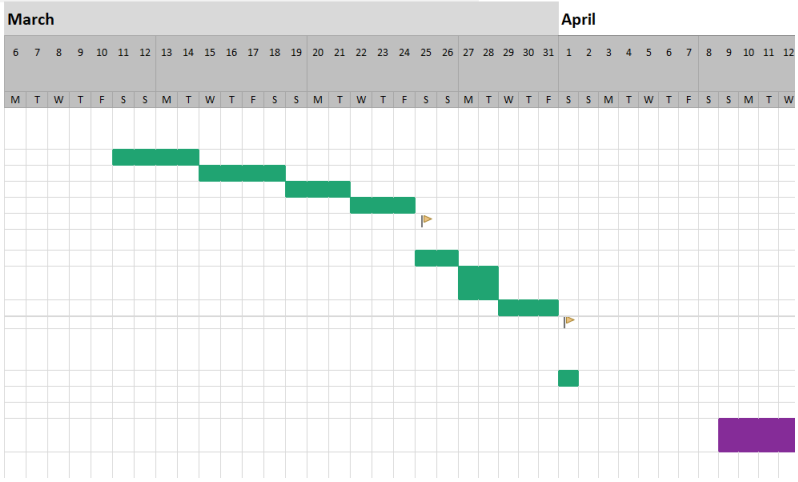



**Figure 5.1.2– Project Timeline (2/6)**

**PROJECT: Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking**

Project start date: 30/1/2023

By Tan Su Hua

| Milestone description | Category | Start | Days |
|---|---|---|---|
| **Chapter 3: System Methodology / Approach** | | | |
| Planning | On Track | 11/3/2023 | 4 |
| System Architecture Diagram | On Track | 15/3/2023 | 4 |
| Use Case Diagram | On Track | 19/3/2023 | 3 |
| Activity Diagram | On Track | 22/3/2023 | 3 |
| Complete System Methodology | Milestone | **25/3/2023** | **1** |
| **Chapter 4: System Design** | | | |
| System Flow Diagram | On Track | 25/3/2023 | 2 |
| Database Class Diagram | On Track | 27/3/2023 | 2 |
| System Flow Description | On Track | 29/3/2023 | 3 |
| Complete System Design | Milestone | **1/4/2023** | **1** |
| **Chapter 5: System Implementation** | | | |
| Hardware Setup | On Track | 1/4/2023 | 1 |
| Software Setup | On Track | 2/4/2023 | 1 |
| Setting Configuration | On Track | 3/4/2023 | 3 |
| Software Development Life Cycle (SDLC) | High Risk | 6/4/2023 | 136 |
| Complete System Implentation | Milestone | **20/8/2023** | **1** |

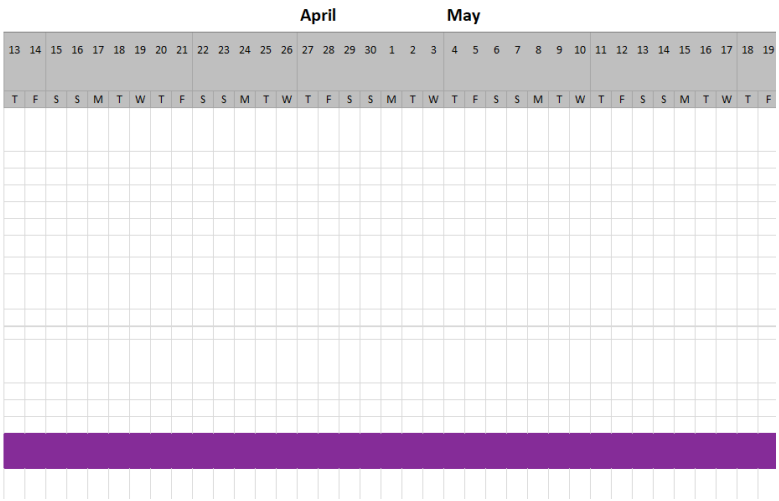



**Figure 5.1.3 – Project Timeline (3/6)**

**PROJECT: Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking**

Project start date: 30/1/2023

By Tan Su Hua

| Milestone description | Category | Start | Days |
|---|---|---|---|
| **Chapter 3: System Methodology / Approach** | | | |
| Planning | On Track | 11/3/2023 | 4 |
| System Architecture Diagram | On Track | 15/3/2023 | 4 |
| Use Case Diagram | On Track | 19/3/2023 | 3 |
| Activity Diagram | On Track | 22/3/2023 | 3 |
| Complete System Methodology | Milestone | **25/3/2023** | **1** |
| **Chapter 4: System Design** | | | |
| System Flow Diagram | On Track | 25/3/2023 | 2 |
| Database Class Diagram | On Track | 27/3/2023 | 2 |
| System Flow Description | On Track | 29/3/2023 | 3 |
| Complete System Design | Milestone | **1/4/2023** | **1** |
| **Chapter 5: System Implementation** | | | |
| Hardware Setup | On Track | 1/4/2023 | 1 |
| Software Setup | On Track | 2/4/2023 | 1 |
| Setting Configuration | On Track | 3/4/2023 | 3 |
| Software Development Life Cycle (SDLC) | High Risk | 6/4/2023 | 136 |
| Complete System Implentation | Milestone | **20/8/2023** | **1** |

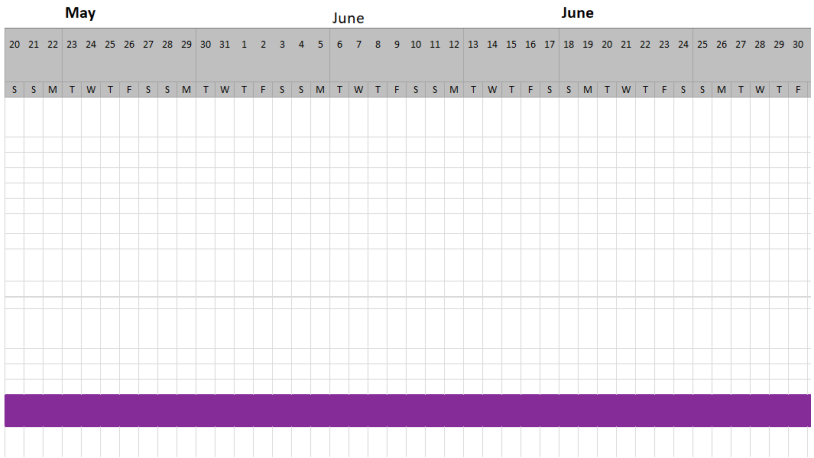



**Figure 5.1.4 – Project Timeline (4/6)**

**PROJECT: Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking**

Project start date: 30/1/2023

By Tan Su Hua

| Milestone description | Category | Start | Days |
|---|---|---|---|
| **Chapter 3: System Methodology / Approach** | | | |
| Planning | On Track | 11/3/2023 | 4 |
| System Architecture Diagram | On Track | 15/3/2023 | 4 |
| Use Case Diagram | On Track | 19/3/2023 | 3 |
| Activity Diagram | On Track | 22/3/2023 | 3 |
| Complete System Methodology | Milestone | **25/3/2023** | **1** |
| **Chapter 4: System Design** | | | |
| System Flow Diagram | On Track | 25/3/2023 | 2 |
| Database Class Diagram | On Track | 27/3/2023 | 2 |
| System Flow Description | On Track | 29/3/2023 | 3 |
| Complete System Design | Milestone | **1/4/2023** | **1** |
| **Chapter 5: System Implementation** | | | |
| Hardware Setup | On Track | 1/4/2023 | 1 |
| Software Setup | On Track | 2/4/2023 | 1 |
| Setting Configuration | On Track | 3/4/2023 | 3 |
| Software Development Life Cycle (SDLC) | High Risk | 6/4/2023 | 136 |
| Complete System Implentation | Milestone | **20/8/2023** | **1** |

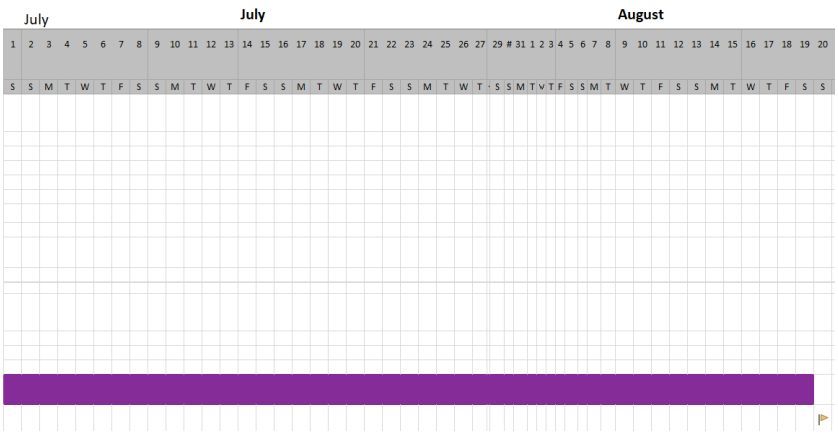
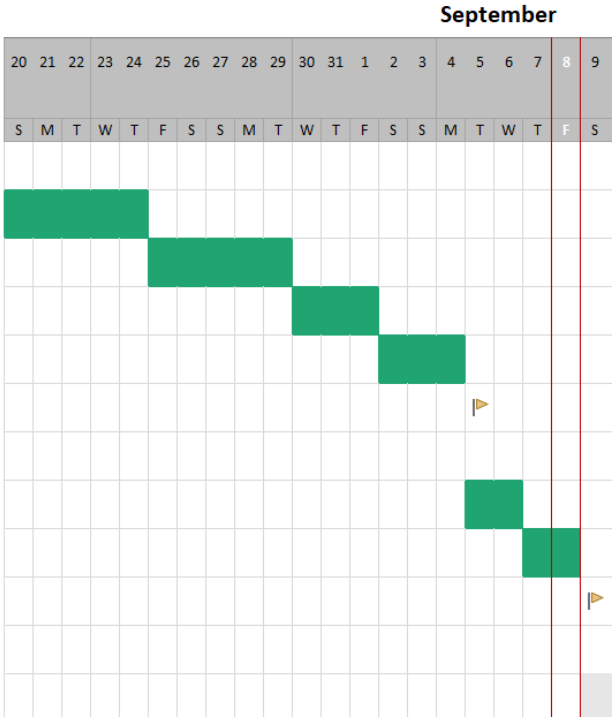



**Figure 5.1.5 – Project Timeline (5/6)**

**PROJECT: Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking**

Project start date: 30/1/2023

By Tan Su Hua




| Milestone description | Category | Start | Days |
|---|---|---|---|
| **Chapter 6: System Evaluation and Discussion** | | | |
| System Testing and Performace Metrics | On Track | 20/8/2023 | 5 |
| Testing Setup and Result | On Track | 25/8/2023 | 5 |
| Project Challenges | On Track | 30/8/2023 | 3 |
| Objective Evaluation | On Track | 2/9/2023 | 3 |
| Complete System Evaluation | Milestone | **5/9/2023** | **1** |
| **Chapter 7: Conclusion and Recommendation** | | | |
| Conclusion | On Track | 5/9/2023 | 2 |
| Recommendation | On Track | 7/9/2023 | 2 |
| Complete Proposed Project | Milestone | **9/9/2023** | **1** |
| | | | |
| **Project End Date** | | **9/9/2023** | **215** |

**Figure 5.1.6 – Project Timeline (6/6)**

## 5.2 Hardware Setup

| Description | Specifications |
|---|---|
| Model | Msi Prestige 14 |
| Processor | Intel(R) Core (TM) i7-10510U CPU @ 1.80GHz 2.30 GHz |
| Operating System | Window 11 |
| Graphic | NVIDIA GeForce MX350 |
| Memory | 16GB DDR4 |
| Storage | 512GB SSD |

**Table 5.2 – Hardware setup**

The table above shows a detailed hardware setup of the laptop used in this project. Moreover, several hardware is also required to make this project a success and these are a physical android-based phone and a USB type C cable. This three hardware is used for testing and debugging of the program.

## 5.3 Software Setup

| Tools | Specification | Description |
|---|---|---|
| Android Studio | Version: Chipmunk (2021.2.1)<br>Programming language: Java<br>SDK version: 32<br>Gradle plugin version: 7.2.1<br>Gradle version: 7.3.3 | An IDE for Google's Android Operating System. |
| Firebase | Real Time Firebase<br>Firebase Storage<br>Firebase Authentication | It provides cloud computing services |
| Asprise OCR API | Receipt OCR | An API that provides OCR function |
| IDLE | Version: 3.8 | A default IDE for Python |
| Chaquopy | Version: 14.0 | A Python SDK for Android |

**Table 5.3 – Software Setup**

## 5.4 Setting and Configuration

build.gradle (:app)

You can use the Project Structure dialog to view and edit your project configuration

```
plugins {
    id 'com.android.application'
    id 'com.google.gms.google-services'
    id 'com.chaquo.python'

}

android {
    compileSdk 32

    packagingOptions {
        exclude("META-INF/DEPENDENCIES")
        exclude("META-INF/LICENSE")
        exclude("META-INF/LICENSE.txt")
        exclude("META-INF/license.txt")
        exclude("META-INF/NOTICE")
        exclude("META-INF/NOTICE.txt")
        exclude("META-INF/notice.txt")
        exclude("META-INF/ASL2.0")
        exclude("META-INF/*.kotlin_module")

        sourceSets {
            main {
                python.srcDir "src/main/python"
            }
        }
    }


    defaultConfig {
        applicationId "com.example.myapplication"
        minSdk 28
        targetSdk 32
        versionCode 1
        versionName "1.0"
        python {
            buildPython "C:/Users/suhua/AppData/Local/Programs/Python/Python38/python.exe"

        }

        python {
            version "3.8"
        }
        ndk {
            abiFilters "armeabi-v7a", "arm64-v8a", "x86", "x86_64"
        }
        python{

                pip {
                    install "requests==2.24.0"
                    install "Image"
                    install "pytesseract"

                }

        }
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }

    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }


}
```

**Figure 5.4.1- Setting and Configuration (1/4)**

```
dependencies {

    implementation 'com.google.firebase:firebase-database:20.2.2'
    implementation 'com.google.firebase:firebase-auth:22.0.0'
    implementation 'androidx.appcompat:appcompat:1.5.1'
    implementation 'com.google.android.material:material:1.7.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.google.firebase:firebase-database:20.1.0'
    implementation 'com.google.firebase:firebase-auth:21.1.0'
    implementation 'com.google.firebase:firebase-storage:20.1.0'
    implementation 'com.google.mlkit:entity-extraction:16.0.0-beta4'
    implementation 'com.google.android.gms:play-services-maps:18.1.0'
    implementation 'com.google.firebase:firebase-firestore:24.4.4'
    implementation 'com.google.firebase:firebase-database-ktx:20.1.0'
    implementation 'com.google.firebase:firebase-storage-ktx:20.1.0'
    implementation 'com.google.firebase:firebase-auth-ktx:21.1.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'com.google.firebase:firebase-firestore-ktx:24.5.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.4'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.0'
    implementation 'com.rmtheis:tess-two:5.4.1'
    implementation 'com.github.denzcoskun:ImageSlideshow:0.1.0'
    implementation 'com.google.android.gms:play-services-auth:20.4.0'
    implementation 'com.github.blackfizz:eazegraph:1.2.5l@aar'
    implementation 'com.nineoldandroids:library:2.4.0'
    implementation 'com.theartofdev.edmodo:android-image-cropper:2.8.+'
    implementation 'com.google.firebase:firebase-firestore:24.7.0'
    implementation 'com.whiteelephant:monthandyearpicker:1.3.0'
    implementation 'com.asprise.ocr:java-ocr-api:15.3.0.3'
    implementation 'com.google.firebase:firebase-auth:22.1.0'
    implementation 'com.google.firebase:firebase-storage:20.2.1'
    implementation 'com.google.android.material:material:1.3.0-alpha03'
    implementation 'com.github.bumptech.glide:glide:4.11.0'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.11.0'
    implementation 'com.google.firebase:firebase-database:20.2.0'
    implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'
    implementation group: 'org.apache.httpcomponents', name: 'httpmime', version: '4.5.6'
    implementation 'org.apache.httpcomponents:httpcore:4.4.16'
    implementation 'org.apache.httpcomponents:httpclient:4.5.14'
}
```

**Figure 5.4.2- Setting and Configuration (2/4)**

build.gradle (:app) settings.gradle (My Application)

You can use the Project Structure dialog to view and edit your project configuration

```
pluginManagement { PluginManagementSpec it ->
    repositories { RepositoryHandler it ->
        gradlePluginPortal()
        google()
        mavenCentral()
    }
}
dependencyResolutionManagement { DependencyResolutionManagement it ->
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories { RepositoryHandler it ->
        google()
        jcenter()
        mavenCentral()
        maven { url 'https://jitpack.io' }
    }
}
rootProject.name = "My Application"
include ':app'
```

**Figure 5.4.3- Setting and Configuration (3/4)**

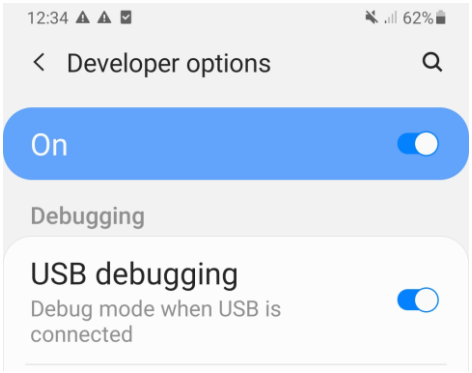



**Figure 5.4.4- Setting and Configuration (4/4)**

To set up the application, several things are required, such as a stable internet connection. Besides that, a physical Android phone is also required, and in this case, a Samsung Note 9 was used for testing and debugging of the application. Moreover, several actions are required to establish a connection between the phone and the Android Studio IDE. For instance, the physical phone should enable the developer option and enable USB debugging option as shown in figure 5.4.4. Hence, this helps establish a connection between the IDE and the phone, making it easier for testing and debugging of the code. Furthermore, to establish and configure the code in the Android Studio IDE, proper IDE setup is necessary. For instance, insertion of the code, as illustrated in Figure 5.4.1 and Figure 5.4.2 within the build.gradle (app) file, is necessary. Additionally, the inclusion of the code within the settings.gradle (My Application) file is also essential.

## 5.5 System Operation (with Screenshot)

### 5.5.1 Login.java

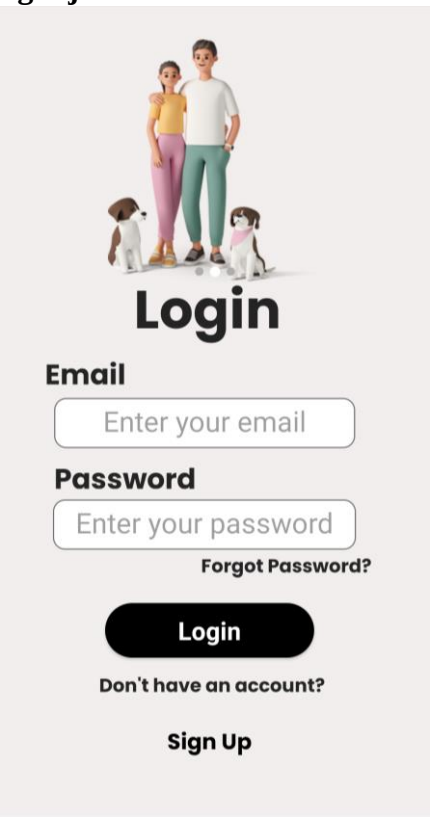



**Figure 5.5.1.1 - Login (1/2)**

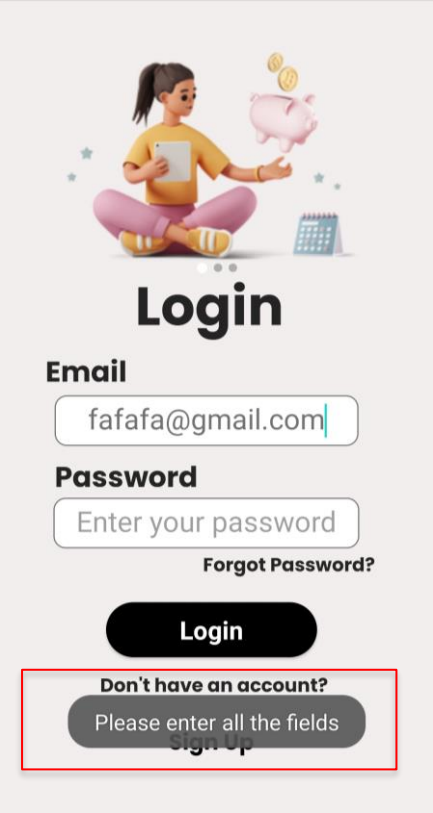



**Figure 5.5.1.2 – Login (2/2)**

Figure 5.5.1.1 displays the login user interface of the application, requiring users to enter their registered email and password to access the application. Additionally, on this screen, users have the option to register a new account by selecting the "Sign Up" text. In case a user forgets their password, they can click on the "Forgot Password?" text, which will redirect them to the ResetPassword page. The application also validates user inputs. If a user enters an invalid or empty field, a toast message will appear, displaying the corresponding error, as illustrated in Figure 5.5.1.2.

### 5.5.2 SignUp.java

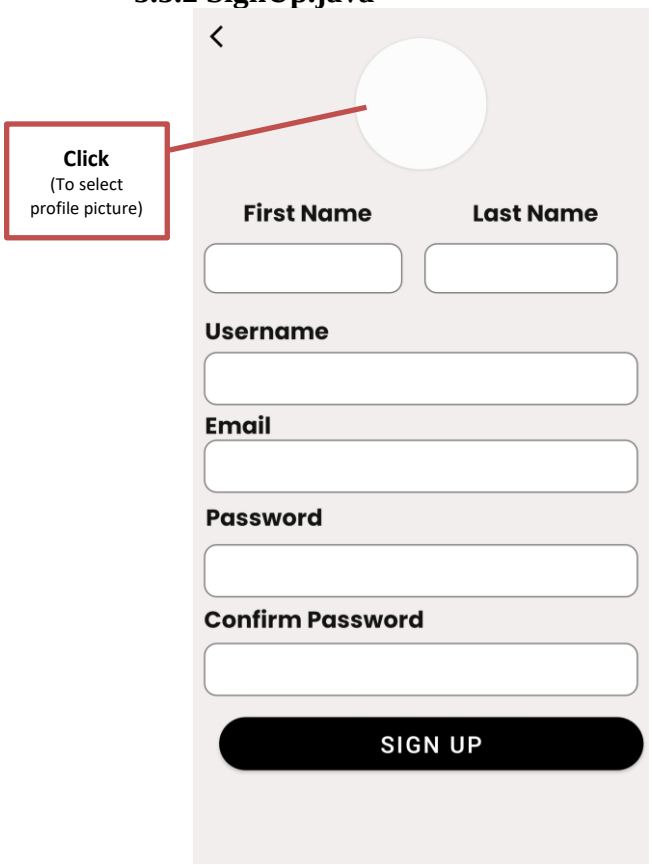



**Figure 5.5.2.1 - Sign Up (1/2)**

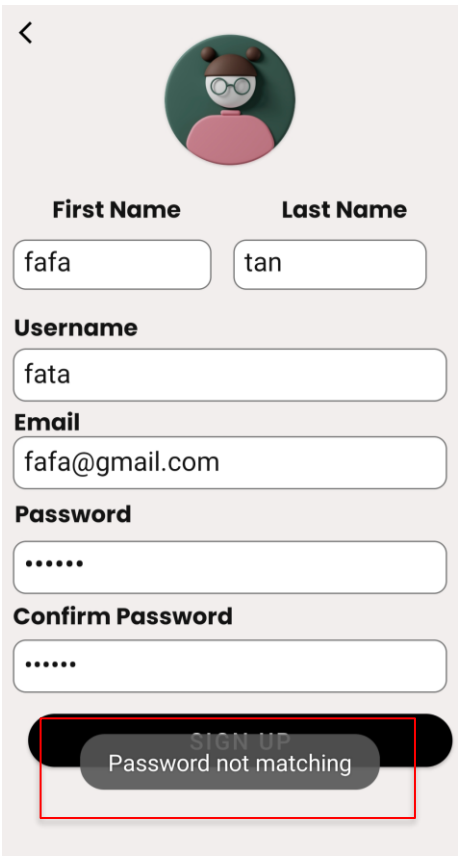



**Figure 5.5.2.2 - Sign Up (2/2)**

Figures 5.5.2.1 and 5.5.2.2 depict the sign-up interface of the application. Figure 5.5.2.1 enables users to input the necessary details to create an account for the application. In contrast, Figure 5.5.2.2 displays a toast message regarding an error made by the user. In this particular instance, the user fails to confirm the password they entered, leading to the appearance of a toast message notifying the user that the entered password is incorrect.

### 5.5.3 MainActivity.java

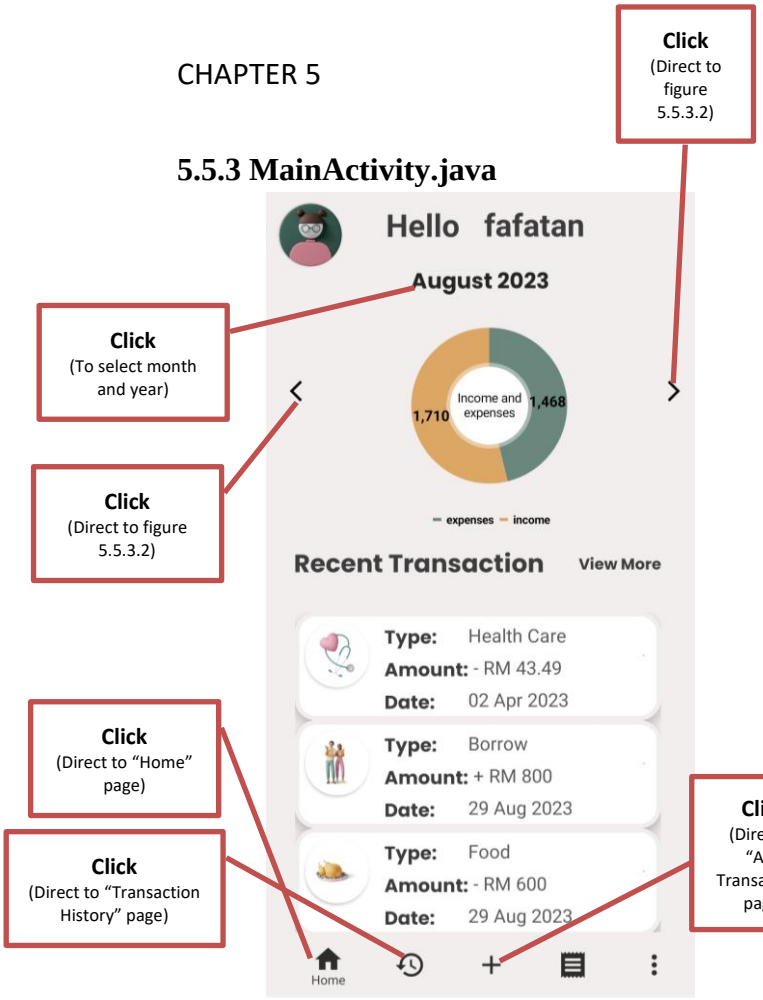



**Figure 5.5.3.1 – Main (1/2)**

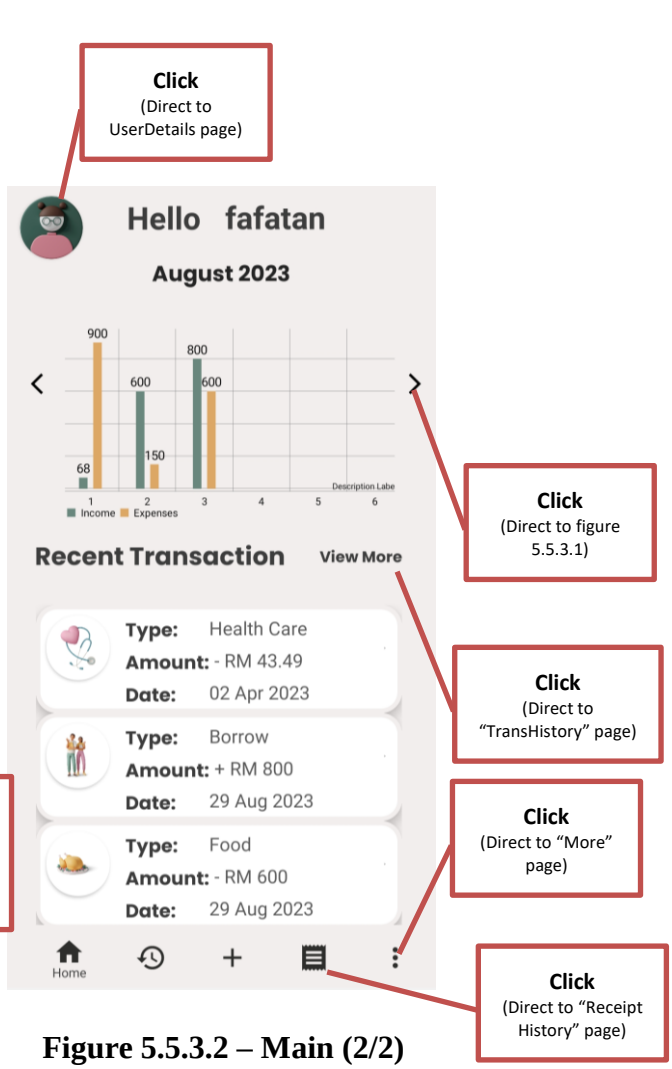



**Figure 5.5.3.2 – Main (2/2)**

Both of the above figures depict the main user interface, which includes various elements such as graphical representations, recent transactions, and a bottom navigation bar featuring functions like home, transaction history, add transaction, receipts history, and more. This screen, as shown in Figures 5.5.3.1 and 5.5.3.2, features two graphical representations: a pie chart and a bar chart. Furthermore, this initial interface is the first screen users encounter upon logging into the application. It adheres to a 3-click rule, allowing users to access the major functions within three clicks. This design ensures that the application's primary features can be easily accessed and used by users, thereby enhancing and guaranteeing a seamless user experience.

### 5.5.4 AddTrans

AddTrans_ManuallyOrScan.java AddTrans_Expenses.java AddTrans_Income.java

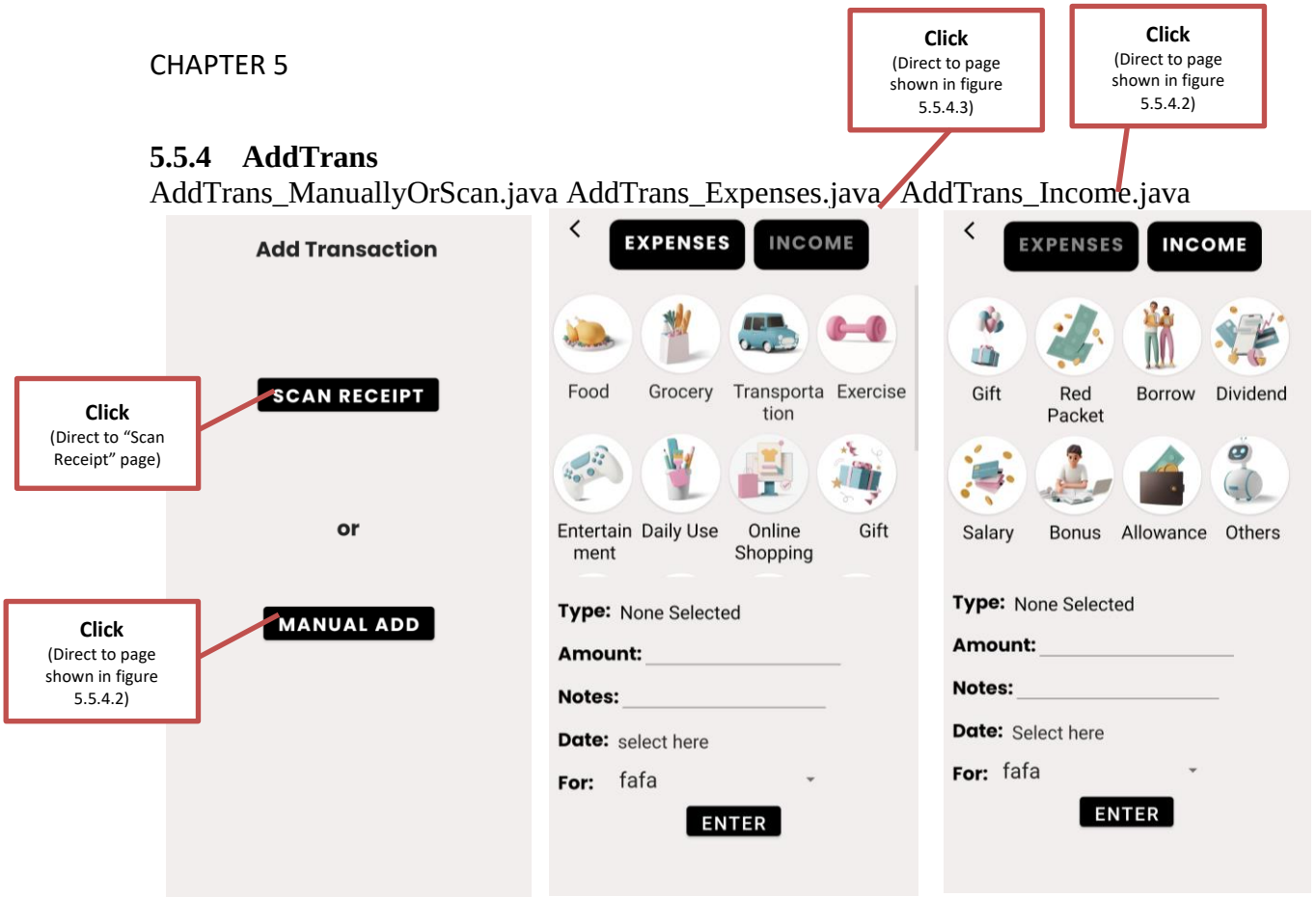



**Figure 5.5.4.1 - AddTrans (1/3) Figure 5.5.4.2 - AddTrans (2/3) Figure 5.5.4.3 – AddTrans (3/3)**

Moving on, Figure 5.5.4.1 displays the user interface of the AddTrans_ManuallyOrScan class. This page allows users to choose how they want to add a transaction, with options including scanning a receipt or manually entering the details. If the user selects "manual add" for an expenses transaction, it will display the screen shown in Figure 5.5.4.2. Conversely, if the user chooses to add an income transaction, they must press the income button within Figure 5.5.4.2. This action directs the user to Figure 5.5.4.3. After inputting all the required details such as type, amount, notes, date, and transaction owner, users can confirm the transaction by pressing the enter button.

### 5.5.5 TransDetail.java

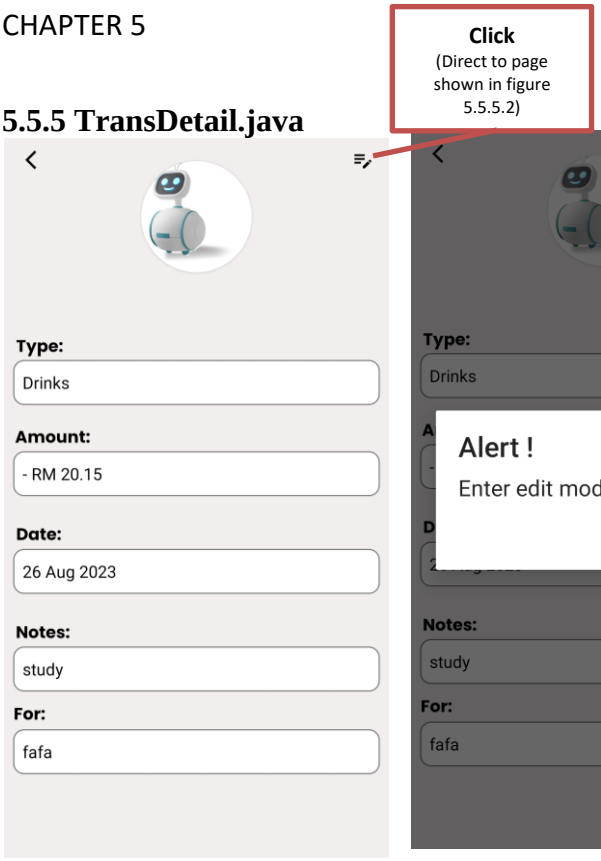



**Figure 5.5.5.1-TransDetails (1/3)**

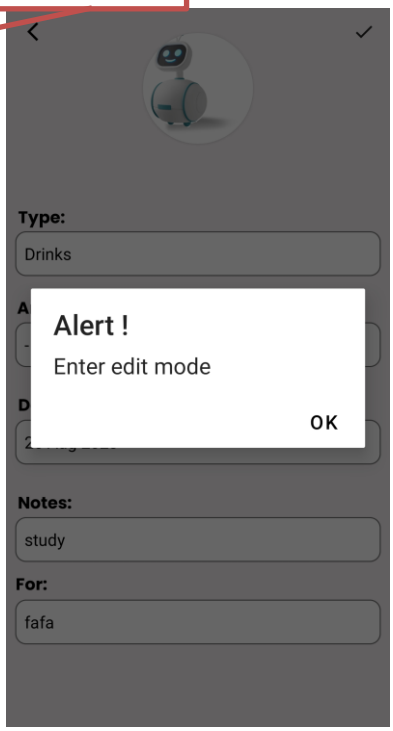



**Figure 5.5.5.2 -TransDetails (2/3)**

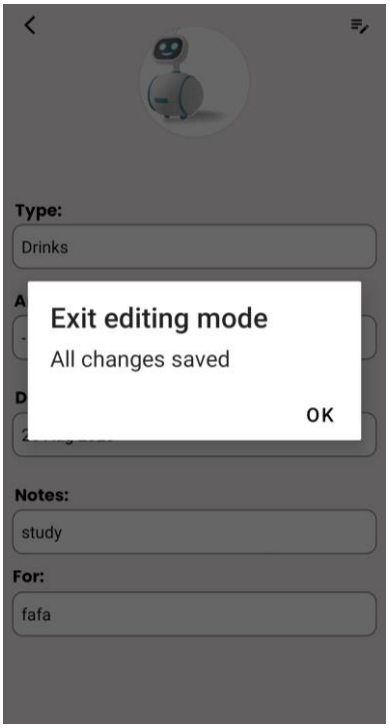
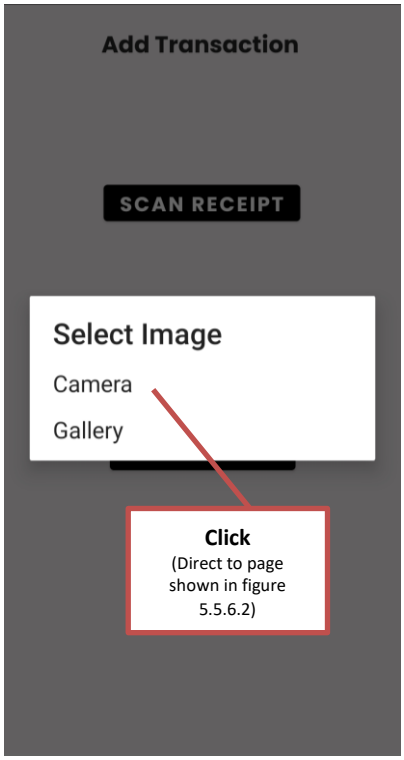
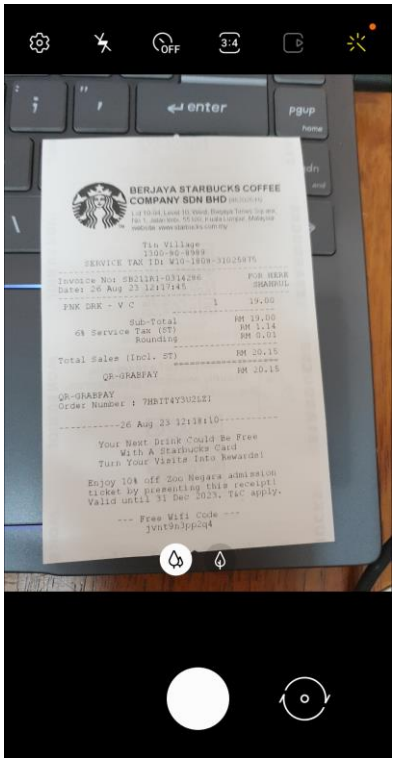



**Figure 5.5.5.3 -TransDetails (3/3)**

In this screen, TransDetail.java, users can view their recorded transactions in a more detailed form. Moreover, users can initiate the editing mode by tapping the top right corner of the screen, which triggers the appearance of an alert dialog box as depicted in Figure 5.5.5.2. This enables users to edit their transaction details. Once users have completed editing, they can exit the editing mode by pressing the same icon, as demonstrated in Figure 5.5.5.3.

### 5.5.6 ReceiptScanner.java



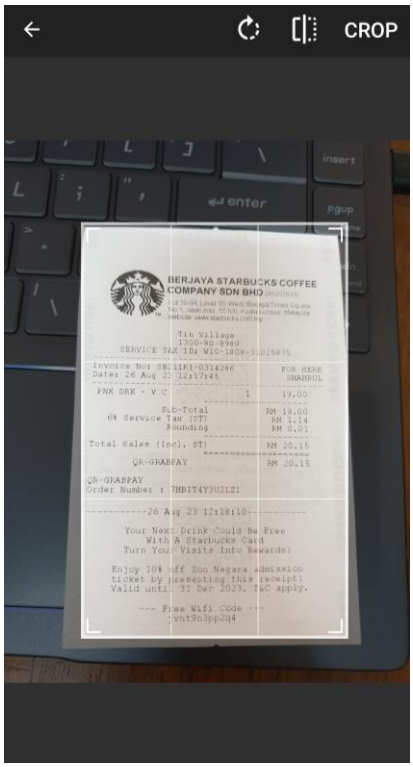



**Figure 5.5.6.1-ReceiptScanner (1/5) Figure 5.5.6.2-ReceiptScanner(2/5 ) Figure5.5.6.3-ReceiptScanner (3/5)**

If the user chooses to add a transaction by scanning a receipt, a pop-up message will appear, prompting the user to select the receipt image either by capturing it using the camera in real time or choosing it from their gallery. Figure 5.5.6.2 is the screen the user will see if they opt to take a real-time photo of the receipt. After that, the user can choose to resize the image, as shown in Figure 5.5.6.3, or they can choose to ignore it. A similar process will apply if the user wants to select an image from the gallery. Instead of directing the user to the camera app, the application will open the gallery application, allowing the user to select the image from their gallery.

### 5.5.6 ReceiptScanner.java

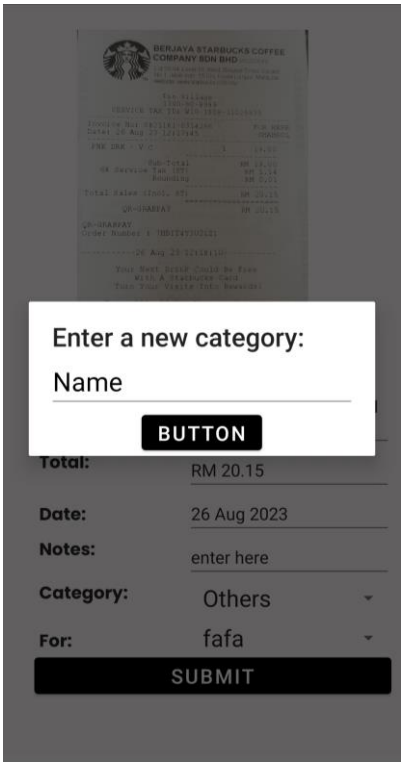



**Figure 5.5.6.4-ReceiptScanner(4/5)**

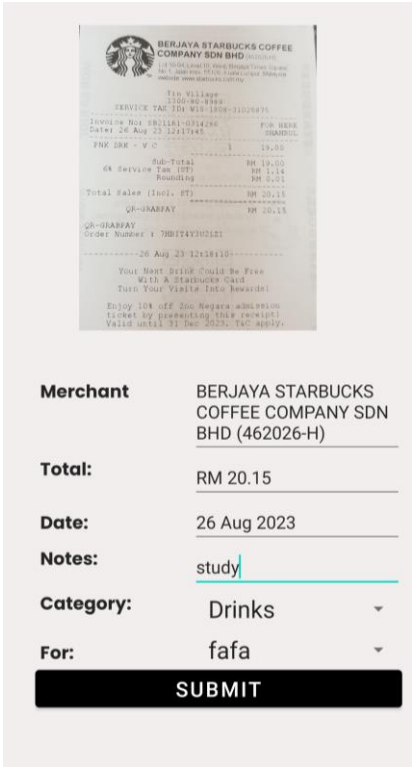



**Figure 5.5.6.5-ReceiptScanner(5/5)**

Moving on, the user can also add a new category based on their preferences, as shown in Figure 5.5.6.4. After the user has entered and edited all the necessary input, they can press the submit button to store their receipts in the application. Meanwhile, it will be recorded as an expense transaction for the user.

### 5.5.7 UserProfile.java

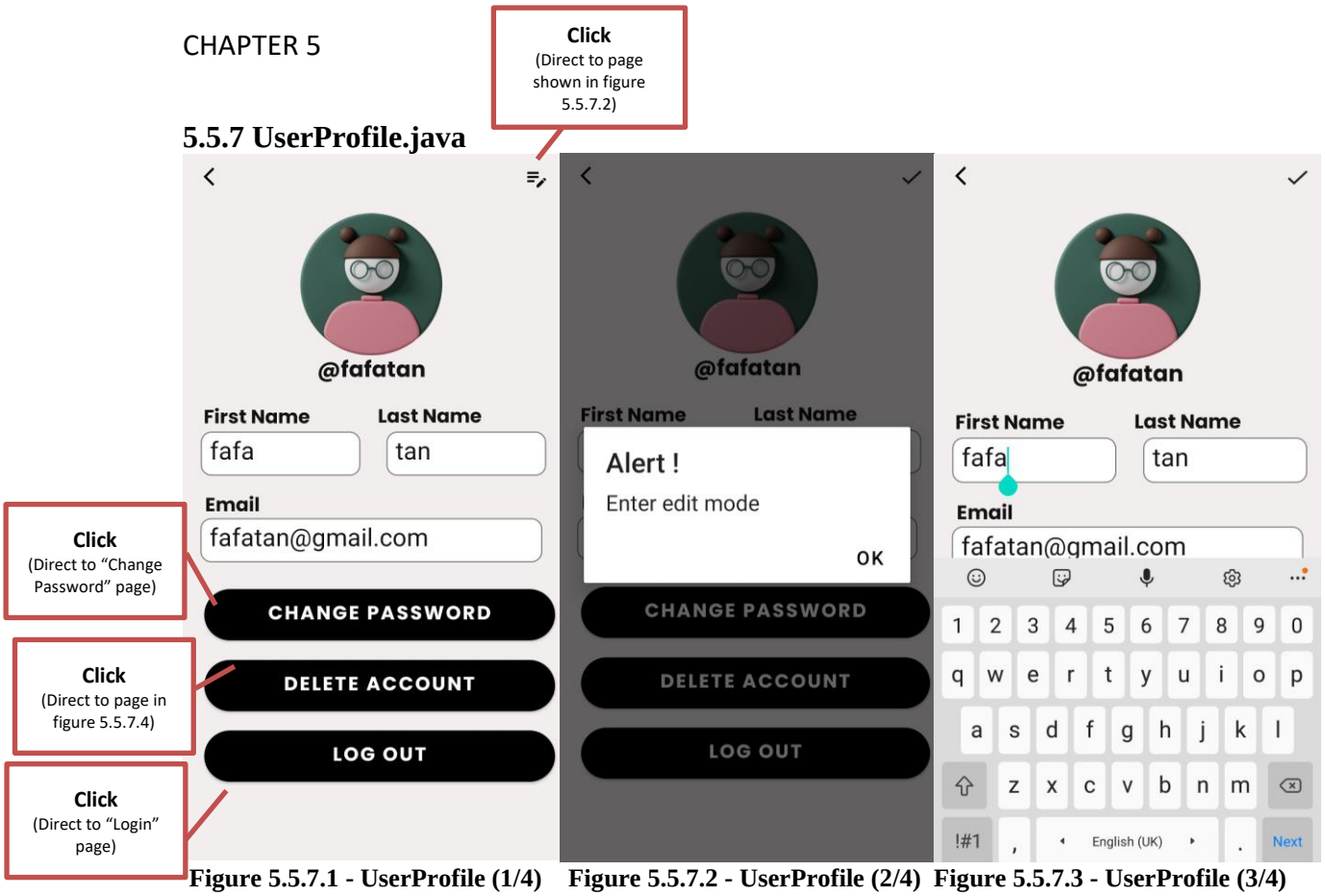



**Figure 5.5.7.1 - UserProfile (1/4)** **Figure 5.5.7.2 - UserProfile (2/4)** **Figure 5.5.7.3 - UserProfile (3/4)**

Figure 5.5.7.1 shows the user details of the application. In this page, the user can see their personal details such as their first name, last name, and email. Moreover, there are other functions as well such as changing their password, delete their account, log out and even allow the user to edit their personal details. The user can start the editing mode as shown in figure 5.5.7.2 by pressing on the top right corner icon. Then the user can start editing their details and once the user finish editing, he or she can press back the top right corner icon to exit from editing mode. All changes made by the user will be updated and saved.

### 5.5.8 ForgotPassword.java


**Figure 5.5.8.1-ForgotPassword (1/3)**

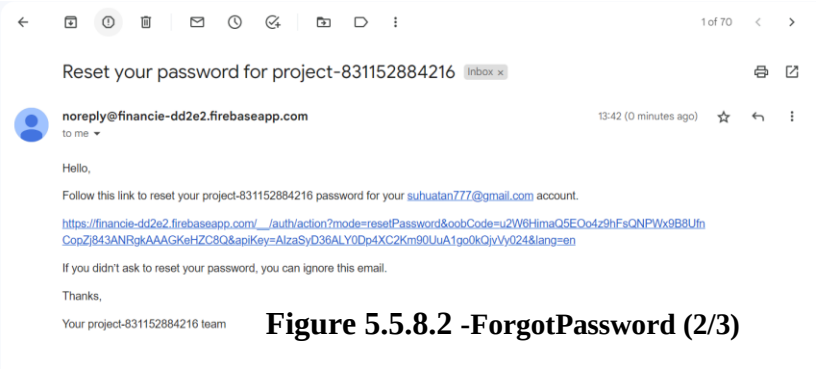



**Figure 5.5.8.2 -ForgotPassword (2/3)**

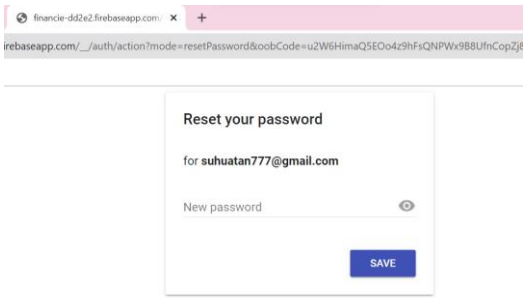



**Figure 5.5.8.3-ForgotPassword (3/3)**

Figure 5.5.8.1 displays the "Forgot Password" screen, which appears when the user clicks on the "Change Password" button in Figure 5.5.7.1 or the "Forgot Password?" text in Figure 5.5.1.1. Clicking on either of these options will direct the user to Figure 5.5.8.1, where they are prompted to enter their email address to verify its registration and validity. Once the user enters a valid email address, an email will be sent to the provided address, as shown in Figure 5.5.8.2. Clicking on the link in the email will lead the user to a screen depicted in Figure 5.5.8.3. Here, the user can enter a new password twice to confirm, which will then redirect them back to the login interface. This enables the user to log in to the system using their new password.

### 5.5.9 UserProfile.java and More.java

UserProfile.java

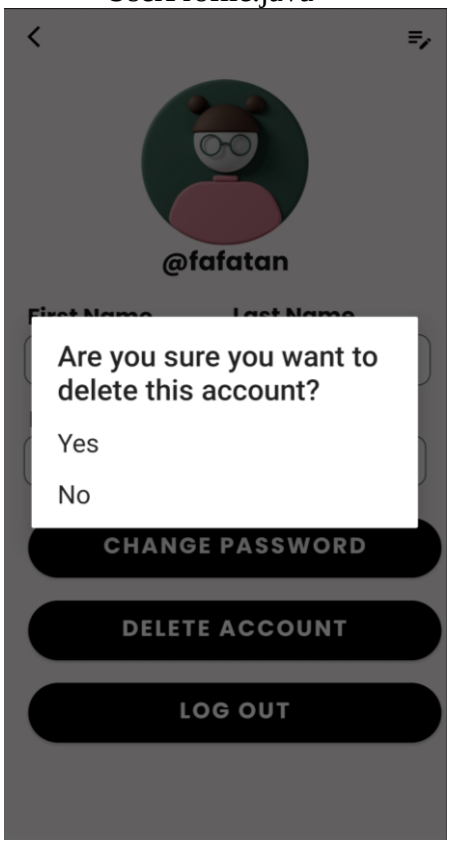



**Figure 5.5.7.4 -UserProfile (4/4)**

More.java

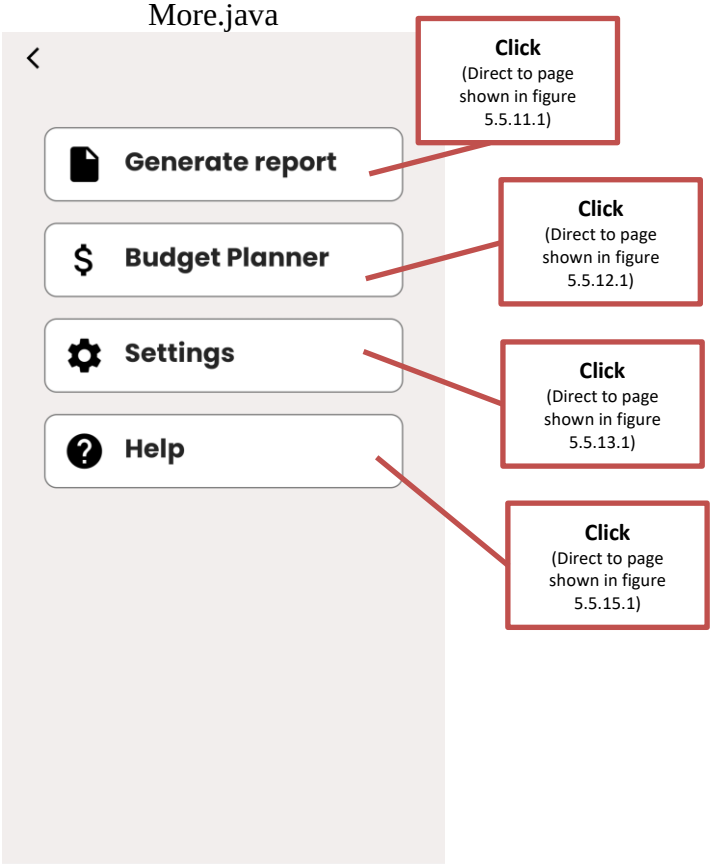
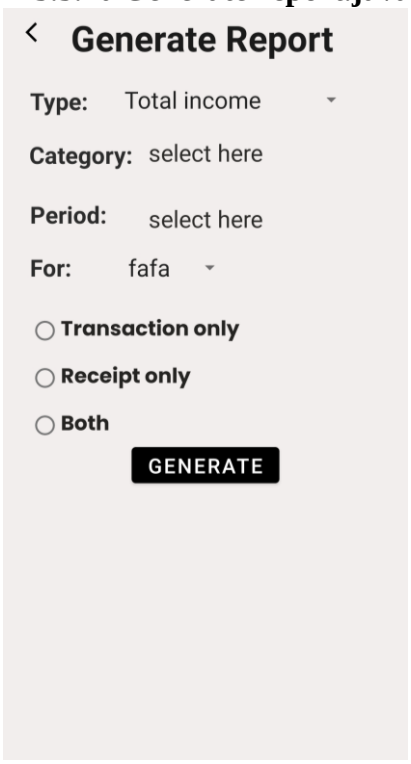
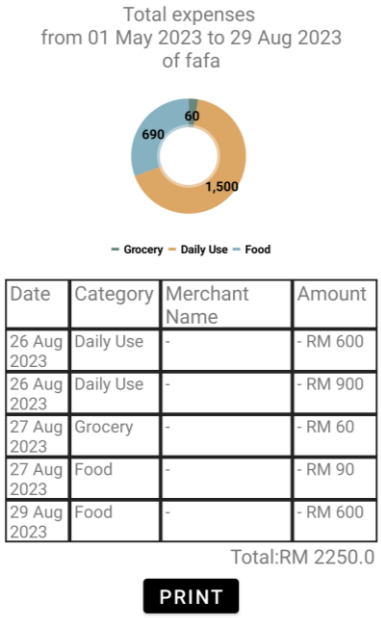



**Figure 5.5.9.1 - More**

The figure 5.5.7.4 shows an alert dialog box to prompt the user whether he or she confirm to delete the current account. If the user press yes, the account will be deleted and direct the user back to the login page. However, if the user press no, then nothing will happen. Moreover, figure 5.5.9.1 shows a list of actions user can choose under the more section when they press on the more icon at the bottom navigation bar in the main page.

### 5.5.10 GenerateReport.java



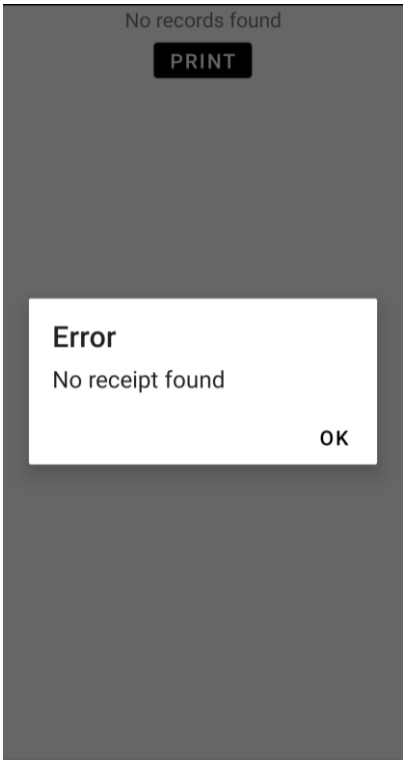




**Figure 5.5.10.1-GenerateReport (1/3) Figure 5.5.10.2-GenerateReport (2/3) Figure 5.5.10.3-GenerateReport (3/3)**

Figure 5.5.10.1 depicts the user interface of "GenerateReport.java" which is displayed when the user clicks on the "Generate Report" button in Figure 5.5.9.1. In the first figure, users can select the type, category, period, owner, and report type. Once the user has made their selections, the report they wish to see will be displayed, as shown in Figure 5.5.10.2. Figure 5.5.10.2 provides an example of the report, including important details about their spending presented through graphical representations. However, if the transaction the user wants to generate is unavailable or invalid, an error message will be shown, as illustrated in Figure 5.5.10.3.

### 5.5.11 BudgetPlanner.java

BudgetPlanner.java BudgetDetails.java AddBudget.java

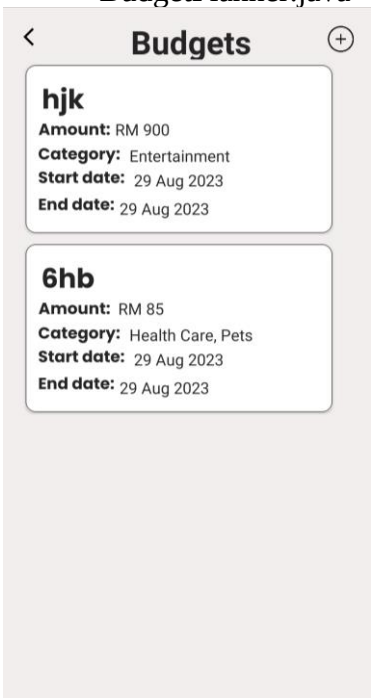



**Figure 5.5.11.1 - BudgetPlanner (1/3)**


**Figure 5.5.11.2 - BudgetPlanner (2/3)**

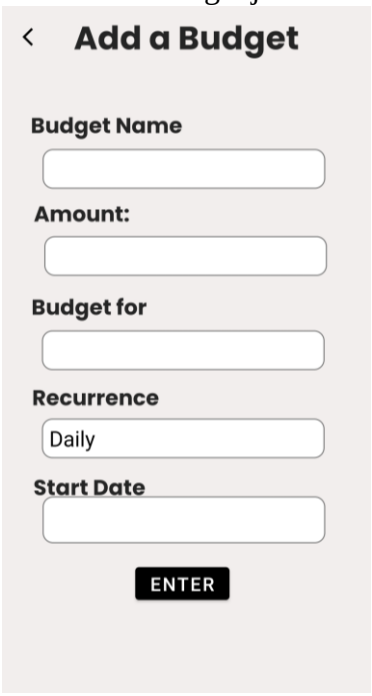
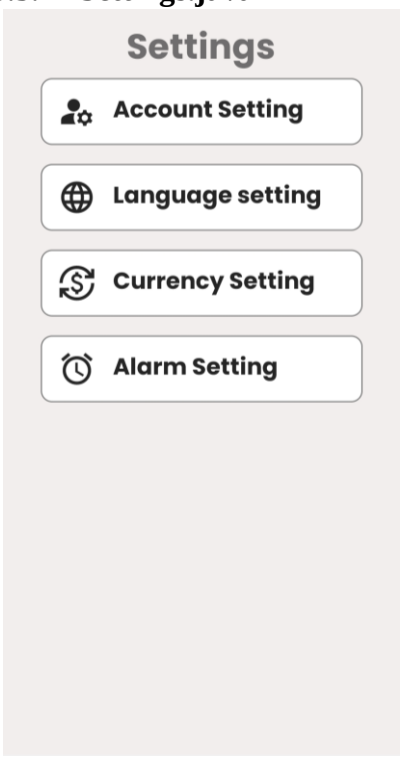



**Figure 5.5.11.3- BudgetPlanner (3/3)**

The three figures above show the budget module within this application. When the user first presses the "Budget planner" button in Figure 5.5.9.1, they will be directed to Figure 5.5.11.1. In Figure 5.5.11.1, the user can view some of the details of each budget in a recycler view. Whenever the user selects an item in the recycler view, it will display more detailed information and insights about the selected budget, as shown in Figure 5.5.11.2. Additionally, if the user taps on the top-right corner icon in Figure 5.5.11.1, it will navigate them to "AddBudget.java", as illustrated in Figure 5.5.11.3.

### 5.5.12 Settings.java



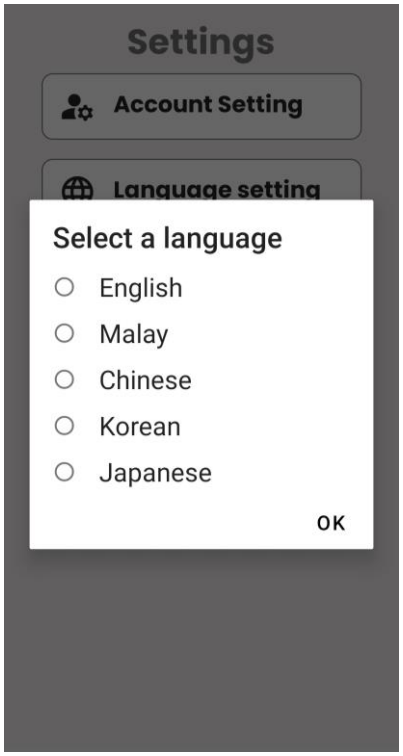





**Figure 5.5.12.1-Settings(1/7)** **Figure 5.5.12.2-Settings(2/7)** **Figure 5.5.12.3-Settings(3/7)**

The figures above shows different settings offered by the application such as account setting, language setting, currency setting and alarm setting. Whenever the user press on the account setting on figure 5.5.12.1, it will direct the user to the UserDetails.java. Whereas if the user choose to modify the language and currency settings, it will pop up a dialog box as shown in both figure 5.5.12.2 and 5.5.12.3. Moreover, there are lots of choices available for the user to choose. Thus, this can make them to have a personalized experience in using this application.


**Figure 5.5.12.4 - Settings(4/7)**


**Figure 5.5.12.5 - Settings(5/7)**


**Figure 5.5.12.6 - Settings(6/7)**


**Figure 5.5.12.7 - Settings(7/7)**

The figures above show the various languages applied to the system, namely Bahasa Melayu, Chinese, Korean, and Japanese, in accordance with the sequence presented in the figures.

### 5.5.13 AlarmSettings.java and Help.java

AlarmSettings.java

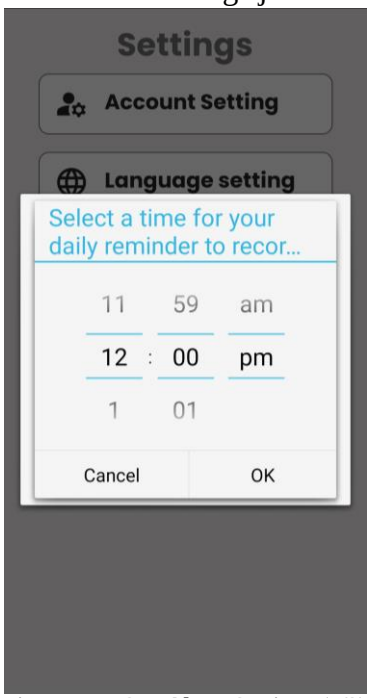



**Figure 5.5.13.1–AlarmSettings (1/2)**

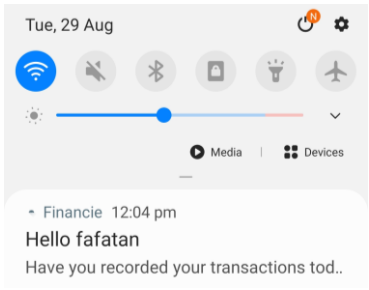



**Figure 5.5.13.2-AlarmSettings(2/2)**

Help.java

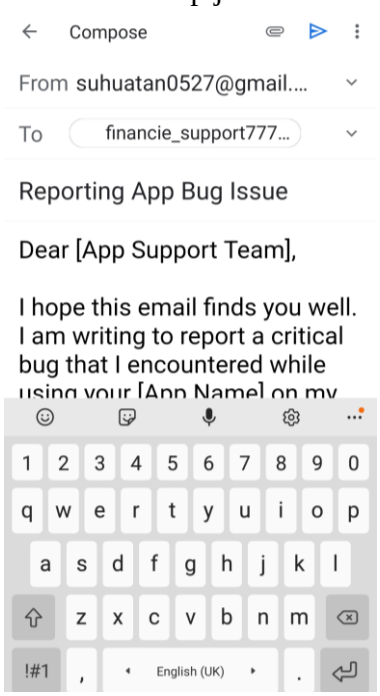



**Figure 5.5.14.1-Help**

Figure 5.5.14.1 displays the screen for selecting a daily reminder time for recording transactions. After setting the time, the application will send daily notifications based on the chosen time. Furthermore, pressing the “Help” button in Figure 5.5.9.1 will open the Gmail

app, as seen in Figure 5.5.14.1, providing a sample feedback email that users can choose to edit as they prefer.

## 5.6 Implementation Issues and Challenges

Implementing this project posed several challenges. To ensure the project's success, we needed to choose existing OCR tools or libraries, as developing OCR technology from scratch would be time-consuming and not cost-effective. Initially, we selected the Google Vision API. However, it did not yield the expected results. It extracted text column by column instead of row by row, which was not suitable for our project's primary goal of detecting receipt information line by line.

After several weeks of research, we discovered and employed another OCR tool called the Asprise Receipt OCR API. This tool automatically detects and categorizes information into key-value pairs. For example, it can automatically categorize details like the merchant's name, amount, and date, and return the result in JSON format. This result then becomes the user's expense transaction details. However, implementing and integrating this tool was not straightforward. We used a Python SDK called Chaquopy to connect the tool to the Android Studio IDE, which is primarily coded in Java. Although the Asprise Receipt OCR API supports various programming languages such as Java, C++, JavaScript, and Python, the Java code provided was outdated, with many functions deprecated or no longer functioning. Fortunately, we discovered Chaquopy, which enabled us to connect the Asprise Receipt OCR API Python code with the Android Studio IDE, written in Java. However, issues arose during the API implementation, such as problems with the Python version, Python API path, and library imports. Fortunately, we resolved all these issues.

Although the Asprise Receipt OCR API saved considerable time in manually preprocessing receipts, it introduced another challenge: hourly and daily usage quotas. If these limits were exceeded, the API would cease functioning, rendering it unusable after scanning receipts multiple times. Instead of providing extracted information, it would display an error message. This increased the difficulty level in implementing the project because, once the limits were exceeded, we had to wait for several hours or days to try the API again.

Apart from coding challenges, another challenge was associated with the use of Firebase. The database architecture was not properly planned and sketched before implementation in the

application. This complicated data retrieval, storage, and updates, requiring more time to sort out reference paths in the database when performing CRUD actions. Additionally, Firebase could only be accessed by connecting to Wi-Fi, limiting data retrieval, storage, and updates in offline scenarios. However, every storage method has its pros and cons. Choosing Firebase as the storage method for this project was ideal because it is user-friendly and allows users to view the database structure clearly.

## 5.7 Concluding Remark

In this chapter, several things are discussed, such as the project timeline, software, hardware setup, user interface, and many more. In short, the project was started on the 30th of January 2023 and ended on the 9th of September 2023. Moreover, several software and tools are required for developing the project, such as Android Studio IDE, Asprise Receipt OCR API, Chaquopy SDK, etc. These software and tools require proper setup before using them, and the configuration and setup process is also mentioned in this chapter. Furthermore, there are several screenshots of the user interface of the application, and each screenshot comes along with a detailed description, whereby most of the user interfaces are designed to increase and enhance the user experience of the application, such as the implementation of the three-click rule. This allows the user to access the major functions within three or fewer clicks. Lastly, the challenges and issues when making the project a success are also discussed in this chapter. Some of the challenges and issues include the choice and implementation of OCR tools, limitations of the chosen OCR tools, the architecture of the storage method using Firebase, and others.

# CHAPTER 6: SYSTEM EVALUATION AND DISCUSSION

## 6.1 System Evaluation

The completed system was tested and evaluated by 22 participants.

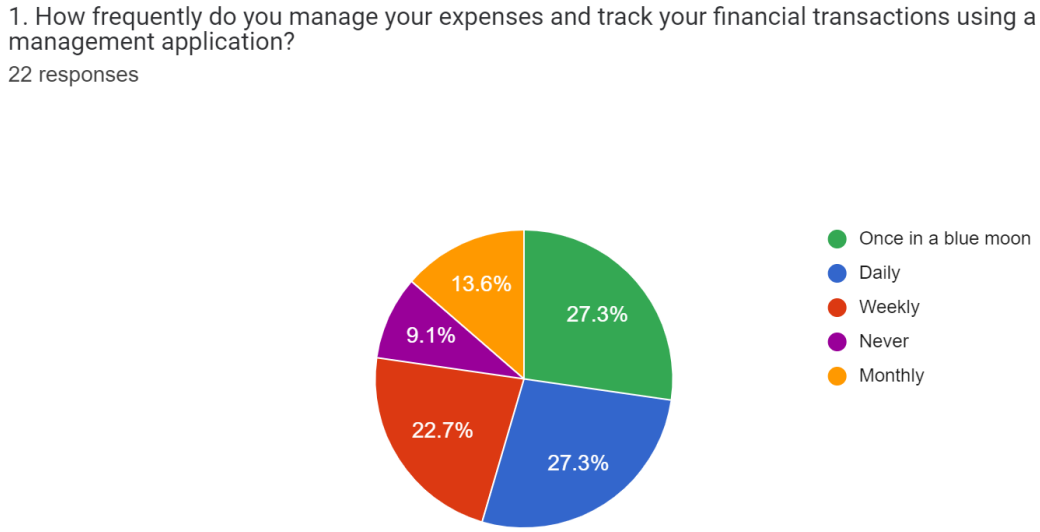



**Figure 6.1.1 – Result of question 1**

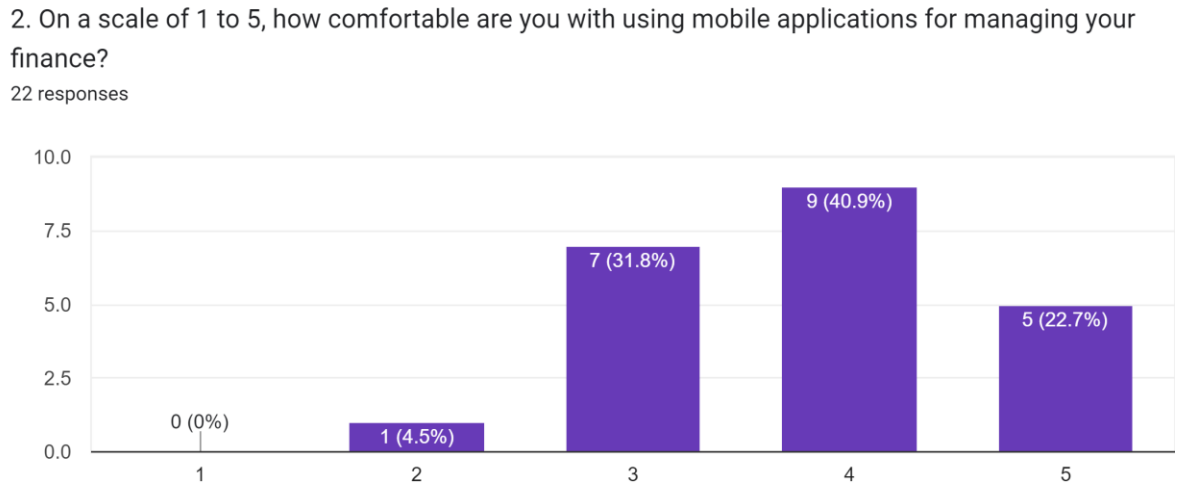



**Figure 6.1.2 -Result of question 2**

Based on the two figures above, we aim to understand how often users manage their finances and identify their experiences when using an existing finance management mobile application. According to Figure 6.1.1, the majority of participants either record their finances daily or only do so once in a while, with both categories accounting for 27.3%. Meanwhile, 22.7% of participants manage their finances weekly, and 13.6% manage them monthly. Additionally, 9.1% of participants never manage their finances. This question is designed to determine the frequency of financial management among the participants.

Moving on to question 2 of this survey, it shows that most participants (14 participants) are

comfortable using a mobile application to manage their finances. Only a minority of participants are not comfortable, and some remain neutral towards using a mobile application for financial management. Consequently, this indicates a potential demand and market for a finance management mobile application.

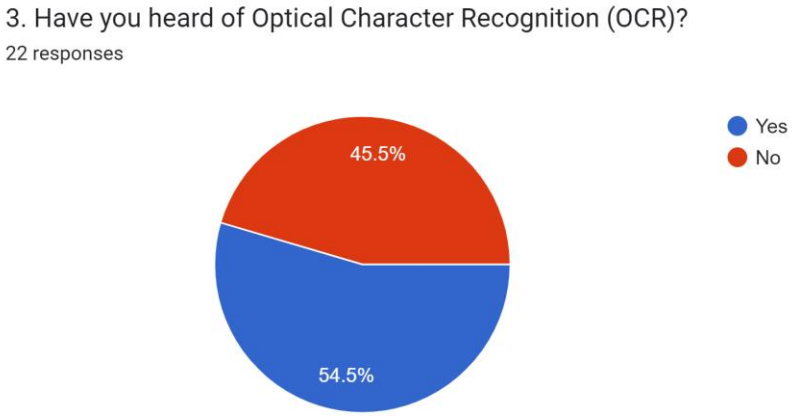



**Figure 6.1.3 – Result of question 3**

The third question of this survey aims to determine the participants' awareness of OCR. Based on the results, the majority of participants (54.5%) have heard of OCR, while the remaining 45.5% have not. If participants are familiar with OCR, they will proceed to question 5. However, if they have not heard of it, they will be directed to question 4.

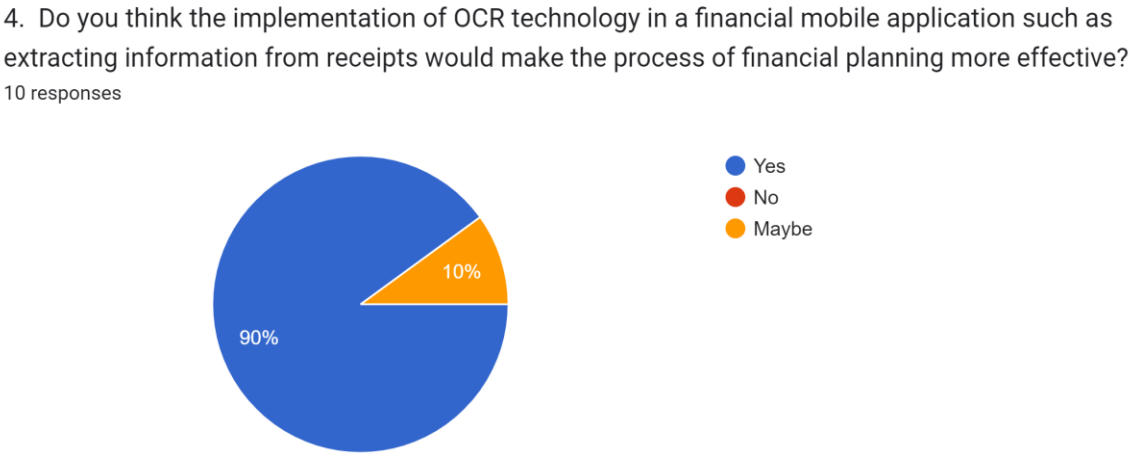



**Figure 6.1.4 – Result of question 4**

If the participants have selected No for the previous question, it will lead the user to this question which is question 4. In this question, OCR technology is explained in a simple and layman term and the aim of this question is to collect the user's thoughts towards the idea

of integrating OCR technology to a financial planning application. As shown in figure 6.1.4, most of the participants agree that by integrating OCR technology to a financial planning application will help the process of managing receipts more effective.

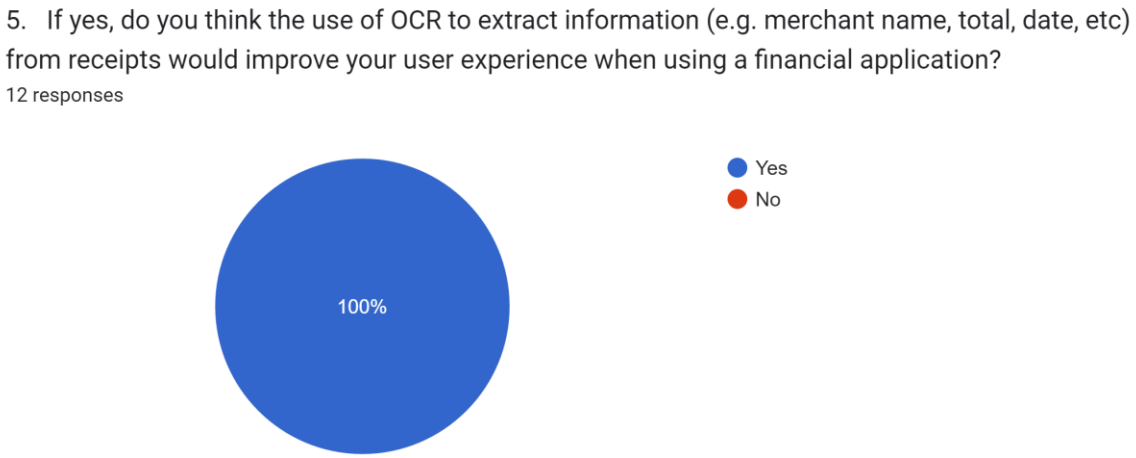



**Figure 6.1.5 – Result of question 5**

Question 5 is the continuous question of question 3 if the user choose "Yes" whereby they acknowledge the existence of OCR technology. Out of 12 responses from this question, all participants agreed that by integrating OCR technology in a financial application will help to enhance the experience of the users.

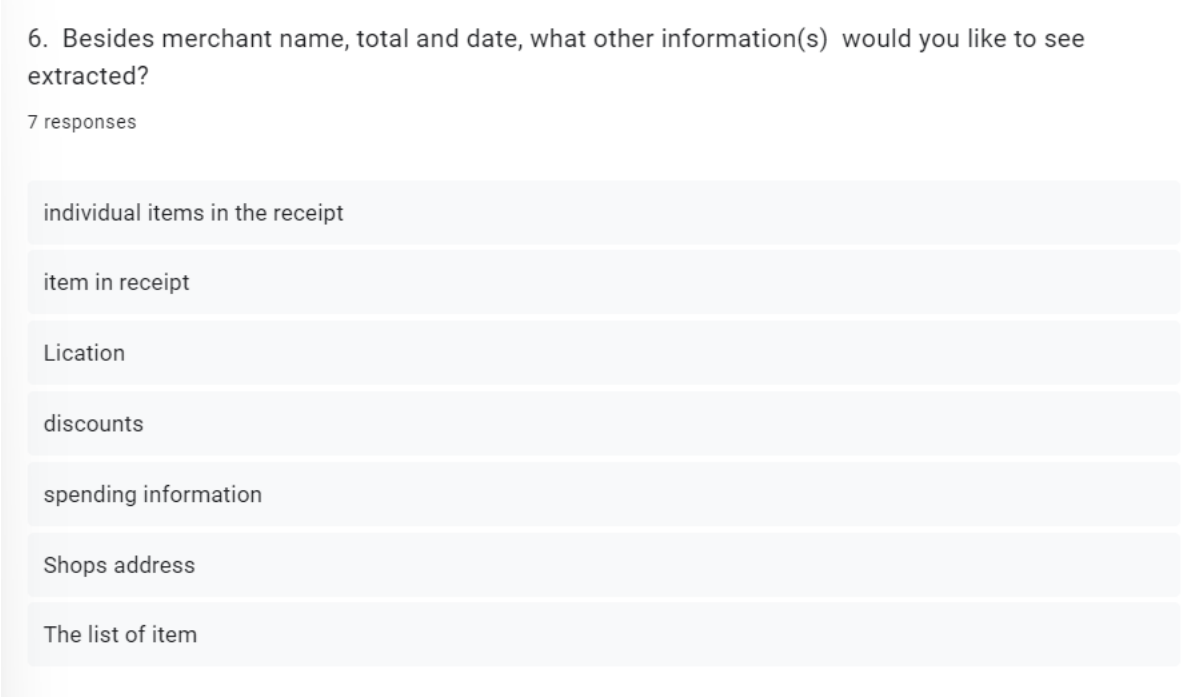



**Figure 6.1.6 – Result of question 6**

Moreover, question 6 aims to discover the participants' opinions and preferences regarding what additional details they would like to see extracted. Some participants have provided opinions, such as the ability to retrieve the list of items on a receipt, the discount percentage and rate, and the address of the merchant shop. Based on the opinions and suggestions provided by the participants, these are relevant and achievable for future development.

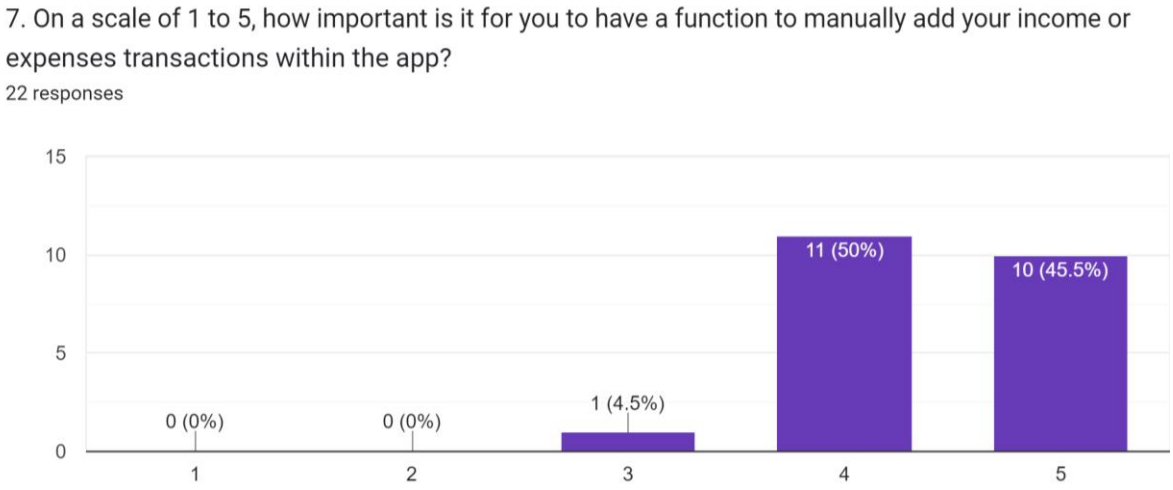



**Figure 6.1.7 – Result of question 7**

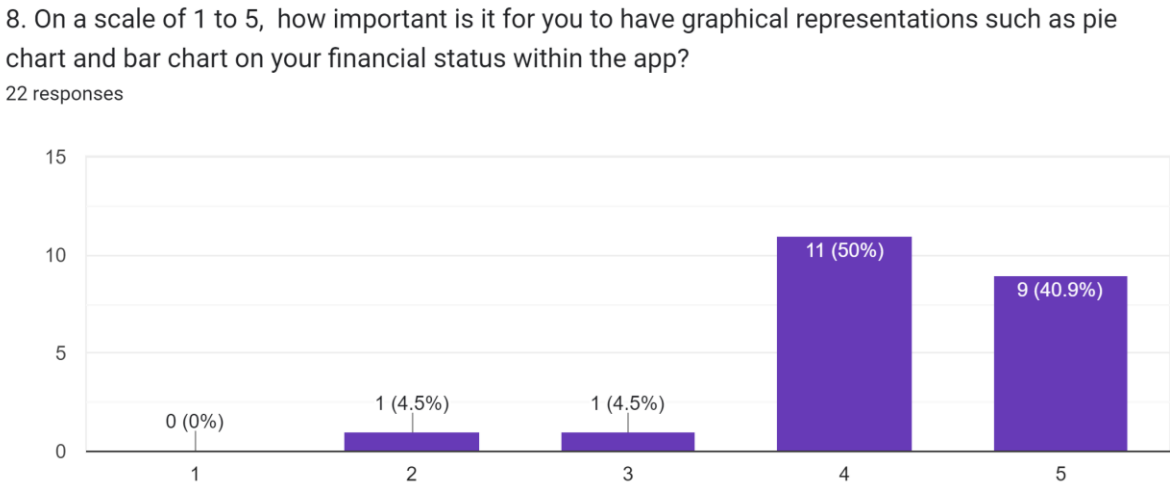



**Figure 6.1.8 – Result of question 8**

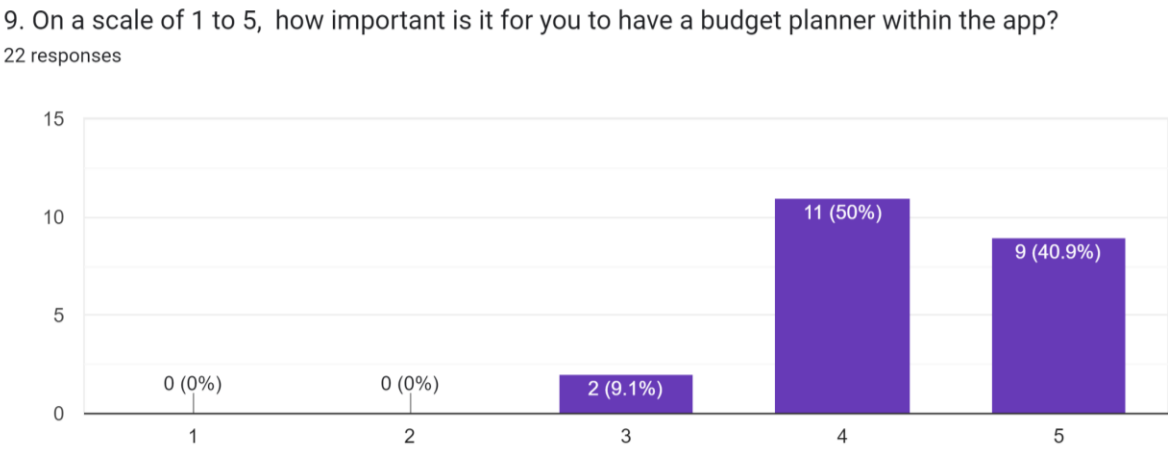



**Figure 6.1.9 – Result of question 9**

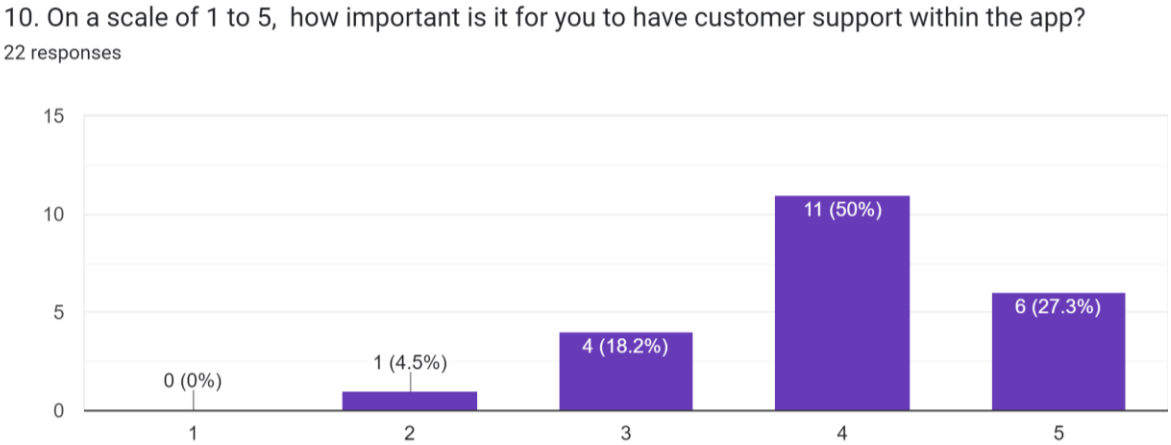



**Figure 6.1.10 – Result of question 10**

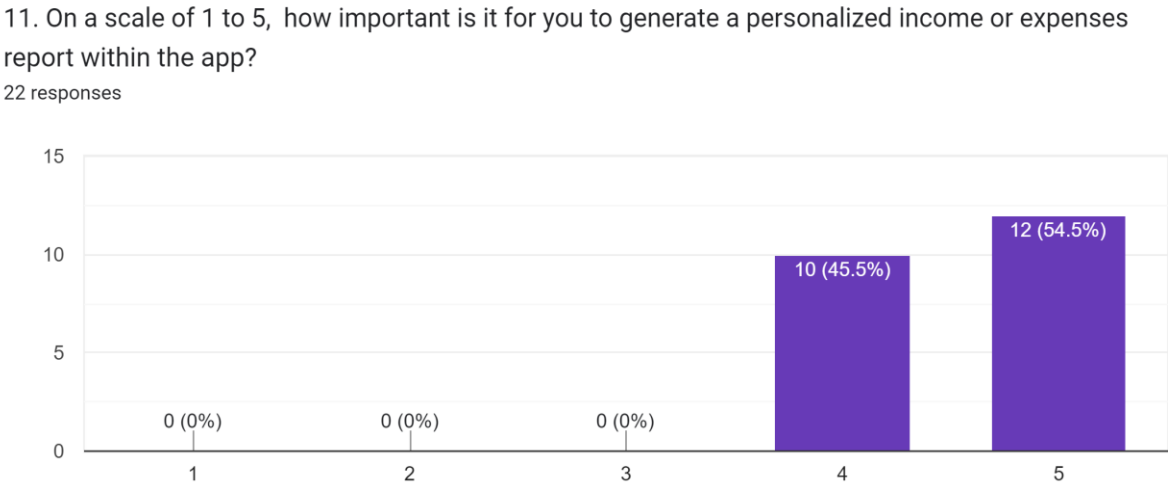



**Figure 6.1.11 – Result of question 11**

12. On a scale of 1 to 5, how important is it for you to choose your desired language and currency within the app?

22 responses

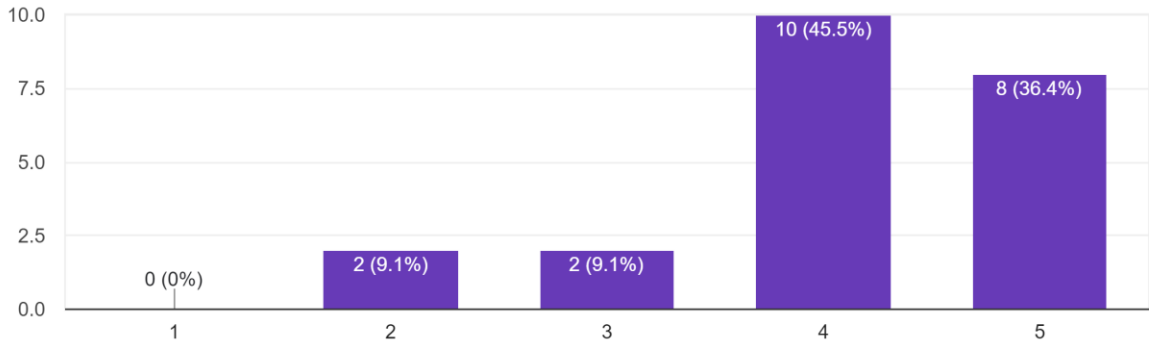



**Figure 6.1.12 – Result of question 12**

13. On a scale of 1 to 5, how important is it for you to generate personalized notifications to remind you to keep track of your transactions within the app?

22 responses

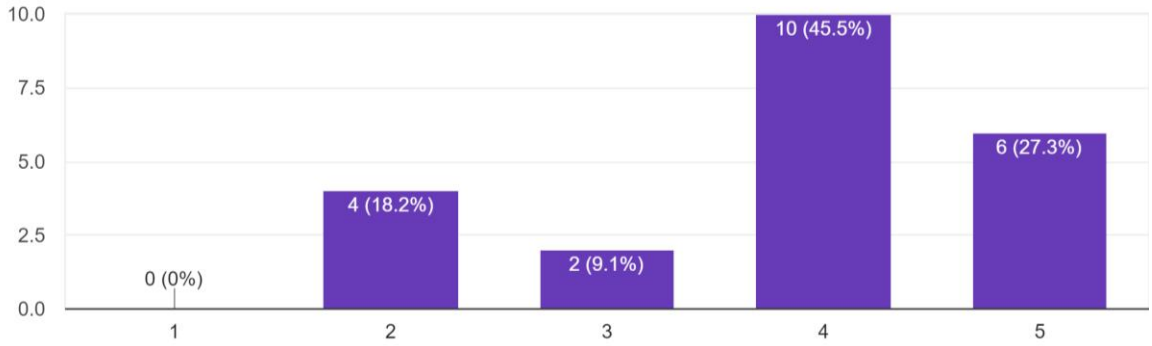



**Figure 6.1.13 – Result of question 13**

14. On a scale of 1 to 5, how would you rate the overall User Interface (UI) of this application?

22 responses

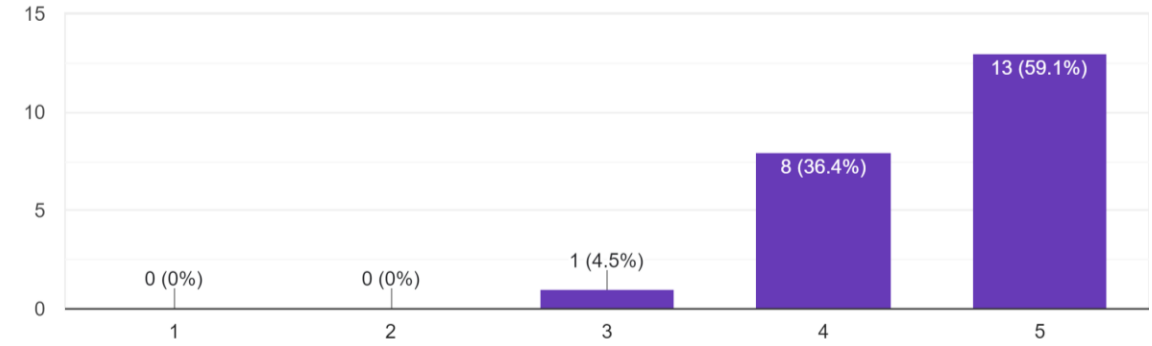



**Figure 6.1.14 – Result of question 14**

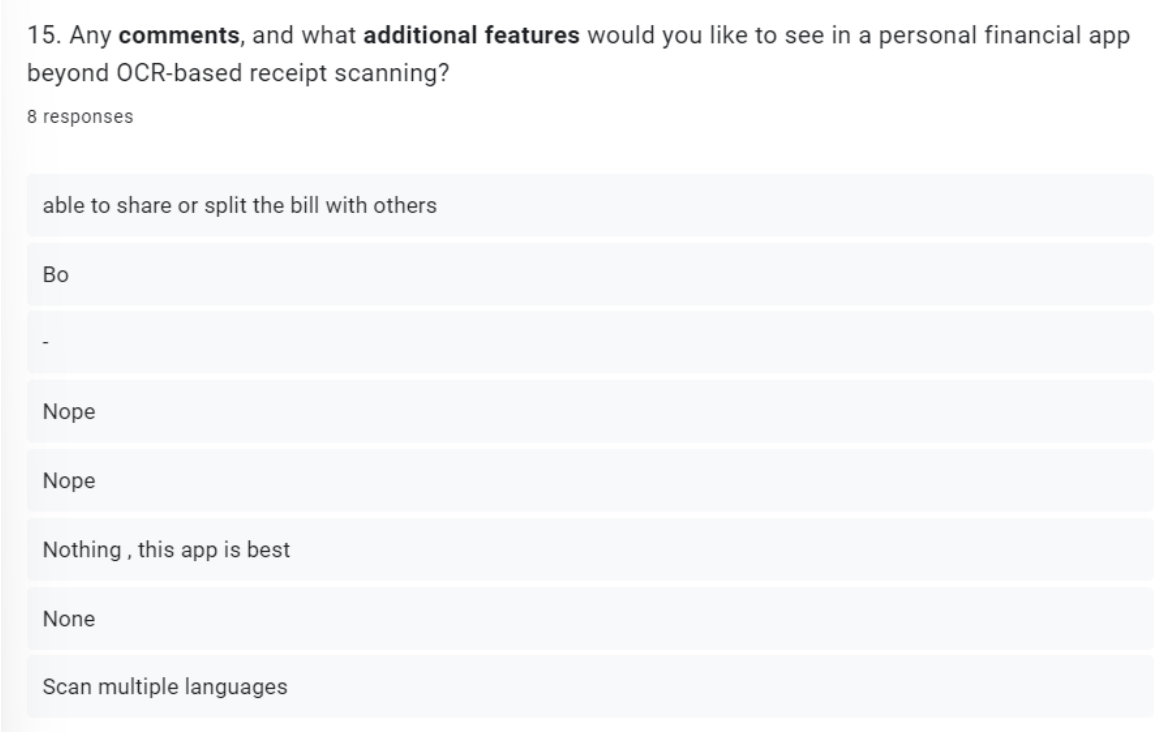



**Figure 6.1.15 – Result of question 15**

Question 7 to 14 are questions about the user's preferences towards the importance of the functionality and features by rating them on a scale of 1 to 5. These functionalities and features include adding transactions manually, having graphical representation, budget planner, customer support, the ability to generate a personalized report and notification, changing language and currency. Based on figures 6.1.7 to 6.1.13, most of the functions and features are considered important and very important to the participants. However, the functions of changing language and currency, and creating a personalized reminder notification, are less valued by the participants. According to figure 6.1.14, most of the participants are satisfied with the overall user interface of the proposed application, and some comments were provided by the participants as well, as shown in figure 6.1.15. Most of the feedback and comments provided have no comment on the application except for one participant; he or she has commented on another interesting function, which is the ability to share or split bills with others. This function can be considered in the future development of the application, as it is quite relevant and an essential function as well.

## 6.2 System Testing and Performance Metrics

**Login module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User enters a valid email and correct password | Direct to the Main page | Direct to the main page |
| User enters an invalid email or password | Show a toast message of invalid input | Show a toast message of invalid input |
| User clicks “Sign Up” | Direct the user to Sign up page | Direct the user to Sign up page |
| User clicks “Forgot Password” | Direct the user to Reset Password page | Direct the user to Reset Password page |

**Table 6.2.1 – Login module**

**Sign Up module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User fills up all fields with valid inputs. | Direct to the Main page | Direct to the main page |
| User fills up all fields with one or more invalid inputs | Show a toast message of invalid input | Show a toast message of invalid input |
| User clicks “Enter” | Direct the user to Main page | Direct the user to Main page |

**Table 6.2.2 – Sign Up module**

**Main module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User press on the top left corner profile picture | Direct to user detail page | Direct to user detail page |
| User clicks on the “View more” | Direct to transaction history page | Direct to transaction history page |
| User clicks on the “Month year” at the top center horizontally | Open a pop out box to allow user to select month and year for the pie chart and bar chart. | Open a pop out box to allow user to select month and year for the pie chart and bar chart. |
| User clicks on the “Home” icon at the navigation bar | Direct to main page | Direct to main page |
| User clicks on the “History” icon at the navigation bar | Direct to transaction history page | Direct to transaction history page |
| User clicks on the “Add” icon at the navigation bar | Direct to add transaction manually or scan page | Direct to add transaction manually or scan page |
| User clicks on the “Receipts” icon at the navigation bar | Direct to receipt history page | Direct to receipt history page |
| User clicks on the “More” icon at the navigation bar | Direct to more page | Direct to more page |
| User press on the left button | Change the fragment screen to bar chart | Change the fragment screen to bar chart |
| User press on the right button | Change the fragment screen to bar chart | Change the fragment screen to bar chart |

**Table 6.2.3 – Main module**

**Generate report module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User input value for all fields | Direct to PrintTransactionOnly page/ PrintReceiptOnly page/ PrintBoth page | Direct to PrintTransactionOnly page/ PrintReceiptOnly page/ PrintBoth page |
| User did not input value for all fields | Print error message | Nothing will happen |
| If there is no transaction or receipts meet the condition entered by the user | Pop out error message | Pop out error message |
| User clicks on the "Print" button | Convert the screen to PDF format and stored it in internal storage | Convert the screen to PDF format and stored it in internal storage |

**Table 6.2.4 – Generate Report module**

**Budget Planner module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User input value for all fields | Direct to Budge planner page which display some details of the budget in recycler view | Direct to Budge planner page which display some details of the budget in recycler view |
| User did not input value for all fields | Print error message | Nothing will happen |
| user overspent during the budget period | Send notification to alert user | Send notification to alert user |

**Table 6.2.5 – Budget Planner module**

**Transaction History module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User taps on the top right corner edit icon at Transaction details page | Enable editing mode and send pop out message showing the user can start to edit | Enable editing mode and send pop out message showing the user can start to edit |
| User taps on the item in the recycler view | Direct user to the Transaction details and show more information of the transaction | Direct user to the Transaction details and show more information of the transaction |
| User search transaction from the search view | Display the searched transaction | Display the searched transaction |

**Table 6.2.6 – Transaction History module**

**Receipt History module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User taps on the top right corner edit icon at Receipt details page | Enable editing mode and send pop out message showing the user can start to edit | Enable editing mode and send pop out message showing the user can start to edit |
| User taps on the item in the recycler view | Direct user to the receipt details and show more information of the transaction | Direct user to the receipt details and show more information of the transaction |
| User search receipt from the search view | Display the searched receipt | Nothing displayed |

**Table 6.2.7 – Receipt History module**

**Language setting module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User chooses “English” for the system language | Change the system to English and direct user to main page | Change the system to English and direct user to main page |
| User chooses “Bahasa Melayu” for the system language | Change the system to Bahasa Melayu and direct user to main page | Change the system to Bahasa Melayu and direct user to main page |
| User chooses “Chinese” for the system language | Change the system to Chinese and direct user to main page | Change the system to Chinese and direct user to main page |
| User chooses “Korean” for the system language | Change the system to Korean and direct user to main page | Change the system to Korean and direct user to main page |
| User chooses “Japanese” for the system language | Change the system to Japanese and direct user to main page | Change the system to Japanese and direct user to main page |

**Table 6.2.8 – Language setting module**

**User details module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User taps on the top right corner edit icon at user details page | Enable editing mode and send pop out message showing the user can start to edit | Enable editing mode and send pop out message showing the user can start to edit |
| User clicks on the “Change password” button | Direct user to reset password page | Direct user to reset password page |
| User clicks on the “Log out” button | Direct user to login page | Direct user to login page |
| User clicks on the “Delete account” button | Show a dialog box to confirm to delete the account and direct to login page | Show a dialog box to confirm to delete the account and direct to login page |

**Table 6.2.9 – User details module**

**Alarm setting module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User selects a time for reminder | Send notification to remind the user to record a transaction based on the selected time set by the user | Send notification to remind the user to record a transaction based on the selected time set by the user |
| User did not select a time for reminder | No notification | Notification appear |

**Table 6.2.10 – Alarm setting module**

**Currency setting module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User chooses "RM" for the system currency | Change the whole system's currency to "RM" and direct user to main page | Change only the subsequent system's currency to "RM" and direct user to main page |
| User chooses "$" for the system currency | Change the whole system's currency to "$" and direct user to main page | Change only the subsequent system's currency to "$" and direct user to main page |
| User chooses "€" for the system currency | Change the whole system's currency to "€" and direct user to main page | Change only the subsequent system's currency to "€" and direct user to main page |
| User chooses "¥" for the system currency | Change the whole system's currency to "¥" and direct user to main page | Change only the subsequent system's currency to "¥" and direct user to main page |
| User chooses "£" for the system currency | Change the whole system's currency to "£" and direct user to main page | Change only the subsequent system's currency to "£" and direct user to main page |

**Table 6.2.11 – Currency Settings module**

**Help module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User edits or inputs feedback message | Send feedback email to financie777@gmail.com | Send feedback email to financie777@gmail.com |
| User taps on the "Help" button | Direct user to the Gmail app | Direct user to the Gmail app |

**Table 6.2.12 – Help module**

**Change password module**

| Input | Expected Output | Actual Output |
|---|---|---|
| User enters a valid and registered email | System sends a reset email to the email inputted by the user | System sends a reset email to the email inputted by the user |
| User enters an invalid and registered email | Show an error message | Nothing will happen |

**Table 6.2.13 – Change Password module**

## 6.3 Project Challenges

In addition to the implementation issues and challenges discussed in Chapter 5.6, several other project challenges deserve attention. Firstly, the accuracy of the OCR is heavily reliant on both the quality of the provided image and the smartphone's camera quality. Factors like shadows, noise, and unclear or faded characters on receipts can significantly reduce the OCR's accuracy, consequently impacting the precision of transaction details.

Furthermore, there is a limited availability of free OCR tools on the internet. While some alternatives exist online, they typically come with a price, and many of these solutions are not budget friendly. This presents an additional challenge due to budget constraints in the project.

Moreover, as observed from the survey questions, some participants infrequently manage their finances, and as mentioned in the problem statement from chapter 1, certain individuals may not retain their receipts. Consequently, there is a risk that a percentage of people may not see the need to use a financial planning application with integrated OCR technology. They might use a basic financial planning mobile applications that don't require these advanced functions instead of the proposed application. This poses a challenge in encouraging broader adoption of the project among potential users.

## 6.4 Objectives Evaluation

There are three main objectives outlined in Chapter 1 of the report for this project:

1. To enable individuals to plan their financial matters effectively and conveniently.
2. To manage receipts with the implementation of OCR.
3. To provide users with insightful graphical representation and documentation.

The final output of this project successfully achieves all three objectives. For example, the application incorporates functions and features such as manual transaction entry, a search view for transaction history, and a budget planner. These functions and features enable individuals to plan their finances effectively and conveniently. Additionally, the integration of OCR technology into the application streamlines the process of managing receipts that may become unusable in the future. Users can easily search for and categorize receipts without the need for manual management.

Furthermore, the proposed application utilizes graphical representation, including pie charts, bar charts, and personalized reports, to provide users with valuable insights into their finances. Another noteworthy feature of the application is the use of simple and pleasant icons and images. These visual elements help users reduce tension and stress when using the application for financial management.

## 6.5 Concluding Remark

In this chapter, several things are discussed such as the survey results and evaluation, system performance metrics, project challenges, and evaluation of objectives. In conclusion, there are 22 participants participated in this survey whereby they have tried use the application and have given evaluation and thoughts by filling out the survey. This survey is essential as it helps us to identify the thoughts and perspectives of the potential user of the application and give us insights to improve for future development. Moreover, the performance metrics are also displayed in chapter 6.2 and in a table format whereby it gave an overview of the performance of the application module by module. Next, the project challenges are also mentioned and some of the challenges are the limited choice of OCR tools, accuracy of the OCR result, and the threats from existing competitors. Lastly, the objectives mentioned in chapter 1 are evaluated in detail and the application is able to achieve all three objectives.

# CHAPTER 7 CONCLUSION AND RECOMMENDATION

## 7.1 Conclusion

The final output of this project is an Android-based personal financial planning mobile application that integrates OCR technology for automated receipt management and expense tracking. The primary objective of this project is to empower individuals to effectively plan and manage their finances and receipts. Furthermore, Chapter 1 discusses three problem statements: individuals struggle to keep track of their finances, lack efficient receipt management, and often lack real-time data visualization of their financial status. The project's objectives align with these problems, aiming to enable individuals to plan their finances, efficiently manage receipts using OCR, and provide users with insightful graphical representations and convenient documentation. Additionally, Chapter 1 covers other aspects such as contributions, direction, and the report's structure.

In Chapter 2 of this report, it reviews four popular finance management applications: Spendee, Money Manager, Money Lover, and Expensify. Each of these applications has its own strength and downside. Moreover, this chapter includes a comparison between the functions and features of the proposed application and these existing applications are created and displayed in a table format. This comparison table listed functions and features that helps to identify mandatory features in a finance management application and acknowledge the proposed application's strengths and weaknesses.

Chapter 3 discusses the system methodology of the proposed application, including essential details such as system architecture diagrams, use case diagrams, and activity diagrams. These diagrams provide a comprehensive understanding of how each component of the application is interconnected. In Chapter 4, the system design of the proposed application, such as system block diagrams, system flow diagrams, and descriptions, is presented. This chapter also explains and highlights some of the application's code.

Additionally, Chapter 5 documents the implementation of the system, including the project timeline, setup and configuration, screenshots of the user interface, and challenges encountered during the project's implementation. Lastly, chapter 6 covers performance metrics, system and

objective evaluations, and project challenges whereby it discusses the expected and actual outputs of the project and problems encountered during the whole project period.

In summary, the project's final output is the mobile application "Financie," which includes various functions and features such as manual transaction entry or receipt scanning, budget planning, personalized financial report generation, graphical cash flow overview, language and currency customization, and more. These features enhance the user experience in managing and planning their finances and also promoting the cultivation of financial management habits.

## 7.2 Recommendations

After collecting feedback from potential users of this application, several enhancements can be considered for future development. These include adding a new function that enables users to split their finances or receipts with others, such as friends and family. This can help to attract more users and improve expense management of the individual. In addition, improvements like listing items purchased from receipts in an organized manner can provide users with more insights and easier for future reference.

Apart from the user recommendations, other essential functions that can enhance the user experience include bill reminders, real-time financial advice, possibly through AI chatbot integration, and the ability to manage and track crypto wallets or e-wallets. These recommendations can further provide a seamless and enhanced user experience while using the application.

## REFERENCES

[1] C. Ben Geier, “What is financial planning? definition, meaning and purpose,” SmartAsset.com, https://smartasset.com/financial-advisor/financial-planning-explained (accessed Sep. 10, 2023).

[2] “5 steps to Financial Planning Success: Deloitte Ireland: Deloitte Private: Financial, Planning,” Deloitte Ireland, https://www2.deloitte.com/ie/en/pages/deloitte-private/articles/5-steps-financial-planning-success.html (accessed Sep. 10, 2023).

[3] D. Winer, “10 ways your life improves by saving money,” *Refresh Financial*, 12-May-2020. [Online]. Available: https://refreshfinancial.ca/blog/financial-news-and-advice/10-ways-saving-money-can-better-your-life/. [Accessed: 13-Apr-2023].

[4] “High cost of living, poor financial planning are the main reasons Malaysians fail to repay loans - Ahmad Maslan,” The Edge Malaysia, https://theedgemalaysia.com/node/672551 (accessed Sep. 10, 2023).

[5] E. Tyson, *Personal Finance for Dummies*. Hoboken, NJ: John Wiley & Sons, Inc., 2019.

[6] J. LaBianca, “8 reasons to never throw out your receipts, experts warn,” *Best Life*, 02-Feb-2023. [Online]. Available: https://bestlifeonline.com/keep-receipts-news/. [Accessed: 13-Apr-2023].

[7] WellyBox, “What is a tax receipt?: WellyBox - automated expense management,” *WellyBox*, 25-Nov-2022. [Online]. Available: https://www.wellybox.com/blog/what-is-a-tax-. [Accessed: 13-Apr-2023].

[8] Infogram, “90% of information transmitted to the brain is visual,” *Infogram*. [Online]. Available: https://infogram.com/90-of-information-transmitted-to-the-brain-is-visual-1gdk8pdr6q6gmq0. [Accessed: 13-Apr-2023].

[9] M. Peterman, “15 statistics that prove the power of Data Visualization,” csg solutions, https://blog.csgsolutions.com/15-statistics-prove-power-data-visualization (accessed Sep. 10, 2023).

[10] “50 data visualization statistics that prove its importance,” Visme Blog, https://visme.co/blog/data-visualization-statistics/ (accessed Sep. 10, 2023).

[11] Canara, “What are the benefits of financial planning?,” Sep-2021. [Online]. Available: https://www.canarahsbclife.com/blog/financial-planning/what-are-the-benefits-of-financial-planning.html. [Accessed: 13-Apr-2023].

[12] Y. ZHANG, H. SHU, H. LIU, S. WANG, and X. ZHANG, “Research on application of data visualization in Finance,” *DEStech Transactions on Engineering and Technology Research*, no. acaai, 2020. doi:10.12783/dtetr/acaai2020/34213

[13] M. A. Awel and A. I. Abidi, “REVIEW ON OPTICAL CHARACTER RECOGNITION,” *International Research Journal of Engineering and Technology (IRJET) e*, vol. 6, no. 6, Jun. 2019.

[14] “Optical character recognition (OCR): Definition & how to guide,” Optical Character Recognition (OCR): Definition & How To Guide, https://www.v7labs.com/blog/ocr-guide#:~:text=Optical%20Character%20Recognition%20(OCR)%20is,of%20what%20the%20document%20conveys. (accessed Sep. 10, 2023).

[15] Dhanashree, “Learn about bounding boxes in image processing: Nanonets,” Nanonets Intelligent Automation, and Business Process AI Blog, https://nanonets.com/blog/image-processing-and-bounding-boxes-for-ocr/ (accessed Sep. 10, 2023).

[16] I. Sydorenko, “OCR with deep learning,” High quality data annotation for Machine Learning, https://labelyourdata.com/articles/ocr-with-deep-learning#:~:text=OCR%20uses%20various%20algorithms%20such,to%20recognize%20text%20from%20images. (accessed Sep. 10, 2023).

[17] “OCR for RPA: Bots now understand unstructured data in 2023,” AIMultiple, https://research.aimultiple.com/rpa-ocr/ (accessed Sep. 10, 2023).

[18] IEEE Xplore Full-text PDF:, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7169508 (accessed Sep. 9, 2023).

[19] S. P. E. N. D. E. E. a.s. www.spendee.com, “Money manager & budget planner,” *Spendee*. [Online]. Available: https://www.spendee.com/. [Accessed: 22-Apr-2023].

[20] Spendee, “What is spendee?,” *Spendee Help Center*. [Online]. Available: https://help.spendee.com/article/114-what-is-spendee. [Accessed: 13-Apr-2023].

[21] “Money manager expense & budget,” *Expense & Budget*. [Online]. Available: https://www.realbyteapps.com/. [Accessed: 22-Apr-2023].

[22] “Spend Management software for receipts & expenses,” *Expensify*. [Online]. Available: https://www.expensify.com/. [Accessed: 22-Apr-2023].

[23] L. Finsify Technology Co., “How to be a financially successful person?: Money lover,” *Moneylover*. [Online]. Available: https://moneylover.me/. [Accessed: 22-Apr-2023].

# APPENDIX

## FINAL YEAR PROJECT BIWEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year:** Y3T2 | **Study week no.:**2 |
| **Student Name & ID:** Tan Su Hua & 21ACB00596 | |
| **Supervisor:** Mr Su Lee Seng | |
| **Project Title:** Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking | |

**1. WORK DONE**

- discuss about what can be done to improve the current system
- made some modification on the user interface of the system
- refined my title

**2. WORK TO BE DONE**

- coding for generating personalized report

**3. PROBLEMS ENCOUNTERED**

- none

**4. SELF EVALUATION OF THE PROGRESS**

Progressing well

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT BIWEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year:** Y3T2 | **Study week no.:**4 |
| **Student Name & ID:** Tan Su Hua & 21ACB00596 | |
| **Supervisor:** Mr Su Lee Seng | |
| **Project Title:** Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking | |

**1. WORK DONE**

- made some modification on the user interface of the system
- added new attributes on adding transaction module

**2. WORK TO BE DONE**

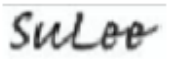
- coding for generating personalized report
- start FYP2 report

**3. PROBLEMS ENCOUNTERED**

- only able to generate certain report

**4. SELF EVALUATION OF THE PROGRESS**

So far so good





Supervisor's signature

Student's signature

# FINAL YEAR PROJECT BIWEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year:** Y3T2 | **Study week no.:**6 |
| **Student Name & ID:** Tan Su Hua & 21ACB00596 | |
| **Supervisor:** Mr Su Lee Seng | |
| **Project Title:** Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking | |

**1. WORK DONE**

- fixed the change password module
- finish generate report module
- finish filling out the brief personal information of the FYP2 report

**2. WORK TO BE DONE**

- coding for budgeting module
- introduction of the FYP2 report

**3. PROBLEMS ENCOUNTERED**

- none

**4. SELF EVALUATION OF THE PROGRESS**

Progressing well

SuLee

Suhua

Supervisor's signature

Student's signature

# FINAL YEAR PROJECT BIWEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year:** Y3T2 | **Study week no.:**8 |
| **Student Name & ID:** Tan Su Hua & 21ACB00596 | |
| **Supervisor:** Mr Su Lee Seng | |
| **Project Title:** Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking | |

**1. WORK DONE**

- finish the chapter 1 of the report
- finish chapter 2 of the report

**2. WORK TO BE DONE**

- finish chapter 3 of the FYP2 report
- coding for budget module
- coding for settings

**3. PROBLEMS ENCOUNTERED**

- none

**4. SELF EVALUATION OF THE PROGRESS**

Still okay

SuLee

Suhua

Supervisor's signature

Student's signature

# FINAL YEAR PROJECT BIWEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year:** Y3T2 | **Study week no.:**10 |
| **Student Name & ID:** Tan Su Hua & 21ACB00596 | |
| **Supervisor:** Mr Su Lee Seng | |
| **Project Title:** Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking | |

**1. WORK DONE**

- finish coding of all modules

**2. WORK TO BE DONE**

- finish chapter 4 of report
- finish chapter 5 of report

**3. PROBLEMS ENCOUNTERED**

- none

**4. SELF EVALUATION OF THE PROGRESS**

Progressing okay

SuLee

Suhua

Supervisor's signature

Student's signature

# FINAL YEAR PROJECT BIWEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year:** Y3T2 | **Study week no.:**12 |
| **Student Name & ID:** Tan Su Hua & 21ACB00596 | |
| **Supervisor:** Mr Su Lee Seng | |
| **Project Title:** Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking | |

**1. WORK DONE**

- finish the report

**2. WORK TO BE DONE**

- prepare for the presentation

**3. PROBLEMS ENCOUNTERED**

- none

**4. SELF EVALUATION OF THE PROGRESS**

Progressing well

SuLee

Suhua

Supervisor's signature

Student's signature

# POSTER

## DEVELOPMENT OF PERSONAL FINANCE MOBILE APPLICATION WITH OCR TECHNOLOGY

for Automated Receipt Management and Expense Tracking

### Introduction

Managing receipts has always been troublesome and time-consuming especially when using them to track our daily financial transactions. Nevertheless, keeping and managing receipts is very important as they can be used for many purposes such as tax audit purposes, as evidence of goods purchased, using it to return or refund goods purchased, etc. Hence, this project aims to help people to manage their receipts using Optical Character Recognition (OCR) technology.

### Objectives

- To enable individuals to effectively and conveniently plan their financial matters
- To manage receipts with the implementation of OCR
- To provide a user with insightful graphical representation and documentation

### Project Discussion

In this project, a personal finance mobile application named "Finceipt" will be developed. The application name comes from the word finance and reciept . It is the combination of both words. Finceipt has several functions such as keeping track of user's cash flow, budgeting, generate report, scanning receipt via OCR, etc.

### Conclusion

As a conclusion, the development of “Financie” aims to help users to manage and plan their finance and receipts in a simple and easy way. Moreover, OCR technology is integrated in the application in order to extract the necessary information from a receipt . Then, these information are then recorded as an expenses automatically. Hence, this has make the process of managing receipts much more efficient and fast.

**Universiti Tunku Abdul Rahman**
Bachelor of Information Systems
(Honours) Business Information System
**Prepared by: Tan Su Hua (21ACB00596)**

# Survey Questions

1. How frequently do you manage your expenses and track your financial transactions using a financial management application?
   A. Daily
   B. Weekly
   C. Monthly
   D. Once in a blue moon
   E. Never

2. On a scale of 1 to 5, how comfortable are you with using mobile applications for managing your finance?
   A. 1 - Not very comfortable
   B. 2 - Not Comfortable
   C. 3 - Neutral
   D. 4 - Comfortable
   E. 5 - Very Comfortable

3. Have you heard of Optical Character Recognition (OCR)?
   A. Yes
   B. No

4. Do you think the implementation of OCR technology in a financial mobile application such as extracting information from receipts would make the process of financial planning more effective?
   A. Yes
   B. No
   C. Maybe

5. If yes, do you think the use of OCR to extract information (e.g., merchant name, total, date, etc) from receipts would improve your user experience when using a financial application?
   A. Yes
   B. No

6. Besides merchant name, total and date, what other information(s) would you like to see extracted?
Short answer question

7. On a scale of 1 to 5, how important is it for you to have a function to manually add your income or expenses transactions within the app?
   A. 1 - Not very important
   B. 2 - Not Important
   C. 3 - Neutral
   D. 4 - Important

E. 5 - Very Important

8. On a scale of 1 to 5, how important is it for you to have graphical representations such as pie chart and bar chart on your financial status within the app?
   A. 1 - Not very important
   B. 2 - Not Important
   C. 3 - Neutral
   D. 4 - Important
   E. 5 - Very Important

9. On a scale of 1 to 5, how important is it for you to have a budget planner within the app?
   A. 1 - Not very important
   B. 2 - Not Important
   C. 3 - Neutral
   D. 4 - Important
   E. 5 - Very Important

10. On a scale of 1 to 5, how important is it for you to have customer support within the app?
   A. 1 - Not very important
   B. 2 - Not Important
   C. 3 - Neutral
   D. 4 - Important
   E. 5 - Very Important

11. On a scale of 1 to 5, how important is it for you to generate a personalized income or expenses report within the app?
   A. 1 - Not very important
   B. 2 - Not Important
   C. 3 - Neutral
   D. 4 - Important
   E. 5 - Very Important

12. On a scale of 1 to 5, how important is it for you to choose your desired language and currency within the app?
   A. 1 - Not very important
   B. 2 - Not Important
   C. 3 - Neutral
   D. 4 - Important
   E. 5 - Very Important

13. On a scale of 1 to 5, how important is it for you to generate personalized notifications to remind you to keep track of your transactions within the app?
   A. 1 - Not very important
   B. 2 - Not Important
   C. 3 - Neutral
   D. 4 - Important
   E. 5 - Very Important

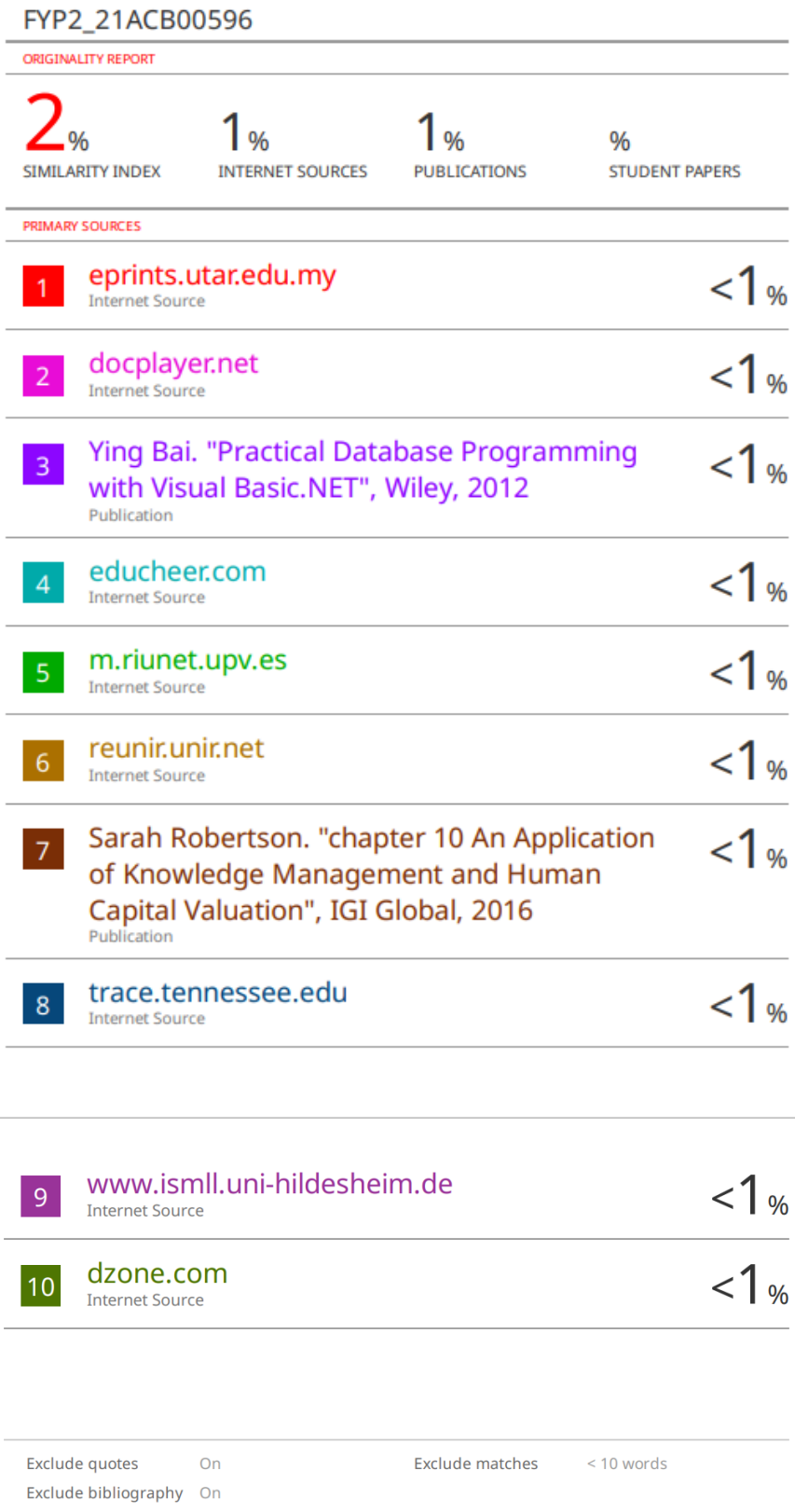
14. On a scale of 1 to 5, how would you rate the overall User Interface (UI) of this application?

A. 1 - Very bad
B. 2 - Bad
C. 3 - Average
D. 4 - Nice
E. 5 - Very nice

15. Any comments, and what additional features would you like to see in a personal financial app beyond OCR-based receipt scanning?
Short answer

# PLAGIARISM CHECK RESULT

APPENDIX



FYP2_21ACB00596

ORIGINALITY REPORT

| 2% | 1% | 1% | % |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | eprints.utar.edu.my<br>Internet Source | <1% |
| 2 | docplayer.net<br>Internet Source | <1% |
| 3 | Ying Bai. "Practical Database Programming with Visual Basic.NET", Wiley, 2012<br>Publication | <1% |
| 4 | educheer.com<br>Internet Source | <1% |
| 5 | m.riunet.upv.es<br>Internet Source | <1% |
| 6 | reunir.unir.net<br>Internet Source | <1% |
| 7 | Sarah Robertson. "chapter 10 An Application of Knowledge Management and Human Capital Valuation", IGI Global, 2016<br>Publication | <1% |
| 8 | trace.tennessee.edu<br>Internet Source | <1% |
| 9 | www.ismll.uni-hildesheim.de<br>Internet Source | <1% |
| 10 | dzone.com<br>Internet Source | <1% |

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | < 10 words |
| Exclude bibliography | On | | |

APPENDIX

| | | | |
|---|---|---|---|
| **Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)** | | | |
| Form Number: FM-IAD-005 | Rev No.: 0 | Effective Date: | Page No.: 1of 1 |

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

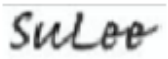
| | |
|---|---|
| **Full Name(s) of Candidate(s)** | Tan Su Hua |
| **ID Number(s)** | 21ACB00596 |
| **Programme / Course** | Bachelor of Information Systems (Honours) Business Information Systems |
| **Title of Final Year Project** | Development of Personal Finance Mobile Application with Optical Character Recognition (OCR) Technology for Automated Receipt Management and Expense Tracking |

| **Similarity** | **Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)** |
|---|---|
| **Overall similarity index: 2%**<br><br>**% Similarity by source**<br>Internet Sources: 1 %<br>Publications: 1 %<br>Student Papers: 0 % | |
| **Number of individual sources listed** of more than 3% similarity: 0 | |
| **Parameters of originality required, and limits approved by UTAR are as Follows:**<br>**(i) Overall similarity index is 20% and below, and**<br>**(ii) Matching of individual sources listed must be less than 3% each, and**<br>**(iii) Matching texts in continuous block must not exceed 8 words**<br>***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.*** | |

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

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

APPENDIX



_____

Signature of Supervisor

Name: Su Lee Seng_____

Date: 11 September 2023_____

APPENDIX

# UNIVERSITI TUNKU ABDUL RAHMAN

## FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

### CHECKLIST FOR FYP2 THESIS SUBMISSION

| | |
|---|---|
| Student Id | 21ACB00596 |
| Student Name | Tan Su Hua |
| Supervisor Name | Su Lee Seng |

| TICK (√) | DOCUMENT ITEMS<br>Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item. |
|---|---|
| √ | Title Page |
| √ | Signed Report Status Declaration Form |
| √ | Signed FYP Thesis Submission Form |
| √ | Signed form of the Declaration of Originality |
| √ | Acknowledgement |
| √ | Abstract |
| √ | Table of Contents |
| √ | List of Figures (if applicable) |
| √ | List of Tables (if applicable) |
| √ | List of Symbols (if applicable) |
| √ | List of Abbreviations (if applicable) |
| √ | Chapters / Content |
| √ | Bibliography (or References) |
| √ | All references in bibliography are cited in the thesis, especially in the chapter of literature review |
| √ | Appendices (if applicable) |
| √ | Weekly Log |
| √ | Poster |
| √ | Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005) |
| √ | I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report. |

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

I, the author, have checked and confirmed all the items listed in the table are included in my report.

Suhua

_____

(Signature of Student)

Date: 11/9/2023