A PERSONAL FINANCIAL MANAGEMENT APPLICATION WITH SPENDING MONITORING

By

Ang Jia Pei

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to take this opportunity to convey my sincere appreciation to everyone who has supported me throughout the development of this project.

I would like to start by expressing my deepest gratitude to my supervisor, Dr. Ng Hui Fuang, for his invaluable guidance, constructive feedback, and continuous support during every stage of the development of this project. I am extremely grateful for his patience and guidance in assisting me for the improvement of both the conceptual and technical parts of this project.

Additionally, I would like to extend my appreciation to the Faculty of Information and Communication Technology (FICT) for providing those essential tools that led to the establishment of this project.

Last but not least, I sincerely thank my family and friends for their unwavering encouragement and support had given me a huge source of motivation. This is because this project may not be achievable if without their emotional and moral support.

ABSTRACT

This project introduces an advanced personal financial management application with spending monitoring capabilities by tracking and monitoring their financial activities. The field of study for this project encompasses financial technology (fintech), mobile application development, and personal finance management. There are services such as effective budgeting, expense tracking, and financial decision-making tools provided to users by using mobile platforms.

Nowadays, many people facing some challenges to keep track of their daily expenses due to busy schedules. Additionally, numerous payment methods have been introduced in the market, including e-wallet platforms such as TNG eWallet, GrabPay, ShopeePay, and embedded wallets within some mobile applications to ease customer payments. However, this variety has further complicated expense tracking, as individuals must keep track of multiple transactions across different platforms and navigate several apps to review their spending history. Consequently, the process of documenting daily expenses has become more difficult and time-consuming. To address this, there are various expense recording applications have appeared in the market, offering features such as receipt or transaction scanning through Optical Character Recognition (OCR). However, many expense tracking apps still fall short in delivering an app that meet the user requirement. To illustrate, the lack of automatic categorization of expenses into different categories accurately and unable to extract the correct amount of expenses. This project able to solve these limitations, by automatically categorizing expenses and allowing users to correct the result, which enhance the flexibility. Further, some rewarding system had been applied in this mobile application which aim to motivate the users to manage their expenses wisely. Additionally, AI-driven savings tips have been introduced to provide users with personalized financial guidance. The main objective of this project is to offer customers a user-friendly interface that simplifies their daily lives and reduce the effort required from users to record their expenses. Users will be able to check their expenses or financial reports anytime such as daily and monthly, helping them achieve their financial goals as well as making better financial planning decisions.

Area of Study (Minimum 1, Maximum 2): Mobile Application Development, Computer Vision

Keywords (Minimum 5, Maximum 10): Personal Finance App, OCR Receipt Scanning, Firebase Integration, Expense Tracking, Data Visualization, AI Savings Tips

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

OCR	Optical Character Recognition
ML	Machine Learning
NLP	Natural Language Processing
AI	Artificial Intelligence

CHAPTER 1 INTRODUCTION

1.1 Background information

Personal finance management is the activity of managing one's expenses, income, savings, and investments with a variety of tools and techniques. Before the digital age, peoples record their expenses and income by using paper or simple spreadsheets such as Excel. But with the advent of technology, personal finance management has evolved significantly. First, paper-based budgeting gave away to desktop-based financial software such as Microsoft Money, which offered basic automation for example, schedules and tracks monthly payments such as rent or subscriptions without manual input. However, by using these traditional methods, it was time-consuming. With the advent of mobile technology, it shifts to digital personal finance application.

Although personal finance management apps brought convenience, such as provide real-time access to financial information, allowing users in budget allocation, and financial planning, thereby enabling users to make more informed decisions. In order to serve the needs of users, different functionalities being provided in an app. Consequently, it overwhelms the users. Additionally, most of the financial management apps does not provide receipt scanning or transaction scanning feature for automatic expenses documentation. Last but not least, most of the existing applications lack of the rewards provided to users, hence failed to motivate users in manage their finance wisely.

In our project, we propose an innovative solution to solve the above problems. By implementing the OCR technology, we are able to convert scanned images of text (like receipts or transaction details from e-wallets) into machine-readable text, enabling data extraction automatically. To illustrate, users will capture the image of a receipt using a smartphone camera, then Google Vision API for text extraction, OpenAI GPT-4 to process the extracted raw text and further analyze the category of each expense and lastly fill it into the appropriate field. As a result, these features able to reduce manual data entry and enhancing accuracy. In addition, basic budgeting tools and rewarding system will be provided for users to plan and allocate funds monthly, as well as assure financial stability when necessary. Lastly, in order to provide financial data visually,

data visualization tools also will be provided for users to identify spending trends and patterns by using pie charts and line graphs.

Modern development platforms offer a responsive user interface, real-time financial data updates. In our project, Firebase is used as the backend to store users' data securely, ensuring seamless synchronization across devices. Last but not least, security measures such as data encryption, Firebase Authentication, Firestore Security Rules are implemented to protect sensitive financial information and prevent unauthorized access.

1.2 Problem Statement and Motivation

In today's fast-paced era, many people are busy with their daily tasks, like studying and working. Consequently, they often felt that expense recording was tedious and time-consuming task as this task need them to manually enter all the expenses details, hence leading to lack of interest in financial management. This bad habit which had led to overspending behaviours and poor saving habits. Additionally, majority of individuals prefer easier recording of their finance without much effort needed such as scanning transaction features, like scan the receipts or scan the e-wallets' transaction details. Thus, according to the challenges mentioned above, it points out there is a need for an application which able to capture the expenses automatically while also offering a clear and comprehensive view of expenses, income, budget and financial goals. Individuals can make use of the information provided to monitor and manage their finances, such as allocating funds for savings, ensuring financial security during emergencies, reducing financial distress, as well as contributing to financial stability.

Although there exist various kinds of personal financial monitoring applications in the market, most of them had some limitations in helping users gain a superior experience. We aim to address and solve three primary problem statements as follows:

1. **Complex interfaces that overwhelm users**

   This problem statement emphasizes the fact that user interfaces in previous applications were too complex which may overwhelm users. Furthermore, many apps have complex interfaces with too many functionalities on one screen,

make it difficult for new users to learn. Hence, it is essential to develop a user-friendly interface that simplifies the recording process while offering suitable functionality to enable users to manage their finances effectively.

2. **Many users struggle to manually record their expenses due to time constraints**

   This problem statement highlights the challenges when users record down all their daily expenses manually. Currently, people often forget to update their expenses due to their busy lifestyle, resulting in inaccurate tracking. Besides that, many financial apps always require users to enter their spending manually instead of capturing them automatically, which is time-consuming and may lead to human errors. Hence, it is essential to provide a transaction scanning feature, which allows immediate and accurate expense recording.

3. **Difficulty organizing and managing paper receipts due to lack of scanning feature**

   This problem statement highlights that individuals may receive a huge number of receipts daily, and the risk of losing receipts is high because they are easily misplaced or damaged, resulting in users having difficulty tracking spending accurately. Hence, it is essential to provide the receipt scanning feature which reduces manual entry and increases efficiency.

4. **Lack of motivation for maintaining financial management**

   When it comes to saving and budgeting, lots of individuals find it challenging to keep up their motivation. Research has shown that majority existing financial apps often do not provide features that motivate excellent financial behavior, causing it difficult for users to remain engaged. Hence, it is essential to provide some rewards such as points in the budgeting section, when users achieve their savings goals. This encourages users to stick to their budgets, which makes financial management feel more motivating and fun at the same time also prevents them from overspending.

1.3 Objectives

In order to address the problem statements, we intend to achieve three key objectives through this project. Three main project objectives are as follows:

1. To develop a **user-friendly financial monitoring application** that simplifies financial management.
2. To **manage and organize the receipts** by implementing **receipt scanning** feature, eliminating manual entry and human errors.
3. To enhance user engagement and motivate them **spend wisely and stick to their budgets** by introducing **gamified rewards for meeting savings goals and daily check-in**.

1.4 Project Scope

Our project intends to deliver a mobile application-based personal finance monitoring system for a wide range of individuals, such as busy workers, students, housewives, and others. This application able to track the users' expenses and at the same time help them manage their finances wisely. Our application's capabilities consist of functions such as **receipt or transaction scanning, budgeting, and comprehensive financial reporting**. For instance, automatic transaction extraction and the OCR technique, which are able to scan and extract the exact amount of spending, both features able to release users from manually recording their expenses. In order to save users' time, multiple transaction images being uploaded at one time is supported by this application. Additionally, the system will automatically categorize each transaction after scanning. Further, this application able to produce financial reports either daily or monthly, which serves as a powerful function that helps users to get a quick overview of their expense distribution visually. In terms of project deliverables, we aim to deliver an intuitive application that keeps the financial recording process simple and encourages users to keep track of their daily spending.

1.5 Contributions

This expenses monitoring application able to provide users a user-friendly tool that simplifies the financial management process. For example, the existing apps frequently

poses common limitations such as some of them may come with complex interfaces, some may lack of transaction or receipt scanning functions, and some of them does not provide any graphs or charts to help users visualize their expenses.

Hence, this application addresses these common challenges. For instance, this app promotes integration with the **receipt scanning with OCR technology**; it can extract the key details from receipts, further reduce the time, users spend in documenting their expenditures. This feature allows users to upload multiple receipts or a single receipt containing multiple transactions at once, enabling efficient extraction while reducing user waiting time. Since there are many different types of payment methods available today, it is common for users to make payments by using different apps throughout the day. This application provides users with the abilities to extract the transaction details when users upload their transaction images such as e-wallet transaction details. By extracting the transaction details, it relieves users from navigating through multiple apps to manually record their daily expenses. Additionally, **AI Tips Suggestions** provides users with personalized financial advice based on their remaining budget, local weather conditions, and total expenses. For example, meal suggestions and weather-related reminder had been provided, to help users reduce unnecessary expenses and make more informed financial decisions. Further, the budget creation function allowed users to allocate budgets for each category, on a monthly basis. To motivate users to record their expenses daily, spend wisely and stick to the budgets, a **reward system** had been applied, such as users able to earn points when they check-in this app daily or when they meet their financial goals. In order to help users better understand their expenditure patterns, a comprehensive financial report is generated. By using **charts and graphs,** user was able to visualize and understand their spending patterns effectively**.** To sum up, this project able to provide a user-friendly interface for users, which help them simplified their daily lives and reduced the effort required to record their expenses.

1.6 Report Organization

This report is organised into **7 chapters**: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology, Chapter 4 System Design, Chapter 5 System Implementation and Testing, Chapter 6 System Evaluation and Discussion, and Chapter

7 Conclusion. The first chapter is an introduction to this project, which contains the problem statement, motivation, project scope, project objectives, impact, significance and project contribution, and report organization. The second chapter is the literature review carried out in order to find out the strengths and weaknesses on several existing personal financial management applications in the market. The third chapter outlines the system methodology or development approach adopted for this project. It focuses on the planning and conceptual design of the system, which further supported by diagrams like the system architecture diagram, use case diagram, and activity diagram. These components describe how it functions and logical structure of the system first before implementation, setting up the groundwork for the comprehensive design addressed in the following chapter. The fourth chapter provides a comprehensive overview of the system design and interaction between system components. It discusses the contribution of each part to the app's functionality by presenting the detailed system block diagrams, which includes key modules such as user login or sign up, income tracking and expense, OCR receipt scanning, AI Tips suggestions and budget management. Additionally, the fifth chapter discusses the system implementation. It includes the software setup, configuration, and operation, screenshots are provided in order to demonstrate key features and functionality. This chapter also discusses all issues or challenges that happened during implementation. Furthermore, the sixth chapter focuses on the evaluation and testing of the application. It covers the testing methodologies employed, the performance metrics assessed and the test environment setup. The chapter also determines whether the initial objectives were reached, providing more insights into the app's overall effectiveness. Lastly, the seventh chapter concludes the report with a summary of the project outcomes and recommendations for future improvements. It reflects on the application's development journey, emphasizing key achievements and lessons learned.

CHAPTER 2 LITERATURE REVIEW

2.1 Personal Finance Management System

The personal finance management system describes how people monitor and manage their personal finances. With the rapid development of digital and mobile banking technology, With the rapid advancement of digital and mobile banking technologies, most individuals now rely on computerized financial assistance to handle their money online or via a smartphone app. The application serves not only as a basic budgeting tool, but also assists users in the planning and tracking of their income and expenses. They can provide an extensive variety of tools, functions, and insights that assist users in the understanding, management, and monitoring of their finances, thus allowing them to make better informed decisions [1]. In order to achieve financial stability and meeting long-term goals, a good personal finance management is a must [2].

2.1.1 Expensify

Expensify [3], a finance management mobile application. As shown in the Figure 2.1.1, it provides a wide range of services for both business and personal finance tasks, including receipt scanning, bill payments, and travel bookings. It provides real-time visibility and control over expense tracking, receipt management, and detailed financial report generation.

Expensify's key functionality is SmartScan, a receipt scanning function. As shown in Figure 2.1.2, it allows users to record their expenses by capturing a photo of the receipts. This feature helps in prevent missing of receipts and eliminate the needs to carry physical copies. Subsequently, the system processes the receipts to extract crucial information, including the merchant, date, and amount. For example, based on Figure 2.1.3, during travel or transactions, SmartScan was able to automatically identify receipts in over 150 currencies by capturing a photo and extracting the merchant, date, and amount. The system can categorize and records such information into the user's account, minimizing the user's effort in manually entering the transaction details and significantly reducing the risk of human errors [4].

Additionally, Expensify provide users with real-time financial report generation that presented in Figure 2.1.4, either for personal budgeting or business expense

management. These reports provide a detailed information of all expenses, categorized by type, date, and merchant. Besides that, flexibility is provided, such as users are allowed to customize reports in order to meet their specific needs. Users can simply create, submit, or export the report in various formats, making the sharing more effectively. Based on the information provided in the financial report, users are able to gain a better understanding towards their spending patterns, allowing them to make more informed financial decisions and identify areas to reduce spending [5].

In summary, Expensify serves as a powerful tool for personal financial management offering functions such as receipt scanning and detailed financial report generation. Its user-friendly interface and powerful functionality fulfil the needs of a wide range of users, from personal to business, to control their finances, monitor expenses, generate reports, perform financial analysis, and help them to be more informed when making financial decisions [3].

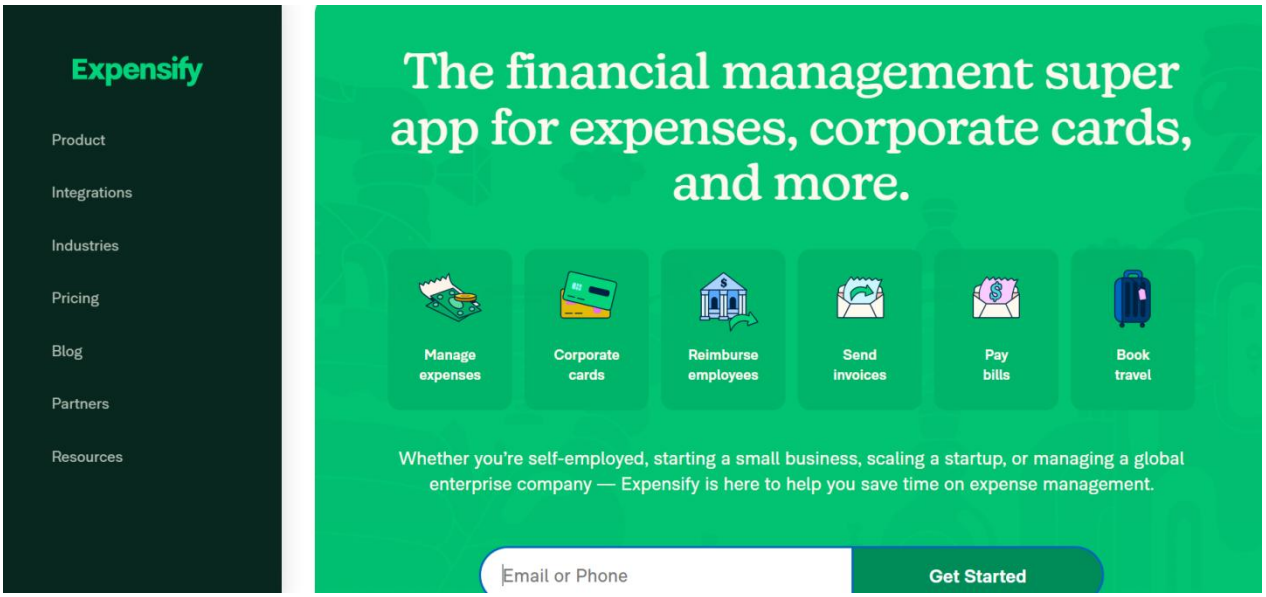


Figure 2.1.1 Functions of Expensify

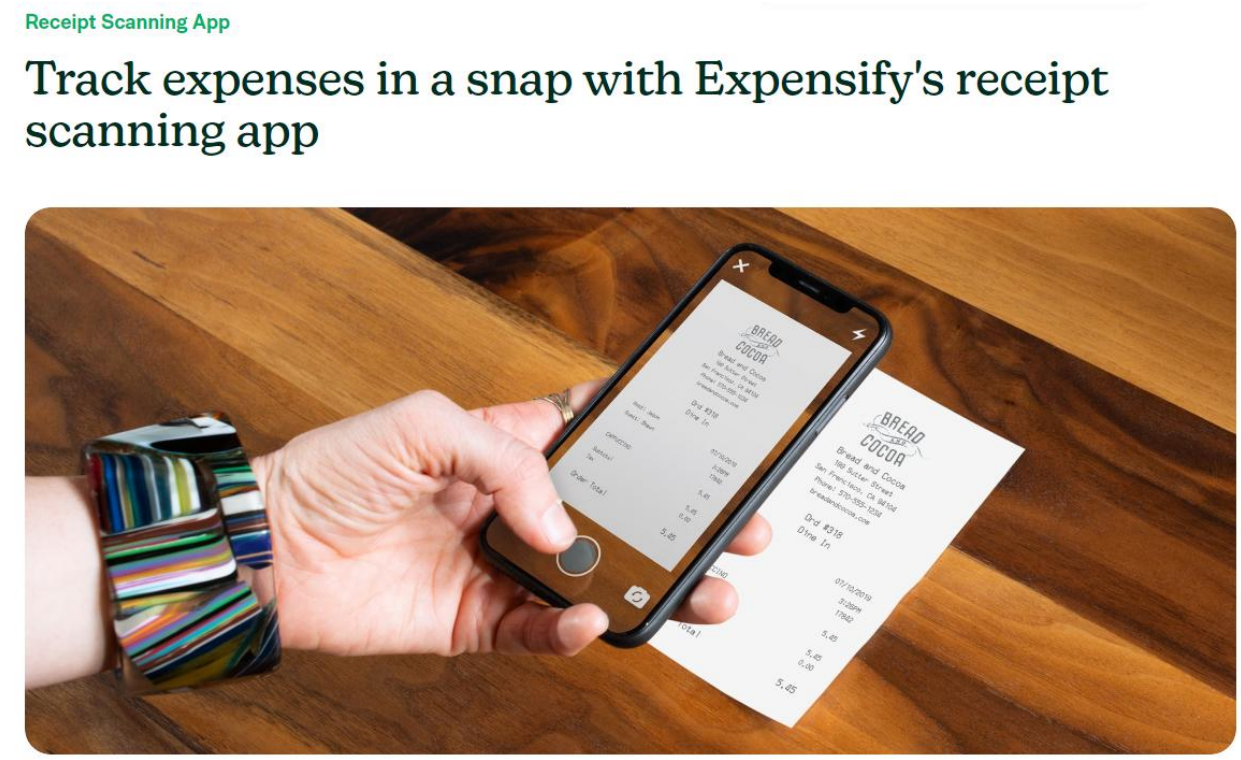


Figure 2.1.2 Receipt Scanning

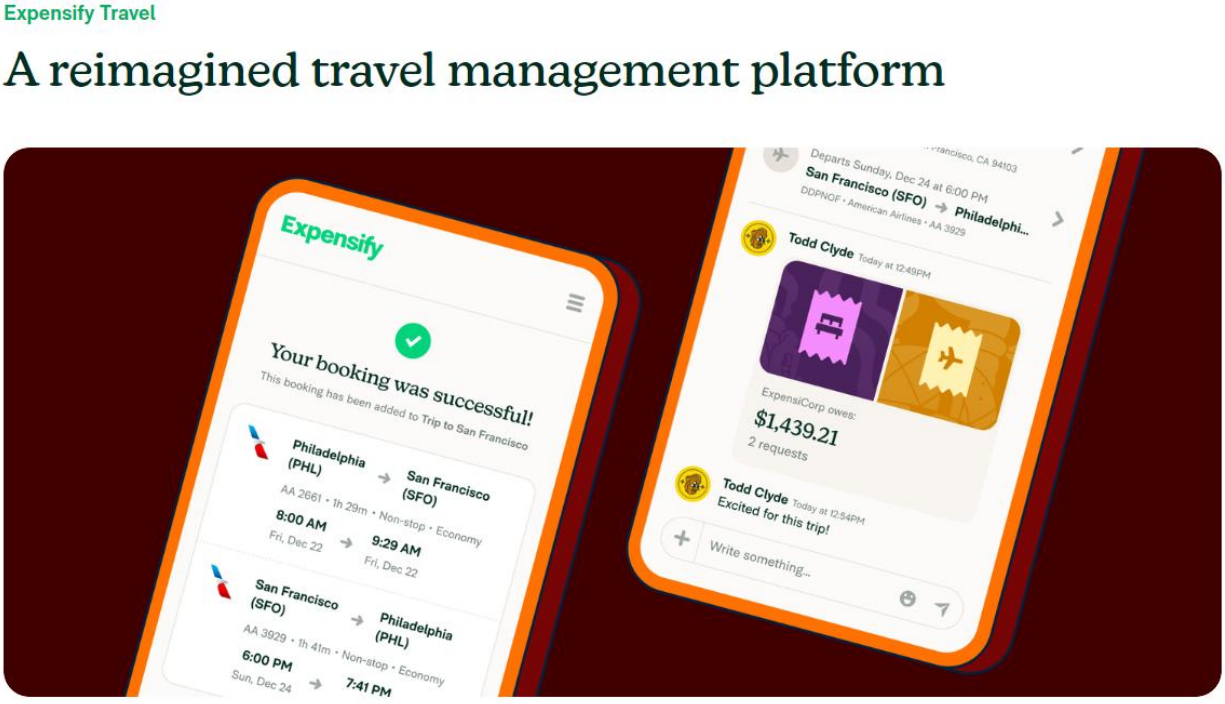


Figure 2.1.3 Expensify Travel

Figure 2.1.4 Insights and Reporting

2.1.2 Zoho Expense

Zoho Expense [6] is an expense monitoring and management software application. According to Figure 2.1.5, it shows that functions provided by Zoho Expense such as expenses tracking, report submission, and approval of expenses for both businesses and users. It enables users to scan the receipts, record their expenses, and submit the financial reports.

First, the feature offered by Zoho Expense is **receipt scanning**. As shown in Figure 2.1.6, it allows the user to scan the receipt by using the camera on their mobile device. It applies OCR to extract key details such as merchant, date, and amount, automatically creating expense entries had shown in Figure 2.1.7. Additionally, the system provides automatically record the user's expenses functions that is presented in Figure 2.1.8, such as when a user forwards the email with invoice to the system's specified email address, it will log in the user's accounts. Another time-saving feature provided by this software had shown in Figure 2.1.9 which is the bulk uploading feature, which allows users post

several receipts at the same time [7]. Besides that, **multi-currency support** had offered in Zoho Expense. It helps the users to convert foreign transactions to the base currency by automatically applying real-time exchange rates. This is a feature that is attractive and useful to those travellers, as it can capture expenses in different currencies, enabling them to manage their expenses when they are traveling around the world without the need to write them down manually or convert the currency rate manually [8].

Furthermore, Zoho Expense has deployed an **analysis and financial reporting feature**. According to Figure 2.1.10, there are various kinds of reports that can be generated, such as spending trends and category-wise expenses, helping users gain a better understanding of their spending habits. Besides that, users can manually create expense reports that include all their transaction details. These reports enable users to customize and group their expenses accordingly [9].

Lastly, as shown in the Figure 2.1.11, the application enables users to set up their own budgets. It has two different types of **budget creation, such as category-based and expense-based**. On top of that, there are few categories provided in the system, such as entertainment, dining. There is also flexibility offered to the users, such **as the ability to set the period of their budgets** either monthly, quarterly, half-yearly, or yearly basis. With this function, users are able to align their financial planning cycles. It offers detailed budget creation, allocation methods, and user-specific budgets, along with tools to make comparisons between budgeted amounts and actual expenses. The system will **send alerts to notify users when they approach or exceed their budget limits**, thereby leading to better financial control and planning [10].

Zoho Expense supports the entire personal financial management process, from initial scanning of receipts to generating financial reports. Most people strongly rely on their powerful functions, and the version is compatible with both iOS and Android phones [6].

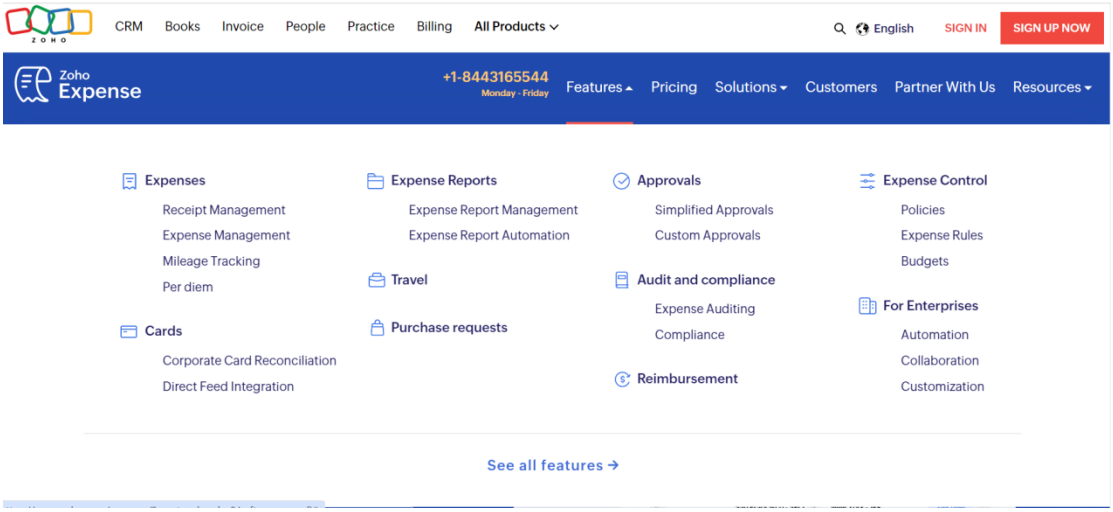


Figure 2.1.5 Functions of Zoho Expense

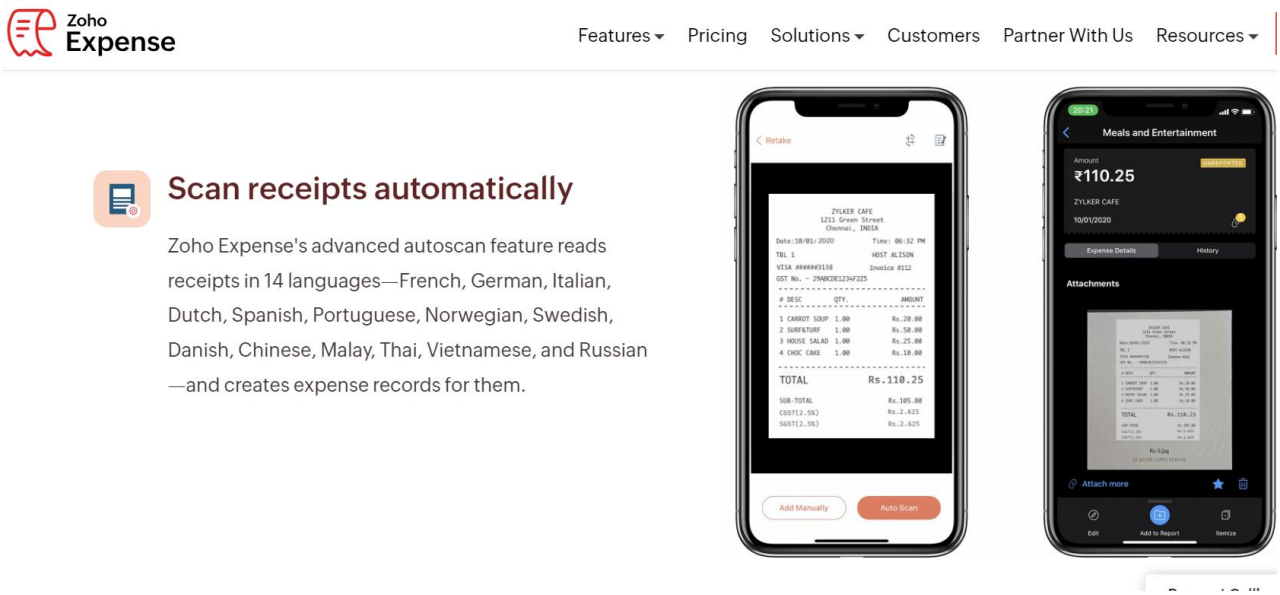


Figure 2.1.6 Scanning receipts

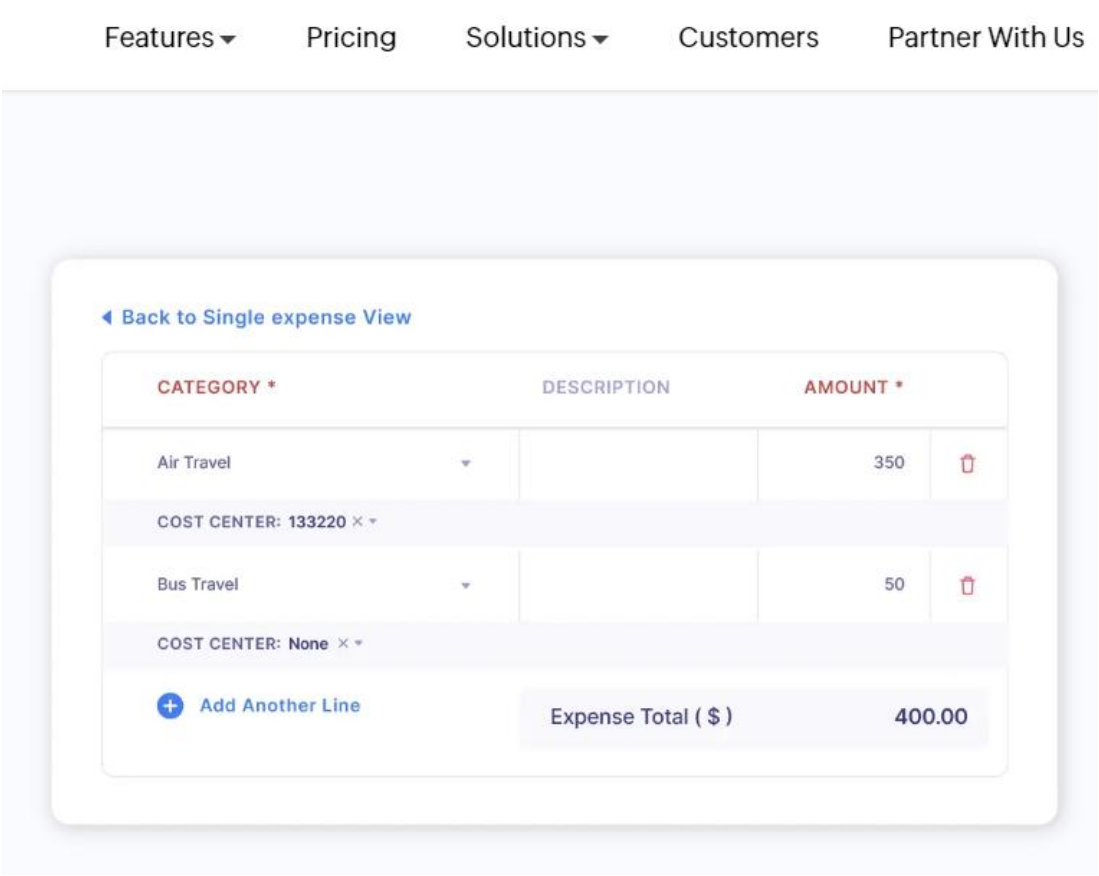


Figure 2.1.7 Categorize the expenses

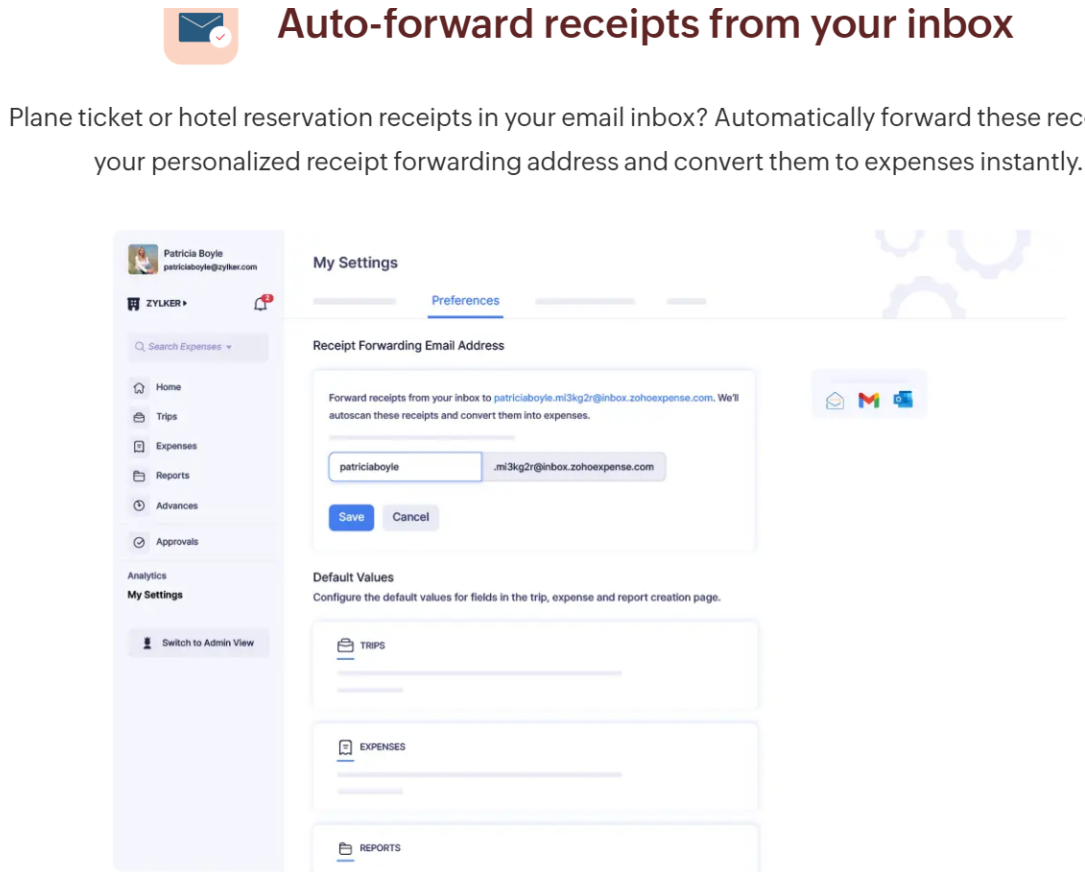


Figure 2.1.8 Record expenses by using invoice forwarded from email automatically

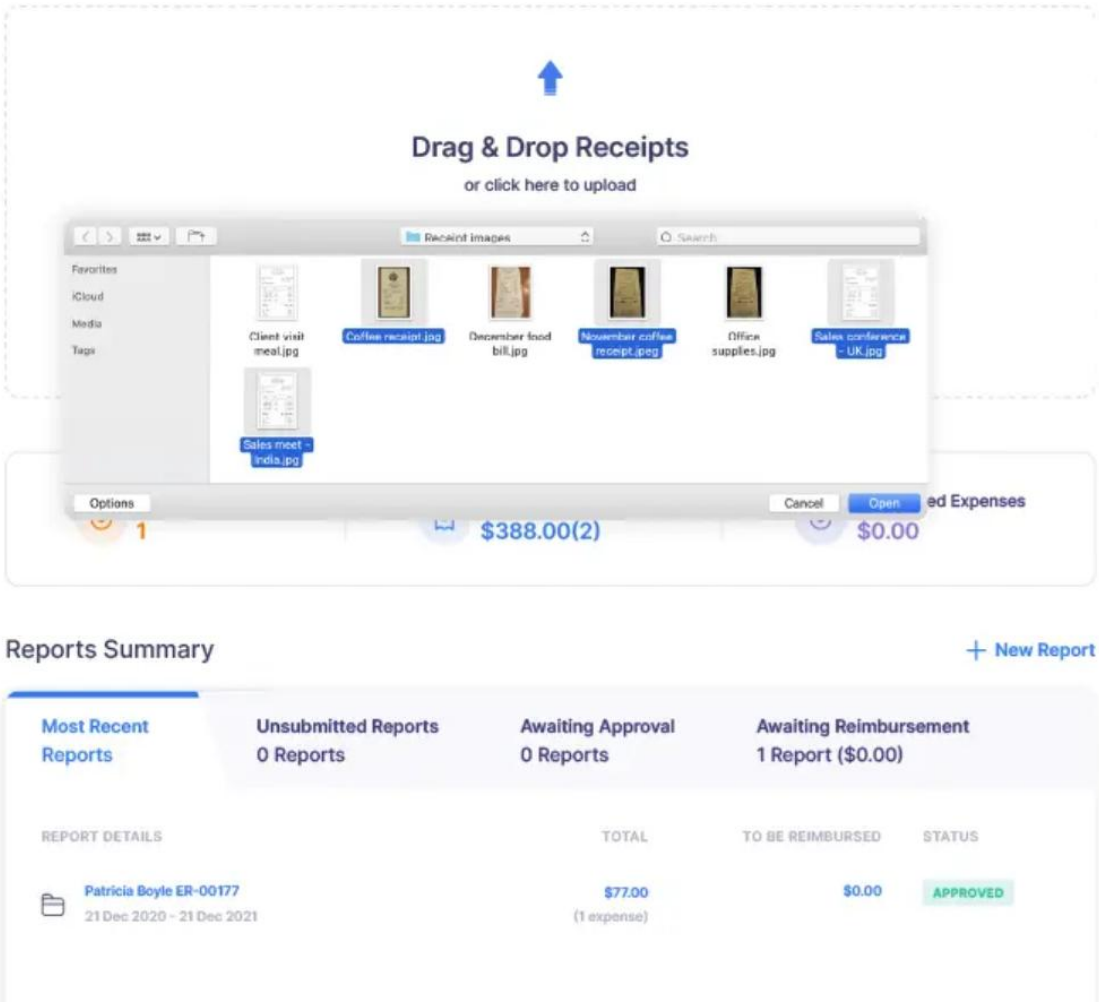


Figure 2.1.9 Upload multiple receipts at once

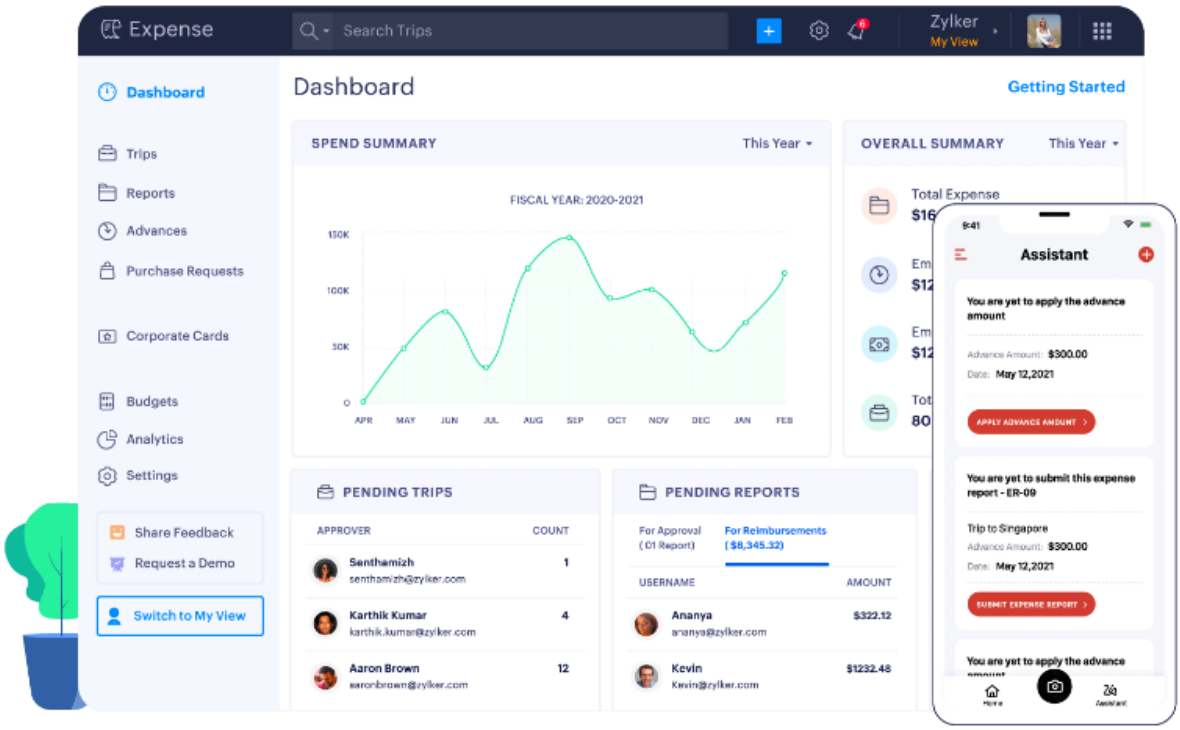


Figure 2.1.10 Reporting

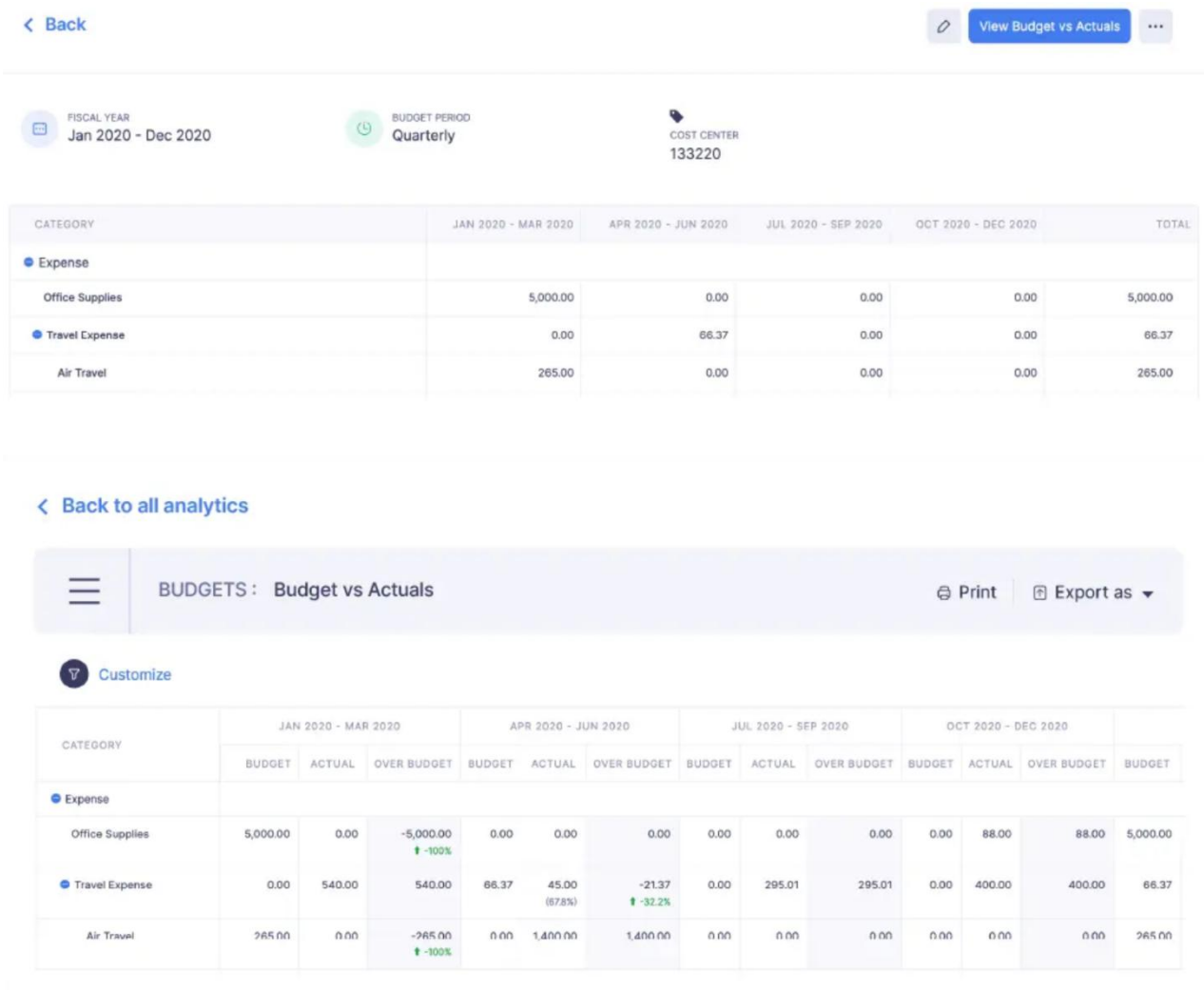


Figure 2.1.11 Budget allocation and comparison between budgets and actual spending

2.1.3 Spendee

Spendee [11] is a mobile application that helps people get their money into shape, such as by managing their money on the go. It supported the Android, iOS, and website versions had been presented in Figure 2.1.12. As shown in Figure 2.1.13, it **supports multiple currencies, automatically converts** them, and displays the conversion rate for users. Hence, it supports a wide range of users from different countries around the world. It has many powerful functions to make their users' lives easier and monitor their daily expenses effectively.

The first feature deployed in Spendee is cash flow tracking. Users can **link their bank accounts to Spendee accounts for automatically synchronise transactions** and

recorded to the Spendee accounts. Hence with this feature, it allows users to obtain a deeper understanding into users' spending patterns and reduce the users' efforts. It also provides the basic cash flow monitoring function, such as users being able to add their transaction details to the app manually. According to Figure 2.1.14, there are a few categories of expenses provided, such as food and drink, shopping, cars, and others [11].

Spendee provide their users with a clear financial insight. It serves a function that allows users to gain a better understanding of their financial habits. According to Figure 2.1.15, a colored graphic along with the trending flow. This feature can enhance the user experience and eliminate the need for financial analysis based on complicated Excel sheets. In addition, users are able to see the cash flow in and out monthly with just one click [11].

Spendee offers smart budgeting. According to Figure 2.1.16, it provides users with budget creation, and users are flexible in selecting the budget category they prefer. To illustrate, the user sets the budget name to travel with an amount of RM1000, and selects three categories: food, entertainment, and beauty, with a spending cap of RM1000 per month starting from July 20, 2024 [11].

To sum up, Spendee provides many powerful functions to ease their users' needs. It helps users gain control over their finances, make better decisions based on the financial reports, and work to achieve financial stability.

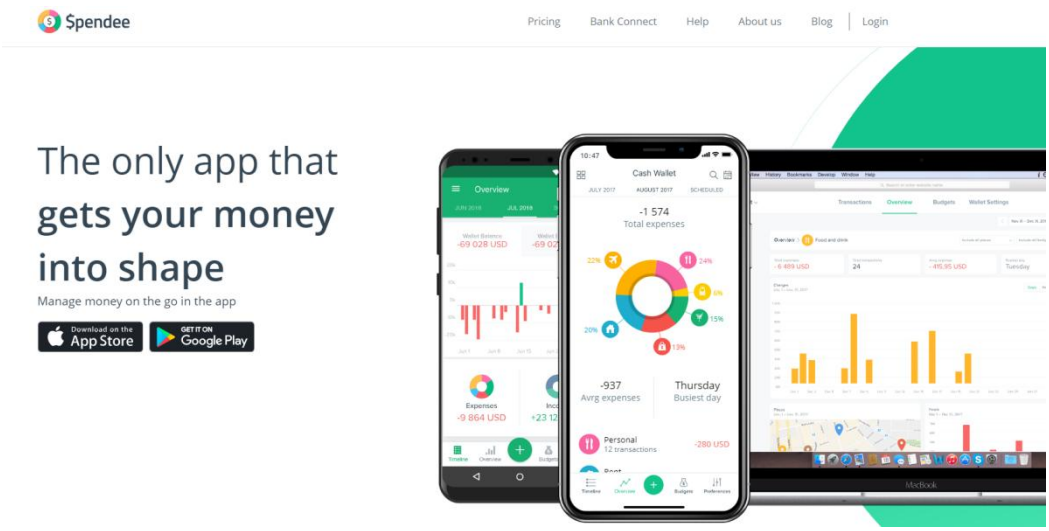


Figure 2.1.12 Spendee available in App Store and Google Play

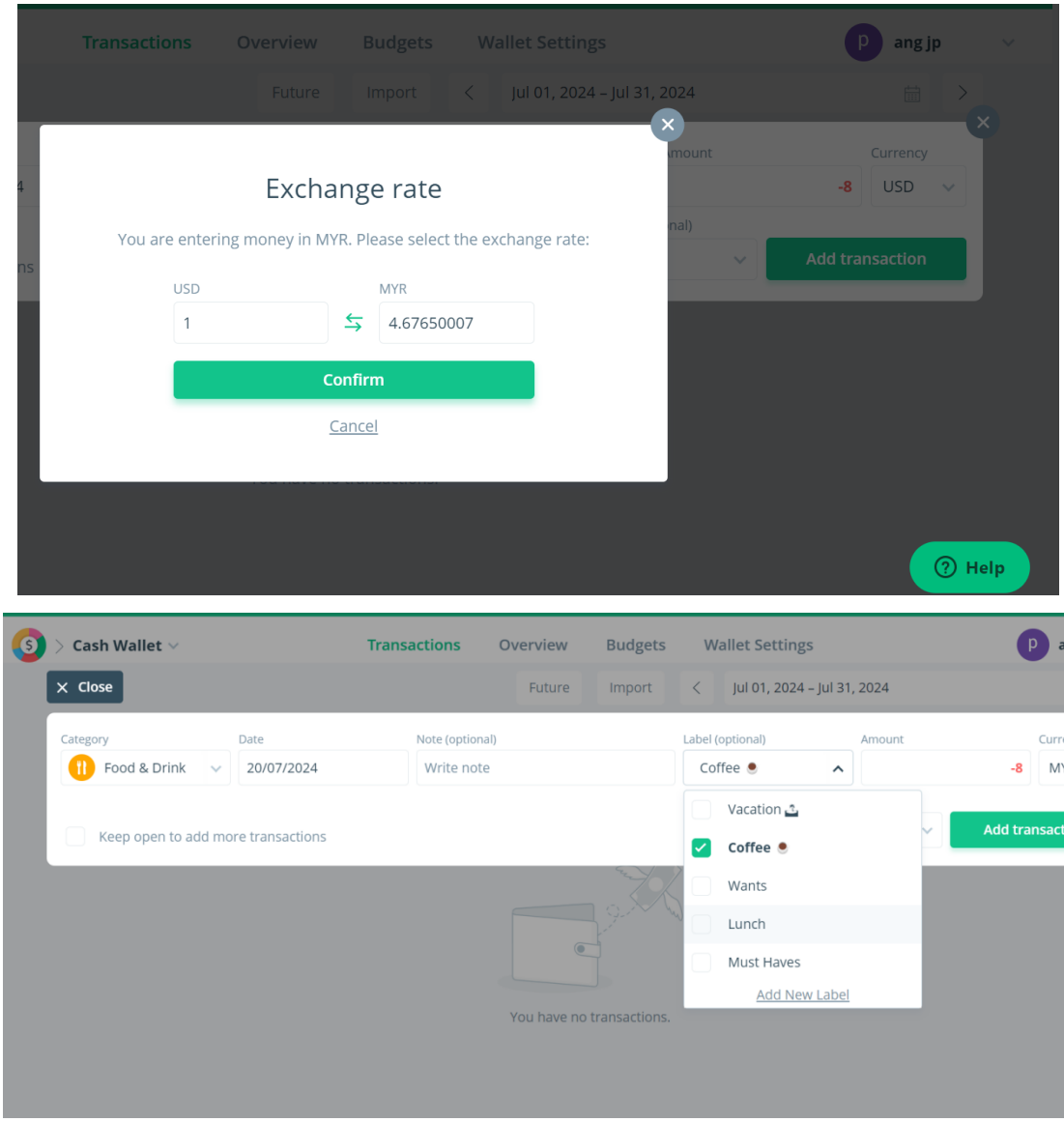


Figure 2.1.13 Multi-currency support and automatically conversion

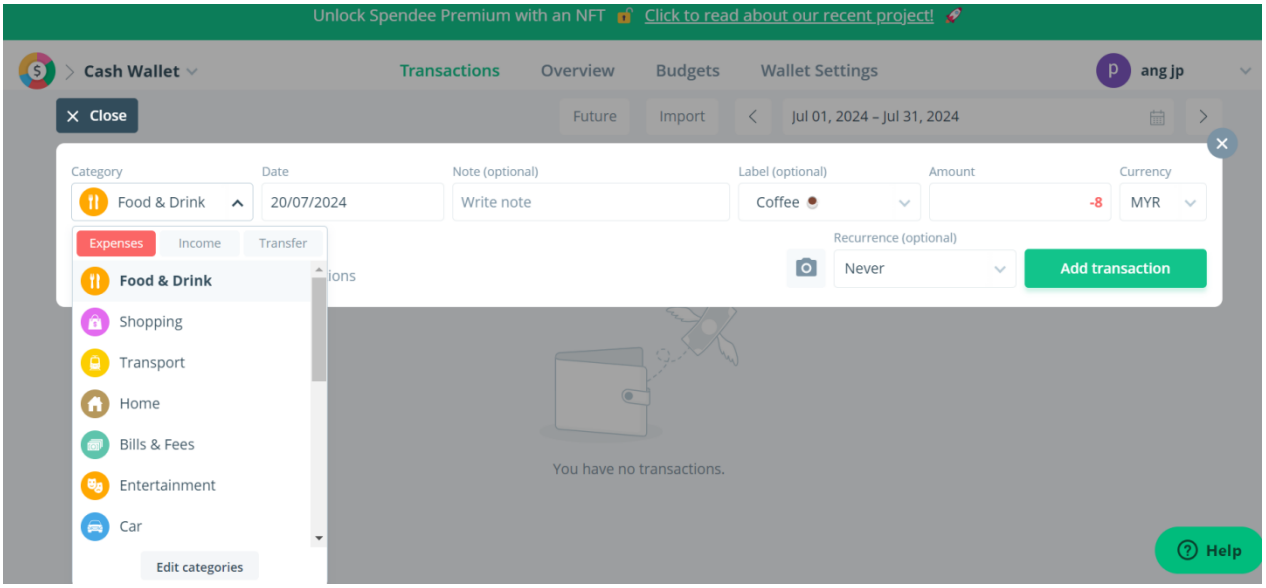


Figure 2.1.14 Different expenses categories provided

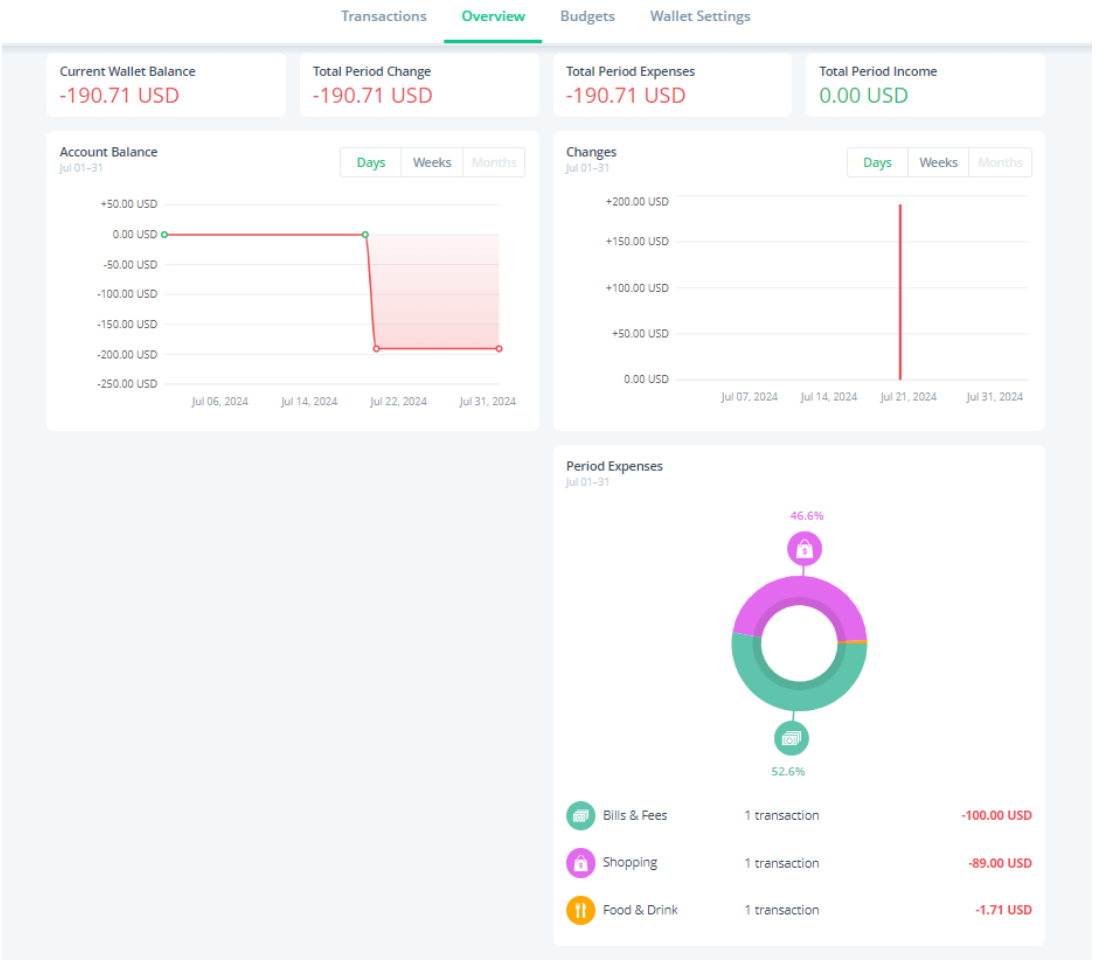


Figure 2.1.15 Expenses Report

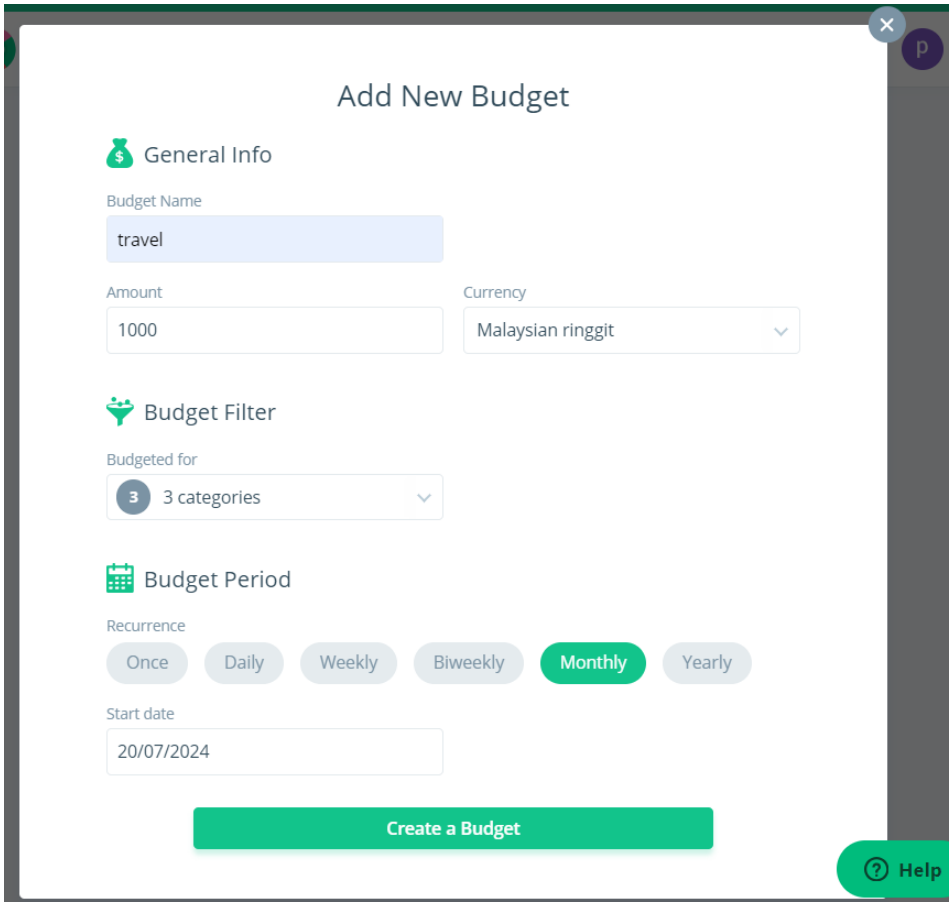


Figure 2.1.16 Set up new budget

2.2 Limitation of Previous Studies

After reviewing the three applications, Expensify, Zoho Expense, and Spendee, three of them are **lack of the point-based reward system to engage their users for continuously recording down the daily expenses or encourage them to spend wisely**.

For **Expensify**, the primary focus is on the business field, which makes it helpful for employees and accountants to submit and approve their claims to monitor business flow. Therefore, it may have some limitations for the users who hope to use for personal finance management. It lacks the income tracking monthly, users need to enter their income every month manually. Besides that, it also does not have the point-based reward system, which less motivation towards users to record their expenses, and spend wisely.

Even though **Spendee** has many features to assist users in effective personal financial management, it does not provide the scanning receipts functionality. Additional effort may be required from the users to manually input, one by one, their transaction details from the receipt into the Spendee account.

The table 2.2.1 below shows the quick summary and comparison between the reviewed application and the proposed system.

Table 2.2.1 Functionality comparison between reviewed application and the proposed system

Software / Functionality	Expensify	Zoho Expense	Spendee	Proposed System
Receipt Scanning and Expense Capture	✓	✓	✗	✓
Budgeting	✓	✓	✓	✓
Analytics and reporting	✓	✓	✓	✓
Reward system for budgeting and daily check-in	✗	✗	✗	✓
Income tracking	✗	✗	✓	✓
AI Tips Suggestion for Savings	✗	✗	✗	✓

2.3 Proposed Solutions

The proposed personal financial management system is perfectly designed for users who wish to use it to record down their daily expenses, monitor their expenses, and generate financial reports to gain a quick overview of their spending patterns. It provides different kinds of functionality, such as receipt scanning, analytics and reporting, and integration with the phone notifications center to filter the transaction details; hence, it suits the users' needs and saves the users' time and efforts.

For **receipt scanning** functionality, it utilizes OCR technology to scan, extract then analyze the details from receipts or transaction details, such as merchant name, date, total amount, and category of the expenses. This feature allows users to upload multiple receipts or a single receipt containing multiple transactions at once, enabling efficient extraction while reducing user waiting time. Additionally, it offers users flexibility to make any changes to the inaccurate categorization, reducing errors and ensuring the accuracy of the financial records.

Secondly, budgeting enables users to **customize their budget** monthly, and the **reward system** is provided when users meet their financial goals or they check-in daily to the app. Besides, it also provides a real-time comparison of expenses against their budget. This functionality helps users keep track of their expenses and avoid overspending.

Moreover, the proposed system can help users **analyze their expenses and create a spending report** in the form of pie charts and bar graphs, and present to users, so they can have a clear overview of their spending and find out which areas have major expenses, and which areas have the least. Besides, users are allowed to customize their financial reports by filtering date and category and then reports will be generated based on the users' requirements.

Additionally, the proposed system includes an **AI-powered feature that provides users with personalized savings suggestions** based on the budget left and weather conditions. For example, the system analyzes the user's current location to determine the weather conditions in their city. Hence, the system will suggest bringing an umbrella on rainy days and recommend budget-friendly meals like home-cooked options.

Lastly, the proposed system allowed users to **enter their income**, and it will try to remember the user's income monthly and later will update automatically every month.

This releases the users from manual enter their income monthly. At the same time, there is also a real time comparison between the expenses and incomes provided to users, which allowed them to identify easily are they overspending.

CHAPTER 3 SYSTEM METHOD/APPROACH

3.1 System Design Specifications

3.1.1 Methodologies and General Work Procedures

The development of this personal finance management application will follow the **five phases of project development lifecycle**, which ensured organized workflows, easier to control and achieving its goals efficiently. These five phases which includes project initiation, project planning, project execution, project monitoring and control, and lastly project closure, that is shown in Figure 3.1.1.

Figure 3.1.1 5 Project Phases of the Project Management Lifecycle

In the **project initiation** phase, it starts with a thorough analysis of the features and limitations of the current personal finance management applications, followed by a feasibility study to assess the project's operational, financial, and technical viability. The feasibility of deploying Google Cloud Vision API for OCR-based receipt scanning will be studied in order to guarantee precise and automatic extraction of transaction details. We will also explore the feasibility of using real-time notification tracking for expense monitoring to ensure seamless transaction updates. Furthermore, we will

identify the key stakeholders, such as potential users, and collect their needs and opinions. Therefore, the project's scope and objectives will be defined to establish a clear vision for this project.

During the **project planning** phase, the project methodology was defined. Figure 3.1.2 shows a prototype model utilized in this project.

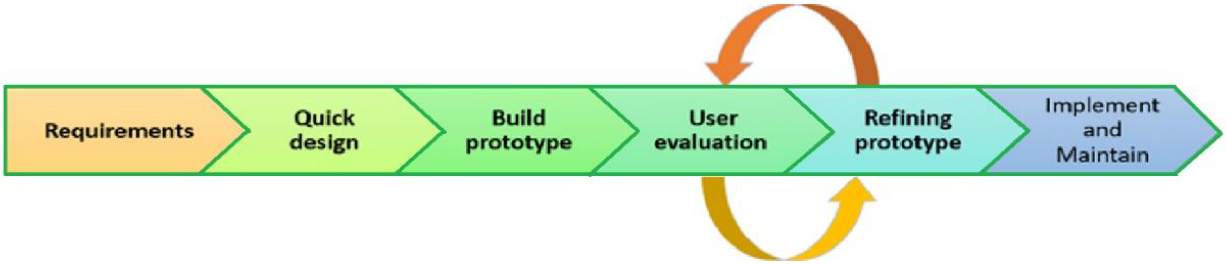


Figure 3.1.2 Prototyping model

Prototyping methodology, it facilitates continuous enhancement based on user feedback and iterative development to create a functional prototype. The development workflow of this application has been divided into several stages which includes **requirement gathering, quick design, prototyping, user evaluation, refining prototype and deployment.** For the first, user requirements had been gathered and analyzed. This step consists of identifying the real user needs and key functionalities, researching on the limitations on current financial apps and improvements that can be applied to our financial apps. Moving forward to the design phase, Figma had been utilized in order to create a simple UI wireframe, which aims to provide users with a clean and simple interface. This prototype will outline the app's navigation flow, feature interactions, and overall user experience. This is because we need to ensure the functionality and usability are designed appropriately, before the full development process begins. During the prototyping phase, a functional prototype will be developed, which consists of features like basic expenses recording, receipt and transaction scanning, which allows for initial testing of these core features. Moving on, the prototype will be distributed to the real users for testing and gather valuable feedback for further improvement. Lastly, on the testing phase, the coding, testing, and refining process will be carried out iteratively, in order to improve the overall performance of

this system. This approach provides flexibility for changes and continuous improvements in order to meet the evolving business needs. The final version of this application will be released and deployed on users' site, when the users are satisfied with the prototype and the system was tested and free from errors. This approach ensures that, by the end of development, the system effectively fulfils the real users' needs and meet their expectations.

During the **project execution** phase, the planned requirements will guide the actual development of this mobile application. This stage involves UI/UX design and feature implementation. We will use Figma to design an interactive prototype that demonstrates the linkage between screens for seamless navigation. Iterative development in stages will be implemented for those key features, like OCR text recognition, automatic expense categorization, and real-time transaction tracking. Flutter (Dart) will be used for frontend development, while Firebase Firestore will handle real-time data storage and user authentication. In order to provide a clearer picture to users regarding their spending patterns, the financial visualization and budgeting tools, including the charts and reports, will be embedded. For receipt scanning, the Google Cloud Vision API, a cloud-based OCR, will be utilized to handle the complex receipt formats. This is because it provides high level of accuracy for text extraction. Further, it also supports structured data extraction, and it efficiently parses the receipt details like amounts, purchased products, and vendor names. Later, we will use GPT-based natural language processing to further analyze and categorize the extracted text. Finally, these transaction details will be stored in Firebase Firestore, which ensures real-time synchronization across devices.

In the **project monitoring and control** phase, we will evaluate the system's accuracy, accessibility and performance. A usability testing will be conducted in order to gather feedback on the app's navigation, OCR precision, and financial monitoring efficiency. For instance, the accuracy of OCR readings must be greater than 90%, and real-time data has to sync within five seconds. There will be security audits carried out in order to check the encryption of data, the authentication of users, and the security of cloud storage. Lastly, before deployment process, the application will undergo numerous

iterations to fix the bugs. As a result, a high-quality application will be sent to the user's hands.

Finally, during the **project closure** phase, a fully functional version of the personal finance management app with spending monitoring will be completed, optimized, and packaged for launching. A final testing will be carried out to verify data security, performance effectiveness, and app stability. To ensure that users understand how to use the app properly, a basic user manual will also be created and proposed. Lastly, an evaluation system will be implemented, enabling users to report issues and suggest improvements.

3.2 System Design Diagram

3.2.1 System Architecture Diagram

This Personal Financial Management Application with Spending Monitoring is built with the combination **of both Client-Server model and Layered architecture approach**. This design divides the system into three main layers which consists of Presentation, Business Logic, and Data layers that had shown in Figure 3.2.1.

At the Presentation Layer, it consists of the Flutter client application, which is the main interface for users to connect and interact with the system. This includes screens for login or sign-up, home screen, charts, budgeting, income and expenses recording. Then, the user input will be handled by the client app and send to the Business Logic Layer for processing. This layer manages the main control flow by using different controllers along with services set up in StatefulWidgets. These controllers check for the validity of user input, process form data, and make sure well integration between the UI and backend operations.

All back-end services are processed by the Data Layer. This includes Firebase Authentication for secure sign-up and login, Cloud Firestore for storing real-time data of income, expenses, and budget entries, and integration with external AI services. A Google Cloud Vision API is used to extract text from receipts or the transaction details. Then, the text that had been extracted is sent to OpenAI GPT and will be further processes in order to analyze the store name, total spending amount, and category of each expense. Later, these details will be return in a structured format for user to check is the scanning result valid or not. As a result, an automatically financial monitoring application had been provided to users with the integration of these features.

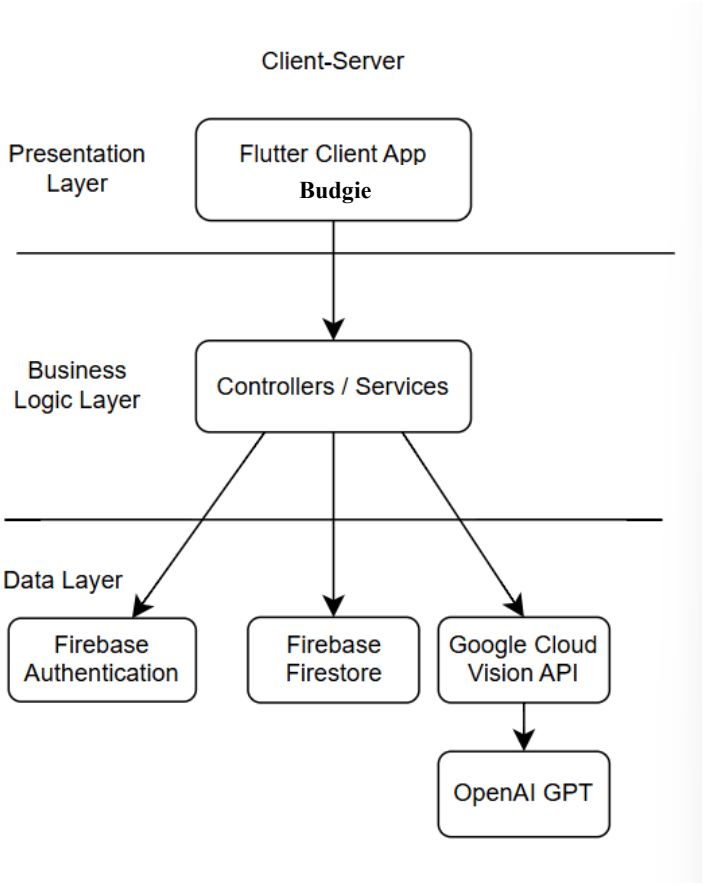


Figure 3.2.1 System Architecture Diagram

3.2.2 Use Case Diagram and Description

Figure 3.2.2 which is the use case diagram illustrates the interactions between users and the functionality inside this Personal Financial Management Application with Spending Monitoring. The primary actor is the **User**, which includes both **New Users** and **Existing Users**. For new users, they need to register an account first before using this application, while for the existing users just needed to login their own accounts to access the system. Once the user login successfully, there are various functionalities provided for them to access which includes adding expenses, adding income, setting budget, viewing transactions, viewing charts, and logging out of the application.

The **Add Transaction** use case is extended by two optional sub-functions. This means that user can either add their income or expenses into their account.

The **Add Expense** use case is extended by two optional sub-functions. This means that the user is able to select either enter their expense details manually or by scanning receipt or transaction details to record the expenses. Finally, the expenses whether recorded through manual entry or by scanning a receipt is passed to the **Update to Database** use case, in order to guarantee that the data is persistently stored.

The **Add Income** use case requires user to enter the income details and select a category. Both are connected via <<include>> relationship, this means that they are essential components of the process. Subsequently, the income details were passed to the **Update to Database** use case, to make sure the users' incomes data are stored persistently.

The **Set Budget** use case is extended by the ability for user to enter a budget and view the current budget. Subsequently, the modified budgets or new budgets details will also include an **update to the database** for storage of the latest budget information.

The **View Transactions** use case is extended by providing options for user to modify their transaction, including both editing or deleting. This is to make sure flexibility provided to users for managing and updating their transaction records effectively. To ensure data consistency, any modifications made will subsequently be included in the **Update to Database** use case.

The **View Charts** use case is extended by providing options for user to visualize their spending habits by using line chart or bar chat format. Additionally, user may apply

filter towards their transaction data based on day or month. By applying such filters, user able to retrieve their financial data quickly for analysis.

The **Logout** use case provides users the ability to quit the app securely. Overall, the diagram provides a straightforward representation of the structured flow within this personal financial management application It illustrates how user interacts with the system and highlights the “extend” and “include” functionalities supported by the system.

The **View AI-powered personalized savings suggestions** use case enables users to receive intelligent tips such as budget-friendly meals and weather-based reminders, like bringing an umbrella to help reduce unnecessary expenses. To ensure variety and prevent repetition, the suggested meals are not repeated within the same week, providing users with diverse, cost-effective meal ideas throughout the week.

The **View Points** use case allows users to check their total accumulated points displayed on the home screen.

The **Spin Wheel** use case allow users to spin the wheel by using their points to exchange for some rewards. To ensure data consistency, any points earned or deducted will be included in the **Update to Database** use case.

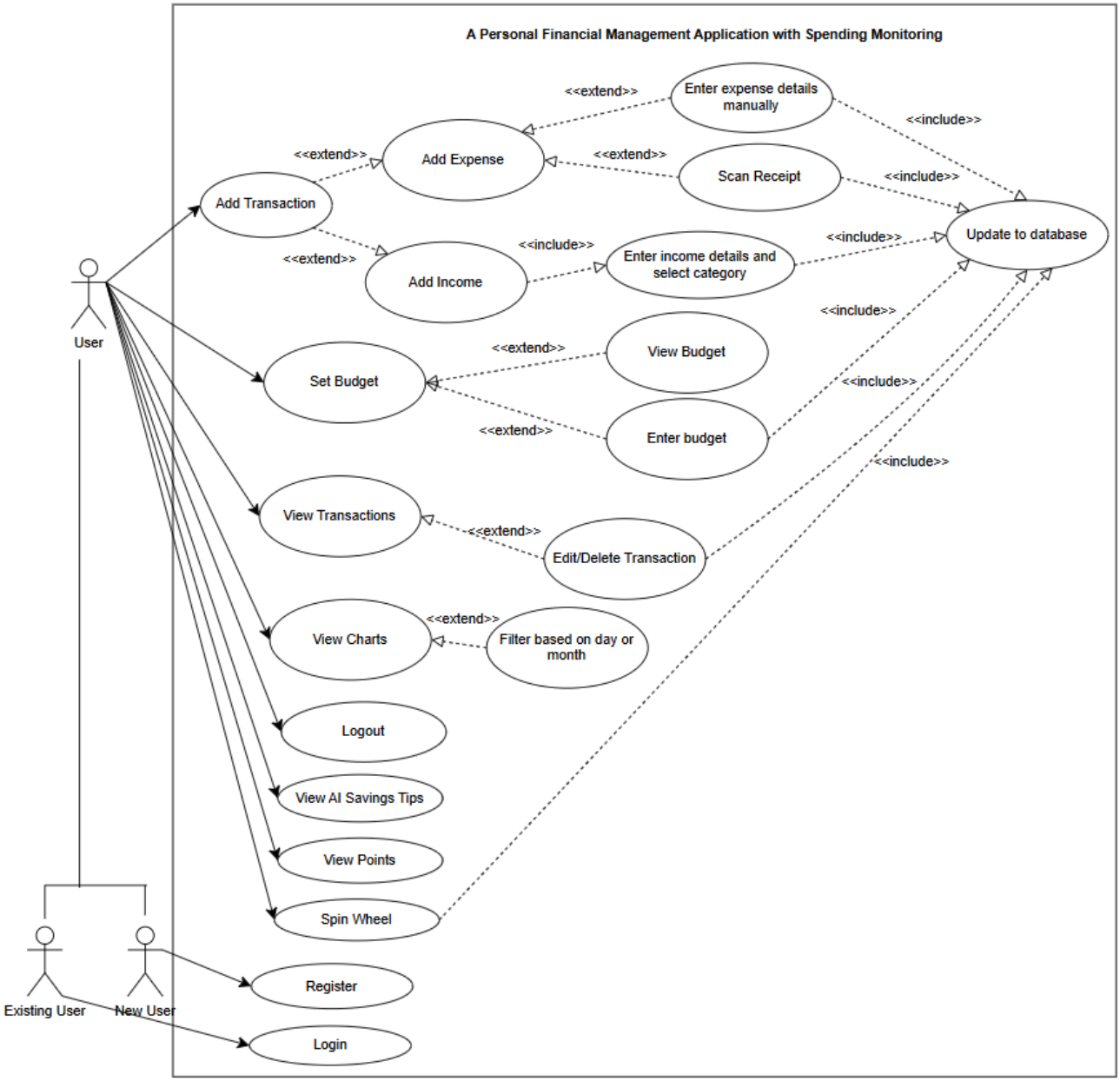


Figure 3.2.2 Use Case Diagram

3.2.3 Activity Diagram

Login and Sign-Up

Figure 3.2.3 illustrates the login and sign-up process for this personal finance management system. First, the user will enter their name, email address, and password to log in or sign up. If he or she is an existing user, then after clicking the login button, the system will validate if all the required fields have been filled in and proceed to credential checking; if it is valid, then log in successfully and redirect to the home screen. If the user enters invalid credentials, the system will display an error message. And if the user is a new user, he or she clicks the sign-up button, and then the system will proceed to check if the email address exists in the database; if yes, then show an error message; otherwise, a successful message about creating a new account is shown, and the user is redirected to the home screen. Later, users will receive points, if they check-in this application daily.

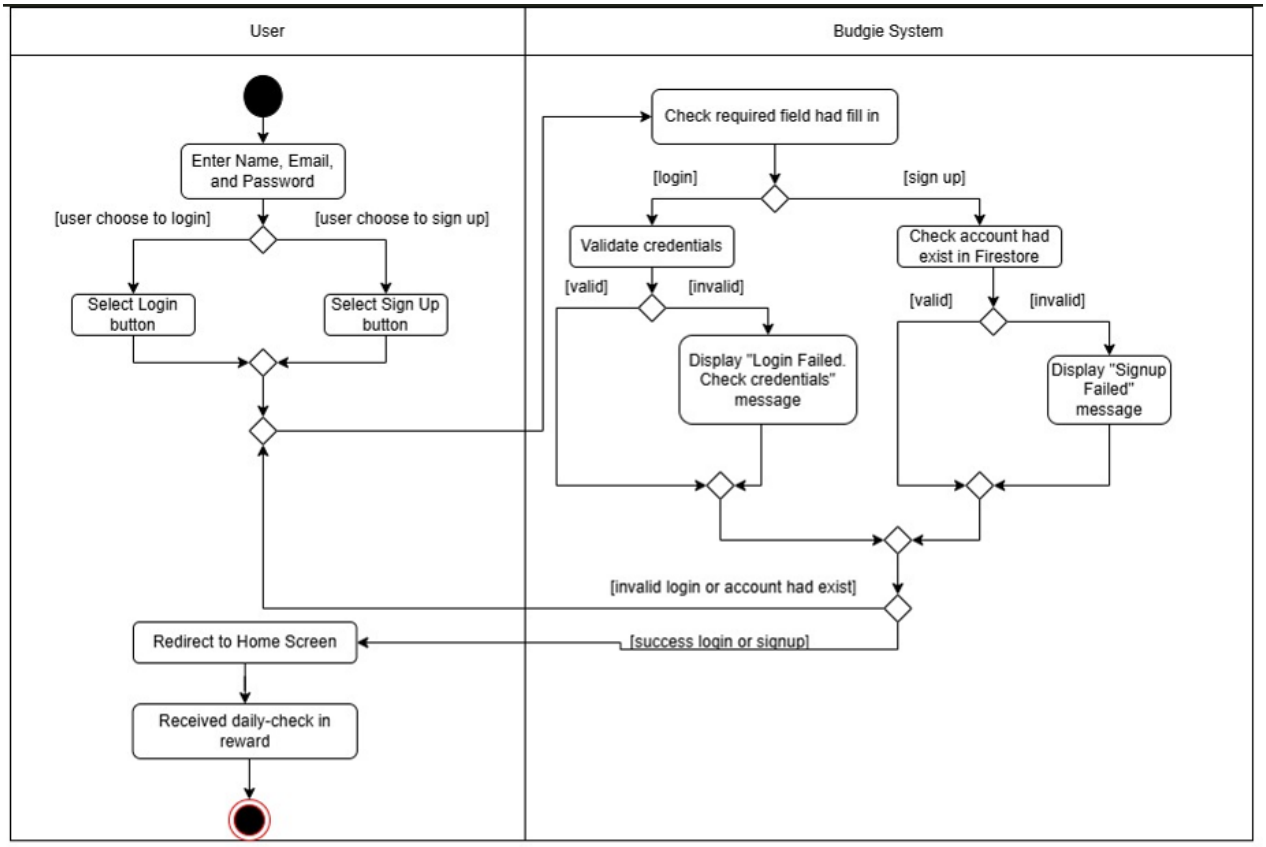


Figure 3.2.3 Login and Sign-Up Activity Diagram

Add Transaction

Figure 3.2.4 shows the add transaction process in this personal finance management system. After the user clicks the "+" button on the home screen, users will be redirected to the Add Transaction screen. There will be a toggle button allowing users to select either add transaction or income. If user select add expenses, then user will be redirected to the add expense form screen, whereas if user select add income, then user will be redirected to the add income form screen. After user had inserted income or expenses successfully, then user will be redirected back to home screen.

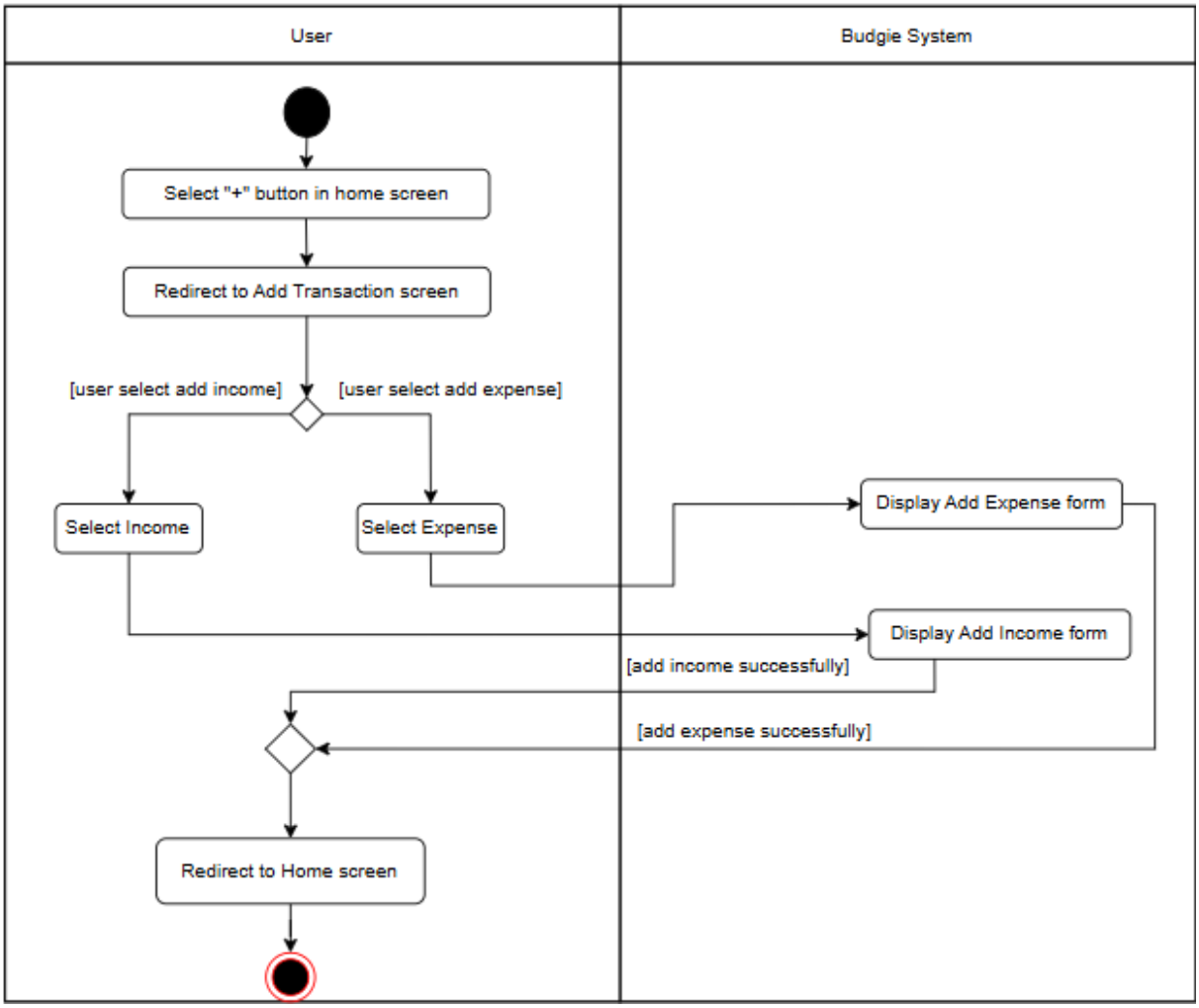


Figure 3.2.4 Add Transaction Activity Diagram

Add Expense (Manual Entry & Receipt Scanning)

Figure 3.2.5 shows the add expense process in this personal finance management system. After the user clicks the "+" button on the home screen, the Add Transaction screen appear, and show add expense form as default. The user has the option to either manually add the expense or scan the receipt. If the user selects to add the expense manually, they are required to input the store name, amount, and select the category of the expenses. Alternatively, the user can select the "Scan Receipt" button, which able to upload receipt or transaction details either by capturing it with the camera or selecting an image from the gallery. Once the receipt is uploaded, the system processes it using Google Cloud Vision API, an OCR tool, to extract key details such as the store name and amount. Later, GPT will used to determine the category of expenses. At this point, there are three conditions for processing the receipts:

1. Single Image with One Transaction

After extraction of text and analyzing the category, the system displays the detected details for the user to review. Once user presses the "Confirm" button. The system will fill in the extracted data into the respective fields in the Add Expense dialog. The user then presses "Save Expense", prompting the system to validate the input and store the data in the database.

2. Multiple Images Selected from Gallery

If the user uploads more than one image, the system automatically enters background processing mode. And a popup is used to inform the user that the receipts will be processed asynchronously. The system then loops through each image, applies OCR and GPT, and automatically saves the extracted transactions without further user input.

3. Single Image with Multiple Transactions (e.g. e-wallet notifications)

The system will prompt background processing if it detects multiple transaction entries in a single image via GPT. And a popup is used to inform the user about this detection, and later the system will extract and save automatically all the transactions that are identified.

After either a manual or scanned entry, once the expenses were saved, the system will update the budget usage, expense list, and total expense amount displayed on the home screen. If the user clicks the “Cancel” button instead, the dialog closes, and they are redirected back to the home screen.

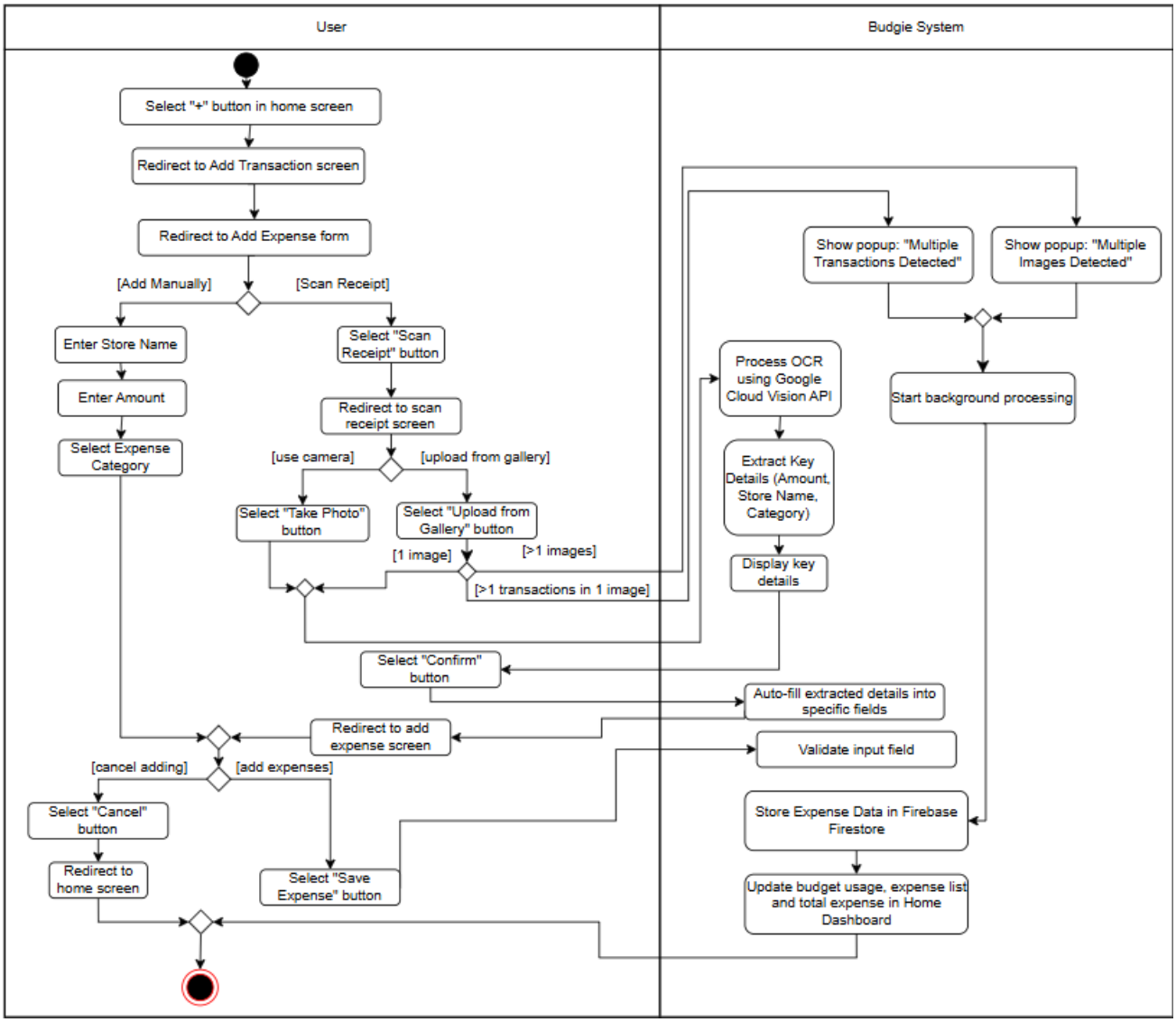


Figure 3.2.5 Add Expense Activity Diagram

View & Manage Expenses

Figure 3.2.6 shows the process of viewing and managing expense listings in this personal finance management system. For existing users, the home screen will show up to four transaction activities. If the user wants to view all transactions, they can click the "See All" button, and the system will redirect them to the Activity Listing Screen, which provides a detailed overview of both expenses and incomes. However, for a new user, a message “No expense records yet” will be shown. To update an expense, the user can swipe right on an expense row, which will open the Edit Expense screen. After making modifications, the user presses the “Save Expense” button to update the changes. To delete an expense, the user must swipe left the expense entry, after which a Delete button will appear beside the row. The user can then press “Delete” to remove the expense. Once an expense is modified or deleted, the system first validates the changes, then updates the database and application interface to reflect the latest modifications.

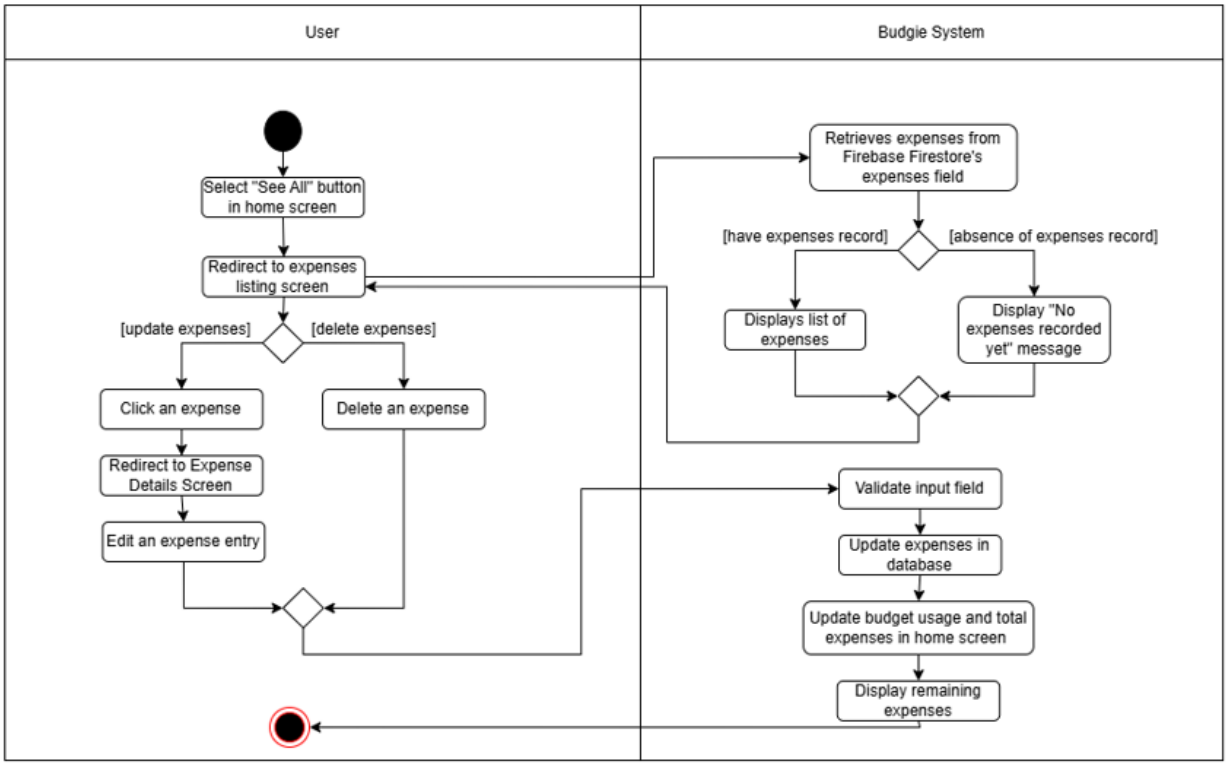


Figure 3.2.6 View and Manage Expenses Activity Diagram

Budget Management

Figure 3.2.7 shows the budget management process in this personal finance management system. The user presses the budget icon located on the rightmost side of the bottom navigation bar, then the user will be redirected to the Budget Screen. For new users, a message stating “No budgets set” will be displayed. The system will show existing users how much of their budget they had been used and offer a savings amount, which is 20% of the remaining budget. The budget is visually represented by a progress bar that refreshes continuously according to the user's spending. If the entire amount spent stays within 80% of the budget, the bar remains blue, reflecting regular expenditure. If the user spends more than 80% of the budget, the bar turns red to indicate overspending. The user was allowed to input a new budget amount and clicks the "Save Budget" button to change the budget. The system then validates the input, verifying it is a positive integer, and updates both the database and the application interface with the latest modifications. If overall expenditure exceeds the chosen budget, the advised savings amount is reduced to RM 0.00, and the system will notify the user with an alert message. Besides, a point-based reward system will also be implemented. One hundred points will be awarded to users if their expenses for the month remain within the budget they set. This feature is used to motivate users to spend wisely.

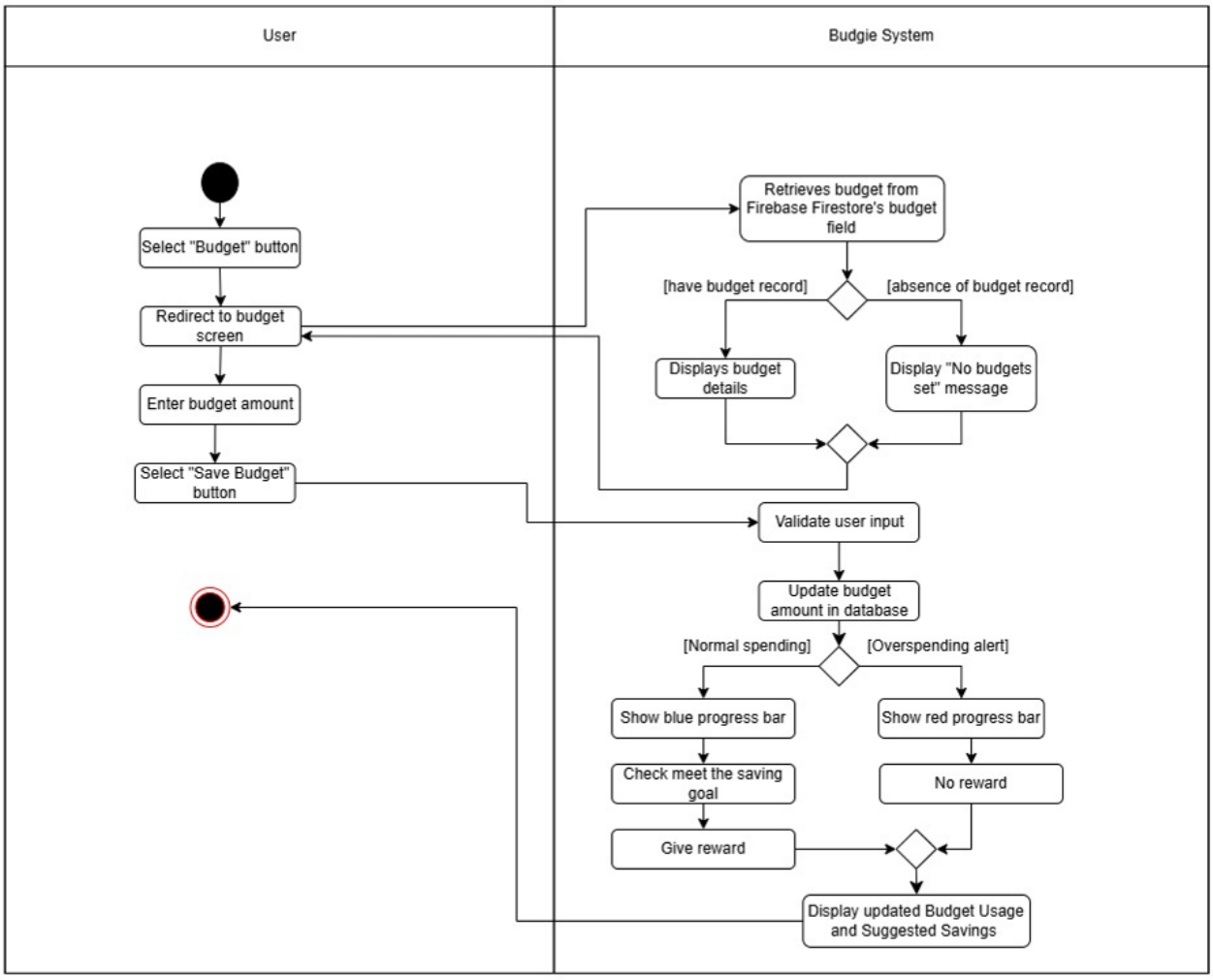


Figure 3.2.7 Budget Management Activity Diagram

Income

Figure 3.2.8 the process of inputting income in this personal finance management application. The user begins by clicking the income box on the home screen, which triggers the Income Dialog to pop up. The user then enters the income amount and selects the income category from a dropdown list. After user presses the “Save Income” button, the system will validate the input and later update in database and application interface to reflect the latest financial information. Once successfully saved, the user is automatically redirected back to the home screen. In order to provide greater flexibility and convenience to user, a Monthly Income Repeat feature will be implemented in the future, which allowing users to set their fixed income to be automatically recorded each month.

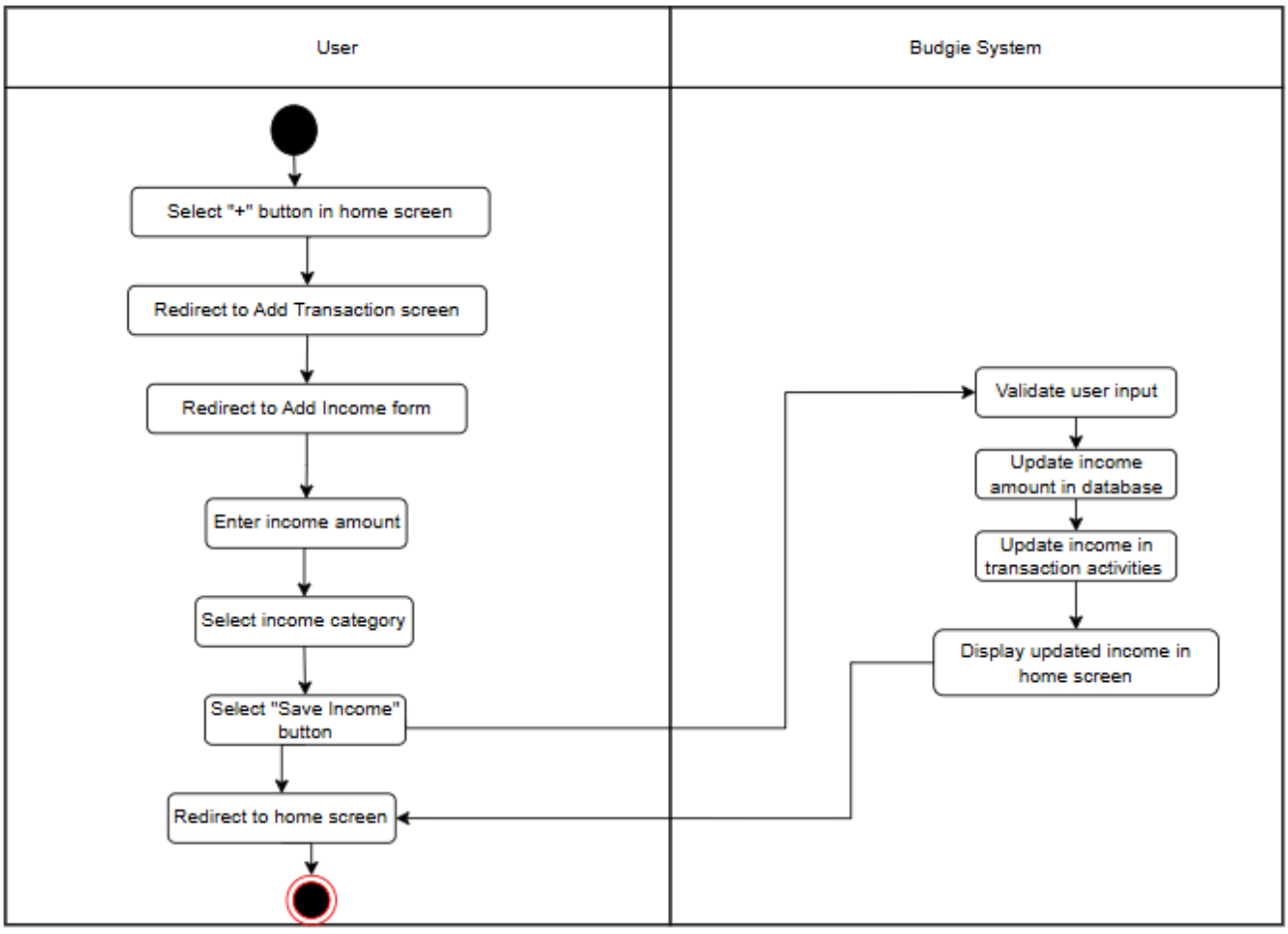


Figure 3.2.8 Income Activity Diagram

Data Visualization (Pie Chart & Expense Trends)

Figure 3.2.9 shows the data visualization process in this personal finance management application. This feature provides users with charts to help them better understand their spending patterns. The process starts with the user pressing the Total Expenses box on the home screen, then user will be redirected to the Chart Analysis Screen. The system then concurrently retrieves and verifies the availability of expense and income records from the database. On the Day Chart Screen, the system displays a vertical bar chart representing daily expenses and a horizontal bar chart showing category-based expense distribution. If the user switches to the Month Chart Screen, a line chart is presented, illustrating income vs. expenses from January to December. Different colors are used to distinguish income from expenses, making it easier for users to analyze financial trends over time.

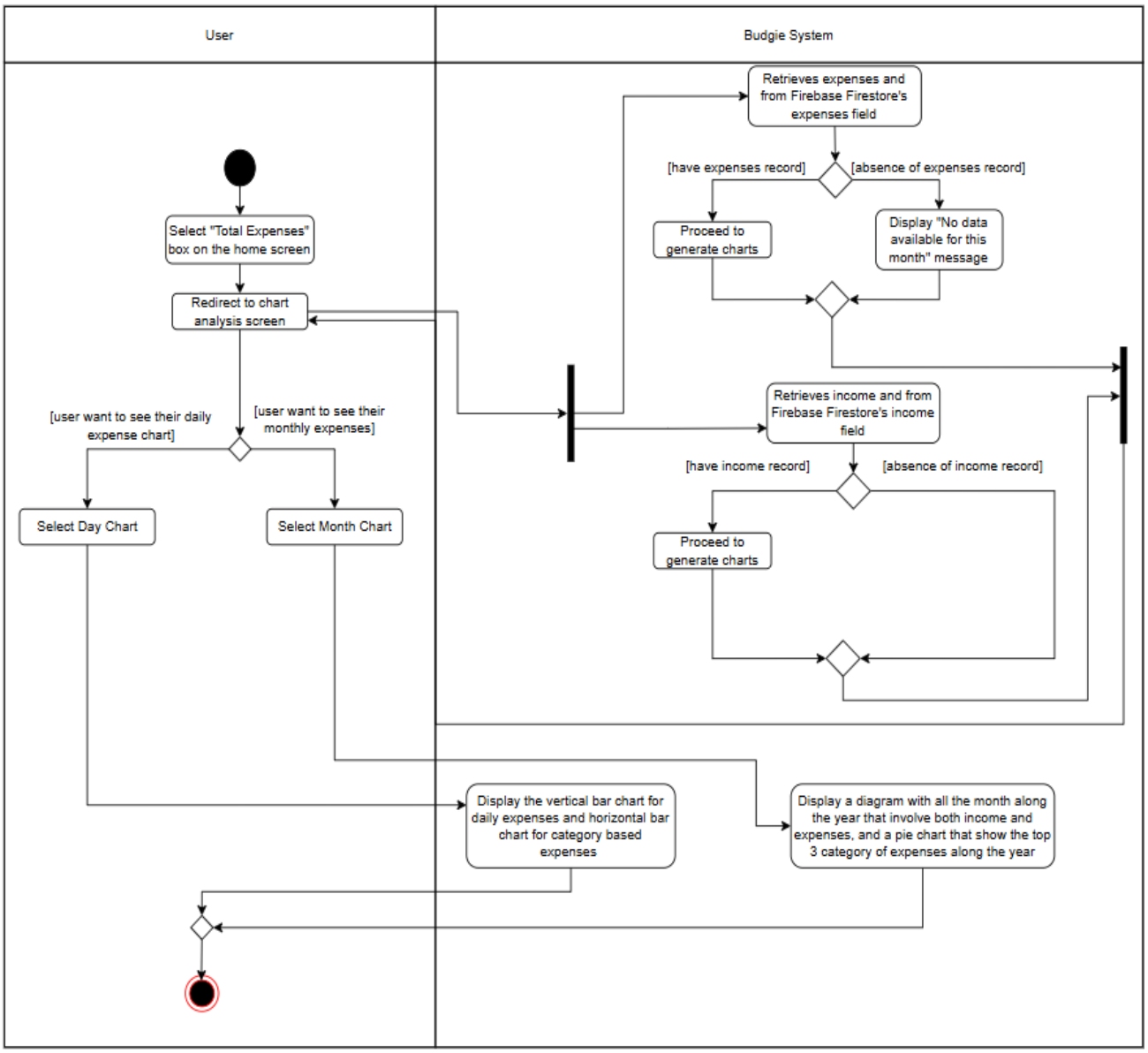


Figure 3.2.9 Data Visualization Activity Diagram

Spin Wheel Reward System

Figure 3.2.10 shows the Spin Wheel Reward System process in this application. This feature motivates users to record their expenses or income consistently, by allows users to earn points when they check in daily into this app. Then, users spin a virtual wheel, and 50 points will be deducted for each spin. There are few rewards provided which includes TNG cash voucher, points, or no rewards at all. The process starts with the user pressing the Reward Point box on the home screen, which redirects them to the Spin Wheel Reward Screen. The system then displays the spin wheel interface along with the user's current points balance. When the user clicks the Spin Wheel button, the system will first determine whether the user has enough points to spin. However, if the user does not have sufficient points to start a spin, then an "Insufficient points" message will be shown to users. Otherwise, the system deducts the required points and starts to spin. After a random result had been generated, the system starts to determine whether the user has won a reward or not at all. Then, the system will display pop up message accordingly. If the outcome is reward points, the points awarded will be update to the user's points balance, and a "Congratulations, you earned X points!" popup will be shown to user. If the outcome is a TNG cash voucher, a popup message informs the user of their win along with instructions to redeem the voucher. If the user does not win any reward, a "Better luck next time" popup is shown. Finally, the system updates the database to reflect any modifications to the user's reward status or point balance, in order to maintain data consistency.

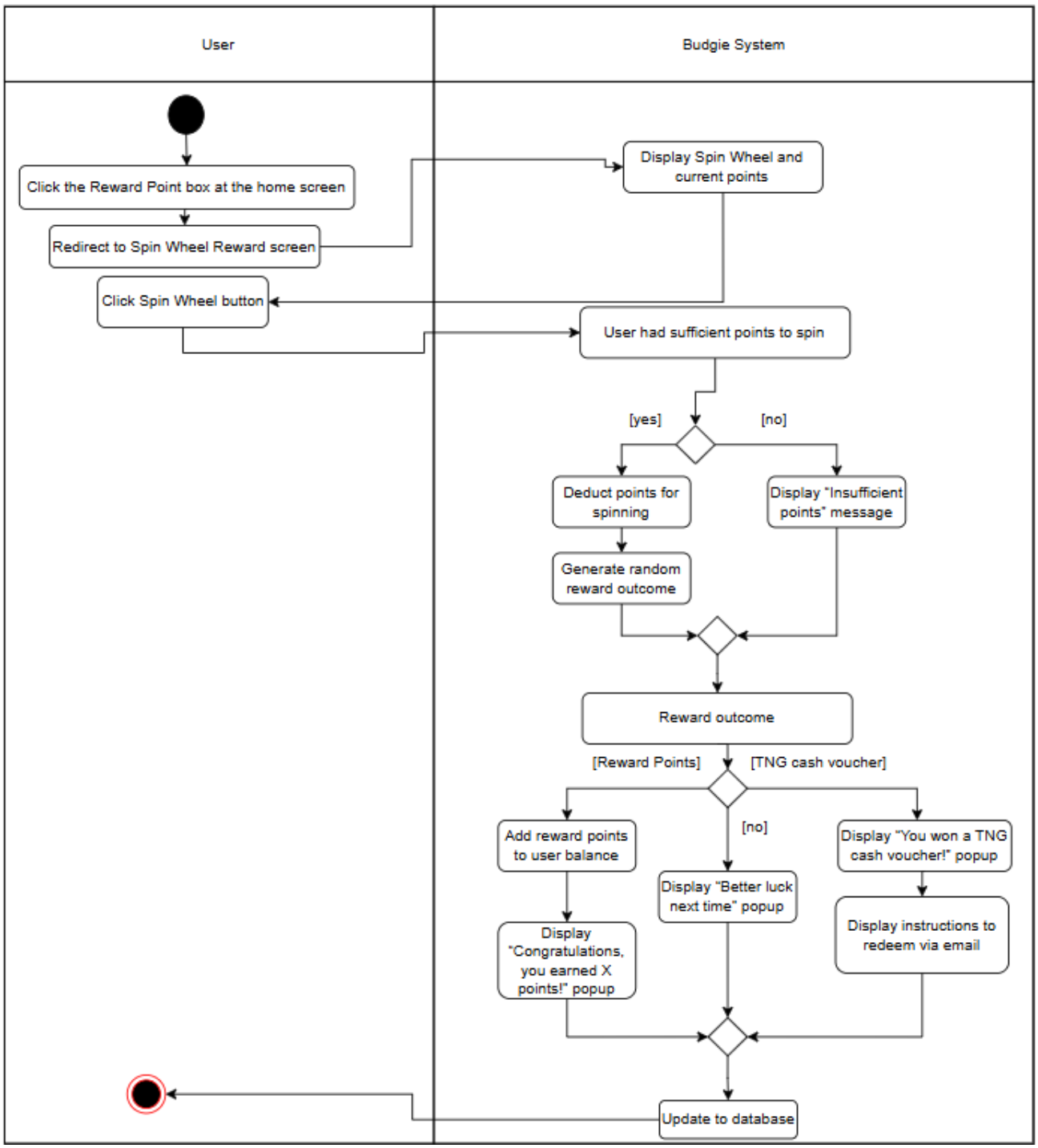


Figure 3.2.10 Spin Wheel Reward System Activity Diagram

AI Tips Suggestion

Figure 3.2.11 shows the process flow of the AI-powered savings suggestion feature in this personal finance management application. First, the user selects the "AI Tips" option from the bottom navigation bar, then user will be redirected to the AI Tips screen. Once user enter this screen, user's current location will be retrieved by the system which is then used to fetch real-time weather information from the OpenWeatherMap API. Simultaneously, the system retrieved the total expense for current month from Firestore, in order to calculate the user's remaining budget. These contextual inputs, like location, budget, and weather, are subsequently passed into the GPT model to generates a personalized savings tip. The suggestion generally consists of the weather-related reminder like bring an umbrella if rain is detected, as well as a budget-friendly meal recommendation. The tip will be shown to the user once it was successfully generated. If the user chooses to save the suggestion, the system stores the tip in Firestore, along with a timestamp and content details for future reference. To prevent the repetition of the same meal suggestions which within a two-week period, hence the tips record is used to provide users with diverse suggestion. Finally, the user is redirected back to the Home Screen, completing the flow.

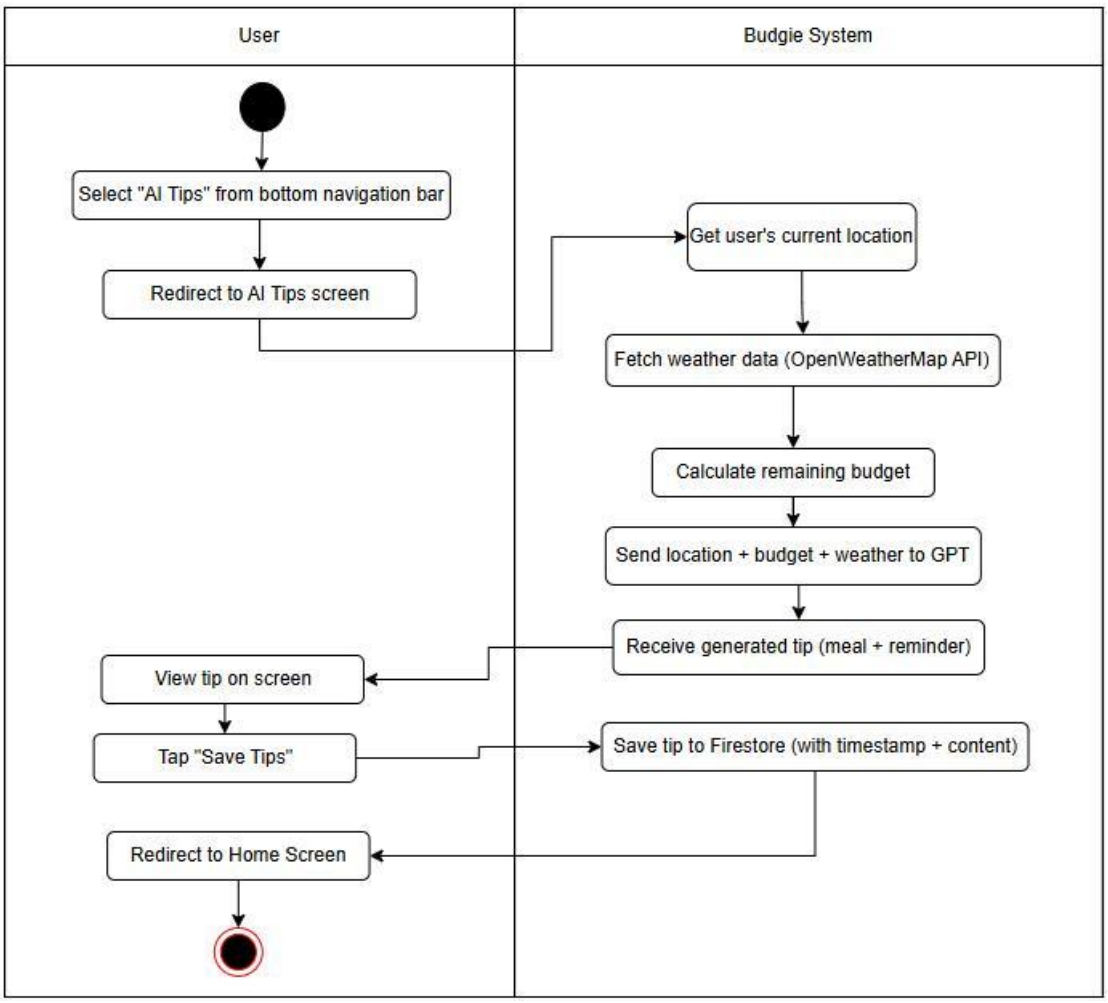


Figure 3.2.11 AI Tips Suggestion Activity Diagram

User Logout and Account Management

Figure 3.2.12 shows the logout process of this personal finance management application. The user initiates the logout by pressing the logout icon located at the top right corner of the home screen. Once the button is pressed, the system receives the logout request, and Firebase Authentication proceeds to clear the current user session. After the session is successfully terminated, the user is redirected back to the login screen.

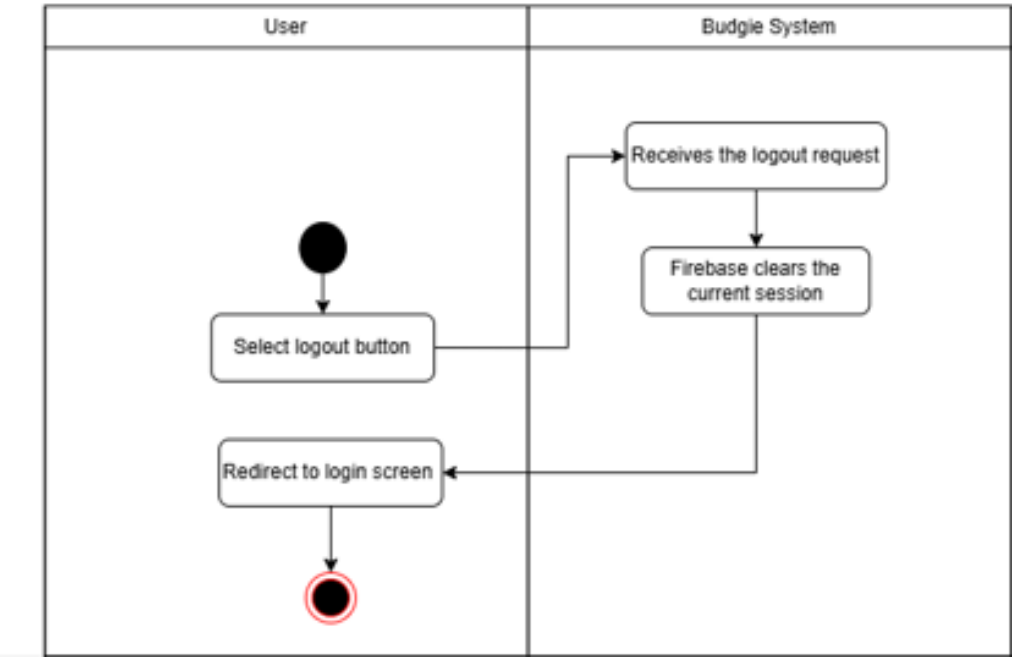


Figure 3.2.12 User Logout and Account Management Activity Diagram

3.3 Project Timeline

	Name	Duration	Start	Finish
1	**Project Initiation Phase**	**10 days?**	**2/10/25 8:00 AM**	**2/21/25 5:00 PM**
2	Conduct feasibility study (OCR + GPT + Firebase)	4 days?	2/10/25 8:00 AM	2/13/25 5:00 PM
3	Research existing personal finance apps	2 days?	2/14/25 8:00 AM	2/17/25 5:00 PM
4	Identify user needs	2 days?	2/18/25 8:00 AM	2/19/25 5:00 PM
5	Define project scope and objectives	2 days?	2/20/25 8:00 AM	2/21/25 5:00 PM
6	**Project Planning Phase**	**4 days?**	**2/24/25 8:00 AM**	**2/27/25 5:00 PM**
7	Define project methodology (Prototyping model)	1 day?	2/24/25 8:00 AM	2/24/25 5:00 PM
8	Create initial project plan and timeline	1 day?	2/25/25 8:00 AM	2/25/25 5:00 PM
9	Identify key functional features	2 days?	2/26/25 8:00 AM	2/27/25 5:00 PM
10	**Project Execution Phase**	**24 days?**	**2/28/25 8:00 AM**	**4/2/25 5:00 PM**
11	**Design**	**4 days?**	**2/28/25 8:00 AM**	**3/5/25 5:00 PM**
12	Design UI wireframe using Figma	2 days?	2/28/25 8:00 AM	3/3/25 5:00 PM
13	Create navigation flow and user experience plan	1 day?	3/4/25 8:00 AM	3/4/25 5:00 PM
14	Finalize architecture (Client-Server + Layered)	1 day?	3/5/25 8:00 AM	3/5/25 5:00 PM
15	**Development**	**20 days?**	**3/6/25 8:00 AM**	**4/2/25 5:00 PM**
16	Setup Flutter project + Firebase integration	5 days?	3/6/25 8:00 AM	3/12/25 5:00 PM
17	Develop key functionalities	15 days?	3/13/25 8:00 AM	4/2/25 5:00 PM
18	**Monitoring and Control Phase**	**4 days?**	**4/3/25 8:00 AM**	**4/8/25 5:00 PM**
19	Conduct usability testing (UI/UX + OCR accuracy)	3 days?	4/3/25 8:00 AM	4/7/25 5:00 PM
20	Perform functional testing (all features)	3 days?	4/3/25 8:00 AM	4/7/25 5:00 PM
21	Debug and refine features iteratively	4 days?	4/3/25 8:00 AM	4/8/25 5:00 PM
22	**Project Closure Phase**	**13 days?**	**4/9/25 8:00 AM**	**4/25/25 5:00 PM**
23	Finalize FYP 1 report	10 days?	4/9/25 8:00 AM	4/22/25 5:00 PM
24	Prepare for presentation	3 days?	4/23/25 8:00 AM	4/25/25 5:00 PM

Figure 3.3.1 Project Timeline for Final Year Project 1

	Name	Duration	Start	Finish
1	**FYP2 Initialization and Planning Phase**	**2 days?**	**6/23/25 8:00 AM**	**6/24/25 5:00 PM**
2	Review feedback from FYP1, set FYP2 objectives	1 day?	6/23/25 8:00 AM	6/23/25 5:00 PM
3	Review system scope and architecture	1 day?	6/24/25 8:00 AM	6/24/25 5:00 PM
4	**Core Development Phase**	**16 days?**	**6/25/25 8:00 AM**	**7/16/25 5:00 PM**
5	Implement advanced features (e.g. OCR multi-receipt, GPT refinement)	8 days?	6/25/25 8:00 AM	7/4/25 5:00 PM
6	Enhance reward systems and budget logic	2 days?	7/5/25 8:00 AM	7/8/25 5:00 PM
7	Improve mobile UI/UX design responsiveness	3 days?	7/9/25 8:00 AM	7/11/25 5:00 PM
8	Add AI tips suggestion features	3 days?	7/12/25 8:00 AM	7/16/25 5:00 PM
9	**Testing and Debugging Phase**	**12 days?**	**7/17/25 8:00 AM**	**8/1/25 5:00 PM**
10	Perform functional testing (add, edit, chart, reward)	2 days?	7/17/25 8:00 AM	7/18/25 5:00 PM
11	Conduct usability testing (UI/UX & OCR)	2 days?	7/20/25 8:00 AM	7/22/25 5:00 PM
12	Distribute app to users for testing	3 days?	7/23/25 8:00 AM	7/25/25 5:00 PM
13	Collect feedback through survey	1 day?	7/26/25 8:00 AM	7/28/25 5:00 PM
14	Fix bugs & optimize performance	4 days?	7/29/25 8:00 AM	8/1/25 5:00 PM
15	**Evaluation and Documentation**	**29 days?**	**8/2/25 8:00 AM**	**9/11/25 5:00 PM**
16	Analyze feedback, finalize system performance metrics	5 days?	8/2/25 8:00 AM	8/8/25 5:00 PM
17	FYP 2 Report Writing	20 days?	8/11/25 8:00 AM	9/5/25 5:00 PM
18	Proofread and finalize full report	4 days?	9/8/25 8:00 AM	9/11/25 5:00 PM
19	**Final Preparation Phase**	**7 days?**	**9/15/25 8:00 AM**	**9/23/25 5:00 PM**
20	Design poster and prepare slides	3 days?	9/15/25 8:00 AM	9/17/25 5:00 PM
21	Prepare for presentation	4 days?	9/18/25 8:00 AM	9/23/25 5:00 PM

Figure 3.3.2 Project Timeline for Final Year Project 2

CHAPTER 4 SYSTEM DESIGN

4.1 Database Design

The following Table 4.1.1 to Table 4.1.8 outlines the database design for the personal financial management application, which employs Firebase Firestore as its cloud-based NoSQL database. There are a total of eight key collections being included for this application. Each table below includes the column name, data type, constraints, a brief description, and an example value to provide a clear picture of the data structure.

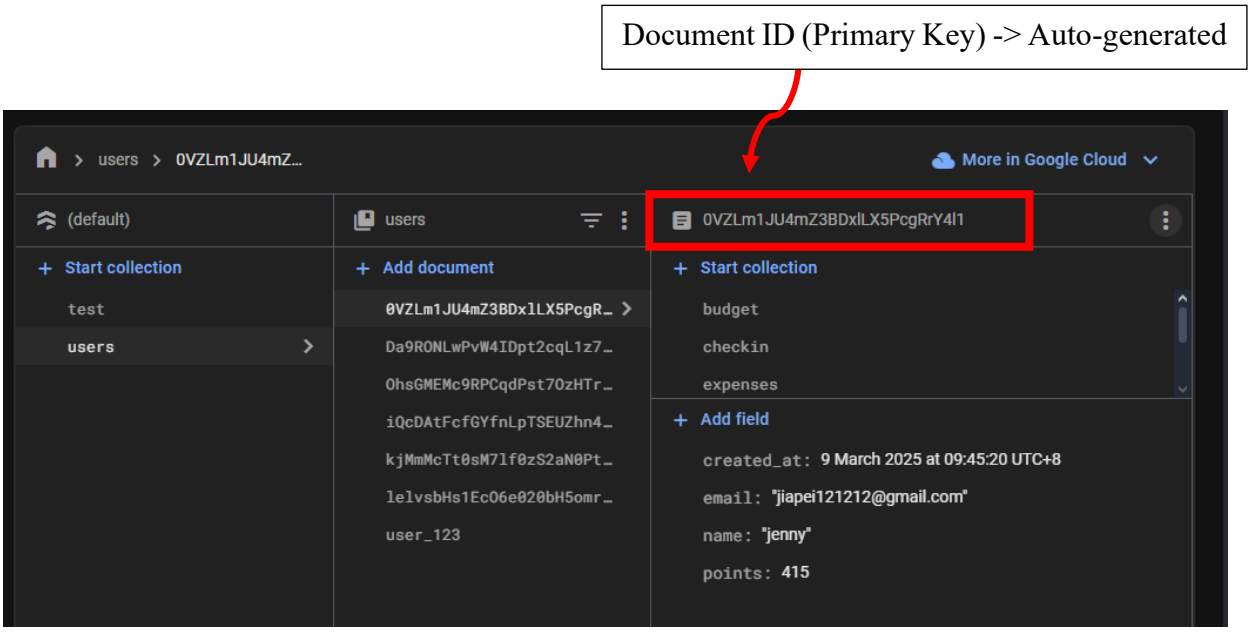
a) Users Collection

*Table 4.1.1 Database Design for the **users** Collection*

Column Name	Data Type	Constraints	Description	Example Value
name	String	Required	Username inside this application	“jenny”
email	String	Required, Unique	User’s email address	"jiapei121212@gmail.com"
created_at	Timestamp	Auto-generated	Account creation time	9 March 2025 at 09:45:20
points	Integer	Required, Default 0	Total reward points had accumulated	415

Figure 4.1.1 shows the Firestore structure of the users collection. The primary key is the document ID, which is used to represent each unique user in the application. According to the Figure 4.1.1, the user "jenny" had been identified with a Document ID 0VZLm1JU4mZ3BDxlLX5PcgRY4l1, and inside her profile, it consists of these fields such as name, email, points, and created_at. Subcollections like budget, checkin, expenses, income, recurring income, rewards, receipts are nested under this document to store related financial data.

Figure 4.1.1 Firestore Structure for Users collection



b) Subcollections under Users Collection

i) Expenses

*Table 4.1.2 Database Design for the **expenses** Collection*

Column Name	Data Type	Constraints	Description	Example Value
store_name	String/null	Optional	The location where user spend their money	"KFC"
total_amount	Double	Required, value > 0	User's spending amount	23.75
category	String	Required	Category of expenses	"Food"
timestamp	Timestamp	Auto-generated	Date and time the expense was recorded	2025-07-01T13:45:00Z

ii) Income

*Table 4.1.3 Database Design for the **income** Collection*

Column Name	Data Type	Constraints	Description	Example Value
amount	Double	Required, value > 0	Amount of income recorded by the user	3000.00
category	String	Required	Income category	"Salary"
timestamp	Timestamp	Auto-generated	Date and time when the income was added	2025-07-01T10:00:00Z

iii) Budget

*Table 4.1.4 Database Design for the **budget** Collection*

Column Name	Data Type	Constraints	Description	Example Value
amount	Double	Required, value > 0	The monthly budget amount set by the user, can only be set once per month	2500.00
created_at	Timestamp	Auto-generated	The time the budget was last updated	2025-07-01T00:00:00Z

iv) Checkin

*Table 4.1.5 Database Design for the **checkin** Collection*

Column Name	Data Type	Constraints	Description	Example Value
day	Integer	-	The day of the month the user checked in	3
month	Integer	-	The month the check-in occurred	7
year	Integer	-	The year of the check-in	2025
check_in_at	Timestamp	Auto-generated	The exact time the check-in was recorded	2025-07-01T10:00:00Z
points_awarded	Integer	-	Reward points given for this check-in, calculated on a weekly cycle based on consecutive daily check-ins (e.g., Day 1 = 1 pt, Day 2 = 2 pts ... Day 7 = 7 pts)	3

v) Rewards

*Table 4.1.6 Database Design for the **rewards** Collection*

Column Name	Data Type	Constraints	Description	Example Value
total	Integer	value ≥ 0	User's accumulated reward points	150
last_rewarded_month	Integer	-	The month when the last reward was issued	7
updated_at	Timestamp	Auto-generated	The last time this document was updated	2025-07-01T10:00:00Z

vi) Receipts

*Table 4.1.7 Database Design for the **Receipts** Collection*

Column Name	Data Type	Constraints	Description	Example Value
image_url	String	Required	URL of the scanned receipt image stored	3000.00
category	String	-	Detected or user-assigned expense category	"Food"
store_name	String	-	Store name extracted from the receipt (via OCR/GPT)	"Starbucks"
total_amount	Double	-	Total amount extracted from the receipt	23.50

vii) AI Tips

*Table 4.1.8 Database Design for the **AI Tips** Collection*

Column Name	Data Type	Constraints	Description	Example Value
timestamp	Timestamp	Required	Date and time the tip was generated	27 July 2025 at 22:41:59 UTC+8
tip	String	Required	Combined weather and meal suggestion message	"Weather: cloudy Dinner suggestion: Fried Fish + Soup, home-cooked, RM13.00 Drink suggestion: Iced Milo, RM2.50"

4.2 System Flowchart

Figure 4.2.1 shows the system flowchart for the personal finance management application, which includes features like user authentication, expense tracking, budget management, receipt scanning via OCR, transaction notification capture, AI Savings Tips, point-based rewards system and financial data visualization. The details of each function will be further explained and described in the following subsections.

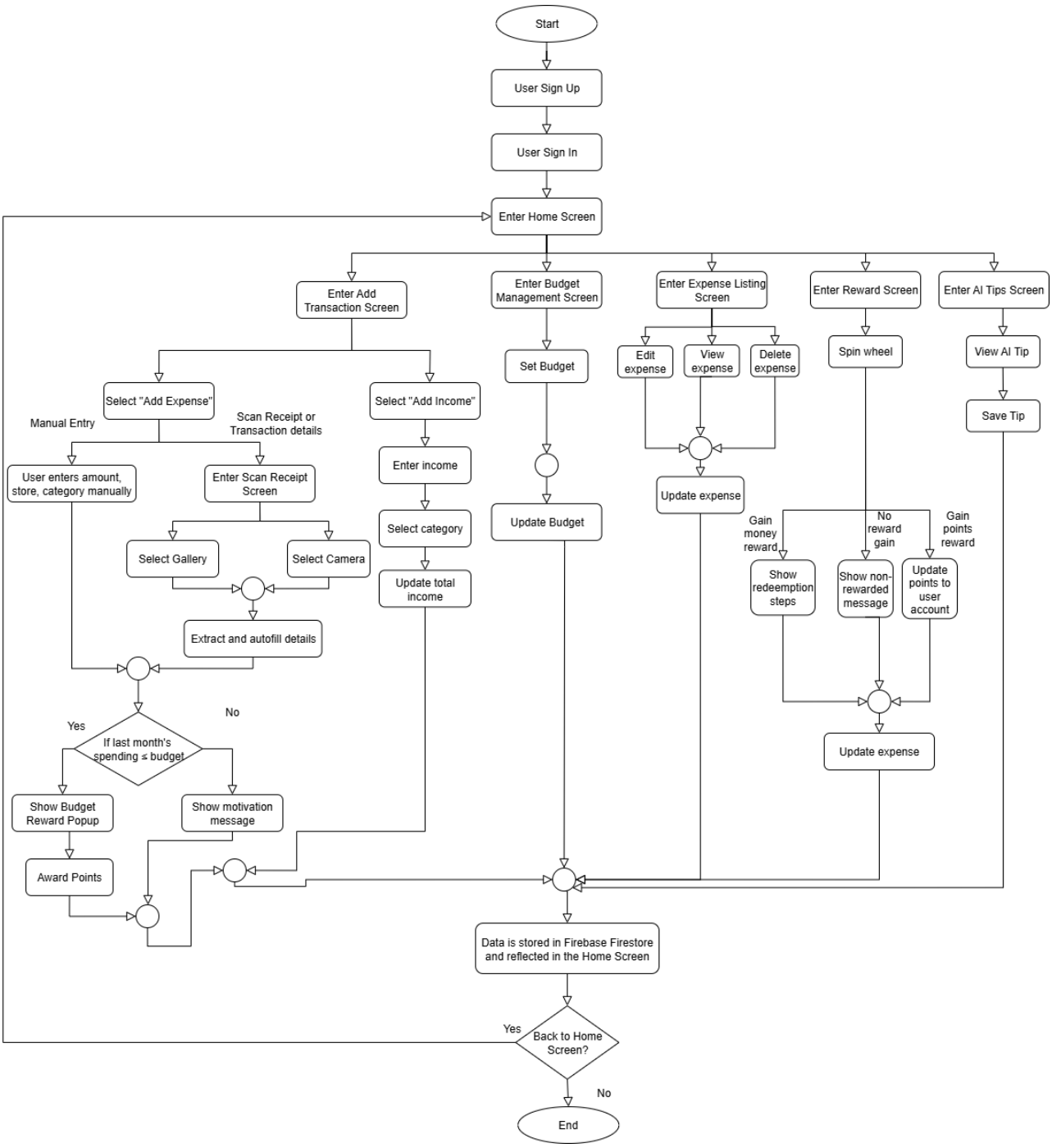


Figure 4.2.1 System Flowchart

4.3 System Block Diagram

According to Figure 4.3.1, it shows the system block diagram for this personal financial management application. The system begins with the first key modules which is the user sign-in (User Login Screen) or sign-up (User Registration Screen) module. After successfully registered, user will proceed to login with their email address and password. After successfully logging in, user will be redirected to the Main Dashboard, which is the Home Screen, it served as the central navigation point. From here, user able to access the six key modules which include Add Transaction Module, Budget Module, Transaction Listing Module, Reward System, and Insights Module, AI Tips Suggestion Module.

In the **Add Transaction Module**, users able to manage both **expenses and income**. User able to add expenses manually or by scanning receipts in the Expense Module. For The Income Module, it allows users to record and manage their income.

For the **Budget Module**, it allows users to set a monthly spending limit and try to track their usage in real time. This module was integrated with the Reward system, as it will award the users with points if their expenses remain within the set budget.

For the **Transaction Listing Module**, a list of past transactions has been provided to users as well as with the ability to modify or delete the expense and income entries. In order to motivate users to record their expenses and income regularly, the **Reward System** has been introduced, which provides the daily check-ins to gain points, and a spin wheel has been used for users to redeem the rewards with their accumulated points.

For the **Insights Module** which allows users to analyze their spending pattern by presenting users with interactive visualizations, including Day Charts, Month Charts, and Category Distribution. Hence, user able to make informed financial decisions.

Last but not least, **AI Tips Suggestion Module** provides users with personalized financial advice based on their remaining budget, local weather conditions, and total expenses. For example, meal suggestions and weather-related reminder had been provided, to help users reduce unnecessary expenses and make more informed financial decisions.

To sum up, this block diagram provides a streamline graphical representation for this personal financial management system. It illustrated how major components are structured and the relationships between those components.

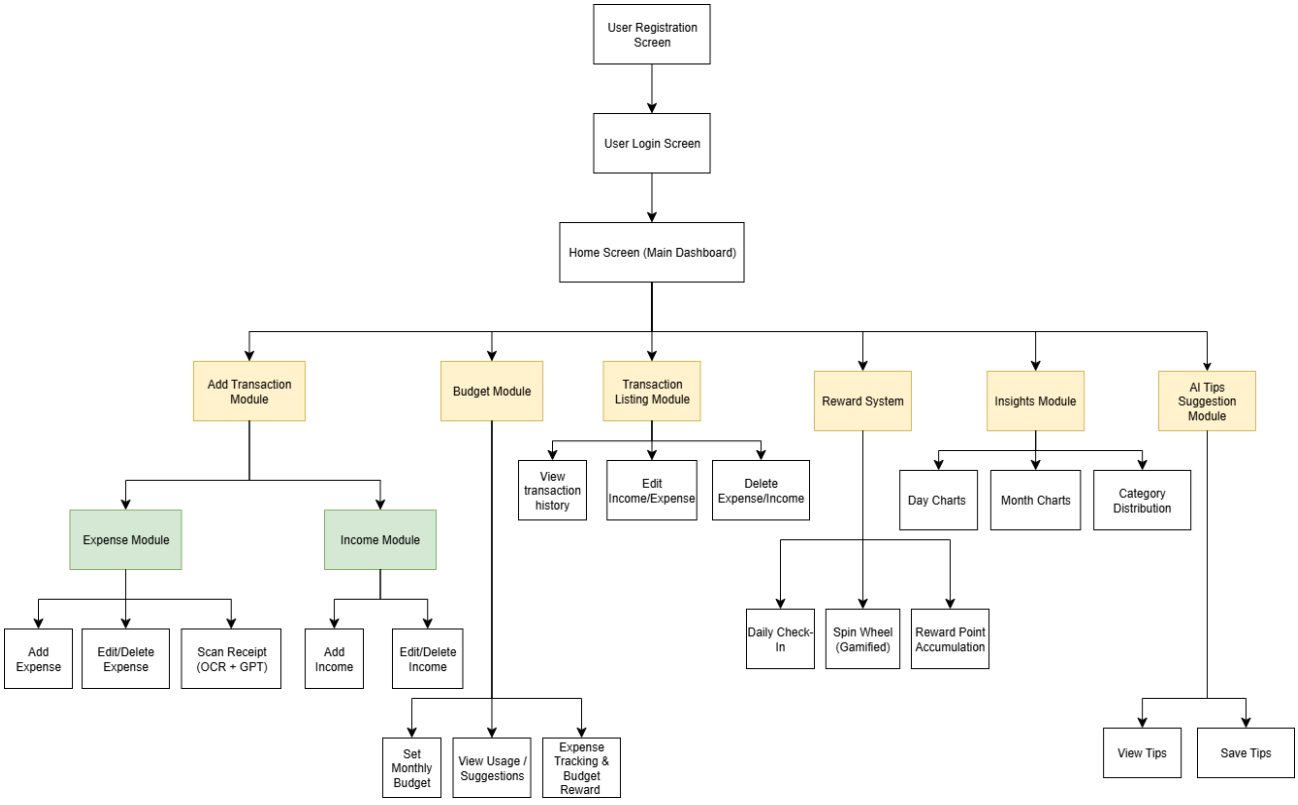


Figure 4.3.1 System block diagram

4.4 System Component Specifications

Below are the implementation details of each module, including an overview of its core functionality, data flow, and how components interact between one another to deliver specific features.

a) User Registration Screen

Description:

Enable new users to register an account by entering their email address, password and name. Users will be redirected to the home screen after they successfully sign up.

Component Requirements:

- **Firebase Authentication** enables new users to create an account securely. It ensures identity verification and user management within the application.
- **Cloud Firestore** is used to store the user profile details, such as name, email address, and created_at (timestamp).
- Once successful registration, the user will be redirected to the Home screen by utilizing the **Navigation components**.

b) User Login Screen

Description:

Enable users to login to their account by entering email address, and password. Users will be redirected to the home screen after they successfully sign in.

Component Requirements:

- **Firebase Authentication** to validate user email and password credentials.
- Once successful login, the user will be redirected to Home screen by utilizing the **Navigation components**.

c) Home Screen (Main Dashboard)

Description:

Serves as the central navigation hub for the application, which allows users to move to other screens, and provide users with quick summarized views toward their financial status such as total expenses, income, and latest transaction activities.

Component Requirements:

- **Cloud Firestore to retrieve and aggregate financial data in real time.**
- **Firebase Authentication** is used to identify the user and fetch their financial data.
- **Stream-based widgets** to reflect live data updates.
- **Navigational control** to allow user to move from home screen to other modules.

d) Add Transaction Screen

Description:

Allow users to add income or expenses.

Component Requirements:

- **Navigational control** to allow user to switch to either add income or expenses.

e) Add Income Screen

Description:

Allow user to add income.

Component Requirements:

- **Cloud Firestore** is used to manage the income entries.

f) Add Expense Screen

Description:

Allow users to add expenses either manually or scanning their receipts.

Component Requirements:

- **Cloud Firestore** is used to manage the expenses entries.
- **Navigational control** allow user to move to scanning receipt screen.

g) Receipt Scanning Screen

Description:

Allow users to add expenses by scanning their receipts, by utilizing the OCR and GPT technology.

Component Requirements:

- **Cloud Firestore** is used to manage the expenses entries, receipts images and details.
- **Device camera and ImagePicker** are used for capturing receipt images.
- **Google Cloud Vision API** is used for optical character recognition (OCR).
- **OpenAI GPT API** to process unstructured text into structured data like amount, category and store name.
- **Navigational control** allow user to redirect back to add expense screen.

h) Budget Screen

Description:

Allow users to set a monthly spending limit, and users are able to see how much they spend in relation to the budget they have set. There are also savings suggestions provided if users' spending stays below the threshold.

Component Requirements:

- **Cloud Firestore** is used to store and update budget information under the user's document. It fetches and sums expense data to calculate budget usage.
- A **conditional logic check** was used to decide whether to award a user points if their expenditure fell within the budget that had been set.

i) Transaction Listing Screen

Description:

Provide users with a list of historical transactions, which involve both income and expenses. Besides, users are allowed to edit or delete the transactions.

Component Requirements:

- **Cloud Firestore** is responsible for fetching records from both the income and expenses subcollections and handling the modifications or updates of the transactions.
- **Firebase Authentication** to filter data by user.

j) Reward System Screen

Description:

Allow user to check in the application daily in order to earn the points, and users may use the accumulated points to spin the wheel for exchange of rewards.

Component Requirements:

- **Cloud Firestore** stores user check-in activity and reward point data in the check-in and rewards subcollections.
- **Firebase Authentication** is used to ensure that each user's reward history is stored under their own account.
- The **spin wheel** allows users to spin and receive random rewards.

k) Chart / Insights Screen

Description:

Offers financial visualization tools that summarize the user's transaction data in daily, monthly, and categorically based, in order to help user make better budgeting decisions.

Component Requirements:

- **Cloud Firestore** is used to retrieve and manage historical income and expense data.
- **fl_chart** package is used to visualize financial data, such as bar chart and pie chart.
- **Firebase Authentication** is used to ensure that the visualized data is user specific.

l) AI Tips Suggestion Screen

Description:

Provides users with personalized financial advice based on their remaining budget, local weather conditions, and total expenses. Hence, the system will suggest bringing an umbrella on rainy days and recommend budget-friendly meals like home-cooked options, to prevent unnecessary expenses.

Component Requirements:

- **Cloud Firestore** is used to save and retrieve the AI-generated tips, enabling the system to track previous suggestions and prevent repeating the same meal suggestions within a specific time period.
- **OpenWeatherMap API** offers the real-time weather data based on the current location of user.
- **OpenAI API (GPT)** generates the contextually aware and intelligent savings advice.
- **Firebase Authentication** is used to ensure the tips and saved suggestions are user specific.

4.5 System Functional Overview

Table 4.5.1 which is the System Functional Overview Table, had outlined the main functionalities provided by this personal financial management application. Each function represents a significant component of the system's capabilities, which enables users to carry out tasks such as managing transactions, reward points accumulation and rewards redemption, creating budgets and analyzing financial data. Besides that, an in-depth breakdown of how each system function is implemented will be presented in Chapter 5.6.

Table 4.5.1 System Functional Overview Table

Function ID	Function Name	Feature Description
F001	User Authentication	Enable users to login or sign-up by entering their email address, username and password.
F002	Add Transaction	Enable users to access to add income or expense modules.
F003	Add Expense (manually)	Enable users to add their expenses manually.
F004	Add Expense (receipt scanning)	Enable users to add their expenses by scanning the receipts or transaction details.
F005	Add Income	Enable users to add their recurring income, or other categorized income entries.
F006	Transaction Listing	Enables users to review past financial transactions, including both income and expenses, arranged from the most recent to the oldest.
F007	Edit/Delete Transaction	Enable users to make changes toward past transaction or delete the transaction.
F008	Financial Insights	Enable users to visualize their financial data in pie chart (category based), line chart and bar chart mode.
F009	Daily Check-In	Motivate users to check in daily to record their transactions and earn some reward points.
F010	Budget Management	Enable users to set a monthly budget, and system will track expenses against budget in real-time, and awarded users based on their saving status monthly.
F011	Spin Wheel Game	Enable users to spin the wheel by using their points in exchange for reward.
F012	Reward Accumulation	Enable users to view their accumulated reward points, which are updated and managed in real time.
F013	AI Tips Suggestions	Enable users to view the suggested tips for saving, like budget-friendly meals or weather condition reminders.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.1 Tools to Use

The development of this personal financial management application with spending monitoring required tools such as software and hardware.

5.1.1 Hardware Architecture

The development of this personal financial management application with spending monitoring involves the utilization of various hardware components such as a laptop and smartphone. As shown in the Table 5.1.1, a **HP Laptop 14s-dq2515TU**, equipped with 11th Gen Intel Core i5-1135G7 processor, 16GB DDR4 RAM, Intel Iris Xe Graphics, and 512 GB NVMe SSD storage, served as the main workstation for programming and development tasks.

In addition, to ensure the apps are functional and meet the users' requirements, **Huawei Y9 2019 (Android)** in Table 5.1.2, were used as the primary platform for user experience (UX) evaluation and testing in a real-world environment.

Table 5.1.1 Specifications of laptop

Description	Specifications
Model	HP Laptop 14s-dq2515TU
Processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
Operating System	Windows 11
Graphic	Intel(R) Iris(R) Xe Graphics
Memory	16GB DDR4 RAM
Storage	512 GB NVMe SSD

Table 5.1.2 Smartphone Specification (Android)

Description	Specifications
Model	Huawei Y9 2019
Processor	HiSilicon Kirin 710, Octa-core (4x2.2 GHz Cortex-A73 & 4x1.7 GHz Cortex-A53)
Operating System	Android 8.1 (Oreo) with EMUI 8.2
Graphic	Mali-G51 MP4 GPU
Camera	Dual rear camera • Primary Camera: 13 MP sensor with an f/1.8 aperture • Secondary Camera: 2 MP depth sensor with an f/2.4 aperture
Memory	4GB RAM
Storage	64GB

5.1.2 Software Architecture

To develop a personal financial management application with spending monitoring, software components play a crucial role in working with hardware tools to create and develop the mobile application. Various software tools had been utilized in order to provide different functionalities during the development process, including Visual Studio Code, Flutter, Dart programming language, Figma, Google Cloud Vision API, Firebase, Notification Listener Service (Android).

a) Integrated Development Environments (IDEs)

- Visual Studio Code

  Visual Studio Code is the major IDE used for developing a personal finance app. It is used for authoring and managing the application code, detecting and fixing bugs, supporting many programming languages, as well provide a variety of extensions including Flutter.

b) Mobile App Development Frameworks

- Flutter

Flutter is used to develop a uniform and responsive user interface for the personal finance mobile application, which enables users to perform tasks like budgeting, income and expense tracking, and receipt scanning. Further, it supports cross-platform development, such as its ability to operate on both iOS and Android platforms using a single codebase. This approach significantly reduces development time and costs.

c) Programming Language

- Dart

The primary programming language used to create this mobile application is Dart. This is due to it allows for seamless integration with the Flutter's UI framework. Further, Dart may be compiled into native machine code, enabling the application to function smoothly and efficiently. Dart's async-await model can handle tasks like, obtaining receipt data from the Google Cloud Vision API, storing and retrieving data from Firebase Firestore, and processing real-time notifications more efficiently. Thus, making it ideal for building modern mobile applications. Since Flutter is built on Dart, it supports hot reload, which allows for real-time UI updates without restarting the app.

d) User interface (UI) design

- Figma

Figma is used to design and provide an initial look for the user interface (UI) and user experience (UX) for the personal finance application. This is because it supports the high-fidelity of UI designs and interactive prototypes. Further, Figma allows stakeholders to provide direct feedback on design files, hence we able to get a deeper understanding towards the real user's needs. Additionally, it also supports responsive interface development, which enables testing across different screen sizes and orientations.

e) Google Cloud Vision API and GPT Integration

Google Cloud Vision API is integrated into the app to support receipt scanning. It extracts text from receipts that are either scanned using the camera or imported from the gallery. The extracted text is then sent to GPT-4.0 for carried

out analysis and processes the data to identify key transaction details such as the amount spent, store name, and purchase category. Then, the system will automatically analyze the expenses belonging to which categories, later store them in the database. This function able to save the users' time in manually input the transaction details, at the same time increase efficiency, due to eliminating the manual input can reduce human errors.

f) Backend Development and Database Management

- Firebase

  The real-time and cloud-hosted database with the abilities to store and sync financial data had been provided by Firebase. Besides, it also authenticates the users for secure sign-in to the app. It also offers cloud storage for secure file management, such as receipts and financial documents. Moreover, the cloud function facilitated the execution of backend codes. Additionally, Firebase is able to integrate with Flutter and provide support for the real-time data synchronization capabilities. As a result, it ensures real-time handling of financial data, including expense tracking, transaction updates, and instantaneous database updates.

g) Data Visualization and Financial Reporting Tools

- fl_chart

  The fl_chart package will be used for financial data visualization, which enables features such as pie charts for expense tracking, budget-expenditure comparisons, and monthly expense classification. For example, a pie chart will be used to illustrate the spending distribution across different categories, which enables users to get a quick understanding towards their financial patterns. At the same time, it also allows users to assess budget adherence, as well as make informed financial decisions effectively.

5.2 Hardware Setup

Android phone setup:

To start with debugging by using Huawei Y9 phone, developer mode and the USB debugging mode must be enabled as shown in Figure 5.2.1 and Figure 5.2.2.

Figure 5.2.1 Developer mode enabled

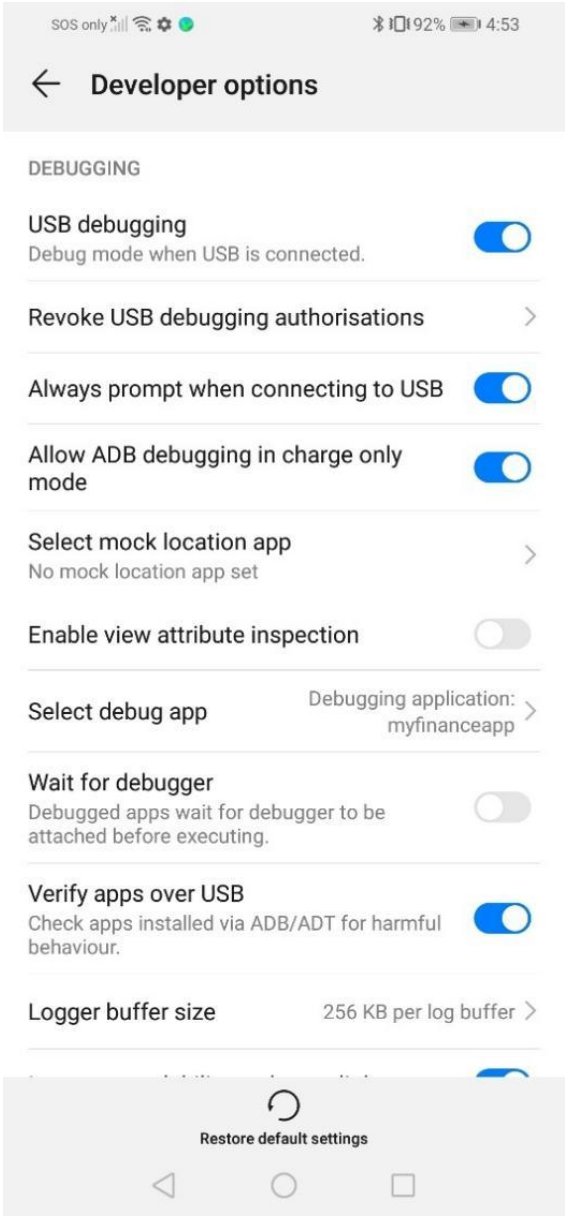


Figure 5.2.2 USB Debugging toggle turned on

5.3 Software Setup

The software setup process comprises two main parts: establishing a connection between the Android smartphone and the computer for development and testing purposes, and configuring Firebase for user data storage.

a) Connection between Android Smartphone with Computer

Next, connect the Huawei Y9 with with our computer by using USB cable. Then, the detail connection process had shown from Figure 5.2.3 to Figure 5.2.8.

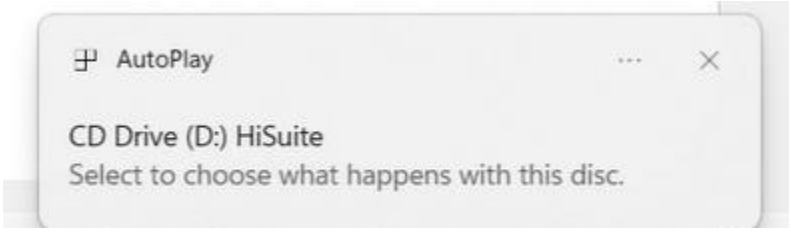


Figure 5.2.3 Notification pop up in laptop to show phone connection

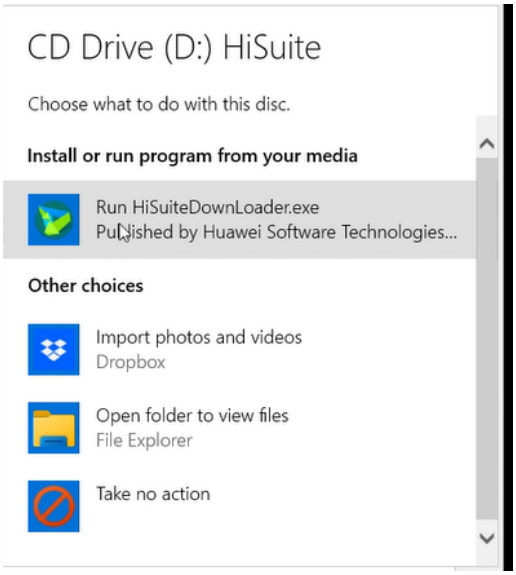


Figure 5.2.4 Choose to run the phone with HiSuite

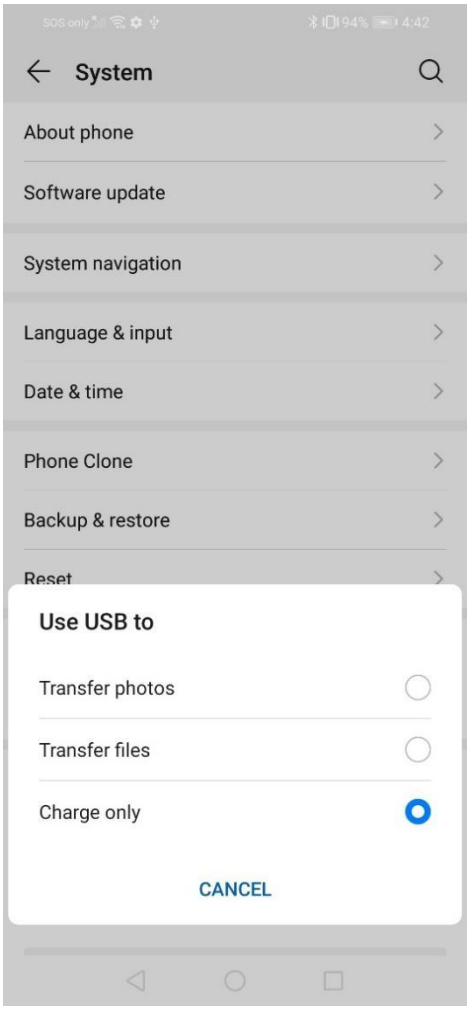


Figure 5.2.5 Select the “Transfer files” for debugging

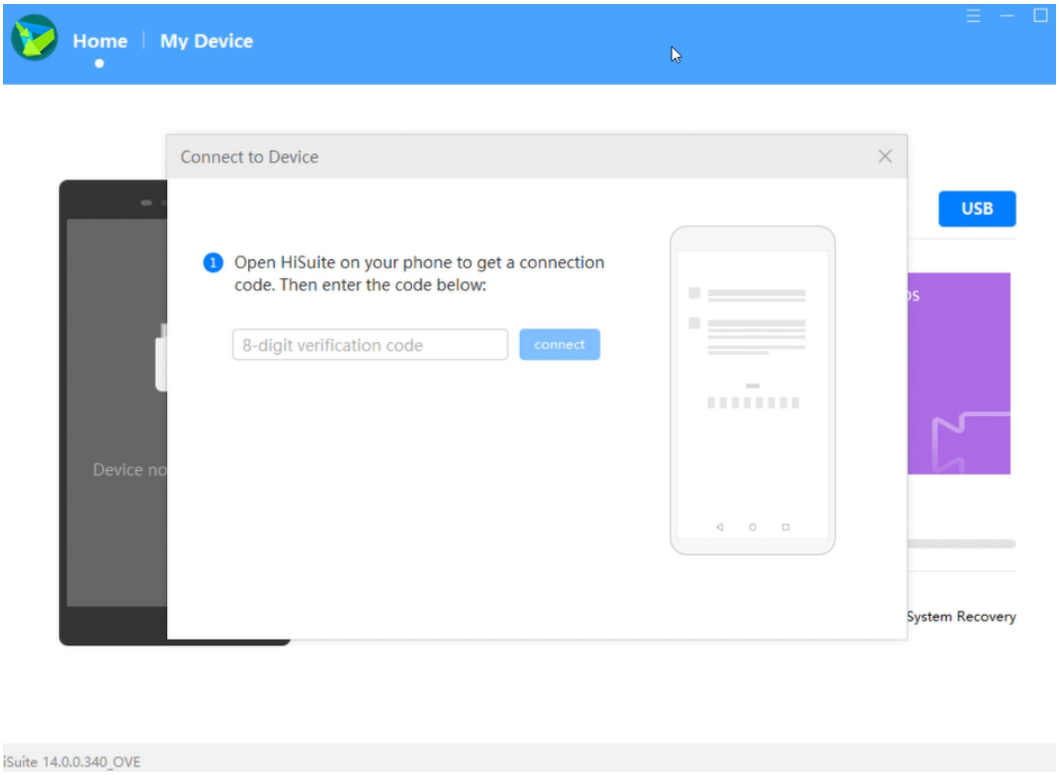


Figure 5.2.6 Connection between laptop and Huawei phone via Huawei HiSuite app

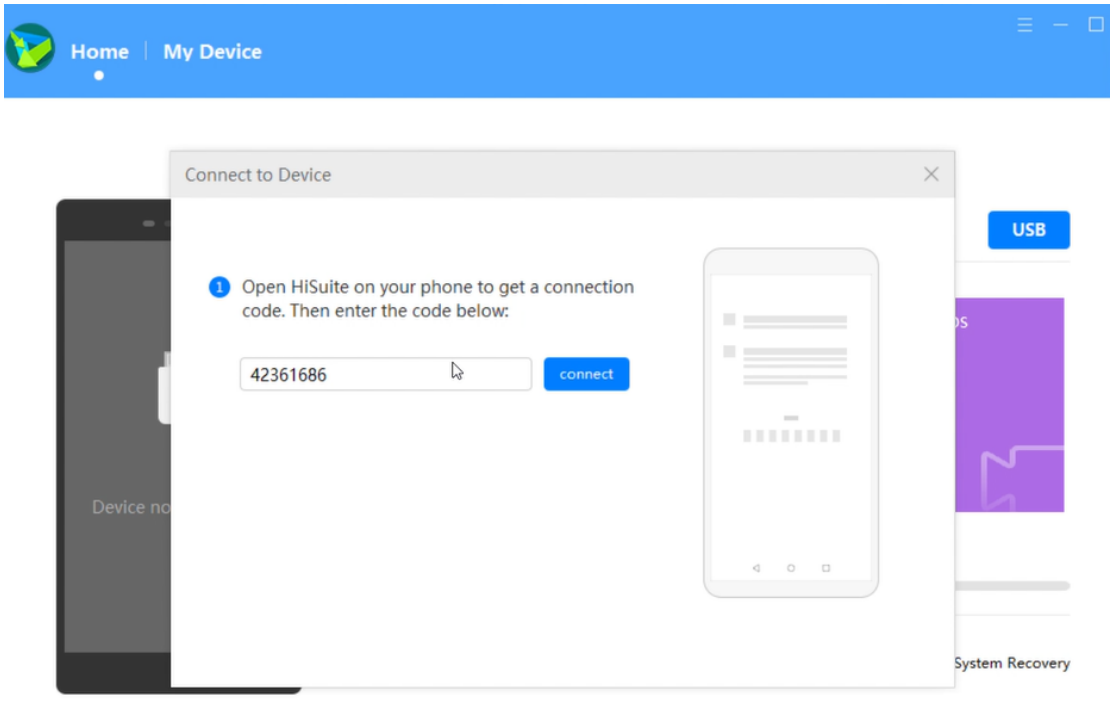


Figure 5.2.7 Enter the 8-digit verification code shown in Huawei phone

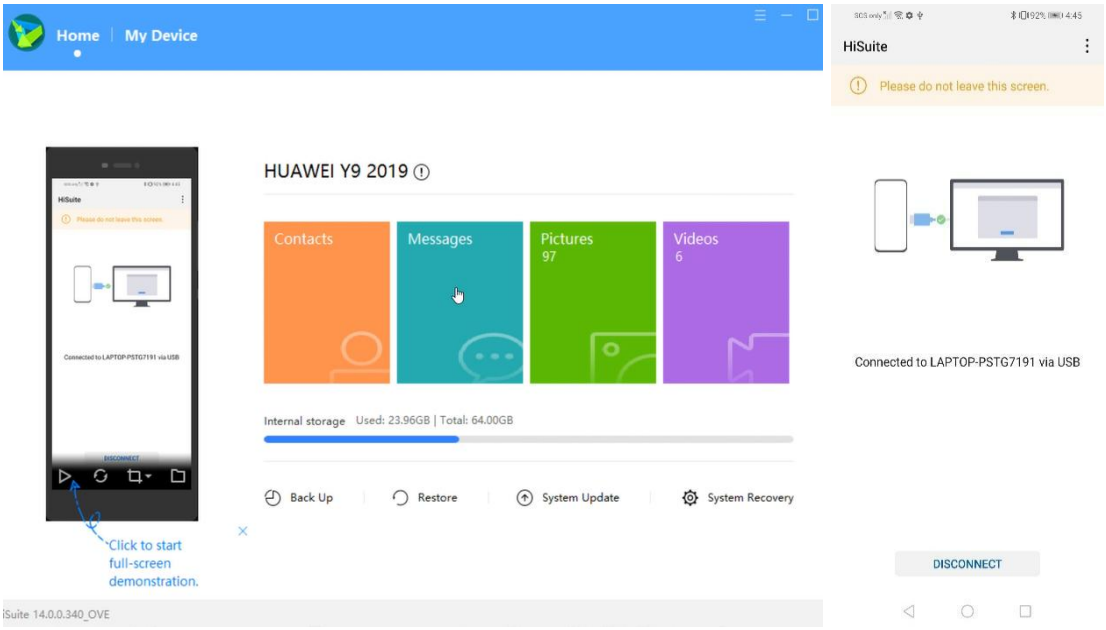


Figure 5.2.8 Successful connection between laptop and Huawei phone

b) Firebase configuration:

Firebase was then configured and connected to the Flutter project. The Firebase Authentication and Firestore had been set up, which was used to store the real-time user data and transactions. Later, the google-services.json was also downloaded and placed inside the development folder. Next, Figure 5.2.9 shows the authentication method had been set up. This method allowed the user to sign up or log in by using an email and password. As shown in Figure 5.2.10, there are few collections being created, which include users, budget, expenses, income, and receipts.

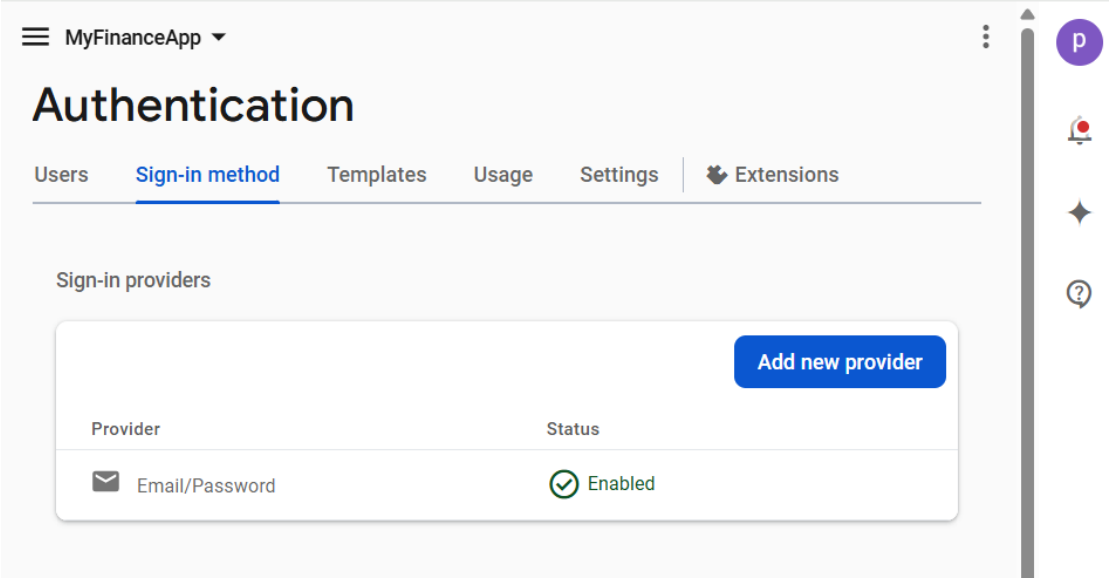


Figure 5.2.9 “Email/Password” Sign-In Method allowed by using Firebase Authentication

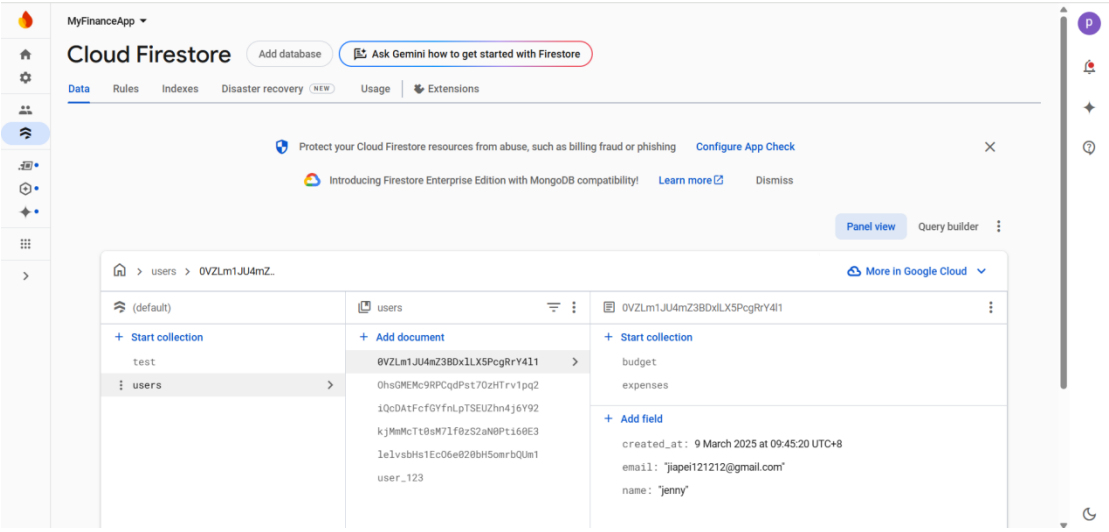


Figure 5.2.10 Firestore Database Structure

Next, Figure 5.2.11 and Figure 5.2.12 showed the Google Cloud Vision API and OpenAI GPT had been integrated for the receipt scanning capabilities. Both of them work together to extract the text from receipts and categorize each expense.

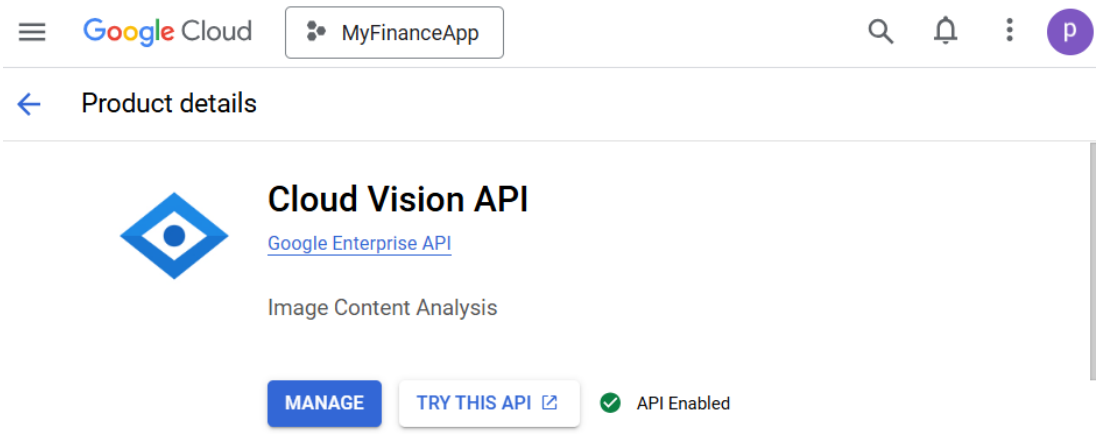


Figure 5.2.11 Cloud Vision API Enabled

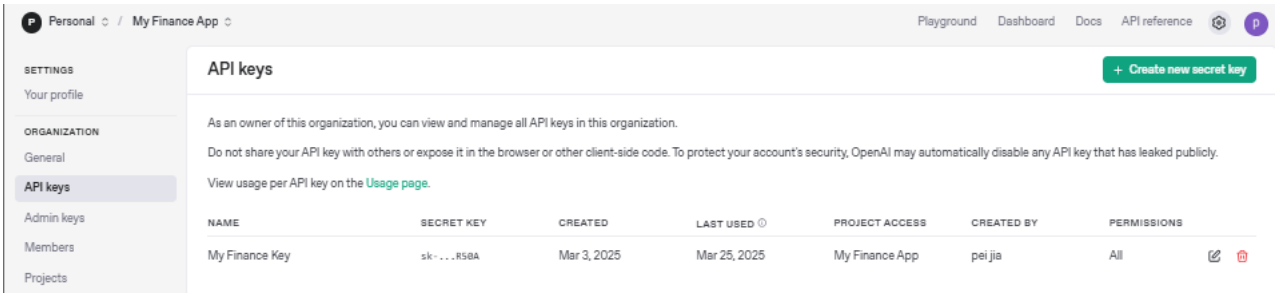


Figure 5.2.12 API Key Generated for OpenAI GPT Integration

After all the preliminary works above had been set up. The development for key functionalities process began, which includes login or sign-up process, adding expenses, incomes, setting budgets, transaction listing and logout. Then, testing was conducted to ensure the application was error free and work smoothly.

Finally, we try to integrate all the modules and proceed to the development of the overall UI/UX design. To ensure visual consistency across the entire application, we had applied dark theme to the home screen, navigation structure, and individual feature screens. This is because a dark theme with bright-colored elements able to help users focus better on the key information.

5.4 System Requirements

5.4.1 System Requirements for Users

Table 5.4.1 shows the system requirements for users.

Table 5.4.1 System Requirements for Users

Hardware Requirements	Software Requirements
✓ Smartphone (Android) with a minimum of 4GB RAM ✓ Camera with at least 8MP resolution ✓ Minimum 64GB storage ✓ Internet connection required for Google Cloud Vision API and Firebase synchronization.	Operating System: ✓ Android 8.0 (Oreo) or later

5.4.2 Functional

Sign up:

1. Users shall be able to create a new account by entering their username, valid email and password.

Log in:

1. Users shall be able to login to the application with valid email and password.

Home Screen:

1. User shall be able to view the welcome message, total accumulated points, total income, total expenses, and latest four transaction activities.
2. User shall be able to enter the Add Transaction screen by clicking the "+" button.
3. User shall be able to enter the Transaction Listing screen by clicking the "See All" button.
4. User shall be able to enter the Reward screen by clicking the point box.

5. User shall be able to perform a daily check-in, and the system shall assign points based on the current streak; updated points shall be displayed on the Home Screen.
6. User shall be able to check in daily to earn the daily check-in points.
7. User shall be able to enter the Transaction Insight screen by clicking the "Total Expenses" box.
8. User shall be able to log out by tapping the logout icon in the app bar, which redirects them to the Login Screen.

Add Transaction Screen:

1. User shall be able to record the income or expenses.
2. User shall be able to select from the categories of expenses or incomes provided.
3. User shall be able to enter the Receipt Scanning screen.
4. User shall be able to select the date of expenses or incomes to be recorded.
5. User shall be able to enter the store name and amount of expenses for expenses recording.
6. User shall be able to enter the income amount for income recording.

OCR Receipt Scanning and GPT Integration:

1. User shall be able to take a photo of their receipts.
2. User shall be able to upload the receipts from the gallery.
3. User shall be able to confirm the extraction details are correct.
4. User shall be able to upload multiple receipt at one time.
5. User shall be able to upload a single piece of receipt which contain multiple transactions.
6. User shall be able to redirect back to Add Transaction screen, with key details of transaction being automatically filled in by the system, and user shall perform latest checking.

Budget Creation and Tracking:

1. User shall be able to enter and save a monthly budget amount once a month.
2. User shall be able to view the suggested savings amount.

3. User shall be able to view the current budget and the total spending for the month.
4. User shall be able to see a horizontal progress bar that visually indicates the proportion of the budget that has been spent.
5. User shall be notified with a color change in the progress bar if spending exceeds 80% of the budget.
6. User shall be provided with a recommended savings amount if the remaining budget allows for it, calculated as 20% of the unspent amount.
7. User shall receive a reward pop-up when their total monthly spending is less than the set budget.

Transaction Insight Screen:

1. User shall be able to view visual insights into their expenses through charts, including daily and monthly trends.
2. User shall be able to toggle between day and month views using the segmented toggle buttons.
3. User shall be able to view a horizontal bar chart which shows the expenses distribution based on category for the selected month.
4. User shall be able to move to the previous month to view their historical expenses.
5. User shall be able to view the line chart which shows daily expenses.
6. User shall be able to click the line in line chart, to view the particular day total expenses.
7. User shall be able to see the total expenses and incomes for each month in a year.
8. User shall be able to see the top 3 categories of high expenses categories in the year.

Transaction Listing Screen:

1. User shall be able to delete the transaction.
2. User shall be able to edit the transaction.
3. User shall be able to view the transaction date and time.

4. User shall be able to view the category of transactions, spending or income amount.
5. User shall be able to see the amount of money, if display in green color indicating income, whereas display in red color indicate expenditure.

Point-based Reward Screen:

1. User shall be able to spin the wheel.
2. User shall be able to receive the rewards.
3. User shall be able to see their accumulated points decrease or increase in real-time.

AI Tips Suggestions Screen:

1. User shall be able to view the suggested saving tips.
2. User shall be able to save the saving tips.

5.4.3 Non-functional

Non-functional requirements define the quality attributes of the system. These benchmarks define the expected system performance in terms of speed, accuracy, responsiveness, and efficiency when handling user actions and background operations.

- ✓ OCR Accuracy: The Google Cloud Vision API was able to support the receipt scanning feature, and extract text with accuracy up to 95%.
- ✓ Transaction Processing Time: The details extracted from receipts will be processed and categorized within 30 seconds.
- ✓ App Responsiveness: All UI interactions like screen transitions will complete within 500 milliseconds to ensure a smooth user experience.
- ✓ Data Sync Efficiency: In order to maintain data consistency, Firebase synchronization will sync expense data across devices within 5 seconds.

5.5 System Operation

5.5.1 Login & Signup

Figure 5.5.1 shows the login and signup screen. Users will be redirected to home screen if they enter the email and password correctly; else the red border will appear around the password field to indicate invalid password entered. Besides that, the name enter by user in the Name field will be used for representing the username in the home screen. Further, we had set the minimum length of password to 10, mix of uppercase and lowercase are required, to ensure passwords are complex and prevent unauthorized access.

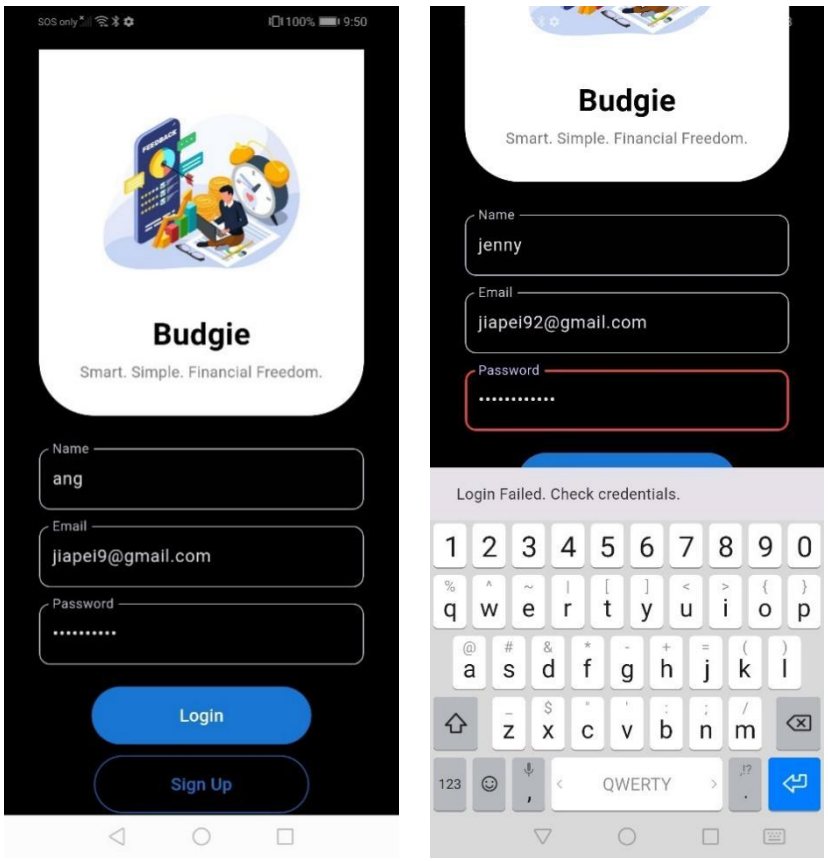


Figure 5.5.1 Login and Signup screen for User Authentication

5.5.2 Home Screen (After Login) and Logout

Figure 5.5.2 shows the home screen for this mobile application. For the new user (Picture at right side), the home screen will display RM 0.00 for both Total Expenses and Income. However, for existing users (Picture at left side), the Total Expenses and Income will show based on what they had recorded into the system, and will update in real-time, which not more than 3 seconds. Besides that, the "Hello, [username]", the username will change based on the name they entered in the login and signup screen. Furthermore, for the Activities side, there will show the latest 4 transactions to users, and if users want to have a look on full transaction history, they may just click on the "See All" button. Moreover, there are 3 buttons provided in the bottom panel, which allowed users to switch among the screen, such as the left side is for navigate back to home screen; the center "+" is for adding expenses; and the right side is for budgeting. In the future, a point-based reward system will also be implemented. This feature is used to engage users to record their daily expenses, which maintain a good financial habits. And lastly the total accumulated points will be displayed on the top of home screen, which users able to know how much points they had earned and keep motivated.

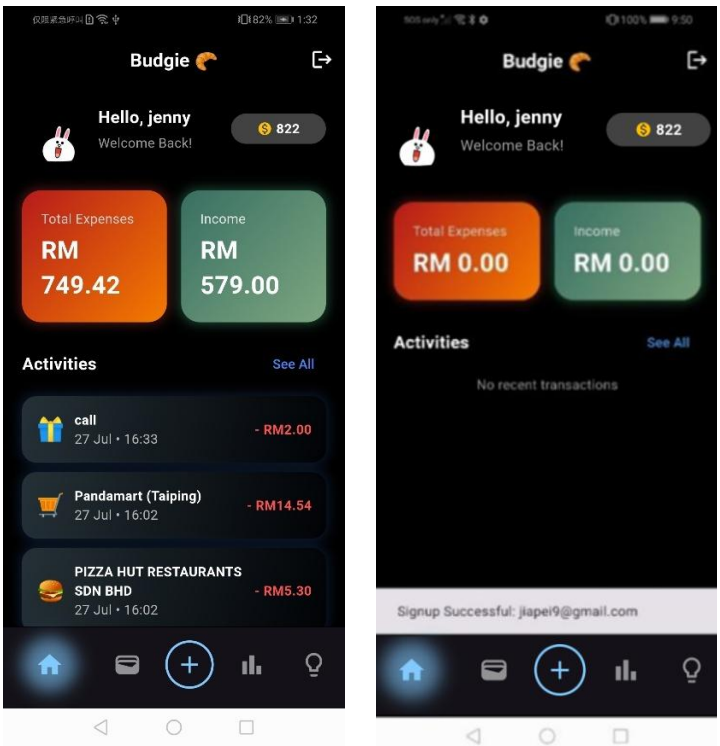


Figure 5.5.2 Home Screen

5.5.3 Add Transaction

Figure 5.5.3 shows that users will be redirected to Add Transaction screen when they press the "+" button from the home screen. After entering the Add Transaction screen, users can easily add either an expense or an income by simply switching between the two options using the toggle button.

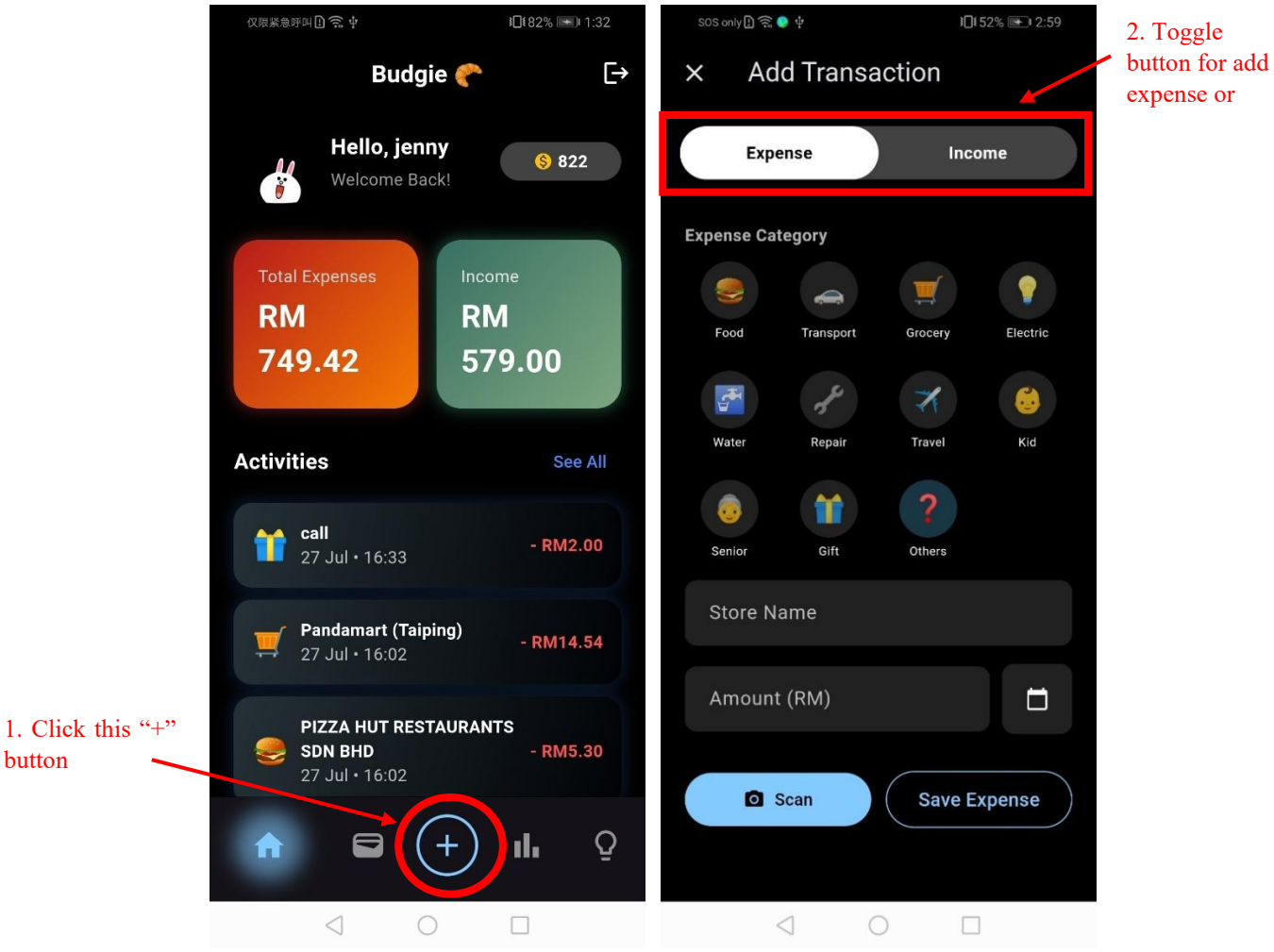


Figure 5.5.3 Add Transaction screen

Figure 5.5.4 shows that users are allowed to choose a specific date when adding an income or expenses. However, if no date is selected by user, the current date will be used by the system as default.

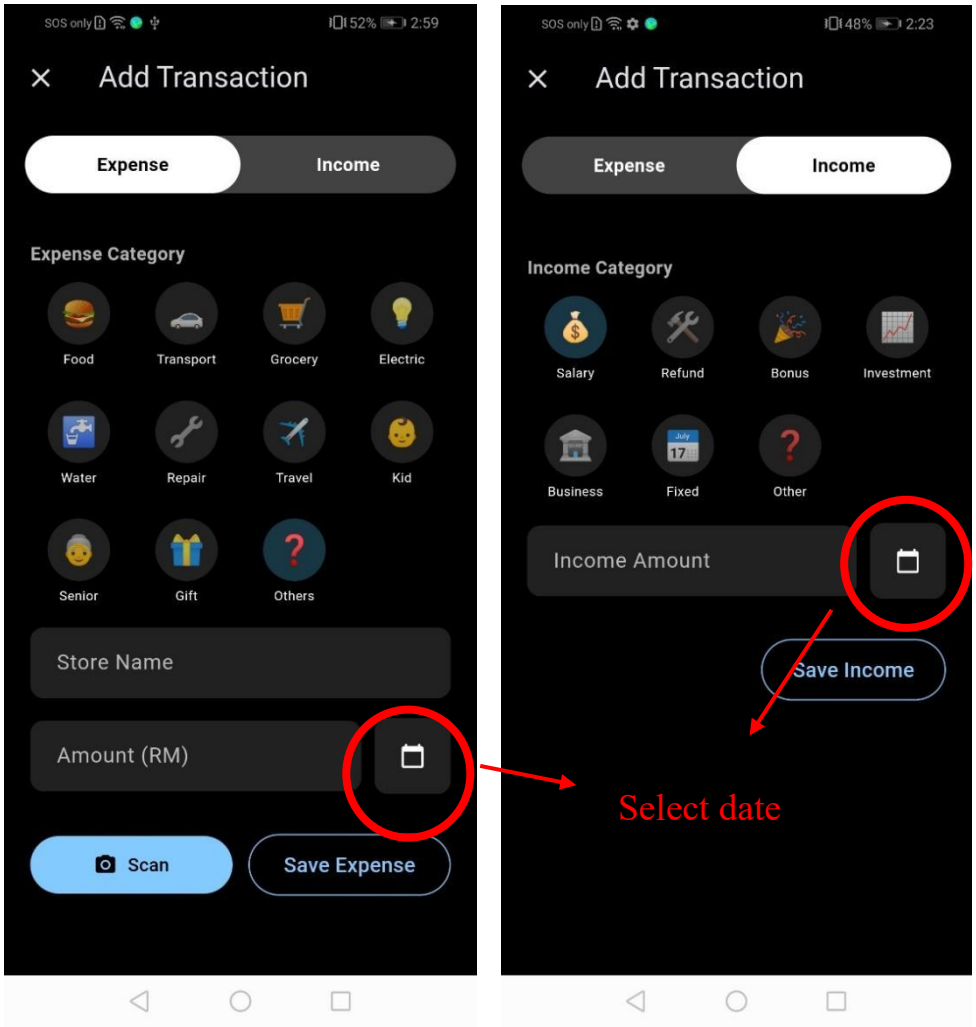


Figure 5.5.4 Allow users to select specific date for expenses or income

Figure 5.5.5 shows the date picker, which will always set the current date as default and users are allowed to select a specific date. If no date is selected, the system will use the default current date as the timestamp for the record.

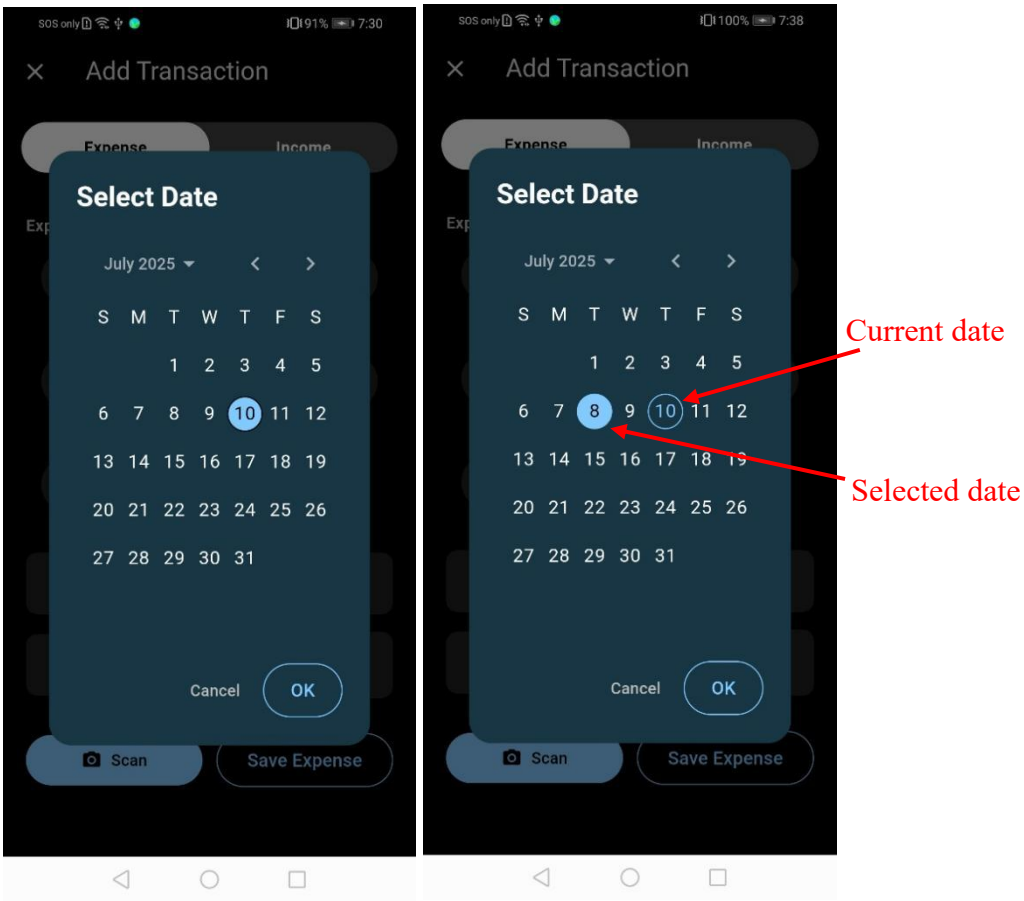


Figure 5.5.5 Date Picker

5.5.4 Add Expense (Manual)

Figure 5.5.6 shows the add expenses manually by user. First, there are various categories of expenses provided for the user to select. And we had set the default category to "Others", and the background color for selected category will change to dark blue. Next, user may enter the store name, if the store name field are not provided by user, then "Unknown" will be used as the store name. For the amount of spending, the user is required to enter the amount, else error will be prompt, and the transaction recording process will fail.

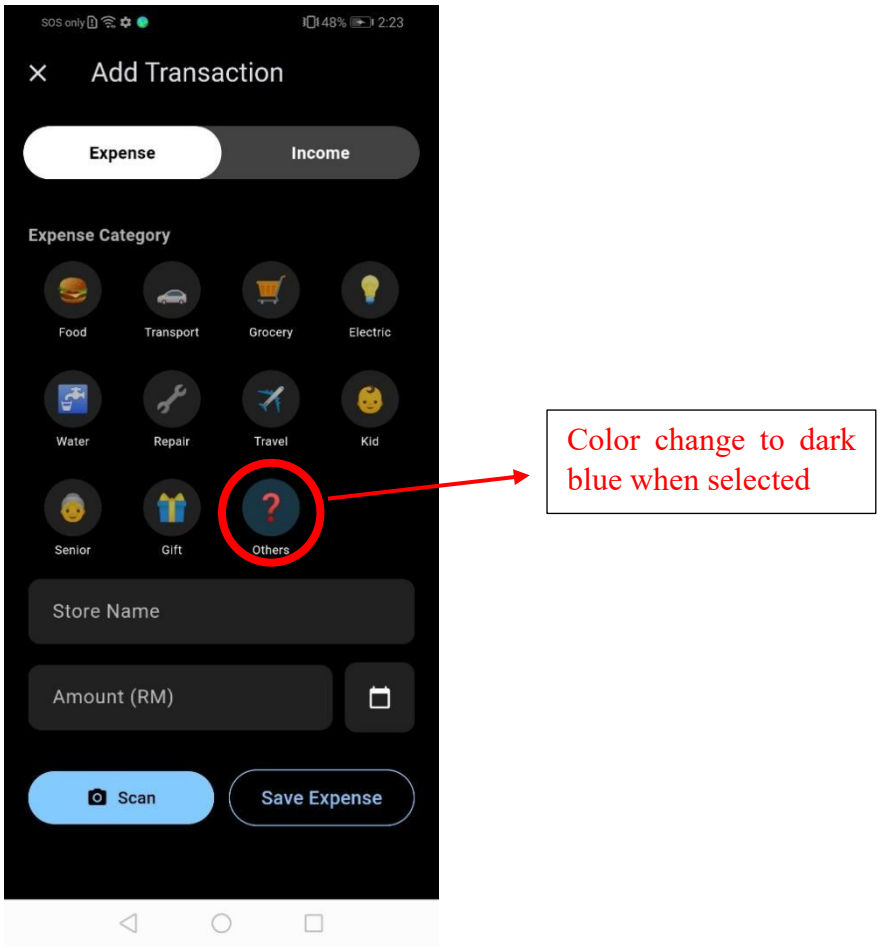


Figure 5.5.6 Add Expense screen

Figure 5.5.7 shows an example of manually entered for the expenses. First, the category of expenses being select is Food, store name is kfc and amount of spending is RM20. The expense will be successfully updated once the "Save Expense" button had clicked. Besides that, the amount is automatically converted to two decimal places to ensure consistency.

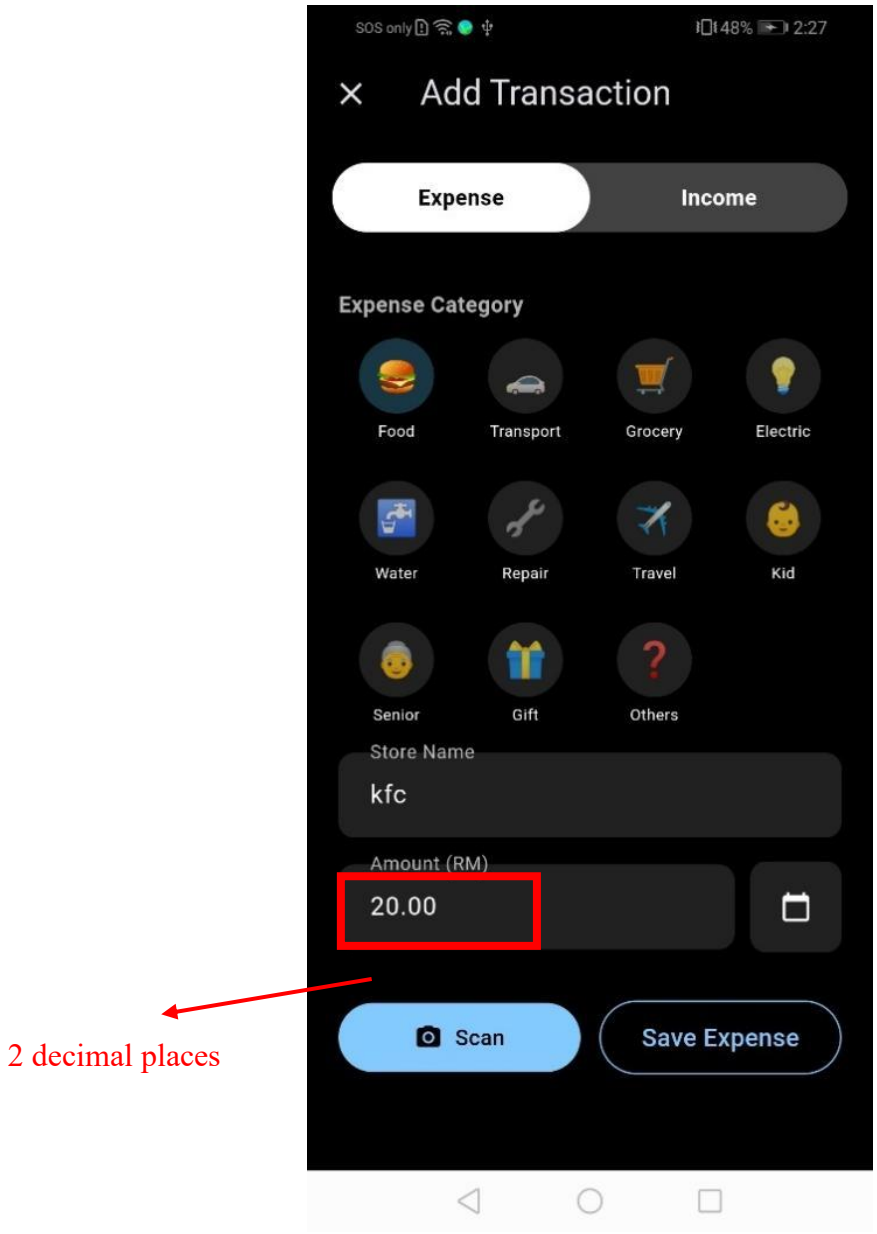


Figure 5.5.7 Add Expense screen with value filled in

5.5.5 Scan Receipt and Transaction (OCR + GPT)

Once the user clicks on the "Scan Receipt" button on the Add Expense screen, they will be redirected to the Scan Receipt screen. There are 2 options provided to user, such as take the photo of receipt directly using camera scanning or upload the receipts or transactions from their gallery as illustrated in Figure 5.5.8.

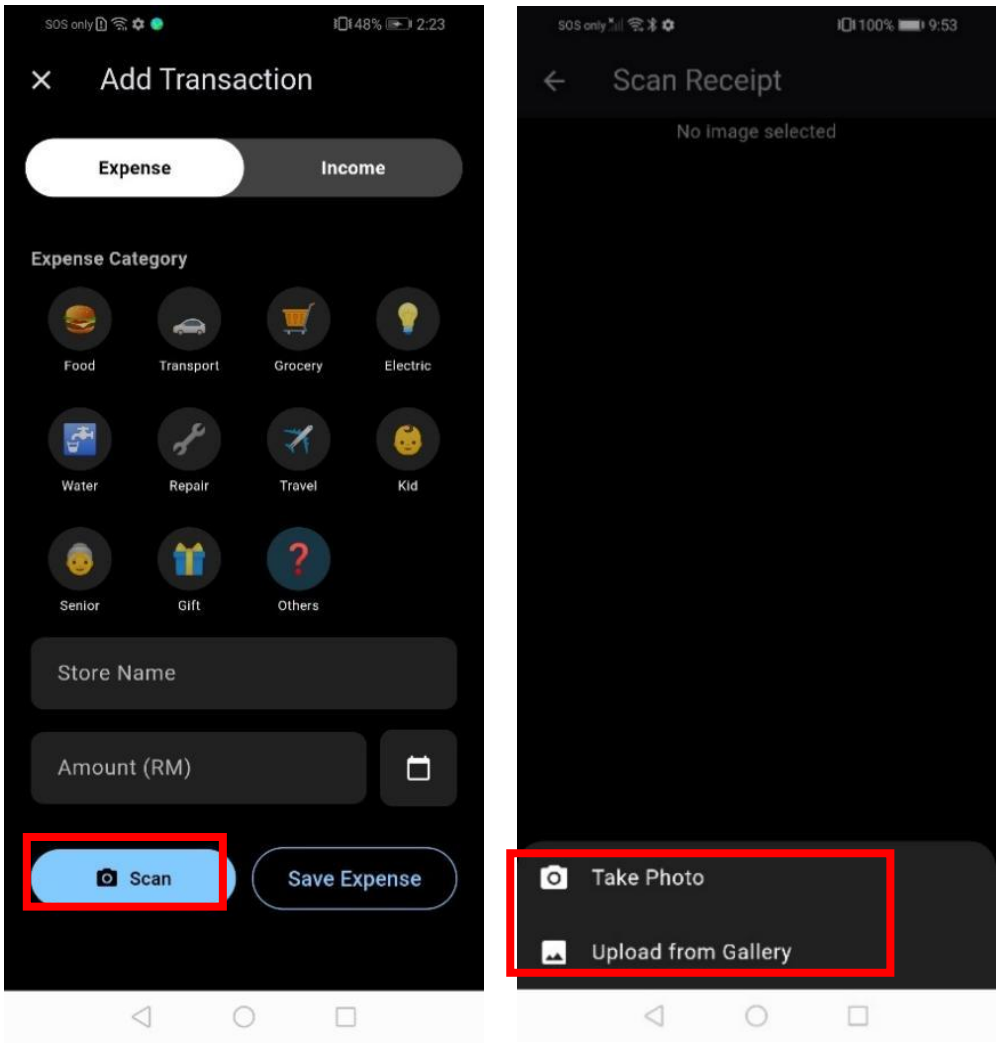


Figure 5.5.8 Receipt or Transaction Scanning screen

Figure 5.5.9 shows that the scanning function is able to extract transaction details from a screenshot of the user's notification center. When the user uploads a screenshot, the system uses OCR (Optical Character Recognition) to read the text and then GPT being used to analyze and extract expenses details, including the store name, amount, and category. In this example, the system successfully identified the category as "Food", the store as "PIZZA HUT RESTAURANT", and the amount as RM 5.30.

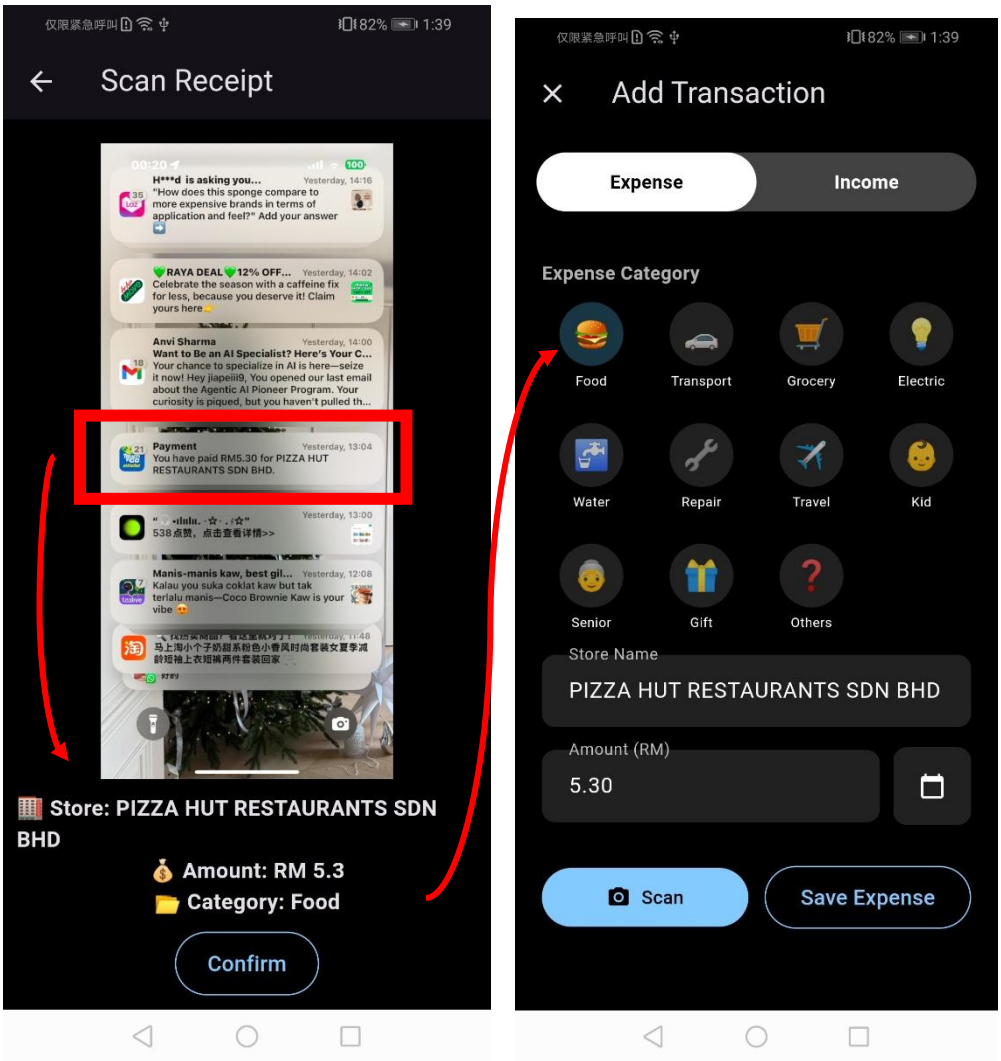


Figure 5.5.9 Scanning Transaction from notification center

Figure 5.5.10 shows the scanning function when deal with a real piece of receipt. In this example, we had used a Starbucks receipt, the system able to scan and extract accurately for the category as “Food”, the store as “BERJAYA STARBUCKS COFFEE COMPANY SDN BHD”, and the amount as RM 23.25. Once the user clicks the “Confirm” button, the key details will fill in to the specific field automatically. If there is an error in scanning, users may correct it manually. This is to ensure each expense being recorded accurately and flexibility being provided for user to modify.

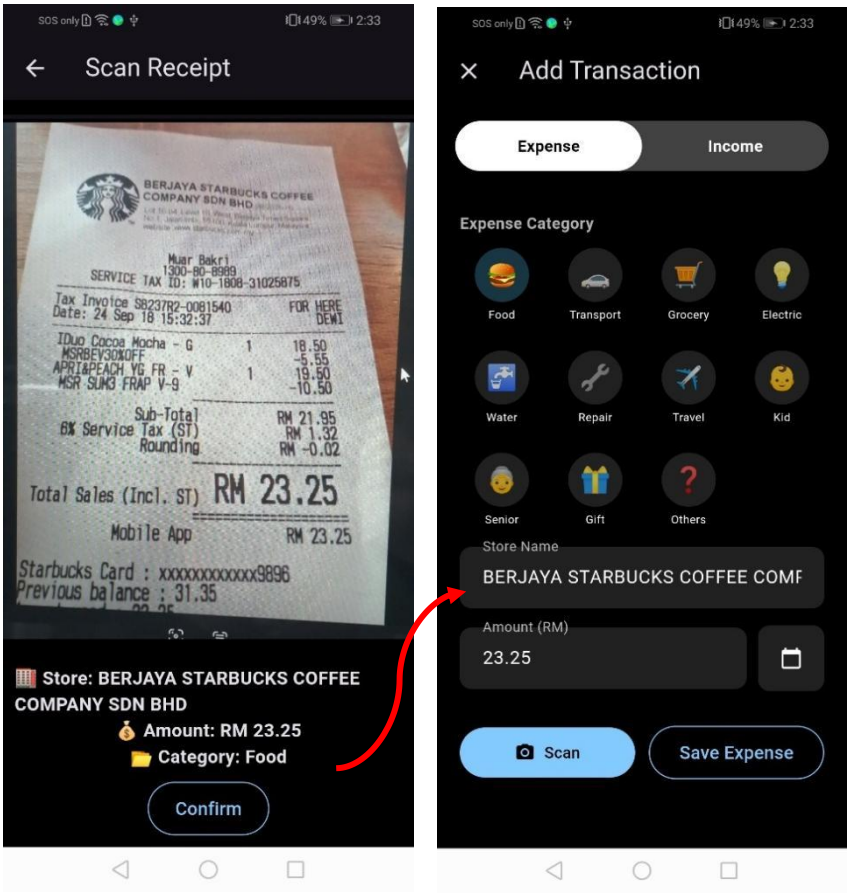


Figure 5.5.10 Scanning for a physical receipt

Figure 5.5.11 shows that scanning of the E-receipt from Grab. The system able to scan and extract accurately for the category as “Transport”, the store as “GRAB”, and the amount as RM 9.00.

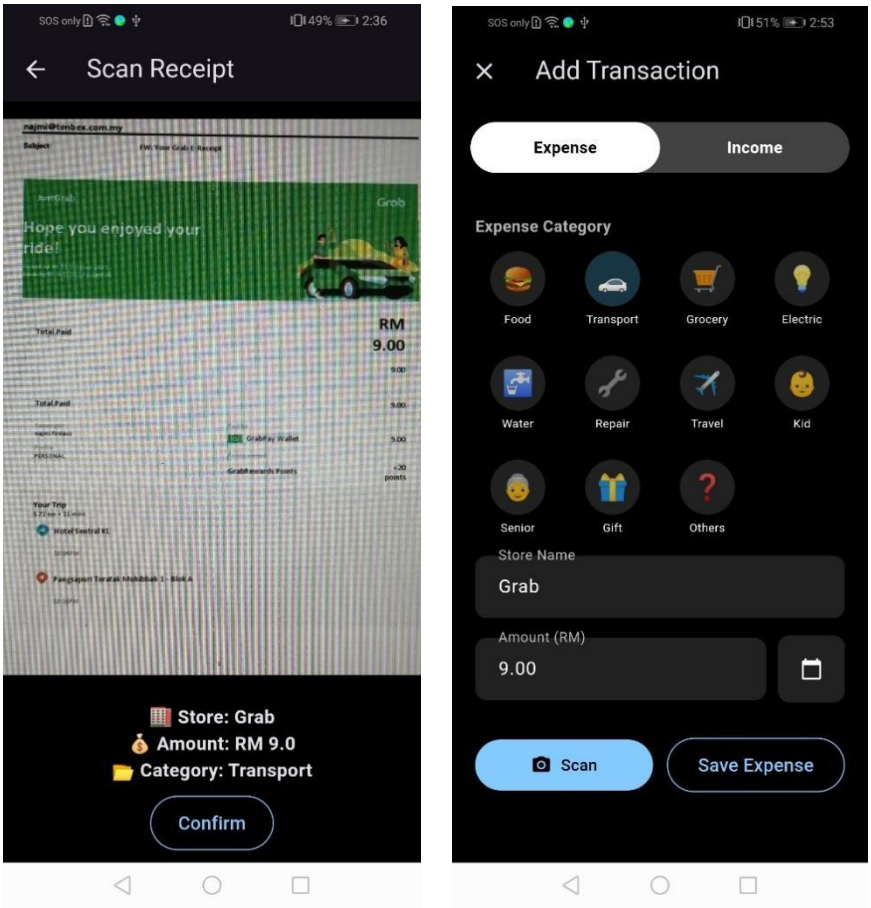


Figure 5.5.11 Scanning of E-receipt

Figure 5.5.12 shows that users are allowed to upload multiple receipts at one time for the system to process and record automatically.

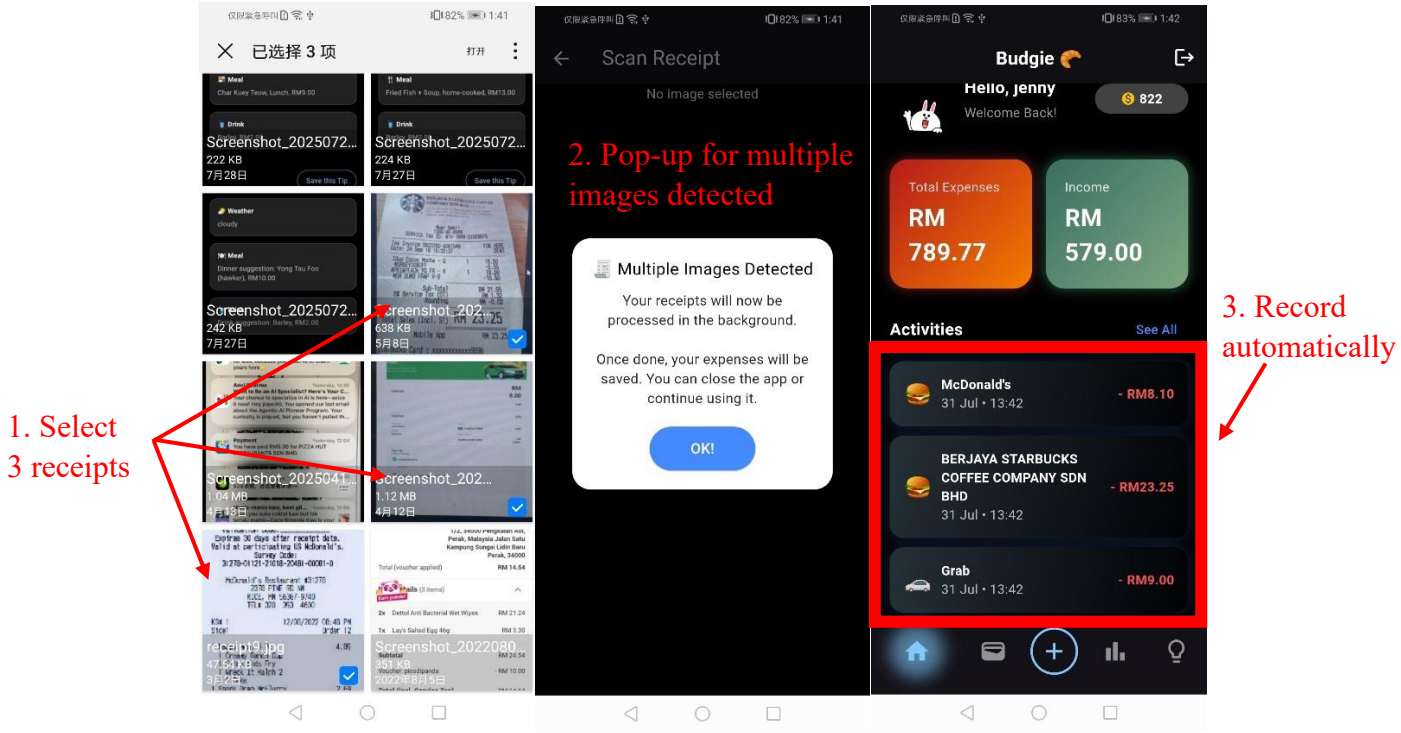


Figure 5.5.12 Scanning multiple receipts

Figure 5.5.13 shows that users are allowed to upload a single receipt with multiple transactions inside, and the system able to process it accordingly.

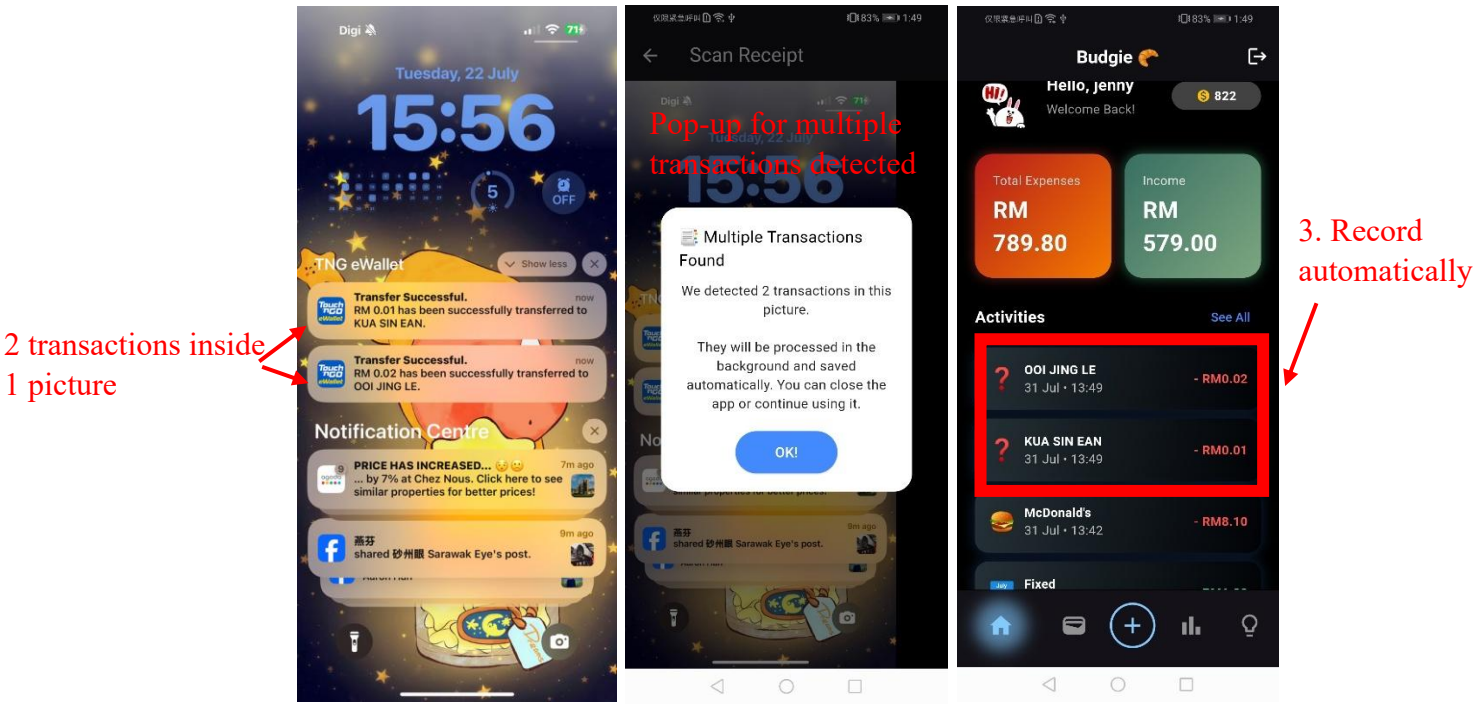


Figure 5.5.13 Scanning single piece of receipt with multiple transactions inside

5.5.6 Add Income

Figure 5.5.14 shows there are multiple categories of income being provided for users to choose. First, the user needs to enter the amount of income, then select the respective category. Lastly, once the user clicks “Save Income” button, the income will be successfully saved and updated to the database.

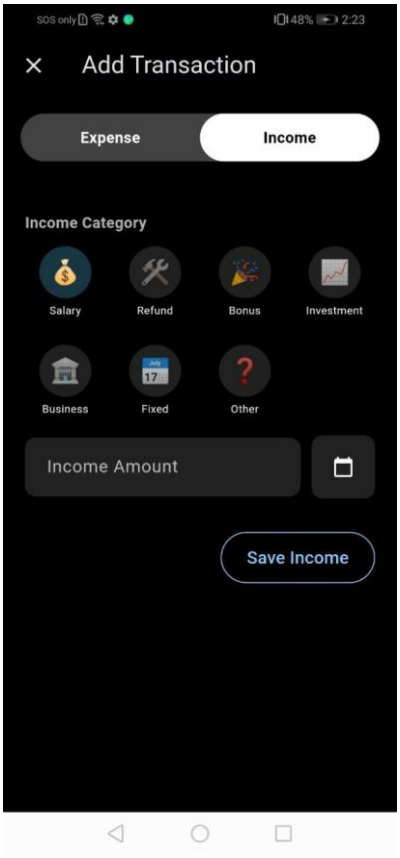


Figure 5.5.14 Add Income screen

Figure 5.5.15 shows the income recurring function, which was designed to automatically record the fixed income amount into the system at the beginning of every new month. This is to help users keep their income records consistent, at the same time, eliminating the user to enter it manually every month. Hence, the user can select the “Fixed” category for their income. Once set, this category will automatically add the fixed income amount at the start of each month when the user opens the app. For example, when the user opens the app on 9 September 2025, the system will instantly record all fixed income amounts for September and display them when the user

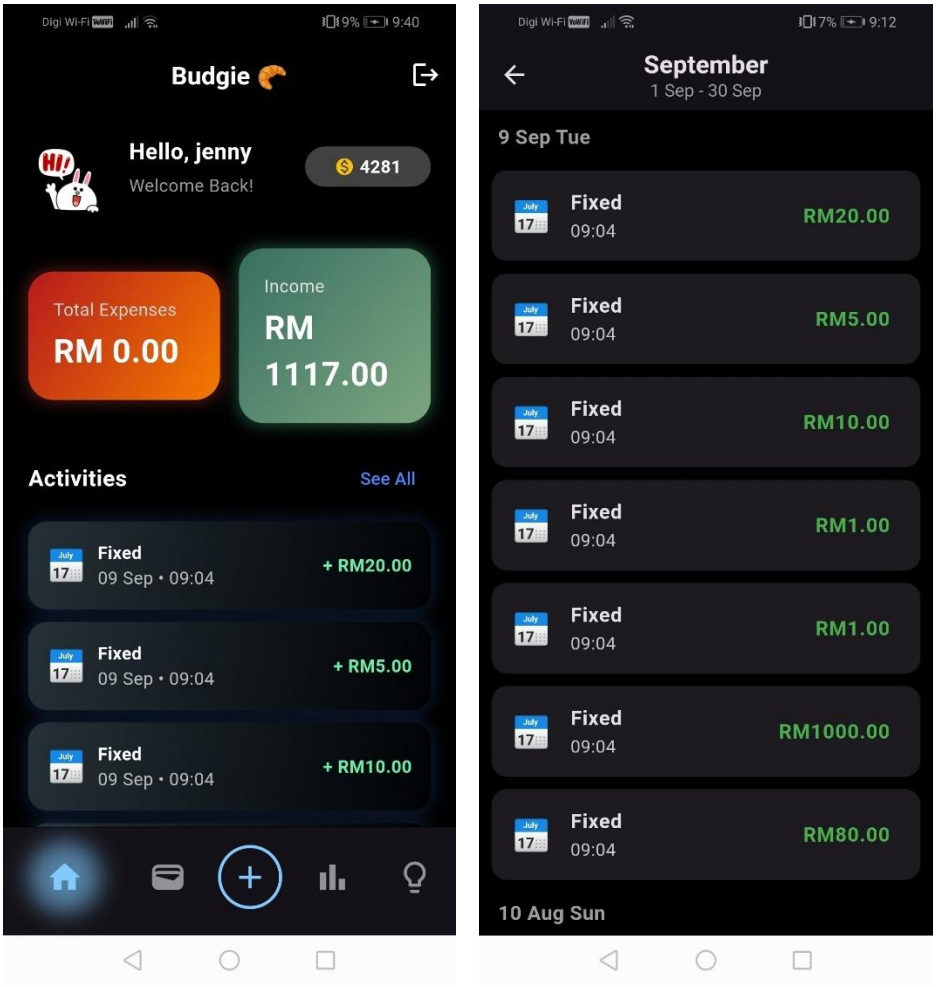


Figure 5.5.15 Auto-Add Income (Start of Month) screen

5.5.7 View Budget & Set Budget

Figure 5.5.16 shows the budgeting screen. User are allowed to enter their budget amount and press the "Save Budget" button to update their budget. However, budgets can only be set once per month and cannot be set multiple times within that month. There are 2 situations that will happen. First is when the expenses exceed the budget set by users, the line chart indicates budget usage will turn to red color, and show alert messages "Careful, you're near your budget". Second is when the user's expenses are within the budget they had set, then a blue line will show to indicate normal budget usage and show messages "Keep going!". Besides, there are also suggested savings provided to the user, which are calculated from the 20% of the remaining budget amount. However, if the user's expenses exceed the budget, hence the suggested savings will be updated to RM

0.00. Besides, a point-based reward system has also been implemented to motivate users to spend wisely. Then, on the first day of each month, the system will analyze the previous month's total expenses and compare them with the budget set by the user. If the total expenses are within the budget, 100 points will be awarded to the user, and later a pop-up message will be displayed to user when they first login of the new month. Lastly, the point will be update to the user's account balance.

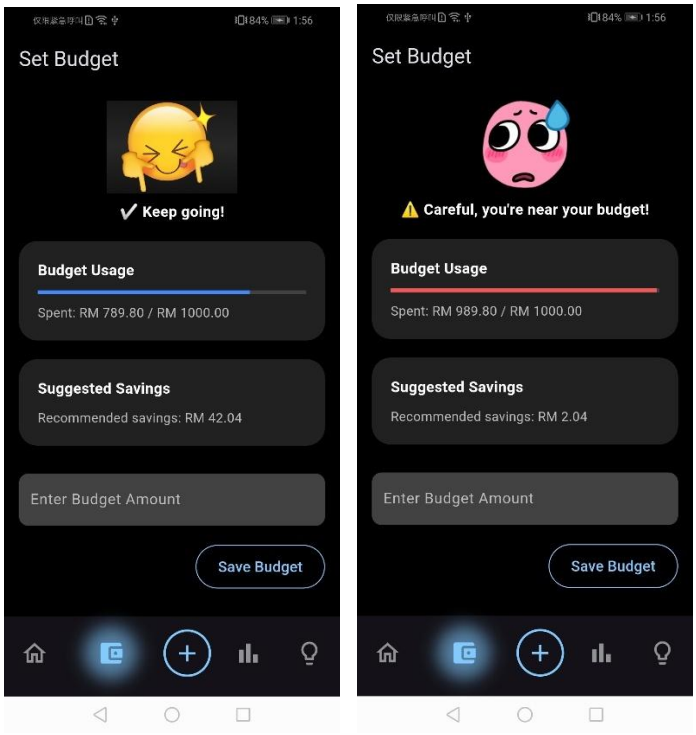


Figure 5.5.16 Budgeting screen

Figure 5.5.17 shows the budget reward pop-up that appears when the user successfully stays within their monthly budget.

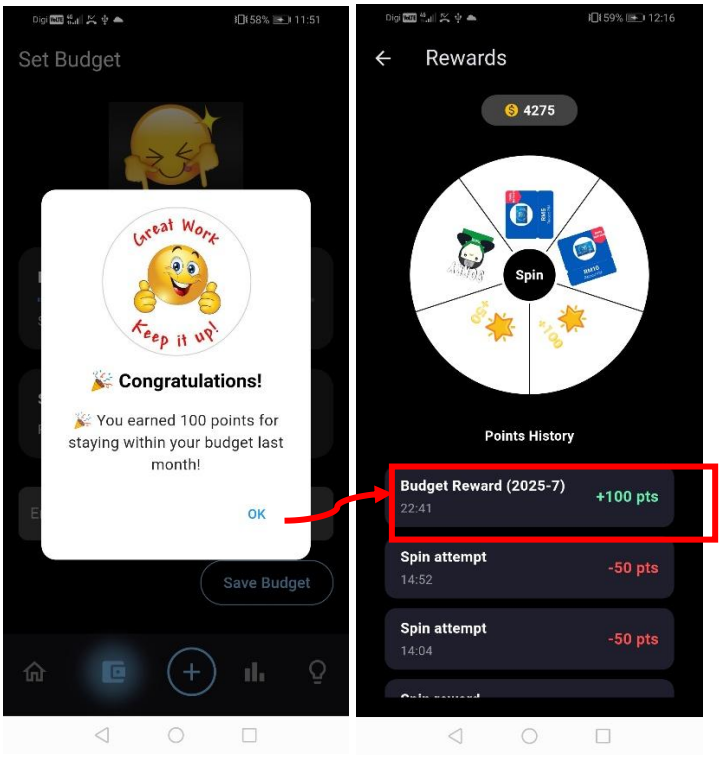


Figure 5.5.17 Budget Reward Pop-Up "Congratulations"

Figure 5.5.18 shows a motivation message aimed at encouraging the user to spend wisely, due to the user's budget goals not being met.

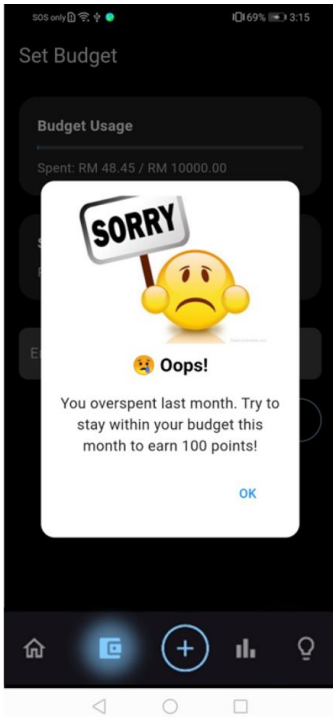


Figure 5.5.18 Budget Reward Pop-Up "Sorry"

5.5.8 View Charts

Users need to click the “Chart” icon from bottom navigation bar to enter the chart screen, as shown in Figure 5.5.19. Figure 5.5.20 shows the chart screen in the Insights section, where users can select either daily or monthly views of their financial data. For the daily view, a bar chart had been utilized. By default, it shows a small tooltip above the bar, which represents the total expenses for that specific day. In the example (left side picture), such as today is 27 July 2025, and the total expense for this day is RM303.63 had shown. Besides, user also can click on any bar, and the respective total expenses for that day will be shown. Further, user able to press the “<” and “>” to switch among different months. For the Monthly Expenses Distribution part (left side picture), a breakdown of monthly expenses by category is provided including the total expenses for each category monthly follow with the percentage, which helps user to visualize which categories they spent the most. For the monthly view, user able to make comparison between monthly income and expenses for each month. The total expenses for each month were shown in red color, and the total income was shown in green color.

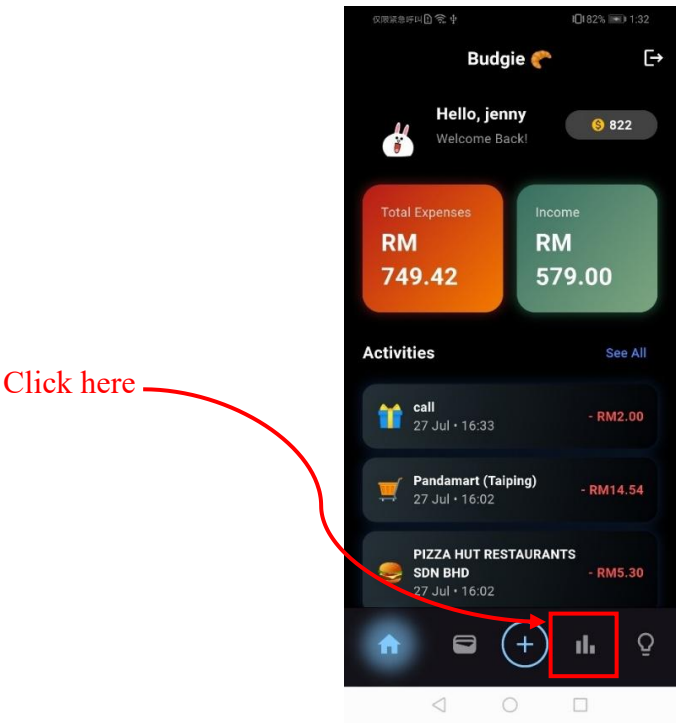


Figure 5.5.19 Enter the chart screen from home screen

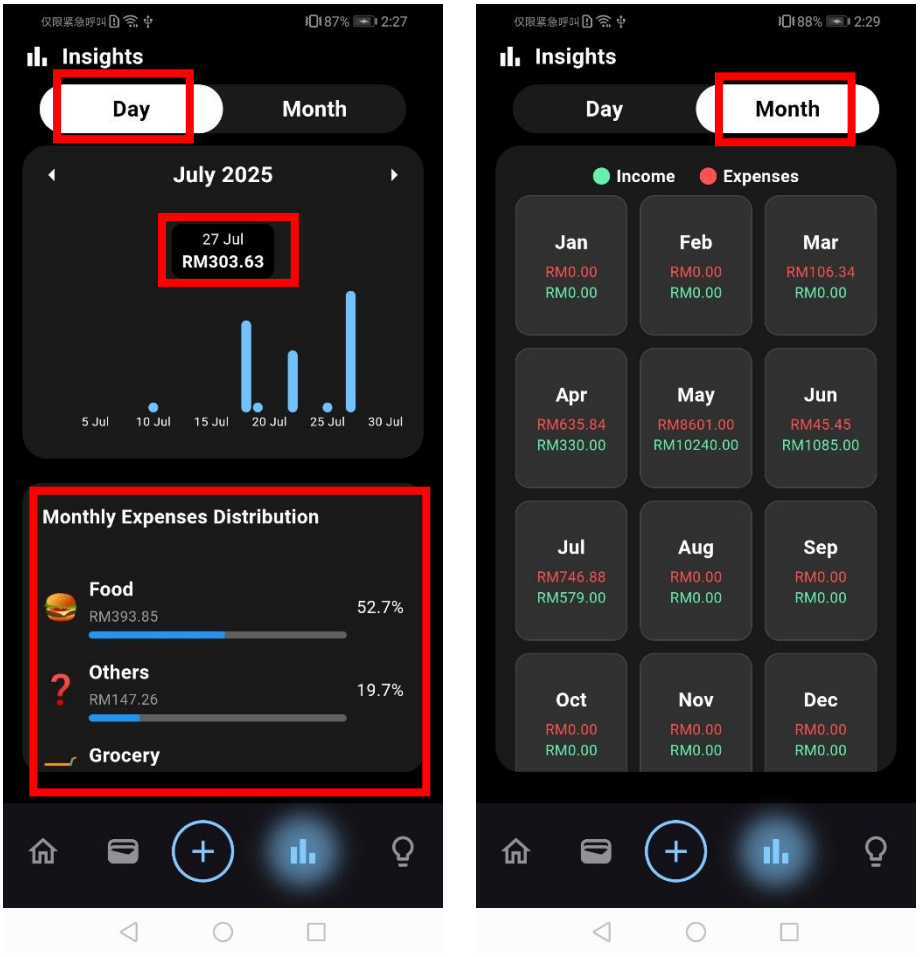


Figure 5.5.20 Transaction Chart screen

5.5.9 View Transaction History

Users need to click the "See All" button from home screen to enter the transaction listing screen, as shown in Figure 5.5.21.

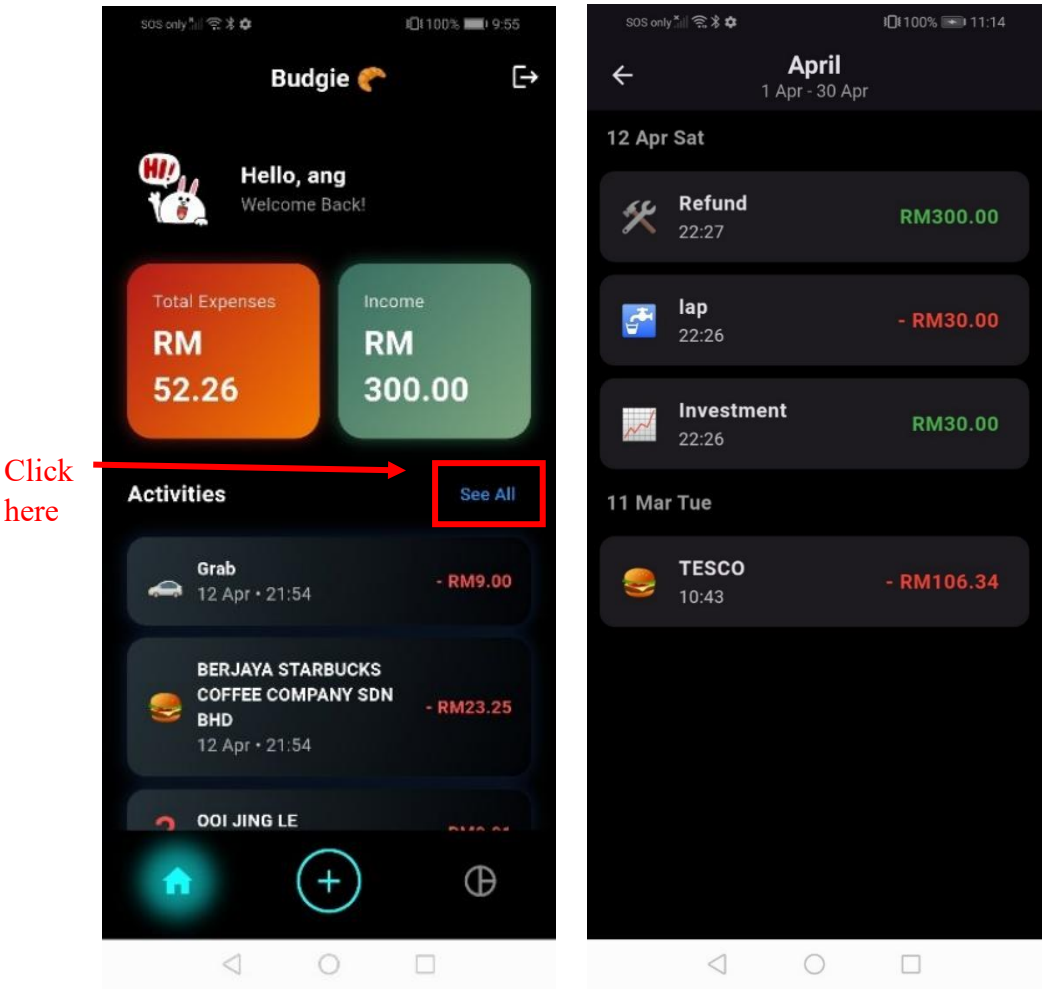


Figure 5.5.21 Transaction Listing screen

Figure 5.5.22 shows the make changes towards a specific transaction. For modification of an existing transaction, user swipes right on an expense row, which will redirect user to the Edit Expense screen. After making modifications, the user presses the “Save Expense” button, then the changes will update to database which shown in Figure 5.5.23.

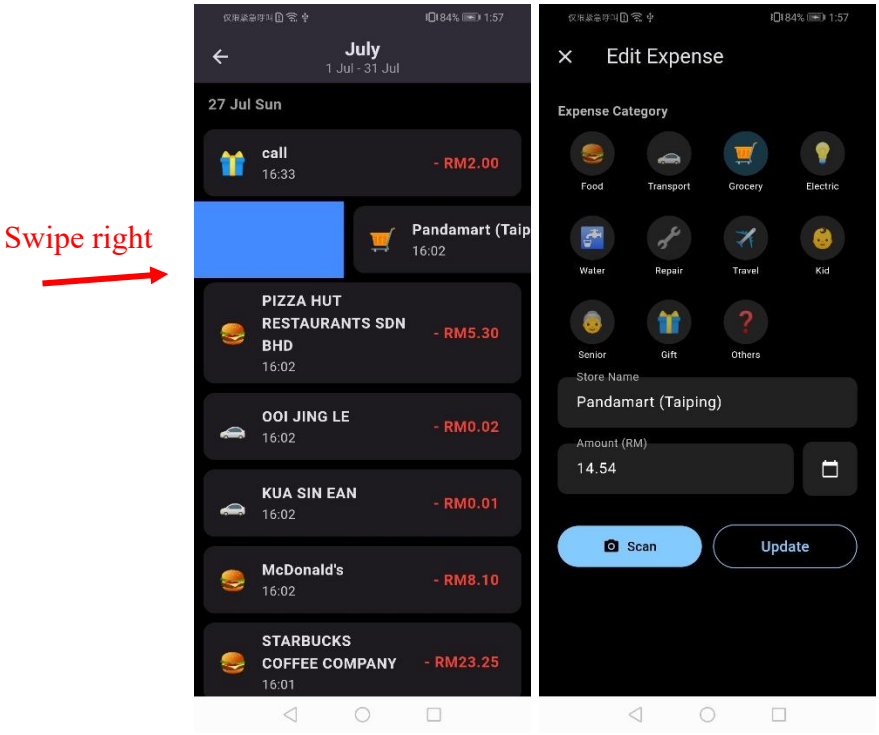


Figure 5.5.22 Swipe Right to Modify Transactions

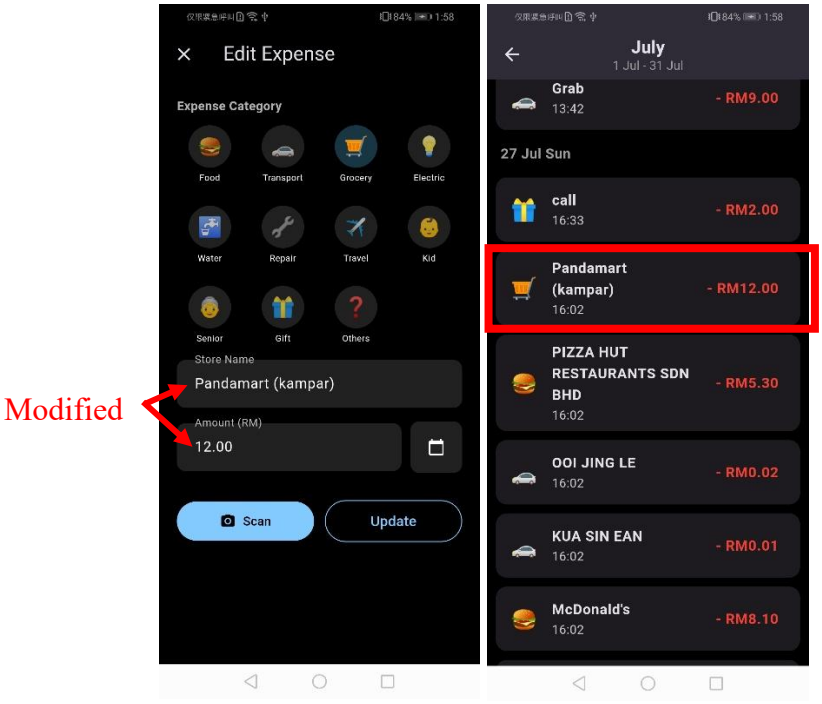


Figure 5.5.23 Changes Saved

Figure 5.5.24 shows the deletion of the transaction. To delete an expense, the user must swipe left the expense entry, then an alert of delete pop up shown, to reconfirm user action, if user press deletes, then success delete. The user can then press “Delete” to remove the expense, and later the changes will update to the database in real-time.

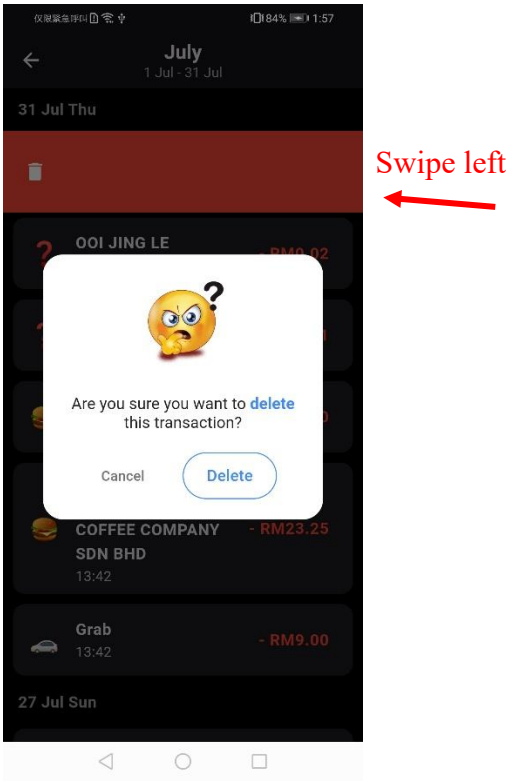


Figure 5.5.24 Delete Transaction with Pop-up Alert

5.5.10 Spin Wheel Game

Figure 5.5.25 shows the Spin Wheel Reward Game feature. This feature enables users to spin the wheel and deduct 50 points per spin. There are three reward options including extra points, a TNG reload cash voucher, or no reward at all. Besides that, the points transaction history, such as spin reward, spin attempt, or budget related reward had also been recorded and shown to user.

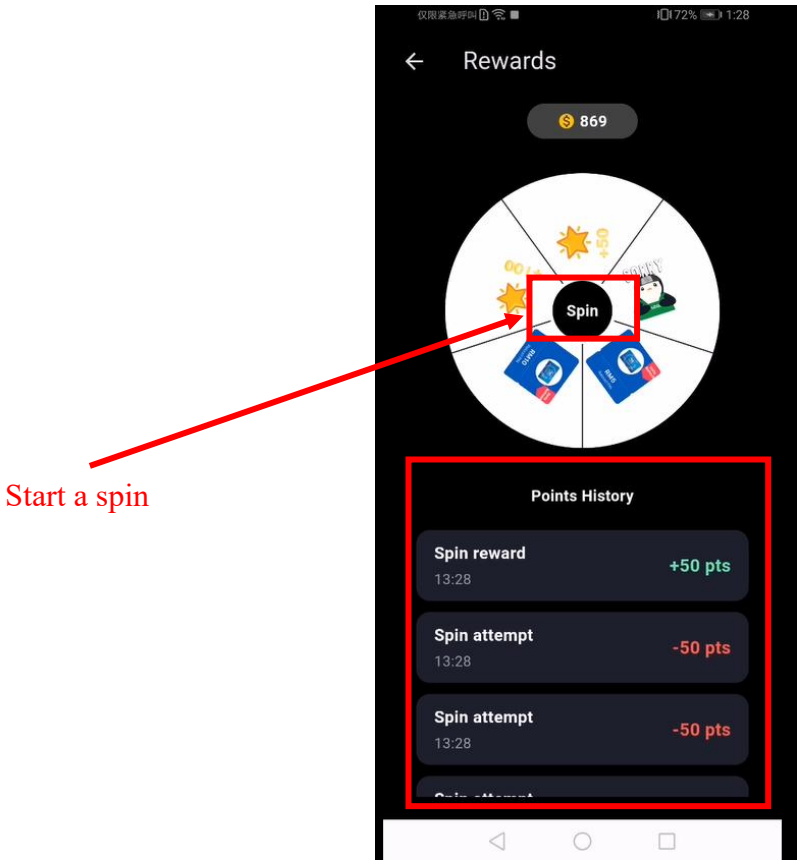


Figure 5.5.25 Spin Wheel Reward

As shown in Figure 5.5.26, for users who win the TNG reload cash voucher, two pop-up messages will appear. The user receives congratulations for winning the voucher in the first pop-up, and the second pop-up will detail the instructions for redeeming it. To redeem the voucher, users are required to drop us an email with a screenshot of the winning screen, which serves as proof, in order to redeem the reward.

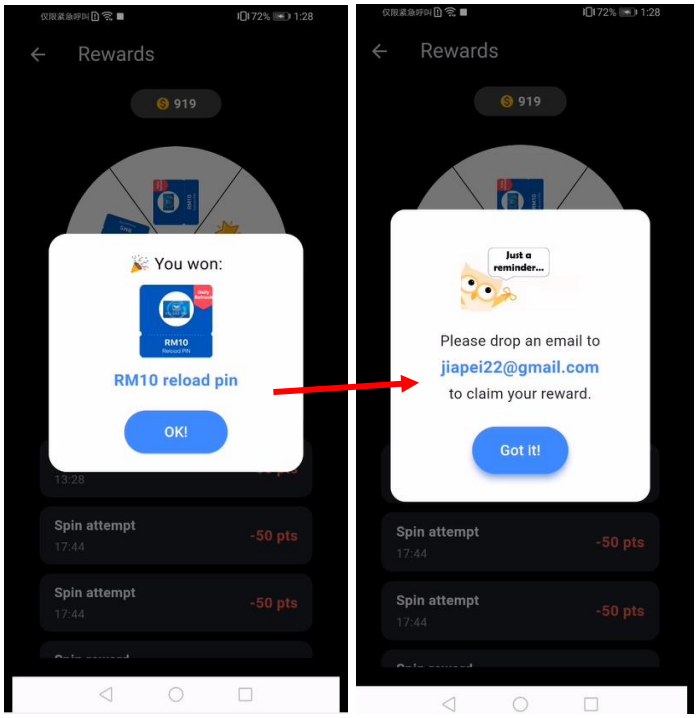


Figure 5.5.26 User wins the TNG Reload PIN

If the user wins extra points, these points will be automatically credited to their account as shown in Figure 5.5.27.

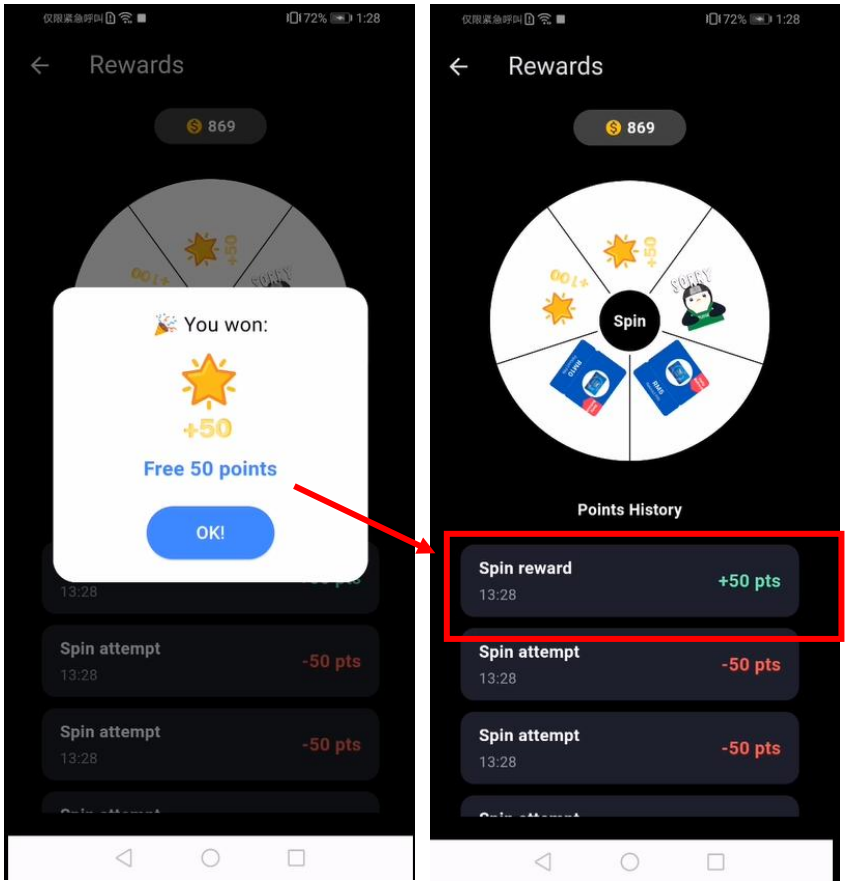


Figure 5.5.27 User wins extra points

Figure 5.5.28 shows the user does not win a prize, a motivational pop-up message with an apology will appear, prompting the user to spin again or try their luck next time.

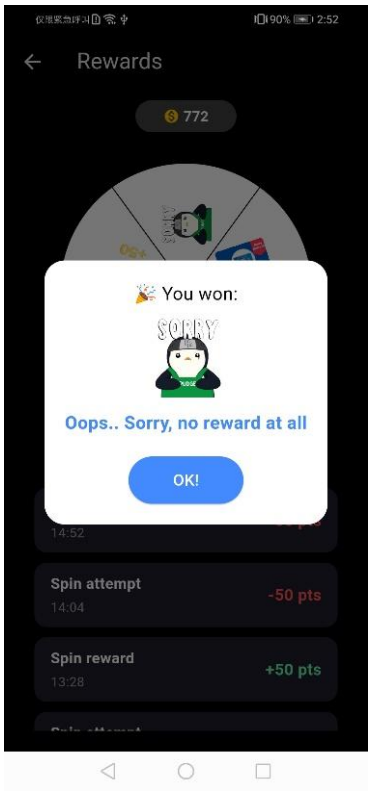


Figure 5.5.28 User does not win a prize

5.5.11 AI Tips Suggestion

Figure 5.5.29 shows the AI Tips suggestion, which provided users with personalized financial advice based on their remaining budget, local weather conditions, and total expenses. There are weather-related reminder, meal (for example, breakfast, lunch or dinner) and drink suggestions with an estimated price had been provided, to help users reduce unnecessary expenses and make more informed financial decisions. Such as, if the current weather is detected as Rain, the app will display a reminder to the user, such as bringing an umbrella or avoiding outdoor activities. For example, the system recommends users to have Yong Tau Foo (hawker) for RM10.00 and Barley drink for RM2.00 as their dinner. The "Save this Tip" button is used to let the user store today's AI-generated suggestions (Weather, Meal, Drink) into Firestore so the system able to avoid repeating the same tips too often. When the info icon is tapped, it displays the remaining food budget for the month. The food budget is calculated as 55% of the total monthly budget set by the user.

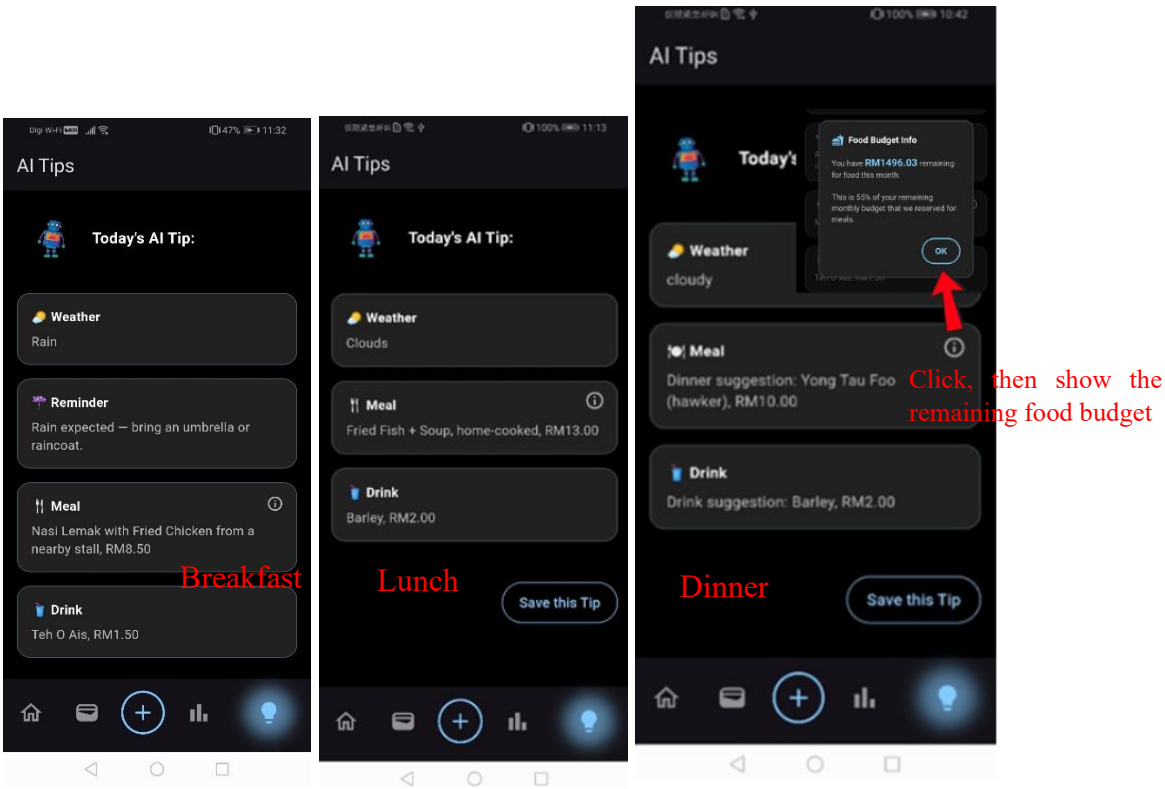


Figure 5.5.29 AI Tips Suggestion

5.5.12 Daily Check-In Reward

Figure 5.5.30 shows the Daily Check-In Reward, rewarding users with points each day, when they open the app and check in. When users check in every day for a maximum of seven days in a row, they are awarded with increasing streak points. This feature motivates users to log into the app daily. Hence, when they open the application to check-in, they will also record their expenses or income, which improves transaction tracking and increases the accuracy of their financial records.

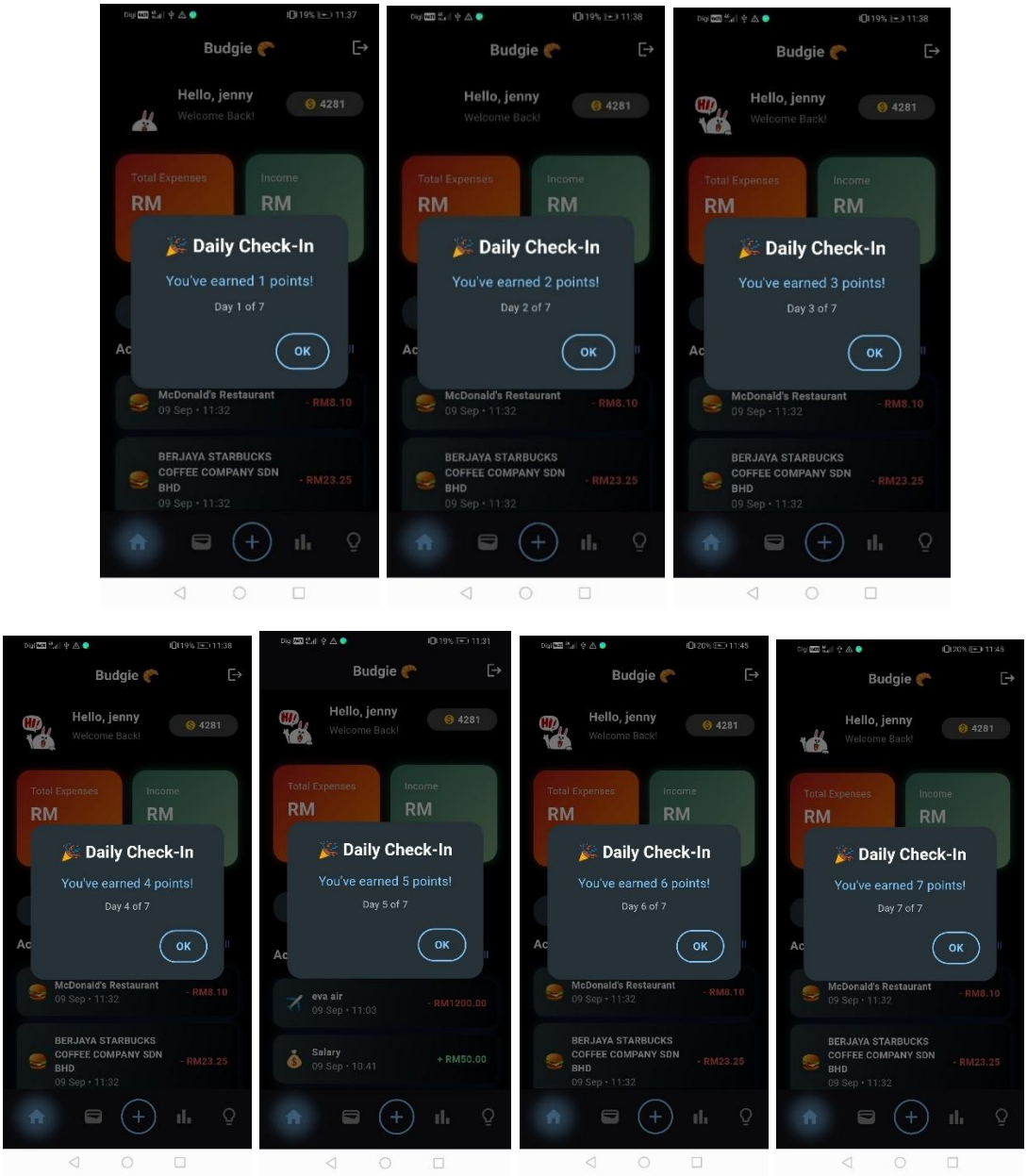


Figure 5.5.30 Daily Check-In Reward

5.6 Module Implementation Overview

5.6.1 User Authentication Module

Functionality Description

In this system, the User Authentication Module serves as an authentication tool. It combines both the registration and login functionalities to handle user accounts effectively. The registration (sign-up) function enables new users to create their personal accounts, establishing their own identities within the application. And for the login (sign-in) function, it allows existing users to access their accounts securely by entering their registered credentials. Together, these functions guarantee that only authorized users can access the system, protect personal information while allowing them to utilize all the application's functionalities.

Data Handling Process

The User Authentication module manages user credentials securely within the system. Upon successfully signed up, Firebase Authentication will handle the users' credentials which includes email and password securely. Whereas Firestore database will only store the email and account creation timestamp for user management purposes as shown in Figure 5.6.1.

```
final User? user = userCredential.user;
if (user != null) {
  print("✅ Sign-Up Successful: ${user.email}");

  // Add user to Firestore
  await FirebaseFirestore.instance.collection("users").doc(user.uid).set({
    'email': user.email,
    'created_at': FieldValue.serverTimestamp(),
  });
```

Figure 5.6.1 Data Handling Process for User Sign Up

Figure 5.6.2 shows the sign-in process, Firebase Authentication had been using by the system in order to verify the entered email and password. Once user login successfully, then it will retrieve the user details.

```
UserCredential userCredential = await _auth.signInWithEmailAndPassword(
  email: email.trim(),
  password: password.trim(),
);

final User? user = userCredential.user;
if (user != null) {
  print("✅ Login Successful: ${user.email}");
  return user;
```

Figure 5.6.2 Data Handling Process for User Sign In

5.6.2 Add Transaction

Functionality Description

In this system, Add Transaction module allows users to add either an expense entry or an income entry. A toggle button is provided for users to easily switch between the two options.

Data Handling Process

This module works based on listening to user's toggle selection, it will display the expense or income form respectively. According to Figure 5.6.3, there are two buttons which is "Expense" and "Income" that used to build a toggle switch. When the user selects an option, the system conditionally renders either the AddExpenseForm or AddIncomeForm widget accordingly.

```
// Toggle Switch
Container(
  margin: EdgeInsets.all(16),
  decoration: BoxDecoration(
    color: Colors.grey.shade800,
    borderRadius: BorderRadius.circular(30),
  ), // BoxDecoration
  child: Row(
    children: [
      _buildToggleButton("Expense", true),
      _buildToggleButton("Income", false),
    ],
  ), // Row
), // Container
// Conditional Form Display
Expanded(
  child: isExpense
      ? AddExpenseForm(userId: widget.userId)
      : AddIncomeForm(userId: widget.userId),
), // Expanded
```

Figure 5.6.3 Data Handling Process for Add Transaction

5.6.3 Add Expense (manually)

Functionality Description

In this system, the Add Expense Module allows users to manually record their daily expenses. First, users are required to select a category of expenses, then followed by entering the amount of expenses, the store name, and lastly users can pick the date of the expenses. This is to ensure that users log their expenses consistently, hence the system able to generate the analysis chart accurately.

Data Handling Process

In this module, the _saveExpense() function had been used to handle the data. It receives the user input, such as the category of expenses, date, store name, and amount. And for the store name field, it is optional to be filled in. If left empty, the default value "Unknown" will be filled in. For the category selection, "Others" will be used as default, if the user does not select a category. Then, a CalendarDatePicker had been integrated through a custom dialog for date selection. By default, the system uses the current date, but users are allowed to select any date they preferred. Next, for the amount value, it will automatically format to two decimal places to ensure consistency. Once all inputs are validated, the data is passed to Firestore and stored in fields such as total_amount, store_name, timestamp and category.

In addition, this module also supports editing existing expense records from the transaction listing screen. The logic checks whether the entry is a new record or an existing one—if it is a new entry, the system creates a new document using the add() method. If the entry is being edited, the existing document is updated using the update() method.

```
Future<void> _saveExpense() async {
  double? amount = double.tryParse(amountController.text.trim());
  if (amount != null) {
    amount = double.parse(amount.toStringAsFixed(2));
  }

  String storeName = storeController.text.trim();
  User? user = FirebaseAuth.instance.currentUser;

  if (amount == null || user == null) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("Please enter a valid amount.")),
    );
    return;
  }

  setState(() => isLoading = true);

  try {
    final expenseData = {
      'store_name': storeName.isNotEmpty ? storeName : null,
      'total_amount': amount,
      'category': selectedCategory,
      'timestamp': Timestamp.fromDate(selectedDate ?? DateTime.now()),
    };

    if (widget.isEditing && widget.docId != null) {
      // Update existing
      await FirebaseFirestore.instance
          .collection('users')
          .doc(user.uid)
          .collection('expenses')
          .doc(widget.docId)
          .update(expenseData);
    } else {
      // Add new
      await FirebaseFirestore.instance
          .collection('users')
          .doc(user.uid)
          .collection('expenses')
          .add(expenseData);
    }
```

Figure 5.6.4 _saveExpense() function used to store manual expense records in Firestore

5.6.4 Add Expense (receipt scanning)

Functionality Description

In this system, the Add Expense Module also provide users with the receipt scanning capability. It allows users to scan their receipts or transaction details by using their phone's camera or upload from their gallery. The extracted information is then filled into the input fields accordingly. Then, users are also provided with flexibility to manually modify the details in case any mistakes during the scanning process.

Data Handling Process

In this module, OCR (Optical Character Recognition) and GPT technologies have been integrated to extract key details, such as store name, total amount, and expense category. First, the image_picker packages, enables users to either take a picture or upload the receipt or transaction images from their gallery. Next, the http package is used to communicate with external services. As shown in Figure 5.6.5, the receipt or transaction images will be sent to the Google Cloud Vision API for OCR processing in order to extract raw text from the image. According to Figure 5.6.6, the extracted text later will be forwarded to the OpenAI GPT API in order to perform analysis and return the structured expense data in JSON format, which includes store name, total amount, and automatically identified category. Later, the details will be shown to users, in order to get their confirmation. Upon received the confirmation from users, then then date will be passed back to the Add Expense form, and the value will fill into the input fields accordingly. Lastly, the existing _saveExpense() function is used to store the data in Firestore.

```
final response = await http.post(
  Uri.parse(endpoint),
  headers: {
    "Content-Type": "application/json",
  },
  body: jsonEncode({
    "requests": [
      {
        "image": {"content": base64Image},
        "features": [
          {"type": "TEXT_DETECTION"}
        ]
      }
    ]
  }),
);
```

Figure 5.6.5 Google Cloud Vision API for OCR text extraction

```
final response = await http.post(
  Uri.parse("https://api.openai.com/v1/chat/completions"),
  headers: {
    "Authorization": "Bearer $apiKey",
    "Content-Type": "application/json",
  },
  body: jsonEncode({
    "model": "gpt-4o-mini",
    "messages": [
      {
        "role": "system",
        "content": """
```

Figure 5.6.6 Data handling process by OpenAI GPT API

5.6.5 Add Income

Functionality Description

In this system, the Add Income module enables users to record their income manually. There are several categories of income provided for users to choose, such as Bonus, Salary, Investment, Refund, Fixed, Business, and Other. In addition, a recurring income category name as “Fixed” had been provided for user, this is to allow users to input their fixed income that will be occur monthly. This feature able to relieve users from inputting the income monthly by themselves. Users can also select the date of the income; if the user does not choose, then the current date will be applied. To ensure consistency, the income is automatically rounded to two decimal places.

Data Handling Process

First, the CalendarDatePicker had been integrated through a custom dialog to allow users to select the income date. By default, the system uses the current date, but users are allowed to select any date they preferred. As shown in Figure 5.6.7, the system also saves a record under the recurring_income collection, which includes field likes, amount, category, start_date and last_inserted_month. This field is used to ensure that fixed income is automatically updated to the system at the beginning of each new month, assisting users to track their recurring income without manual entry.

According to Figure 5.6.8, this module also supports editing existing income records from the transaction listing screen. The logic checks whether the entry is a new record or an existing one—if it is a new entry, the system creates a new document using the add() method. If the entry is being edited, the existing document is updated using the update() method.

```
// Also add to recurring income if "Fixed"
if (categoryName == "Fixed") {
  await FirebaseFirestore.instance
      .collection("users")
      .doc(user.uid)
      .collection("recurring_income")
      .add({
    "amount": incomeAmount,
    "category": "Fixed",
    "start_date": Timestamp.fromDate(selectedDate ?? DateTime.now()),
    "last_inserted_month": DateFormat('yyyy-MM').format(DateTime.now()),
  });
}
```

Figure 5.6.7 Data handling process by OpenAI GPT API

```
try {
  if (widget.isEditing && widget.docId != null) {
    // Update income
    await FirebaseFirestore.instance
        .collection("users")
        .doc(user.uid)
        .collection("income")
        .doc(widget.docId)
        .update(incomeData);
  } else {
    // Add income
    final docRef = await FirebaseFirestore.instance
        .collection("users")
        .doc(user.uid)
        .collection("income")
        .add(incomeData);
```

Figure 5.6.8 Update Income or Add New Income

5.6.6 Transaction Listing and Edit/Delete Transaction

Functionality Description

In this system, the Transaction Listing module displays all of the transaction by the users, which including both expenses and income. To ensure users able to track easily their expenditures and income, the transactions had been set to displayed from the latest to the oldest date. From this listing, users able to have an overview towards the total amount of expenses or income, category, and the time of recorded for each expense or income entry. By implementing this module, it able to provide users with a detailed and well-organized view towards their financial activities in the current month. Additionally, users are allowed to swipe right to edit their transactions or swipe left to delete a transaction inside this listing screen.

Data Handling Process

First, all the transactions data is retrieved from the database and combined into a single list of transactions. According to Figure 5.6.9, the transactions are displayed in descending order, which are sorted by their timestamp field. In addition, when user wanted to edit a transaction, the system will load the form either AddExpenseForm or AddIncomeForm in edit mode accordingly as shown in Figure 5.6.10. As shown in Figure 5.6.11, if the user chooses to delete a transaction, the corresponding data will be removed from Firestore.

```
DateTime date = (data['timestamp'] as Timestamp).toDate();
String formattedDate =
    DateFormat('d MMM EEE').format(date);

if (!groupedTransactions.containsKey(formattedDate)) {
  groupedTransactions[formattedDate] = [];
}
groupedTransactions[formattedDate]!.add(doc);
```

Figure 5.6.9 Grouping Transactions by Date

```
// Navigate to appropriate form based on transaction type
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => Scaffold(
      backgroundColor: ■Colors.black,
      appBar: AppBar(
        backgroundColor: ■Colors.black,
        title: Text(collectionName == "expenses"
            ? "Edit Expense"
            : "Edit Income"), // Text
        leading: IconButton(
          icon: Icon(Icons.close),
          onPressed: () => Navigator.pop(context),
        ), // IconButton
      ), // AppBar
      body: collectionName == "expenses"
          ? AddExpenseForm(
              userId: userId,
              isEditing: true,
              docId: docId,
              initialCategory: category,
              initialStoreName: storeName,
              initialAmount: amount.replaceAll(RegExp(r'[-+ RM]'), ''),
              initialDate: initialDate,
            ) // AddExpenseForm
          : AddIncomeForm(
              userId: userId,
              isEditing: true,
              docId: docId,
              initialCategory: category,
              initialAmount: amount.replaceAll(RegExp(r'[-+ RM]'), ''),
              initialDate: initialDate,
            ), // AddIncomeForm
    ), // Scaffold
  ), // MaterialPageRoute
);
```

Figure 5.6.10 Edit Transaction Functionality

```
await FirebaseFirestore.instance
    .collection('users')
    .doc(userId)
    .collection(collectionName)
    .doc(docId)
    .delete();
```

Figure 5.6.11 Delete Transaction

5.6.7 Financial Insights

Functionality Description

In this system, the Financial Insights module provide charts for users to visualize their income and expenses. There is a toggle button provided to users, that allows users to switch between Day Chart or Month Chart view.

For the day chart screen a vertical bar chart shows the total expenses for each day within the selected month. Besides that, users allowed to switch between the month to view different month daily expenses details. Additionally, a category breakdown is shown, listing the total amount spent in each category for the current month in descending order. This helps users quickly identify high-expense areas. By default, a tooltip displays the current day's expense, and it updates dynamically when a user taps on any bar.

For the month chart screen, it consists of 12 boxed that display monthly total expenses from January to December. Thus, users able to track income versus expense trends and better understand their income-to-expense balance over time.

Data Handling Process

First, cloud_firestore is used to retrieve the expense and income data for visualization. Then, fl_chart had been integrated for rendering the bar charts. For the toggle button, the system will listen to user's action and display either ChartDay() or ChartMonth() accordingly.

5.6.8 Daily Check-In

Functionality Description

In order to promotes user engagement, the Daily Check-In module rewarding users with points each day, when they open the app and check in. When users check in every day for a maximum of seven days in a row, they are awarded with increasing streak points.

Data Handling Process

First, firebase_auth had been integrated for user identification. Follow with cloud_firestore, as it is used to read and update check-in streak data stored under users/{uid}/checkin/checkin_status. When the user launches the app and redirects to the home screen, the system determines whether the user has already checked in for the day. Based on Figure 5.6.12, an AlertDialog is displayed as a reward popup when the user successfully checks in for the first time that day.

```
@override
Widget build(BuildContext context) {
  return AlertDialog(
    backgroundColor: Color(0xFF1E1E2C), // Dark background like spin reward
    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
    title: Center(
      child: Text(
        "🎉 Daily Check-In",
        style: TextStyle(
          fontSize: 22,
          fontWeight: FontWeight.bold,
          color: Colors.white,
        ), // TextStyle
      ), // Text
    ), // Center
    content: Column(
      mainAxisSize: MainAxisSize.min,
      children: [
        Text(
          "You've earned ${dayStreak.clamp(1, 7)} points!",
          style: TextStyle(
            fontSize: 24,
            fontWeight: FontWeight.bold,
            color: Colors.amberAccent,
          ), // TextStyle
        ), // Text
        SizedBox(height: 12),
        Text(
          "Day $dayStreak of 7",
          style: TextStyle(
            color: Colors.white70,
            fontSize: 16,
          ), // TextStyle
        ), // Text
      ],
    ), // Column

    actions: [
      Center(
        child: ElevatedButton(
          onPressed: () => Navigator.pop(context),
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.cyanAccent[400],
            padding: EdgeInsets.symmetric(horizontal: 32, vertical: 12),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(30),
            ), // RoundedRectangleBorder
          ),
          child: Text(
            "OK",
            style: TextStyle(
              fontSize: 16,
              fontWeight: FontWeight.bold,
              color: Colors.black,
            ), // TextStyle
          ), // Text
        ), // ElevatedButton
      ), // Center
    ],
  ); // AlertDialog
```

Figure 5.6.12 Popup displaying check-in streak and awarded points

There are three conditions of logic checking had shown in Figure 5.6.13: If the last check-in date matches today, no action is taken. Else if the user last check-in was yesterday, the streak is incremented by one up to a maximum of 7. Otherwise, the streak is reset to 1. After determining the correct streak value, the system updates the Firestore document with the new last_checkin date and streak. The user then earns points equal to the current streak value, and the updated total points earned will be shown on the home screen.

```
Future<void> _checkDailyCheckIn(BuildContext context) async {
  if (user == null) return;

  final checkinRef = FirebaseFirestore.instance
      .collection("users")
      .doc(user!.uid)
      .collection("checkin")
      .doc("checkin_status");

  final doc = await checkinRef.get();
  final today = DateFormat('yyyy-MM-dd').format(DateTime.now());

  int newStreak = 1;

  if (doc.exists) {
    final data = doc.data()!;
    final lastCheckIn = data["last_checkin"];
    final lastStreak = data["streak"] ?? 1;

    if (lastCheckIn == today) return;

    final yesterday = DateFormat('yyyy-MM-dd')
        .format(DateTime.now().subtract(Duration(days: 1)));
    if (lastCheckIn == yesterday) {
      newStreak = (lastStreak + 1).clamp(1, 7);
    }
  }

  await checkinRef.set({
    "last_checkin": today,
    "streak": newStreak,
  });
```

Figure 5.6.13 Code snippet showing daily check-in streak logic and Firestore update

5.6.9 Budget Management

Functionality Description

In this system, the Budget Management module enables users to set a monthly spending limit and track their expenditure in real time against that limit. First, users need to manually enter their intended budget for the month, then a suggested savings amount will be provided for user to have an idea towards the amount they need to save. Hence, they can plan wisely when making purchases. Apart from that, the system will automatically reward users with 100 points if users stay within their set budget for the month.

Data Handling Process

As shown in Figure 5.6.14, the user defines a value for their monthly budget, then this data will be stored into the Firestore under the path users/{uid}/budget/{year-month}. For real-time expense tracking, a progress bar was used to show the user's budget usage. If the total expenses exceed the budget, the progress bar turns red to alert the user; otherwise, it remains blue, indicating that the expenses are still within the budget. The system will show existing users how much of their budget they had been used and offer a savings amount, which is 20% of the remaining budget. Besides that, the system only allows users to set budget once per month.

```
Future<void> _saveBudget() async {
  if (user == null) {
    _showMessage("User not authenticated!");
    return;
  }

  if (_budgetController.text.isEmpty) return;

  double budgetAmount = double.tryParse(_budgetController.text) ?? 0.0;
  if (budgetAmount <= 0) {
    _showMessage("Please enter a valid budget amount.");
    return;
  }

  setState(() => isSaving = true);

  try {
    String currentDocId = "${DateTime.now().year}-${DateTime.now().month}";
    final docRef = FirebaseFirestore.instance
        .collection("users")
        .doc(user!.uid)
        .collection("budget")
        .doc(currentDocId);

    final existingDoc = await docRef.get();

    if (existingDoc.exists) {
      _showMessage("Budget already set for this month.");
    } else {
      await docRef.set({
        "amount": budgetAmount,
        "created_at": FieldValue.serverTimestamp(),
      });
      _budgetController.clear();
      FocusScope.of(context).unfocus();
      _showMessage("✅ Budget set successfully!");
    }
  } catch (e) {
    _showMessage("Error saving budget: ${e.toString()}");
  }

  setState(() => isSaving = false);
}
```

Figure 5.6.14 Code snippet for saving monthly budget to Firestore

As shown in Figure 5.6.15, during the first week of a new month, the system checks for the user's total expenses from the previous month under the Monthly Reward Logic.

```dart
Future<double> _getTotalExpenses() async {
  double totalSpent = 0.0;
  DateTime now = DateTime.now();
  DateTime startOfMonth = DateTime(now.year, now.month, 1);
  DateTime startOfNextMonth = DateTime(now.year, now.month + 1, 1);

  QuerySnapshot snapshot = await FirebaseFirestore.instance
      .collection("users")
      .doc(user?.uid)
      .collection("expenses")
      .where("timestamp", isGreaterThanOrEqualTo: startOfMonth)
      .where("timestamp", isLessThan: startOfNextMonth)
      .get();

  for (var doc in snapshot.docs) {
    var data = doc.data() as Map<String, dynamic>;
    totalSpent += (data["total_amount"] as num?)?.toDouble() ?? 0.0;
  }

  return totalSpent;
}
```

Figure 5.6.15 Code snippet for calculating and displaying total expenses

Figure 5.6.16 shows that if the user's expenditure was within the allocated budget, they will receive 100 points as rewards. A popup message had been used to notify the user whether they received a reward or not and ensures that the reward is only shown once by using a Firestore popup_shown flag.

```dart
Future<void> _rewardPointsIfWithinBudget(double budget, double spent) async {
  if (spent > budget) return; // No reward if over budget

  final rewardRef = FirebaseFirestore.instance
      .collection("users")
      .doc(user!.uid)
      .collection("rewards")
      .doc("points");

  final snapshot = await rewardRef.get();
  int currentPoints = 0;
  int currentMonth = DateTime.now().month;

  if (snapshot.exists) {
    currentPoints = (snapshot.data()?['total'] ?? 0);
    int? lastRewardedMonth = snapshot.data()?['last_rewarded_month'];

    if (lastRewardedMonth == currentMonth) {
      print("❌ Already rewarded this month in _rewardPointsIfWithinBudget");
      return; // Skip if already rewarded
    }
  }

  // Only now reward
  await rewardRef.set({
    'total': currentPoints + 100,
    'last_rewarded_month': currentMonth,
    'updated_at': FieldValue.serverTimestamp(),
  }, SetOptions(merge: true));

  _showMessage("🎉 You earned 100 points for staying within budget!");
}
```

Figure 5.6.16 Reward points logic if within budget

5.6.10 Spin Wheel Game & Reward Accumulation

Functionality Description

In this system, the Spin Wheel module provides a gamified reward experience for users. This module is integrated with both the Budget Management and Daily Check-In modules, where users can collect points from these two modules. Then, users can spend fifty points to spin a fortune wheel. As shown in Figure 5.6.17, all users stand a chance to win various rewards such as bonus points, TNG reload pin or nothing at all. This feature motivates users to check-in daily into the app to record down their expenses or income regularly.

```
final List<String> rewards = [
  "RM5 reload pin",
  "RM10 reload pin",
  "Free 100 points",
  "Free 50 points",
  "Oops.. Sorry, no reward at all",
];
```

Figure 5.6.17 Reward Listing

Data Handling Process

First, the flutter_fortune_wheel package had been employed to handle the spinning wheel animation and reward selection. The user's current points are retrieved from Firestore as the screen loads, and for any updates, like point increments or deductions are reflected in real time. When the user taps the “Spin” button, the system first determines whether the user has at least 50 points. If sufficient, 50 points are deducted using FieldValue.increment(-50) as shown in Figure 5.6.18. To determine the reward, the dart:math library is used to generate a random number between 0 and 4 via Random().nextInt(5), which corresponds to one of five reward options. The system handles three reward types differently: If the reward includes bonus points such as “Free 100 points”, the points will be added to the user’s balance according by using FieldValue.increment(...). Else If the reward is a reload pin such as “RM5 reload pin”, a congratulation message is shown with instructions to claim it via email. Else the result is no reward, the system displays a “Sorry” message, and no points are granted. Finally, an AlertDialog is used to display the reward result in a popup format, along with a message or instruction on how to claim the reward.

```dart
Future<void> _deductPointsAndSpin() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user == null || userPoints < 1) return;

  setState(() {
    isSpinning = true;
    result = Random().nextInt(5); // 5 rewards
  });

  controller.add(result!);

  await FirebaseFirestore.instance
      .collection("users")
      .doc(user.uid)
      .update({"points": FieldValue.increment(-50)});

  setState(() {
    userPoints -= 50;
  });
  await FirebaseFirestore.instance
      .collection("users")
      .doc(user.uid)
      .collection("points_history")
      .add({
    "points": -50,
    "description": "Spin attempt",
    "timestamp": Timestamp.now(),
  });

  await Future.delayed(Duration(seconds: 4));

  _showResultDialog(result!);
  setState(() => isSpinning = false);
}
```

Figure 5.6.18 Point Deduction and Spin Logic Using Random Reward Generation

5.6.11 AI Tips Suggestions

Functionality Description

In this system, personalized savings tips had been provided by the AI Tips Suggestions module. These tips were generated based on the local weather conditions and user's remaining budget. The goal of this module is to help users make more informed purchases decisions and avoid unnecessary expenditures. The system integrates real-time weather data and AI-generated responds to provide context-aware suggestions, for example budget-friendly meal plans, including breakfast, lunch and dinner. Besides that, weather-related reminders provided, such as the system will show either sunny, cloudy or raining, thus remind user to bring an umbrella, when predicted there will be raining later. As a result, users receive these suggestions as part of their daily experience, encouraging them to manage their money more wisely.

Data Handling Process

The AI Tips Suggestions module had integrated multiple services for generating and displaying suggestions. First, the system had integrated Firestore to store and retrieve the AI-generated tips. Figure 5.6.19 shows that the tip content and timestamp had been stored under users > [userId] > ai_tips.

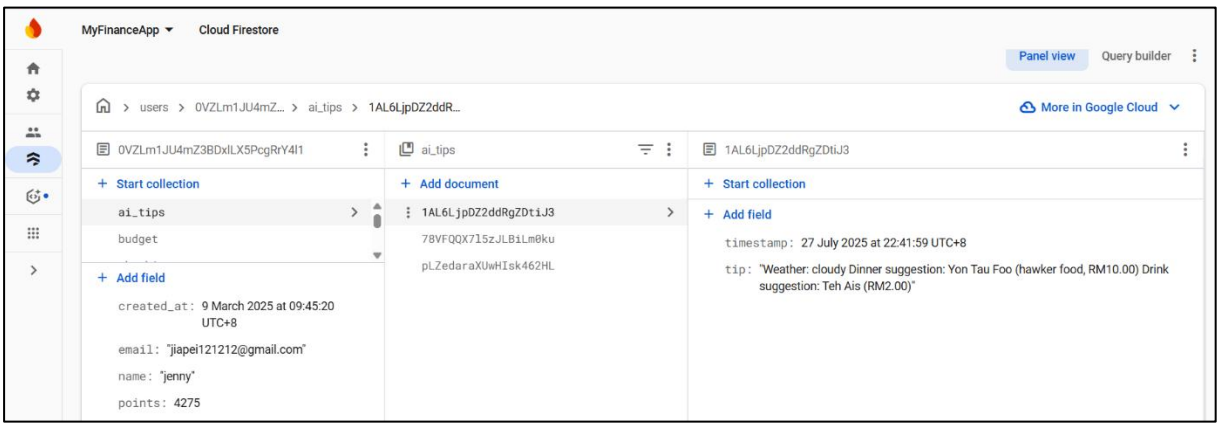


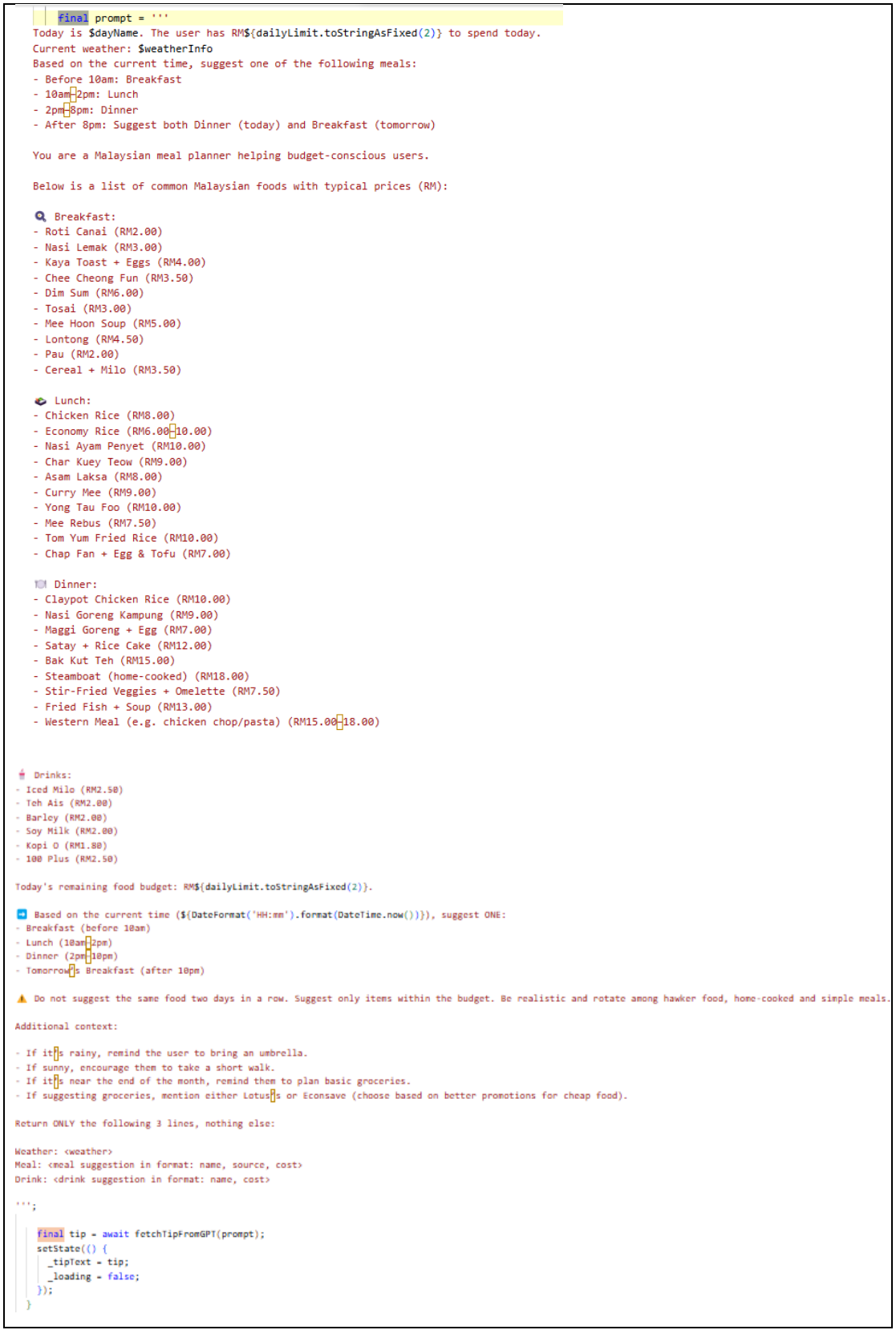
Figure 5.6.19 Firebase Firestore Structure for AI Tips

In order to provide the weather-related reminders, Figure 5.6.20 shows the OpenWeatherMap API had been integrated in the getCurrentWeather() function, to retrieve current weather data according to the user's current location. For user's current location, the system integrates the geolocator package to access the device's GPS, hence able to detect user's location accurately.

```
Future<String> getCurrentWeather() async {
  final apiKey = 'e6c54d43c4b2fb342cdb8e311e9316ec';

  try {
    // Request location permission
    LocationPermission permission = await Geolocator.requestPermission();
    if (permission == LocationPermission.denied ||
        permission == LocationPermission.deniedForever) {
      return "Location permission denied";
    }

    // Get current location
    Position position = await Geolocator.getCurrentPosition(
        desiredAccuracy: LocationAccuracy.low);

    final lat = position.latitude;
    final lon = position.longitude;

    final url = Uri.parse(
        'https://api.openweathermap.org/data/2.5/weather?lat=$lat&lon=$lon&appid=$apiKey&units=metric');

    final response = await http.get(url);

    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      final condition = data['weather'][0]['main'];
      final temp = data['main']['temp'].toDouble();
      return "$condition, ${temp.toStringAsFixed(1)}°C";
    } else {
      return "Weather unavailable";
    }
  } catch (e) {
    return "Failed to retrieve weather";
  }
}
```

Figure 5.6.20 Firebase Firestore Tip Storage

Next, OpenAI's GPT model had been employed to generate personalized financial advice which using the user's budget, weather, day, and time context. The prompt that showed in Figure 5.6.21 is then sent to OpenAI's GPT model via a secure API call which generates a suitable food and beverage suggestion for the user.



```
    final prompt = '''
Today is $dayName. The user has RM${dailyLimit.toStringAsFixed(2)} to spend today.
Current weather: $weatherInfo
Based on the current time, suggest one of the following meals:
- Before 10am: Breakfast
- 10am–2pm: Lunch
- 2pm–8pm: Dinner
- After 8pm: Suggest both Dinner (today) and Breakfast (tomorrow)

You are a Malaysian meal planner helping budget-conscious users.

Below is a list of common Malaysian foods with typical prices (RM):

🍳 Breakfast:
- Roti Canai (RM2.00)
- Nasi Lemak (RM3.00)
- Kaya Toast + Eggs (RM4.00)
- Chee Cheong Fun (RM3.50)
- Dim Sum (RM6.00)
- Tosai (RM3.00)
- Mee Hoon Soup (RM5.00)
- Lontong (RM4.50)
- Pau (RM2.00)
- Cereal + Milo (RM3.50)

🍛 Lunch:
- Chicken Rice (RM8.00)
- Economy Rice (RM6.00–10.00)
- Nasi Ayam Penyet (RM10.00)
- Char Kuey Teow (RM9.00)
- Asam Laksa (RM8.00)
- Curry Mee (RM9.00)
- Yong Tau Foo (RM10.00)
- Mee Rebus (RM7.50)
- Tom Yum Fried Rice (RM10.00)
- Chap Fan + Egg & Tofu (RM7.00)

🍽️ Dinner:
- Claypot Chicken Rice (RM10.00)
- Nasi Goreng Kampung (RM9.00)
- Maggi Goreng + Egg (RM7.00)
- Satay + Rice Cake (RM12.00)
- Bak Kut Teh (RM15.00)
- Steamboat (home-cooked) (RM18.00)
- Stir-Fried Veggies + Omelette (RM7.50)
- Fried Fish + Soup (RM13.00)
- Western Meal (e.g. chicken chop/pasta) (RM15.00–18.00)


🥤 Drinks:
- Iced Milo (RM2.50)
- Teh Ais (RM2.00)
- Barley (RM2.00)
- Soy Milk (RM2.00)
- Kopi O (RM1.80)
- 100 Plus (RM2.50)

Today's remaining food budget: RM${dailyLimit.toStringAsFixed(2)}.

➡️ Based on the current time (${DateFormat('HH:mm').format(DateTime.now())}), suggest ONE:
- Breakfast (before 10am)
- Lunch (10am–2pm)
- Dinner (2pm–10pm)
- Tomorrow's Breakfast (after 10pm)

⚠️ Do not suggest the same food two days in a row. Suggest only items within the budget. Be realistic and rotate among hawker food, home-cooked and simple meals.

Additional context:

- If it's rainy, remind the user to bring an umbrella.
- If sunny, encourage them to take a short walk.
- If it's near the end of the month, remind them to plan basic groceries.
- If suggesting groceries, mention either Lotus's or Econsave (choose based on better promotions for cheap food).

Return ONLY the following 3 lines, nothing else:

Weather: <weather>
Meal: <meal suggestion in format: name, source, cost>
Drink: <drink suggestion in format: name, cost>

''';

    final tip = await fetchTipFromGPT(prompt);
    setState(() {
      _tipText = tip;
      _loading = false;
    });
  }
```

Figure 5.6.21 AI Prompt Construction and GPT API Call

5.7 Concluding Remark

In summary, the system successfully integrates Firebase Authentication and Cloud Firestore to ensure secure, real-time data handling. Overall, the system not only promotes better money management via releasing users from manually recording their expenses but also enhances user experience with the gamified elements, like the spin wheel and daily check-in.

This chapter has outlined the implementation details of each module within the personal finance management system based on the system design. It began with an overview of the required hardware and software tools, followed by the setup processes needed for development. It then described the system requirements, both functional and non-functional requirements, and continued with the system operation, showcasing how the entire system works, covering every screen flow in detail.

And lastly, each module was discussed in detail, including how it functions, the user interactions, and how data is handled, and code snippets had provided as supporting documents to enhance the overall clarification process.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

6.1 System Testing and Performance Metrics

To ensure all the system features behave as expected and meet their functional requirements, black-box testing had been employed to test the whole system and how the system behaves based on user inputs. Black box testing is a software testing method that focuses on testing the system's functionality without inspecting its internal code structure. This method focuses on inputs, expected outputs, and system behavior.

System testing means testing the whole app to ensure it works correctly as a complete system. First, functional testing had been carried out to verify that main functions such as login or sign-up, adding expenses or income, budget management, viewing charts, daily check-in calculations, and triggering rewards worked accurately based on the input provided. The detailed test cases for each function had been carried out and described in detail in Section 6.2 below to ensure that the outputs match the expected results. Additionally, integration testing was also carried out to ensure that different modules, such as the Firestore database, Firebase Authentication, and the budget reward system, interact properly.

Apart from system testing, multiple metrics were also discovered. First, we discovered that the system took less than one second to record the transaction, either income or expenses, into Firestore, whereas the average app launch time was about 2.1 seconds. Then, the latest amount of total expense or income will be displayed on the Home Screen. While for the OCR transaction scanning feature, it is able to handle the extraction of multiple transactions from a single transaction image. And the system took approximately 3 to 5 seconds to process a single receipt image, depending on image clarity. Next, the budget-based reward system was examined to ensure it calculates accurately whether the previous month’s spending was capped within the budget that had been set, and points were being awarded appropriately at the beginning of a new month. Lastly, the system was monitored to detect any crashes during operation, ensuring it runs smoothly without errors.

6.2 Testing Setup and Results

6.2.1 User Sign-Up Function Testing

Table 6.2.1 User Sign-Up Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Enter username, valid email and password, then press "Sign Up" button Username: jiapei Email: jiapei@gmail.com Password: Angjiapei123	App displays "Signup Successful" and navigates to Home Screen.	Outcome matches the acceptance criteria.	✓
TC02	Enter an email that had registered, then press "Sign Up" button Username: jiapei Email: jiapei@gmail.com Password: Angjiapei123	App displays "Signup Failed", preventing duplicate registration.	Outcome matches the acceptance criteria.	✓
TC03	Leave username blank, then press "Sign Up" button Username: Email: jiapei@gmail.com Password: Angjiapei123	Application displays a "Signup Failed" message, and the border of the unfilled username input field is highlighted in red.	Outcome matches the acceptance criteria.	✓
TC04	Leave email blank, then press "Sign Up" button Username: jiapei Email: Password: Angjiapei123	Application displays a "Signup Failed" message, and the border of the unfilled email input field is highlighted in red.	Outcome matches the acceptance criteria.	✓
TC05	Leave password blank, then press "Sign Up" button Username: jiapei Email: jiapei@gmail.com Password:	Application displays a "Signup Failed" message, and the border of the unfilled password input field is highlighted in red.	Outcome matches the acceptance criteria.	✓
TC06	Click Logout from the Home Screen	App redirects to the Login Screen and clears session data.	Outcome matches the acceptance criteria.	✓

6.2.2 User Login Function Testing

Table 6.2.2 User Login Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Launch the application which user not logged in	User is redirected to the login screen.	Application redirects as expected.	✓
TC02	Enter correct email and password Email: jiapei@gmail.com Password: Angjiapei123	User successfully logs in and message "Login Successful: jiapei@gmail.com" being displayed, then redirected to the Home Screen.	Outcome matches the acceptance criteria.	✓
TC03	Enter incorrect email or password Email: jiapei@gmail.com Password: Angjiapei1234	System displays "Login Failed. Check credentials."	Outcome matches the acceptance criteria.	✓
TC04	Leave username, email or password field empty Email: Password: Angjiapei1234	System displays "Please enter both email and password."	Outcome matches the acceptance criteria.	✓
TC05	Leave username, email or password field empty Email: jiapei@gmail.com Password:	System displays "Please enter both email and password."	Outcome matches the acceptance criteria.	✓

6.2.3 Home Screen Function Testing

Table 6.2.3 Home Screen Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Launch the application which user logged in	User is redirected to the home screen, which displays welcome text, total spending, and expense listing (if data exists).	Application redirects as expected.	✓
TC02	Click the Button (+) to enter transaction (income/expenses) details	User is redirected to Add Transaction screen.	Application redirects as expected.	✓
TC03	Once transactions had recorded successfully	System updates the latest 4 transactions to show in home screen transaction listing side at real-time.	Outcome matches the acceptance criteria.	✓
TC04	Click “See All” button	User is redirected to the Transaction Listing screen within the app.	Outcome matches the acceptance criteria.	✓
TC05	Click the logout icon in the app bar	User is redirected to the Login Screen within the app.	Application redirects as expected.	✓
TC06	User logs in for the first time with no data	System displays “No expenses recorded yet.” Message.	Outcome matches the acceptance criteria.	✓
TC07	Click the Total Expenses box	User is redirected to the Chart screen.	Application redirects as expected.	✓
TC08	Click the Budget button	User is redirected to Budget screen.	Application redirects as expected.	✓
TC09	Click the Point Reward button	User is redirected to Reward screen.	Application redirects as expected.	✓
TC10	Click the Insight button	User is redirected to Chart Insight screen.	Application redirects as expected.	✓
TC11	Click the AI Tips button	User is redirected to AI Tips Suggestions screen.	Application redirects as expected.	✓

6.2.4 Daily Check-In Function Testing

Table 6.2.4 Daily Check-In Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	User checks in for the day	System detects daily check-in and adds the correct number of points based on streak day; updated points appear on home screen	Outcome matches the acceptance criteria.	✓
TC02	User misses a check-in day	System resets streak to Day 1 and starts point count from 1 on next check-in	Outcome matches the acceptance criteria.	✓

6.2.5 Add Transactions Function Testing

Table 6.2.5 Add Transactions Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Click the Expense toggle button	User is redirected to the Add Expense screen.	Application redirects as expected.	✓
TC02	Click the Income toggle button	User is redirected to the Add Income screen.	Application redirects as expected.	✓
TC03	Transaction successfully added	Total expense and income, transactions listing updated in real-time	Outcome matches the acceptance criteria.	✓
TC04	Clicks “X” button	User is redirected to Home screen	Application redirects as expected.	✓

6.2.6 Add Expenses Function Testing

Table 6.2.6 Add Expenses Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Select the expense's category and date, enter store name and amount, then click "Save Expense" Category: Food Store Name: KFC Amount (RM): 20.50 Date: 20/7/2025	System checks that all required details are filled and valid, updates the database, and redirects the user to the Home screen.	Outcome matches the acceptance criteria.	✓
TC02	Select the expense's category, leave the date blank, enter store name and amount, then click "Save Expense" Category: Food Store Name: KFC Amount (RM): 20.50 Date: 20/7/2025	System validates input, records the transaction with the current date, updates the database, and redirects the user to the Home screen.	Outcome matches the acceptance criteria.	✓
TC03	Not select the expense's category Category: Store Name: KFC Amount (RM): 10.50 Date: 20/7/2025	System record category as default which is "Others" and update to database.	Outcome matches the acceptance criteria.	✓
TC04	Leaves store name empty and clicks "Save Expense" Category: Food Store Name: Amount (RM): 10.50 Date: 20/7/2025	System record store name as "Unknown" and update to database.	Outcome matches the acceptance criteria.	✓
TC05	Leaves amount empty and clicks "Save Expense" Category: Food Store Name: KFC Amount (RM): Date: 20/7/2025	System displays error message "Please enter a valid amount."	Outcome matches the acceptance criteria.	✓

TC06	Enter an amount with more than two decimal places. Category: Food Store Name: KFC Amount (RM): 20.54567 Date: 20/7/2025	System automatically converts the amount into 2 decimal places (RM20.5467 to RM20.55) then update to database.	Outcome matches the acceptance criteria.	✓
TC07	Click “Scan” button	User is redirected to the Scan Receipt Screen.	Application redirects as expected.	✓

6.2.7 OCR Receipt Scanning and GPT Integration Function Testing

Table 6.2.7 OCR Receipt Scanning and GPT Integration Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Select upload image	User is redirected to gallery screen	Application redirects as expected.	✓
TC02	Select scan by camera image	User is redirected to camera screen	Application redirects as expected.	✓
TC03	Upload a transaction image	System extracts the amount and store name, analyzes the appropriate category, and allows the user to review their expenses.	Outcome matches the acceptance criteria.	✓
TC04	Check the extracted details and click "Confirm" button	User is redirected to Add Expense screen, and the extracted details are inserted into the specific fields accordingly.	Outcome matches the acceptance criteria.	✓
TC05	Upload multiple transaction images	System displays an alert dialog "Your receipts will now be processed in the background. Once done, your expenses will be automatically saved to the database. You may close the app or continue using it." to user.	Outcome matches the acceptance criteria.	✓
TC06	Upload an image with 2 transactions inside	System displays an alert dialog "We detected 2 transactions in this picture. We'll process them in the background and update your account. You can close the app or continue using it." to user.	Outcome matches the acceptance criteria.	✓
TC07	Upload a transaction image	System extracts the amount and store name, analyzes the appropriate category, and allows the user to review their expenses.	Outcome matches the acceptance criteria.	✓
TC08	Scan a physical receipt image by using the camera	System extracts the amount and store name, analyzes the appropriate category, and allows the user to review their expenses.	Outcome matches the acceptance criteria.	✓
TC09	Upload a digital receipt	System extracts the amount and store name, analyzes the appropriate category, and allows the user to review their expenses.	Outcome matches the acceptance criteria.	✓

6.2.8 Add Income Function Testing

Table 6.2.8 Add Income Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Select the income's category and date, then enter amount, and click "Save Income" Category: Salary Amount (RM): 1000 Date: 25/7/2025	System checks that all required details are filled and valid, updates the database, and redirects the user to the Home screen.	Outcome matches the acceptance criteria.	✓
TC02	Select the income's category, then enter amount, and click "Save Income" Category: Salary Amount (RM): 1000 Date:	System validates input, records the transaction with the current date, updates the database, and redirects the user to the Home screen.	Outcome matches the acceptance criteria.	✓
TC03	Not selects a category and clicks "Save Expense" Category: Amount (RM): 1000 Date: 25/7/2025	System record category as "Other" and update to database.	Outcome matches the acceptance criteria.	✓
TC04	Leaves amount empty and clicks "Save Income" Category: Bonus Amount (RM): Date: 25/7/2025	System displays error message "Please enter a valid amount."	Outcome matches the acceptance criteria.	✓
TC05	Enter amount which more than 2 decimal places Category: Bonus Amount (RM): 1000.667 Date: 25/7/2025	System automatically converts the amount into 2 decimal places (RM1000.667 to RM1000.67) then update to database.	Outcome matches the acceptance criteria.	✓
TC06	Select category "Fixed" and provide valid amount and date.	System will automatically record it as a recurring entry, so the same amount will be added again next month and shown on the Home screen.	Outcome matches the acceptance criteria.	✓

6.2.9 Budget Creation and Tracking Function Testing

Table 6.2.9 Budget Creation and Tracking Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Enter a valid budget amount, then press "Save Budget".	System saves the budget amount is to Firestore, and a success message is shown.	Outcome matches the acceptance criteria.	✓
TC02	Leave budget amount field empty and press "Save Budget".	System does not save budget and no success message shown.	Outcome matches the acceptance criteria.	✓
TC03	Enter a valid budget but already have a set budget for the month, then press "Save Budget".	System shows "Budget already set for this month" message.	Outcome matches the acceptance criteria.	✓
TC04	Verify if the monthly budget progress bar updates as user's expenses increase but still below 80% of the set budget.	The progress bar should remain blue color.	Outcome matches the acceptance criteria.	✓
TC05	Verify if the monthly budget progress bar updates as user's expenses increase but exceed 80% of the set budget.	The progress bar should remain red color.	Outcome matches the acceptance criteria.	✓
TC06	Suggested Savings with Remaining Budget: Budget = RM 500 Expenses = RM 300 Remaining Budget = RM 200	The suggested saving is calculated as 20% of (budget - expenses) and displayed as "Recommended savings: RM 40".	Outcome matches the acceptance criteria.	✓
TC07	Suggested Savings with No Remaining Budget: Budget = RM 500 Expenses = RM 500 Remaining Budget = RM 0	The suggested savings should be displayed as "Recommended savings: RM 0" when there's no remaining budget.	Outcome matches the acceptance criteria.	✓
TC08	Sets a monthly budget and adds expenses throughout the month, and does not overspent	System checks if the user stayed within the budget for the previous month, and a reward popup "You earned 100 points for staying within budget!" will be shown and award points to user's account at the start of the new month.	Outcome matches the acceptance criteria.	✓

TC09	Sets a monthly budget and adds expenses throughout the month, and overspent	System checks the user overspent for the previous month, and a warning popup will show and motivate them to stay within the budget for the current month at the start of the new month.	Outcome matches the acceptance criteria.	✓

6.2.10 Points Reward Function Testing

Table 6.2.10 Points Reward Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Verify if the user points are loaded correctly from Firestore on screen load.	System retrieved the current points value of user from Firestore and display in Home screen.	Outcome matches the acceptance criteria.	✓
TC02	Fail to start a spin	System disables the spin button when user's points are less than 1.	Outcome matches the acceptance criteria.	✓
TC03	Click the "Spin" button	System deducts user's points by 50 points after the spin button is clicked.	Outcome matches the acceptance criteria.	✓
TC04	Reward displayed based on the spin result	System pops up a dialog show the correct reward and corresponding image based on the spin result.	Outcome matches the acceptance criteria.	✓
TC05	Points history	System displays the points increment and decrement history in a list, with timestamps.	Outcome matches the acceptance criteria.	✓
TC06	When reward is the TNG Reload Pin	System displays a secondary popup with instructions on how to claim the reload pin via email.	Outcome matches the acceptance criteria.	✓

6.2.11 Insight Chart Function Testing

Table 6.2.11 Insight Chart Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Select Day chart	User is redirected to Day chart	Outcome matches the acceptance criteria.	✓
TC02	Select Month chart	User is redirected to Month chart	Outcome matches the acceptance criteria.	✓
TC03	Verify that the Month chart displays the total monthly expenses and income for each month of the current year	System shows the total expenses and income for each month from January to December.	Outcome matches the acceptance criteria.	✓
TC04	Verify that the Day chart shows daily expenses for the selected month	System shows expenses for each day in the selected month.	Outcome matches the acceptance criteria.	✓
TC05	Select a day by clicking on the corresponding vertical bar in the Day chart's bar chart	System shows that day's expense in a tooltip.	Outcome matches the acceptance criteria.	✓
TC06	Click on "<" in Day chart	System redirects user to the previous month.	Outcome matches the acceptance criteria.	✓
TC07	Click on ">" in Day chart	System redirects user to the next month.	Outcome matches the acceptance criteria.	✓
TC08	Verify that the Category chart properly displays categories sorted by the highest expense	System sorts and displays the categories in descending order based on total spending, showing both the expense amount and the percentage of the total spent.	Outcome matches the acceptance criteria.	✓

6.2.12 Expense Listing Function Testing

Table 6.2.12 Expense Listing Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Verify if transactions are displayed in descending order based on timestamp.	System displays transactions include both expenses and income which are sorted from the latest to the oldest based on the timestamp.	Outcome matches the acceptance criteria.	✓
TC02	Verify if the transaction row shows correct information, including store name, category, amount, and time.	System displays each transaction row with the store name, category, amount, and datetime.	Outcome matches the acceptance criteria.	✓
TC03	Swipe right to edit the transaction.	System pops up the Edit Transaction screen.	Outcome matches the acceptance criteria.	✓
TC04	Click the “Update”	System updates the modified transaction to Firestore.	Outcome matches the acceptance criteria.	✓
TC05	Swipe left to delete the transaction.	System pops up the Delete transaction confirmation dialog.	Outcome matches the acceptance criteria.	✓

6.2.13 AI Tips Suggestions Function Testing

Table 6.2.13 AI Tips Suggestions Function Testing

Test Case ID	Test and Input Description	Acceptance Criteria	Actual Result	Pass (Yes /No)
TC01	Verify if AI Tips are displayed.	System displays weather, budget, and generates personalized tips (meal + beverages).	Outcome matches the acceptance criteria.	✓
TC02	Verify if the weather condition and related reminder display correctly: Weather Condition: Rain Reminder: Remember to bring umbrella	System displays "Remember to bring umbrella."	Outcome matches the acceptance criteria.	✓
TC03	Verify if the weather condition and related reminder display correctly: Weather Condition: Cloudy/ Sunny Reminder: -	System no display any reminders to user.	Outcome matches the acceptance criteria.	✓
TC04	User taps "Save this Tip" button.	Tip is stored in Firestore under user's ai_tips subcollection.	Outcome matches the acceptance criteria.	✓

6.3 User Testing and Feedback Survey

To evaluate the application's usability, functionality, overall user satisfaction of the application, we had conducted a user testing and feedback survey. The survey was distributed via Google Forms and targeted beta testers who interacted with the system. Respondents were invited to provide feedback on a variety of aspects, including overall experience, interface design, feature usefulness, and ease of navigation. Hence, their responses were compiled and analyzed in order to identify strengths, areas for improvement, and potential future enhancements.

A total of 10 users participated in the user testing survey for this personal finance management application. With the majority of the respondents rating the app an overall score of 4-5 out of 5 for responsiveness, convenience of use, and first impressions. Below is a summary of the strengths and weaknesses identified from the responses submitted by users during testing. The detailed survey questions and results from ten participants had been discussed in the following part.

As shown in Figures 6.3.1 and 6.3.2, the **receipt scanning feature** also being rated as one of the most useful functions. Users described it as accurate in detecting the store name and spending amount, **save their time compared to manual entry, and it also able to handle multiple receipts in a single scan**. However, few users noticed that the **scanning speed could be improved**, especially when processing single receipts.

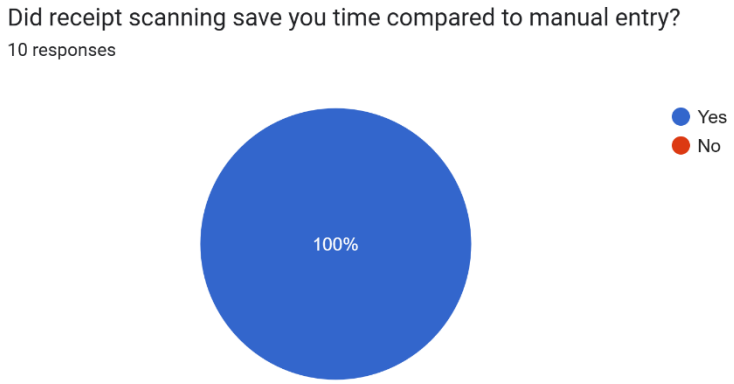


Figure 6.3.1 User Rating on Whether Receipt Scanning Saved Time Compared to Manual Entry

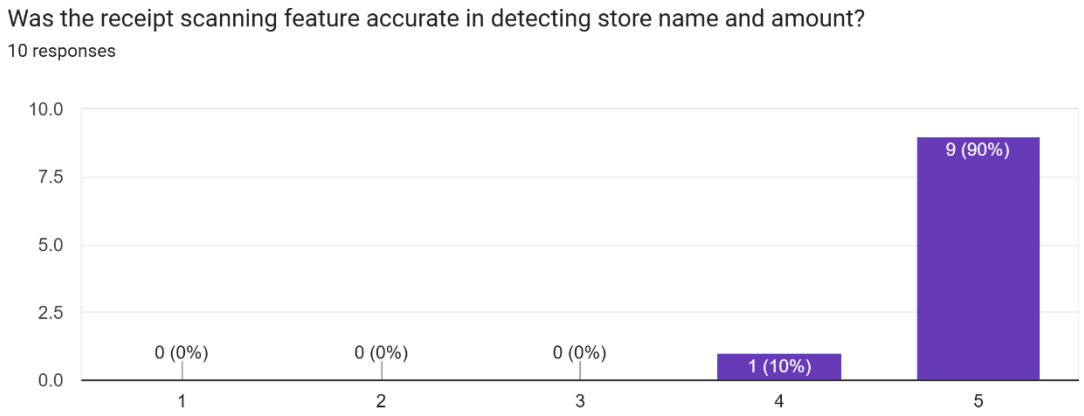


Figure 6.3.2 User Rating on the Accuracy of Receipt Scanning in Detecting Store Name and Amount

Core features such as **budget setting and transaction management** were highly praised. As shown in Figure 6.3.3, **90% of users rated budget setting a 5 out of 5, with the remaining 10% rating it a 4**. This gave the feature received a high average score of 4.9, indicating that users found the process to be straightforward and effective. Additionally, several participants commented positively, with one point out that the feature was "all good," which reflected how user-friendly it was overall.

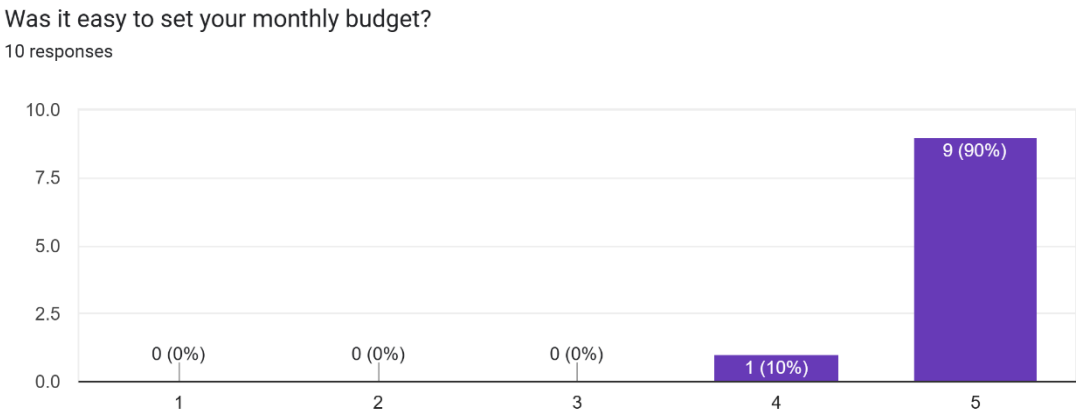


Figure 6.3.3 User Ratings for Monthly Budget Setting

The rewards system was also widely accepted. As shown in Figure 6.3.4, **90% of users had given the rewards system a rating of 5 out of 5, while 10% gave it a 4, resulting in an average rating of 4.7**. Many respondents mentioned that this feature had successfully motivated them to manage their spending more wisely. For example, one of the users mentioned, *"Spin reward, able to earn different prizes, good, motivate me to spend wisely every month."* Another simply selected it as their favourite feature, highlighting its significance in maintaining user engagement.

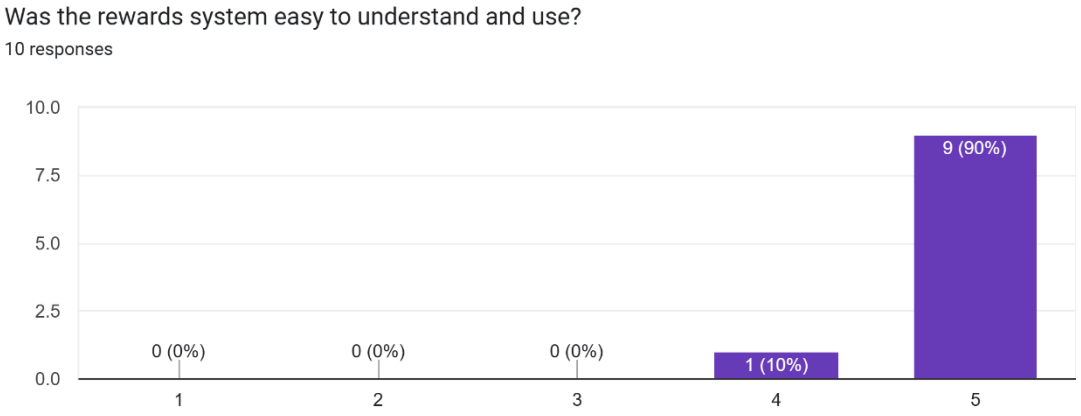


Figure 6.3.4 User Ratings for Rewards System

Figure 6.3.5 shows that most of the users found the **charts and insights was helpful**, which enable them to understand their spending patterns easily. Specifically, 90% of respondents had given a rate of **5 out of 5** for this feature, with the remaining 10% giving it a rating of 4, resulting in an overall strong positive response. This also prove that the visual representation of expenses and incomes was helpful for users to monitor their financial habits. However, one respondent had pointed out a **performance issue**, which is the **chart loading speed**, he mentioned about, *"It would be great if the charts could load faster, especially for users with many transactions."* This implies that even though the chart feature is highly valued, but optimizing their loading speed could further enhance the user experience. Hence, we had performed some adjustments, and the **average loading time was reduced from approximately 40 seconds to 12 seconds**, resulting in a smoother and more responsive interaction.

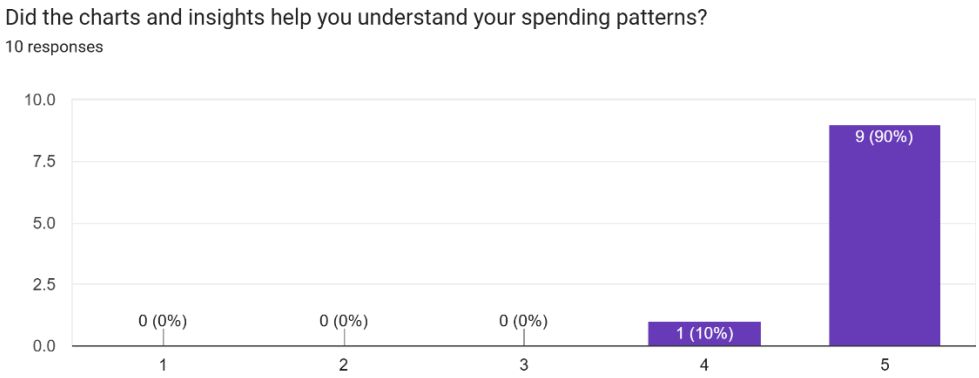


Figure 6.3.5 User Rating on the Helpfulness of Charts and Insights in Understanding Spending Patterns

AI tips also received positive feedback, with few participants had chosen it as their favourite feature. Figure 6.3.6 had shown that **80% of respondents gave a rate of 5 out of 5 towards the AI tips were relevant to their spending habits and local weather, whereas the remaining 20% rated them between 3 and 4**, which proved that the **app is generally accurate** in providing useful recommendations to users.

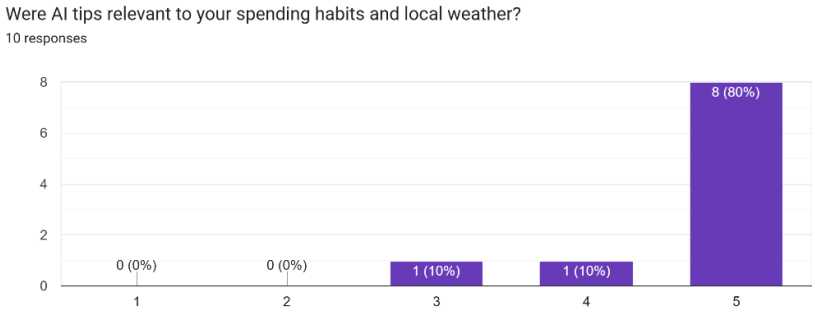


Figure 6.3.6 User Rating on the Relevance of AI Tips to Spending Habits and Local Weather

Besides that, users had also highlighted the usefulness of meal suggestions and money-saving advice, but there are few recommended improvements for this feature, such as **make the tips more varied, less repetitive, and more context-aware**. In response to this feedback, we implemented improvements to **reduce repetition by ensuring that food suggestions are not repeated within a two-week period**. Further, Figure 6.3.7 had shown that the majority of users found the AI helpful and expressed interest in additional features. For example, some suggested that the app could integrate location data to provide relevant suggestions, such as affordable meal options nearby, as well as the accuracy of weather-based advice should be improved. Furthermore, users recommended that the app should incorporate location data to provide relevant suggestions, such as affordable meal options nearby, and also improve the accuracy of weather-based advice.

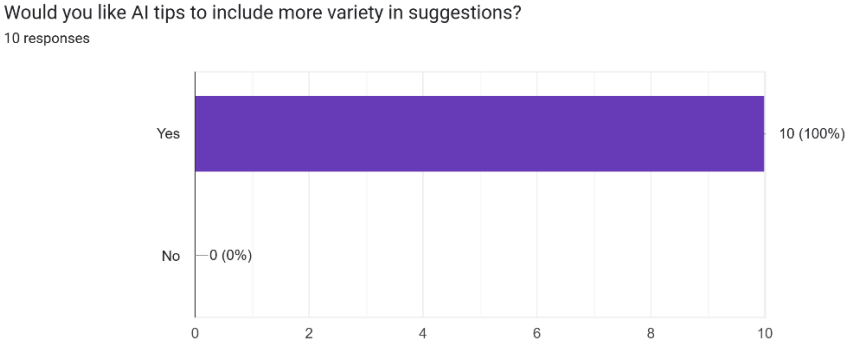


Figure 6.3.7 User Rating on Desire for More Variety in AI Suggestions

As shown in Figures 6.3.8 and 6.3.9, the **speed and responsiveness** of apps had received an average rating of **4.5 out of 5 for performance and stability**. However, there were respondents facing difficulties during receipt scanning, while the majority of users did not encounter issues. One user commented, "*Improve stability as the category analyzed by AI for receipt scanning is sometimes weird,*" while another noted, *"The receipt scan is a bit slow for a single piece of receipt.*" Additionally, some participants had recommended to improve the loading speed of charts and the scanning process, especially when handling larger sets of data. Following this feedback, we had performed improvements on the app, and the app is now able to scan and categorize receipts across all categories more accurately and the overall speed of receipt scanning had improved.

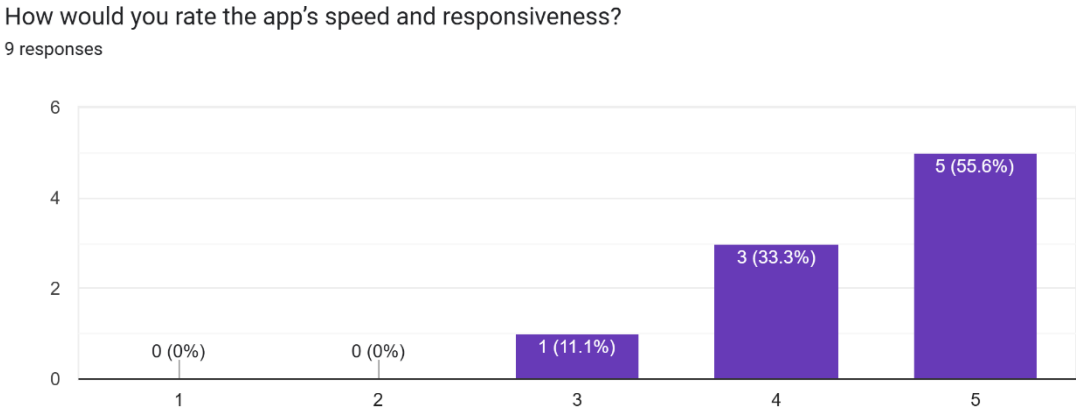


Figure 6.3.8 User Ratings of App Speed and Responsiveness Ratings

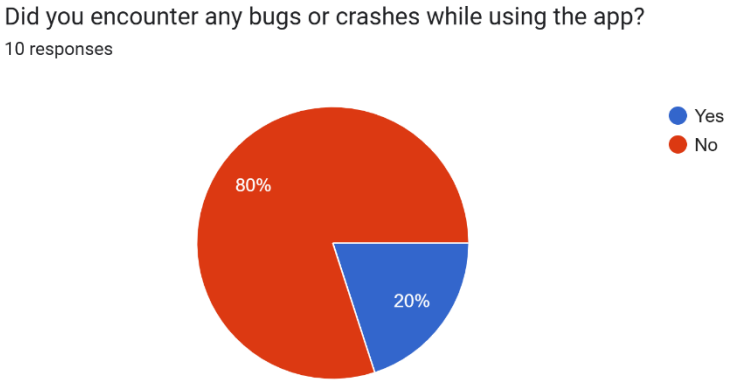


Figure 6.3.9 User Ratings of Bug Reports (Yes vs. No)

Figures 6.3.10 and 6.3.11 show that the **overall user experience** with the Budgie app was very positive. As users gave an **average rating of 4.6 out of 5** for their **first impression of the app**, and the same score was given for the application **ease of navigation**. The majority of respondents commented that the interface was intuitive and straightforward, which suggests that the design decisions were effective in facilitating usability.

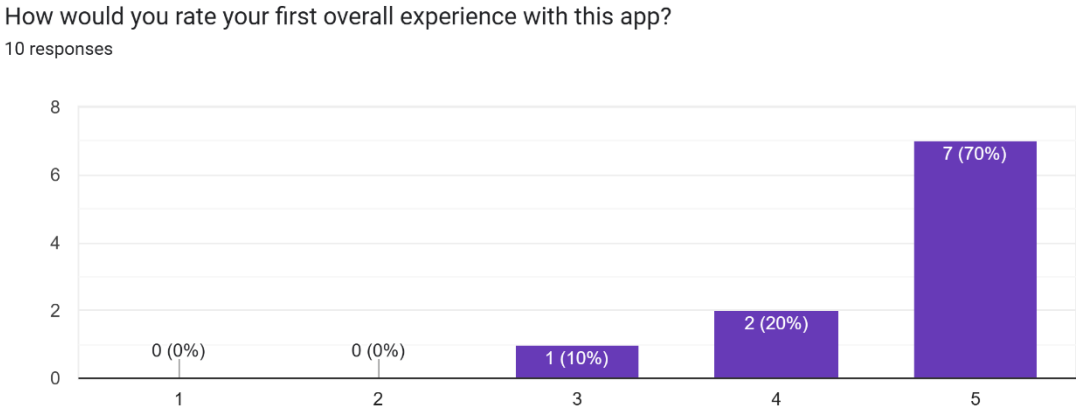


Figure 6.3.10 User Ratings of First Overall Experience with the App

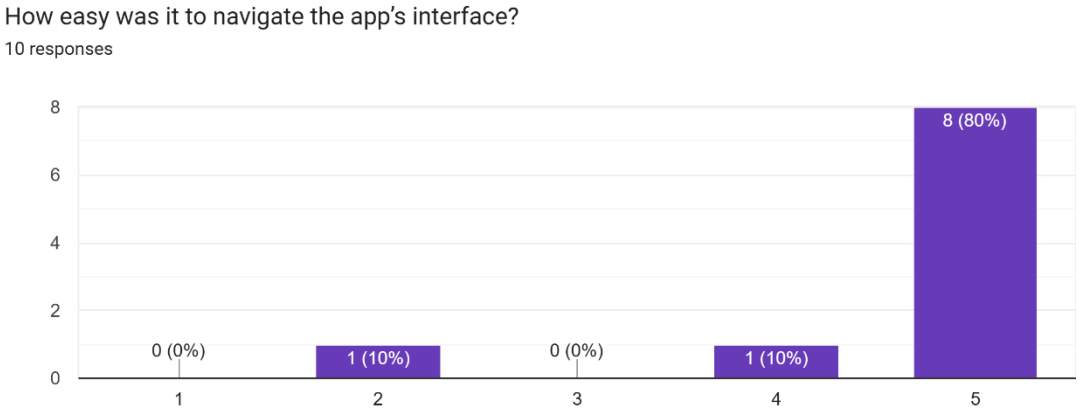


Figure 6.3.11 User Ratings of Ease of Navigating the App's Interface

Figure 6.3.12 presents the **favourite features** selected by users, which including the receipt scanning, the rewards system, AI tips, and budget tracking features. However, the **rewards system gained the highest number of selections**. This shows that even though the app had provided many useful functions, the rewards system is the most engaging and appealing feature to users.

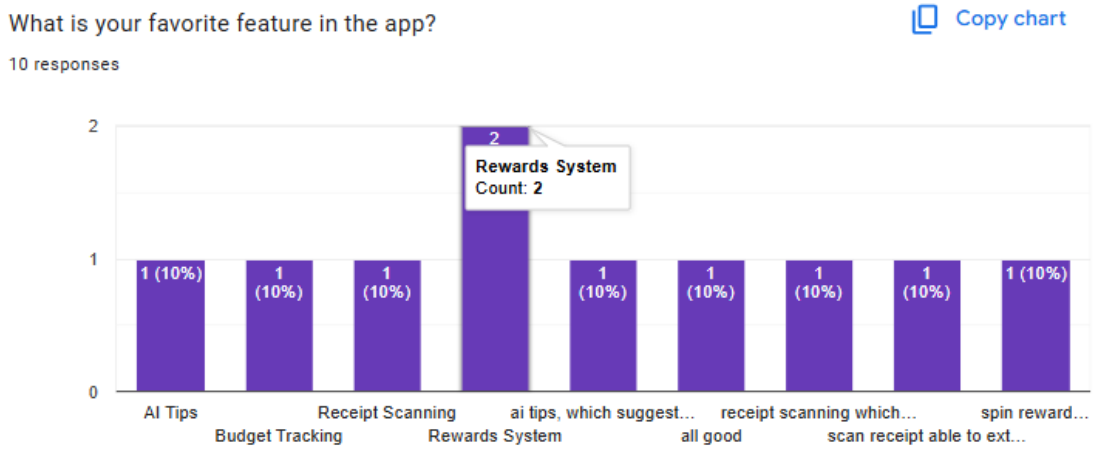


Figure 6.3.12 Distribution of Users' Favourite Features in the App

Overall, the survey results showed positive feedback from users, which indicate that this personal finance management application had meet the users' needs and reducing their difficulties in managing finances. However, there are few improvements that had suggested by respondents could not be implemented within current development timeline and may be carry out in the future. These include enhancing chart loading performance, adding search and filtering options in transaction management, support for income attachments, introducing multi-currency support, and providing smarter AI suggestions. But, the majority of improvement requests were not about usability flaws but rather about enhancing performance and extending functionality. This had proved that the app has established a solid foundation, with opportunities to grow further based on user needs.

6.4 Project Challenges

During the development of this Personal Financial Management Application with Spending Monitoring, I had faced several challenges. As this application had been developed by using the Dart programming language and the Flutter framework, and this is my first time dealing with this language. I had spent a lot of time learning from scratch for the state management, the structure of Flutter and the widget system.

Initially, I planned to create a cross-platform software for both iOS and Android. But, unfortunately due to hardware limitations, I only had a HP laptop,which could only develop the application that was able to be supported by Android. Besides that, I also faced lagging issues when running my application on the virtual phone in Android Studio. Hence, in order to ensure smoother performance, I had switched the testing process to a real device.

Another challenge faced was the implementation of the receipt scanning functions into this application. In order to analyze the receipt data, I first tried utilizing the Firebase ML Kit, however the result of text extracted from the receipt was inaccurate and took a long time to process an image. Hence, I switched to the Google Cloud Vision API, then connected it with OpenAI GPT to analyze the receipt data. At the same time, compatibility issues also arise, as my Flutter's version was incompatible with the API systems version. Hence, I managed to fix it by upgrading dependencies and modifying the code. And lastly for storage of users' details, this is my first-time deal with the Firebase Firestore, hence I spent some time learning and lastly successfully stored all user data into the database.

6.5 Objectives Evaluation

In this section, we evaluate the extent to which the objectives of this project have been achieved. Table 6.5.1 to Table 6.5.4 had shown the detailed analysis for each objective, which includes the expected outcomes, achieved outcome, evaluation criteria, and results.

Table 6.5.1 Objective 1: To develop a user-friendly expenditure monitoring application that simplifies financial management.

Objective 1: To develop a user-friendly financial monitoring application that simplifies financial management.	
Expected Outcome	The application needs to provide an easy-to-use interface which allows users to manage their finances with minimal effort and difficulty.
Evaluation Test	Functional Testing and User Feedback Surveys
Achieved Outcome	The application's interface has been designed to be straightforward and easy to use, with features such as buttons with icons that are identifiable, a simple layout with easily readable font styles, and well-organized sections for tracking expenses and income. To enhance readability and lessen eye strain, we applied bright, contrasting fonts on a black background. Additionally, features like quick-add transaction buttons at the bottom navigation bar make it more accessible for users at different learning levels.
Result	The objective has been successfully fulfilled. From the testing, positive feedback from test users had been received, with most rating it as simple to use and effective for tracking their expenditures and income.

Table 6.5.2 Objective 2: To manage and organize the receipts by implementing receipt scanning feature, eliminating manual entry and human errors.

Objective 2: To manage and organize the receipts by implementing receipt scanning feature, eliminating manual entry and human errors.	
Expected Outcome	The receipt scanning feature should automatically capture and extract the transaction data, which includes the store name and amount, and then further analyze the category of expenses from receipt images, eliminating the need for manual data entry. Besides that, this application should be able to handle multiple transaction images or single images with multiple transactions smoothly.
Evaluation Criteria	Functional Testing and User Feedback Surveys
Achieved Outcome	The receipt scanning feature had been implemented to handle various kinds of receipts, including paper receipts and e-receipts that appeared in different formats. Hence, OCR technology extracts data like the store name and expense amount from receipts, and then the category of expenses is analyzed by OpenAI GPT. Additionally, this application is able to handle multiple transaction images or single images with multiple transactions, which eliminates users' waiting time for transactions to be processed. The system has significantly reduced manual entry and the potential for human error.
Result	The objective has been successfully fulfilled. From the testing, positive feedback from test users had been received; they rated the accuracy in extracting data as high. And users feel that this feature has saved their time and lowered the chances of human errors in data entry.

Table 6.5.3 Objective 3: To enhance user engagement and motivate them to spend wisely and stick to their budgets by introducing gamified rewards for meeting savings goals and daily check-in.

Objective 3: To enhance user engagement and motivate them to spend wisely and stick to their budgets by introducing gamified rewards for meeting savings goals and daily check-in.	
Expected Outcome	To motivate users to spend wisely and enhance user engagement, the application needs to introduce a points-based reward system.
Evaluation Criteria	Functional Testing and User Feedback Surveys
Achieved Outcome	The application successfully introduced a points-based reward system to motivate users to spend wisely and record their transactions consistently. For example, the user may use the points to spin a reward wheel to earn extra points or a TNG reload pin. Furthermore, points will be awarded when users stay within their budget. Besides that, daily check-in rewards will also be implemented to encourage user engagement.
Result	The objective has been successfully fulfilled. From the testing, positive feedback from test users had been received; they rated the rewards system as having successfully motivated them to spend wisely and check in to the app consistently to record their transactions. And users feel that this feature's gamification aspect makes budgeting more fun.

6.6 Concluding Remark

In this chapter, we had carried out a comprehensive review toward the system via a variety of testing procedures, such as functional testing, performance metrics, and user feedback. Important features like user login or sign-up, transaction management, receipt scanning, and a points-based reward system had been undergoing both the functional testing and user feedback survey. Throughout the surveys, we had gained valuable insights into the app's usability and effectiveness. Besides that, the difficulties encountered throughout development were discussed, and each project objective was evaluated based on the achieved outcomes. Overall, the system has achieved its objectives and been implemented successfully, with users showing positive engagement and satisfaction with the features provided by this application.

CHAPTER 7 CONCLUSION AND RECOMMENDATION

7.1 Conclusion

This Personal Financial Management Application with Spending Monitoring is developed to assist users track their income, expenditures, budgeting, and spending behaviour. In contrast to many existing finance apps, this application reduces manual data entry through the integration of AI-based spending categorization with receipt scanning. The application aims to provide a simple, attractive, and visually friendly user experience, which is suitable for a wide range of users, from students to adults who would like to take control of their spending and manage their finance.

A cross-platform user interface (UI) has been developed for the system by using Flutter, and it integrates to Firebase Firestore and Firebase Authentication for the objective of recording data in real time and enabling a secure login. In addition, Google Cloud Vision API is also used for OCR receipt scanning, whereas OpenAI GPT helps analyze the scanned text and categorize the expenses. Hence, by scanning either the receipt or transaction details, this feature saves user time, streamlines the process, and make the expense tracking process more accurate. Besides, with a Layered design and Client-Server architecture, this application able to separate the data management, logic, and user interface in order to enhance the maintainability and scalability. The combination of these frameworks provide supports for real-time synchronization, integration of third-party APIs easily, and efficient data flow.

Further, this application also introduces new functionalities such as AI tip suggestions, recurring income automation, transaction charts, budget monitoring, and a point-based reward system. Hence, user motivation and accessibility had been improved through features like daily check-in rewards, point-based rewards, and monthly rewards for keeping expenses within budget. To conclude, this personal finance management application with spending monitoring not only simplifies personal finance management while also providing a fun and interactive way for users to develop healthier spending habits.

7.2 Recommendation

According to the functional testing carried out on the system as well as the survey responses from users who tested the application, both results clearly show that the current application meets the daily financial management needs of most users. Core features including both expenses and incomes tracking, budget monitoring, point-based reward system, and AI-powered savings tips have been well received, which had built a solid foundation for personal finance management. These features ensure that users able to record and monitor their transactions with minimal difficulty or effort, while also gaining useful guidance that can assist them in developing better financial habits.

For future improvement, one area of enhancement could be the **automatic recording of transactions directly from phone notifications, SMS, or emails**. For example, banks usually send transaction details via SMS, push notifications, or email, while e-wallet providers typically send transaction details via in-app notifications or email. Hence, the application could detect and capture these transaction data in the background, then automatically record and categorize transactions. This would be eliminating the user to manually enter or scan receipts, and improve the accuracy significantly, while also ensure that their financial records are always up to date. However, this feature is currently unable to implement, as the banks and e-wallet providers generally prohibit third-party applications from directly accessing or "listening" to transaction notifications in order to safeguard user privacy and security. With proper collaboration or official API integration in the future, this functionality could be investigated to further improve the overall user experience.

Another useful enhancement is to **expand the AI suggestions to become more varied and context-aware**, such as the application can **integrate location data**, in order to **suggest affordable nearby meal options**. Hence, users able to discover nearby food easily and select that food that within their budget. Besides that, the AI could also provide personalized savings strategies based on each user's income, spending, and goals. In addition, expense forecasting predicts future spending trends based on past data, which allows users to anticipate cash flow problems and adjust in advance.

Another potential improvement is the introduction of **voice input support** for adding transactions. This feature able to help users record expenses or income quickly by speaking. For example, a user may simply say “Lunch at KFC, RM12” and the application would automatically record down the expense. This would increase accessibility for visually impaired users, or those users are on the move or unable to type, help them make expense tracking more convenient.

REFERENCES

[1] MX Technologies, Inc., “What is PFM? Benefits & Use Cases of Personal Financial Management,” mx.com., 23 February 2024. https://www.mx.com/blog/what-is-pfm/#

[2] Peoples Security Bank & Trust, “Why Is Personal Finance Important?,” psbt.com., 25 September 2024. https://www.psbt.com/Learn/Resources/PSBT-Corner-News/Why-Is-Personal-Finance-Important

[3] “Expense Report Software | Manage your company spend,” Expensify, 2023. https://use.expensify.com

[4] Expensify, Inc., “Receipt Scanning App,” use.expensify.com. https://use.expensify.com/receipt-scanning-app

[5] “Expense Reports - Create, submit, approve - Automated expense reporting,” Expensify - Expense Management, 2023. https://use.expensify.com/expense-reports

[6] “Best travel and expense management software - Zoho Expense,” Zoho Corporation Pvt. Ltd., 2024. https://www.zoho.com/expense/

[7] “Receipt Tracking Software - Receipt Management | Zoho Expense,” Zoho Corporation Pvt. Ltd., 2024. https://www.zoho.com/expense/receipt-tracking/

[8] “Manage per diem easily with Zoho Expense,” Zoho Corporation Pvt. Ltd., 2024. https://www.zoho.com/expense/per-diem/

[9] “Expense Report Management | Zoho Expense,” Zoho Corporation Pvt. Ltd., 2024. https://www.zoho.com/expense/expense-reports/

[10] "Budget Management | Zoho Expense," Zoho Corporation Pvt. Ltd., 2024. https://www.zoho.com/expense/budgets/

[11] "Money Manager & Budget Planner | Spendee," SPENDEE, 2024. https://www.spendee.com/

APPENDIX

Poster