Food Waste Mobile Application with Gamification

By

Chai Cun Lin

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 2024

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Food Waste Mobile Application with Gamification

Academic Session: June 2024

I CHAI CUN LIN

(CAPITAL LETTER)

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

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

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

Verified by,

(Author's signature)

(Supervisor's signature)

Address:
E57 Jalan Dua
Taman Bukit Bidor,
35500, Perak.

Tseu Kwan Lee
Supervisor's name

Date: 13/09/2024

Date: 13/09/2024

Universiti Tunku Abdul Rahman			
Form Title : **Submission Sheet for FYP**			
Form Number: **FM-IAD-004**	Rev No.: **0**	Effective Date: **21 JUNE 2011**	Page No.: **1 of 1**

FACULTY/INSTITUTE* OF Information and Communication Technology

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 13/09/2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that ***Chai Cun Lin*** (ID No: ***20ACB03887***) has completed this final year project/ dissertation/ thesis* entitled "***Food Waste Mobile Application With Gamification***" under the supervision of ***Tseu Kwan Lee*** (Supervisor) from the Department of ***Computer Science***, Faculty/Institute* of ***Information and Communication Technology*** , and ***Luke Lee Chee Chien*** (Co-Supervisor)* from the Department of ***Information and Communication Technology***, Faculty/Institute* of ***Computer Science*** .

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

Yours truly,

Chai Cun Lin

DECLARATION OF ORIGINALITY

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

Signature : ____________________

Name : Chai Cun Lin

Date : 13/09/2024

ACKNOWLEDGEMENTS

I am deeply grateful to my project supervisor, Ms Tseu Kwan Lee, for her invaluable guidance and the incredible chance to participate in the Sustainability initiative. This project marks a significant milestone in my journey to develop innovative solutions for food waste. My heartfelt gratitude goes out to you.

To all my companions, for your enduring patience, unwavering support, and affection, and for being my pillars of strength during challenging moments. Lastly, I owe a world of thanks to my parents and family for their boundless love, backing, and constant motivation throughout my academic pursuits.

ABSTRACT

Food waste is a global problem that poses serious environmental and social challenges. Supermarkets are one of the major sources of food waste, as they often discard large amounts of surplus or near-expiry food products. To address this issue, this project aims to design and develop a gamified mobile application that can reduce food waste in supermarkets by engaging both the supply and demand sides. The target users of the application are supermarket managers and customers who are interested in saving money and the environment. The application uses gamification elements such as points, badges, challenges, and rewards to motivate and educate users to adopt sustainable food consumption and disposal habits. The application also provides a management system for supermarkets to monitor and manage their food products, and to offer incentives and discounts for users who buy or donate the surplus or near-expiry food items. The application differs from other existing solutions by integrating both the supply and demand sides, and by using a user-centred and participatory design approach. The project also creates and validates a mid-fidelity prototype of the application through usability testing and feedback sessions with the potential users. The results show that the application is easy to use, useful, and enjoyable, and that it can increase users' awareness and intention to reduce food waste. The project contributes to the field of food waste management by proposing an innovative and engaging solution that addresses the problem from both the supply and demand perspectives.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GHG	Greenhouse Gas
CO_2	Carbon Dioxide
SDGs	Sustainable Development Goals
UNEP	United Nations Environment Programme
BSF	Black Soldier Fly
UX	User Experience
FLW	Food Loss and Waste
TAM	Technologies Acceptance Model
UI	User Interface
IDE	Integrate Development Environment

CHAPTER 1 Introduction

The issue of food waste is vital for sustainability and resource conservation. Many people are aware of the problem, but it is not adequately tackled in various aspects. It leads to serious moral, environmental, and economic challenges. Around the world, each year 931 million tons of food are wasted by households, retail businesses, and the food service sector [1] and one-third of the food that is still edible for humans is wasted, while over 783 million people suffer from hunger [2]. Moreover, the climate change crisis is worsened by food waste, which has a large impact on greenhouse gas (GHG) emissions. Food production, transport, and handling produce a lot of Carbon Dioxide (CO_2), and when food goes to landfills, it creates methane, a more powerful greenhouse gas[3]. Reducing food waste is important for achieving the Sustainable Development Goal 12.3 (SDG 12.3) which is to reduce global food waste in both retail and consumer settings by 50% by the year 2030. Additionally, the aim is to minimize food losses, especially those occurring after the harvest, throughout supply chains[4].

To reduce food waste, various applications have been developed, but most of them target the end consumer rather than the supply chain such as supermarket. These applications aim to reduce food waste by offering discounts, donations, and exchanges surplus food items [5]. However, they do not address the food loss that occurs in supply chain such as food products reach expire date. To address this problem, we need innovative and engaging approach that involves educating people, changing their behaviour, and encourage a culture of sustainability. However, conventional methods of communication and awareness have not been very effective in influencing long-term behaviour change[6], because it may not provide sufficient feedback, reinforcement, or monitoring to sustain behaviour change over time. People may need ongoing encouragement, reminders, incentives, or accountability to keep up their motivation and commitment to change their behaviour. Gamification approach has potential, it applies game concepts and designs to non-game settings to attract users, inspire them, and influence their behaviour[7]. By adding gamification elements such as points, levels, badges, rewards, and challenges to everyday activities, gamification taps into human's natural desire for achievement and competition.

1.1 Problem Statement and Motivation

According to the UNEP Food Waste Index Report 2021 [8], this amounts to 931 million tons of food waste, of which 13% came from retail, therefore reducing the food waste in the retail sector also important. But the existing food waste gamification mobile application are not focused on the food supply chain, the management system of the mobile application is not applicable for the user of food supply chain to manage their food products[9]. For example, the supermarket only provides the food product lists for the consumer but cannot manage to address the food products that expire soon, this causing food waste because consumer not willing to buy a food product that almost expired. This limits the ability of the supermarket to monitor and reduce its food waste, as well as to offer incentives and rewards to the consumers who buy the surplus or near-expiry food items. Therefore, there is a need for a mobile application that can address the food waste problem from both the supply and demand sides and use gamification techniques to motivate and engage the users.

The motivation behind the food waste gamified mobile application is rooted in the urgent need to address the global issue of food waste and its far-reaching consequences. The application seeks to inspire a fundamental shift in user behaviour by leveraging the principles of gamification to make the process of reducing food waste not only effective but also enjoyable. In a world where individuals are constantly bombarded with information and stimuli, the application aims to cut through the noise and capture users' attention through engaging content, interactive challenges, and a visually appealing interface.

The motivation is driven by a recognition of the critical role individuals play in contributing to a sustainable and eco-friendly food ecosystem. By instilling a sense of responsibility and empowerment, the application aims to make users active participants in the fight against food waste. The integration of game mechanics, such as rewards, levels, and challenges, is carefully designed to create a motivating and dynamic experience that sustains user interest over the long term.

Moreover, the application is motivated by the desire to educate users about the environmental impact of food waste. By providing accessible and engaging information, the application aims to raise awareness and promote informed decision-making regarding food consumption and disposal. This educational aspect is crucial for building a deeper understanding of the issue and inspiring users to make meaningful changes in their daily lives.

In essence, the motivation for the food waste gamified mobile application is to transform the typically mundane task of reducing food waste into a compelling and rewarding experience. By making the process enjoyable, educational, and socially impactful, the application seeks to motivate users to embrace sustainable habits, contributing collectively to a significant reduction in global food waste and its associated environmental and societal impacts.

1.2 Research Objectives

A comprehensive review of the existing gamification mobile applications for reducing food waste is essential for developing a successful and innovative solution in this domain. This study can help us identify the weak point, opportunities, and new ideas from the current applications, improvement and apply them to the design. The goal is to leverage the strengths of the existing solutions, and add new gamification elements to our application, to boost user engagement and impact.

Apply gamification can increase user engagement and motivate them to actively participate the food waste activities. Adding game-like elements such as points, leaderboards, and challenges are motivating the users to actively engage and achieve their goals. The use of collections, such as Firebase Storage, Cloud Firestore crucial for efficient data management and organisation. This helps to quickly locate and navigate through large datasets, improving user experience. By integrating these factors, the solution becomes more appealing, user-friendly, and effective, which enhances the end-users' satisfaction and performance.

The mobile application should be designed for multi-market to use it. It may increase the effectiveness of reduce food waste because increasing the user base and the potential revenue of the application by reaching more customers and markets. By designing the application for multi-market, the application can appeal to a wider and more diverse audience, which can increase the number of downloads, registrations, and active users of the application. Additionally, it also improving the user experience and satisfaction of the application. More users can give more feedback to enhance the application since can more understand the user needs.

The user interface will need to be improved, the gamification elements will need to be fine-tuned, and the application will need to be customized to the needs and preferences of the users. Therefore, usability testing is crucial to ensure that the gamification and user interface are no usability issues such as difficulty in interacting with the mobile application. Usability testing can improve the user satisfaction and the performance of the application by get feedback from the users.

Objectives:

- To study existing food waste gamified mobile applications in the domain.
- To propose gamification elements, and data management in address food waste problem.
- To develop a gamification mobile application, implement engaging methods to motivate users and design for multi-market.
- To evaluate the proposed solution with usability testing in gamification elements and UI.

1.3 Project Scope and Direction

The food waste gamification mobile application aims to address the food waste problem by engaging both users and the supermarket. The application will be encouraging users to adopt sustainable practices, make informed decisions, and actively participate in reducing food wastage.

In the scope of this project, the food waste gamification mobile application involves several critical components that highlight its technical complexity and innovation. At the core of the application is the user registration and profile management system, where users create accounts and manage personal data. This requires secure authentication mechanisms, such as Firebase Authentication, to ensure the integrity and privacy of user data. Information such as profile details, game progress, and earned rewards is stored in a database such as Firebase Firestore, which needs to be efficiently structured to handle large volumes of user and supermarket data. Indexing and data normalization play key roles in ensuring efficient data retrieval and query performance, particularly as the application scales to support multiple users and markets.

The application heavily incorporates gamification elements to engage users and motivate sustainable behaviour. Features such as points, rewards, and challenges are built on algorithms that calculate user achievements and progress in real-time. These gamification elements are integral to tracking user behaviour, including tasks like identifying expiring food products or participating in sustainability challenges. To enhance user interaction, a leaderboard system is implemented, allowing users to see their rankings and share them on social media. From a technical standpoint, this involves managing real-time data updates and handling concurrency to ensure the leaderboard reflects the latest scores without delays.

In terms of system architecture, the application is designed to support multiple supermarkets, each able to manage their inventory and track food products nearing expiration. This requires building a modular system with role-based access controls to ensure different types of users—supermarket managers, customers, and administrators—have appropriate permissions to view and modify data. Furthermore, the architecture must be scalable and performant, ensuring that the application can handle an increasing number of users and transactions without compromising speed or reliability.

The game challenges within the application, such as rewarding users for finding expiring products, require algorithms for point system such that how the users to be gaining the point or losing point when finding the expiring soon products. The UI/UX design of the app plays a crucial role in making the gamification engaging and intuitive for users. With UI frameworks like Flutter, the app's interface is designed to be seamless and responsive across multiple platforms, providing an immersive experience that includes visual and auditory feedback for earned points, and other achievements.

Lastly, the application contributes to sustainability by utilizing data analytics to track and evaluate user behaviour. By analysing patterns in how users purchase or donate nearly expired food products, the app can provide insights into how much food waste has been reduced. This data-driven approach helps the application align with SDG 12.3, which focuses on halving global food waste by 2030. In summary, the food waste gamification mobile application brings together database management, gamification elements, system architecture, and data analytics to create a comprehensive and scalable solution for reducing food waste in supermarkets.

1.4 Contributions

The food waste gamified mobile application contributes significantly to addressing the global challenge of food waste through a multifaceted approach. Its primary motivation lies in inspiring positive behavioral change by incorporating engaging content, challenges, and rewards to encourage users to adopt sustainable practices. Also, the application contributes to behavioral change and awareness by fostering a sense of responsibility and providing personalized feedback on users' food consumption and disposal habits. Furthermore, it actively contributes to environmental impact and sustainability by motivating users to minimize individual food waste. The application serves as an educational tool, empowering users with knowledge about the consequences of food waste, and its community-building features amplify the social impact of individual actions. The application also provides a management system for supermarkets to monitor and manage their food products, and to offer incentives and discounts for users who buy or donate the surplus or near-expiry food items.

1.5 Report Organization

The report is structured into five key chapters, each fulfilling a unique role within the project framework. **Chapter 1** sets the stage by providing an overview that includes background context, a clear problem statement, the driving motivations, the aims of the project, a delineation of its scope, and an acknowledgment of its contributions.

Chapter 2 offers an in-depth literature review, structured in two main parts. The first segment presents a comparative evaluation of current systems, analysing their capabilities and limitations. The second segment reviews critical languages pertinent to the project's context.

In **Chapter 3**, the focus shifts to the methodologies and processes implemented in the project's system. This chapter details the strategic approach employed during the development phase.

Chapter 4 documents the early phases of the project, outlining the setup procedures and the initial results that were obtained from these foundational activities.

Concluding the report, **Chapter 5** provides a summarizing overview, weaving together all aspects of the project to present the final discoveries and their broader significance. This chapter offers a cohesive wrap-up of the project's journey and its outcomes.

CHAPTER 2 Literature Reviews

2.1 A Food Waste Mobile Gamified Application Design Model using UX Agile Approach in Malaysia

The paper [10] presents the design and development of a food waste mobile gamified application that aims to improve the collection and decomposition of food waste using black soldier fly (BSF) treatment in Malaysia. The application was created using the agile UX approach, which involved various stakeholders such as households, restaurants, collectors, and BSF farms in the requirement and design phases. The application also incorporated gamification elements such as points, levels, ranks, and rewards to motivate and engage the users to participate in the food waste disposal process. The paper also describes the creation and validation of a high-fidelity prototype of the application through usability testing and feedback sessions with the potential users. Finally, the paper proposes a system design model that consists of five components: motivation, stakeholder, systematic and user-friendly, job opportunity, and eco-friendly approach. This model can serve as a guideline for developing similar applications for food waste management.

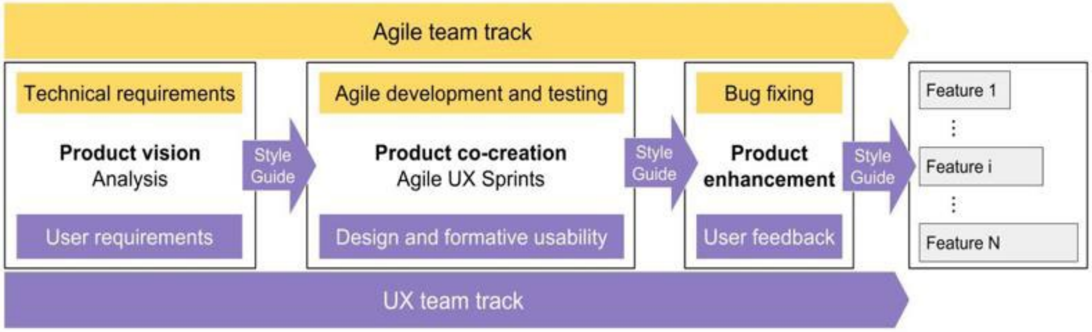


Figure 2.1.1 Agile UX Development Lifecycle

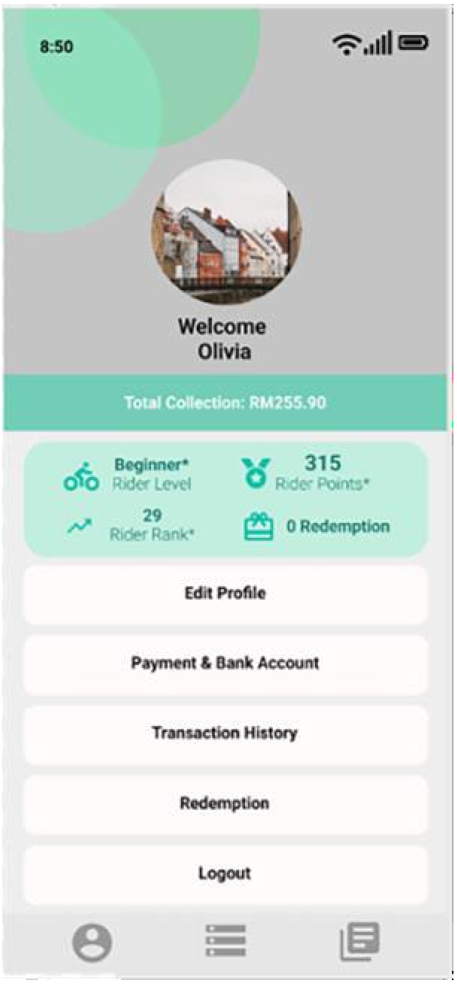


Figure 2.1.2 Rider's Gamification

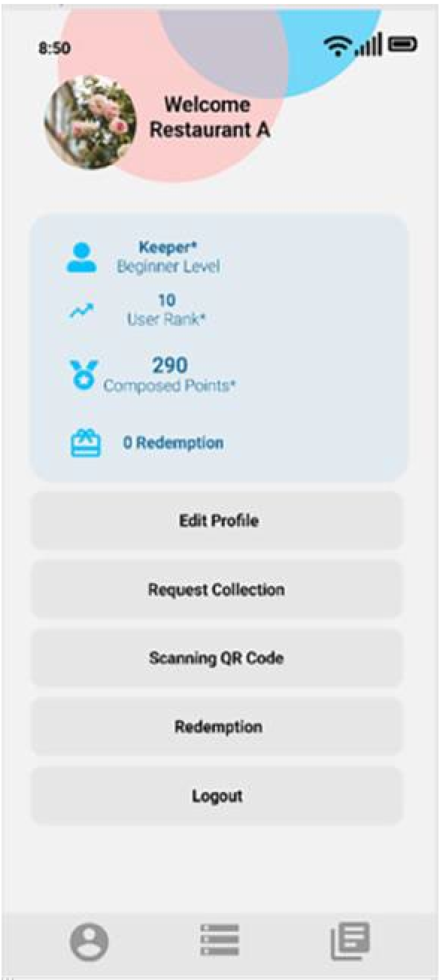


Figure 2.1.3 Household/Restaurant Gamification

2.1.1 Strengths of Previous Studies

In this paper is the Agile UX methodology, which is a hybrid approach that combines agile software development and user experience design. It allows for active involvement and collaboration from all stakeholders, such as developers, designers, researchers, and users, throughout the development process. Also, enables rapid and iterative design and testing of the application, which can improve the quality and usability of the product and reduce the risk of errors and rework. Other than that, incorporates gamification elements to motivate and engage users in the food waste management process, which can enhance their awareness and behavior towards environmental sustainability.

2.2 Employing Gamification to Support Sustainable Food Consumption, Analysis and Redesign of the Too Good To Go Mobile App

The paper [11] explores the potential of gamification in the design of apps intended to support more sustainable food consumption behavior. The paper is centered on the existing Too Good To Go app, which aims to reduce food waste by enabling retailers to sell their surplus food at a reduced price. The paper examines current user behavior and perception through quantitative and qualitative methods, such as Instagram polls, online surveys, diary studies, and interviews.

The researcher employs design frameworks and iterative prototype cycles to redesign the app and add gamification features. The paper uses the Octalysis framework by Yu-Kai Chou to analyze the existing app and the prototypes, and to identify the core drives that motivate users to use the app. Also uses the Player's Journey model to divide the user experience into four phases: Discovery, Onboarding, Scaffolding and Endgame. The paper aims to encourage active sustainable food consumption, which means using the app more often and saving more food from being wasted.

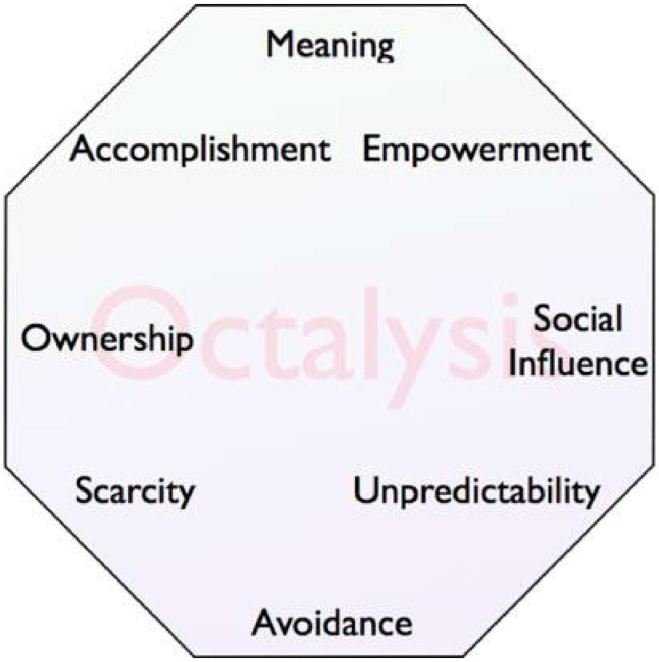


Figure 2.2.1 Octalysis Framework by Yu-Kai Chou [12]

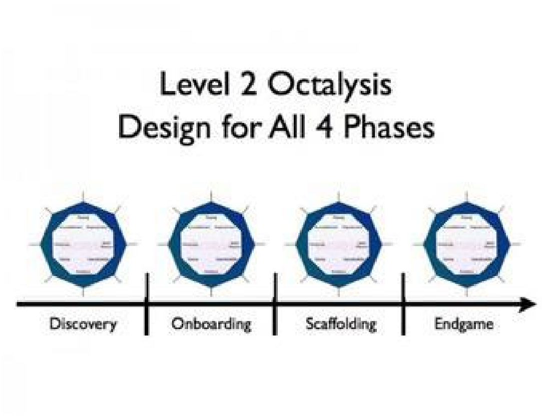


Figure 2.2.2 Player's Journey Model

The paper proposes several gamification elements to enhance the user experience and motivation, such as personalization, progression, feedback, social interaction, unpredictability, and loss avoidance. For example, users can customize their profile, avatar, and preferences; see their progress and achievements in saving food, earning points, and unlocking badges; receive feedback on their actions, such as notifications, tips, and rewards; interact with other users, such as following, messaging, and joining group challenges; experience surprises and curiosity, such as mystery bags, random rewards, and daily quests; and avoid negative consequences, such as losing points, badges, or streaks.

2.2.1 Strengths of Previous Studies

The paper uses existing models and theories, such as the Octalysis framework and the Player's Journey model, to guide the design process and evaluate the effectiveness of the gamification elements. These models help to identify the key factors that influence user behaviour and experience, such as the core drives, the stages of change, and the phases of the journey. They also help to balance the different types of motivation, such as intrinsic and extrinsic, and positive and negative, and to design for different user profiles and scenarios.

The framework applies gamification techniques, which are proven to increase user engagement, motivation, and behaviour change, especially in domains that require long-term commitment and repeated actions, such as sustainability. By using game elements such as challenges, feedback, rewards, and social interaction, the framework aims to make the application more fun, appealing, and rewarding, and to encourage users to actively participate in saving food and reducing waste.

2.3 Usage of Gamification and Mobile Application to Reduce Food Loss and Waste: A Case Study of Indonesia

The researchers in this paper have designed and developed a mobile application that uses gamification to reduce food loss and waste (FLW) in households[13], especially in Indonesia. They have followed the case study method of Cechetti et al. [14] to implement and evaluate the gamification features and the FLW management system. The main features of the application are:

- **Home**: This feature allows users to log their daily food waste level and habits, and earn points, badges, hearts, and streaks based on their performance. The lower the food waste level, the higher the rewards. The home feature also shows the user's rank and level, which are determined by the total points accumulated.

- **Pantry and Shopping List**: These features help users to keep track of their food inventory and plan their grocery shopping. Users can add, edit, and delete food items, and set expiration dates and reminders. Users can also create and manage their shopping lists based on their needs and budget. Users earn points every time they make an entry in these features.

- **Learn**: This feature provides quizzes and tips to educate users about the FLW issue and how to prevent it. Users can choose from different topics and difficulty levels and earn points for each correct answer. Users can also view their quiz history and progress.

- **Leaderboard**: This feature displays the ranking of users based on their points, badges, and levels. Users can compare their performance with other users and compete to achieve higher ranks and levels. The leaderboard also shows the user's impact on the environment, such as the amount of food, water, and money saved by reducing food waste.

The gamification elements in the application are designed to motivate and engage users to reduce their food waste and improve their food management skills. The researchers/developers have used the simplified technologies acceptance model (TAM) to evaluate the acceptance of the application and found that the perceived usefulness and perceived ease of use of the application positively influenced the behavioral intention to use the application. They have also conducted qualitative interviews with the users to collect their feedback and suggestions for improvement. The results of the interviews showed that most users found the application useful, important, fun, and easy to use, and that the gamification elements enhanced their motivation and awareness of the FLW issue.

2.3.1 Strengths of Previous Studies

The paper follows a case study method of Cechetti et al. [14] that is suitable for developing and evaluating a gamification application in a real-world context. The case study method allows the researchers to explore the users' needs, preferences, behaviours, and experiences, and to test the effectiveness and acceptance of the application in a natural setting. Moreover, uses the TAM, which is a widely used and validated model for measuring the acceptance of information systems and technologies. The TAM focuses on the key factors that influence the users' intention to use the application, namely the perceived usefulness and perceived ease of use. The TAM also provides a simple and parsimonious way to explain and predict the users' adoption behaviour. Also, incorporates qualitative interviews as a complementary method to the TAM. The qualitative interviews enable the researchers/developers to gain deeper insights into the users' opinions, perceptions, feelings, and suggestions regarding the application and the gamification elements. The qualitative interviews also allow the researchers/developers to capture the users' stories, experiences, and impacts of using the application, which may not be fully reflected by the quantitative measures of the TAM.

2.4 Limitations of Previous Studies

There is a limitation of these previous studies which is the design of framework and functionality of the mobile application are focus mainly on end consumer, and do not address the food supply chain in food waste management system, such as supermarkets.

In the paper [10], mainly focus on household and restaurants, who generate food waste and request collection services, this may limit the scope and impact of the application in achieving the goal of reducing food waste in supporting the BSF industry in Malaysia. For the paper [11], wants to resolve the problem of consumer food waste at the retail level such as in restaurants. The mobile application is not designed for the retailers, but for the consumers who use the mobile application. The management system is based on the Octalysis framework, which design for user engagement and behavior change, therefore the management system is not applicable for retailers. Moreover, the paper [13] developed a gamification mobile application that focuses on household users. The application aims to changing and reducing wasteful food habits of the households but does not address the address the food supply chain issues such as food products reach expire date, storage, or processing. A more comprehensive inclusive approach may be needed to consider the perspectives of all stakeholders along the food supply chain.

2.5 Proposed Solutions

Create a management system should have a user-friendly interface that allows different users to access and interact with the system according to their roles and permissions and the end consumers and the food supply chain both are applicable to use it. Also, the management system should involve multi-stakeholder in food supply chain including the end consumers, supermarkets, restaurants, and other relevant entities. The system is able for the food supply chain to monitor and manage their food products, provide features to address the food waste issue such as the food products expire soon. For example, end consumers can view the detail food products such as expiration dates. The Supermarkets can manage the food products based on it expire date and prioritize selling food products that are about to expire.

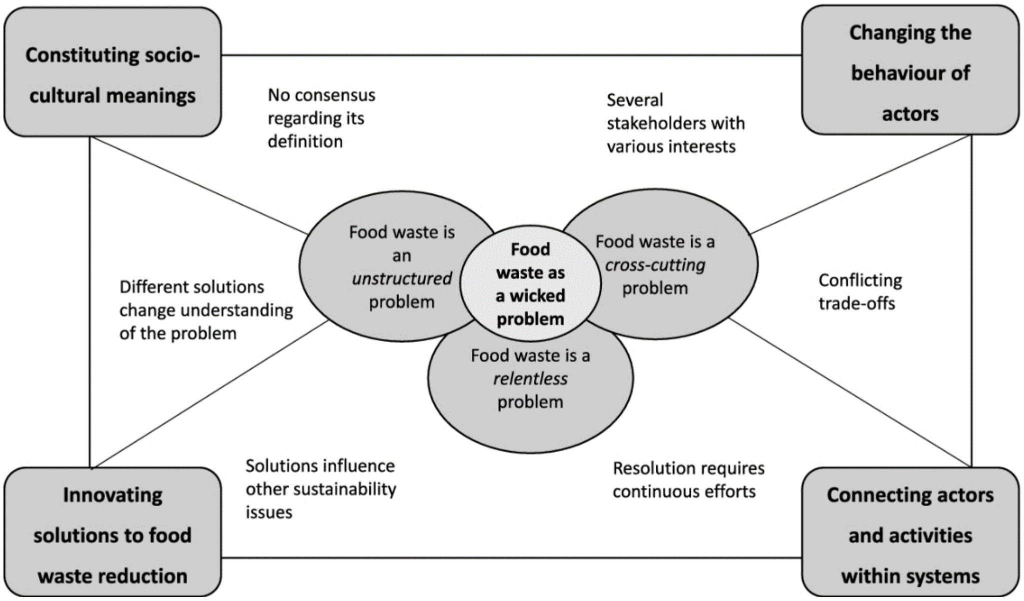

Constituting socio-cultural meanings
Changing the behaviour of actors
Innovating solutions to food waste reduction
Connecting actors and activities within systems
No consensus regarding its definition
Several stakeholders with various interests
Different solutions change understanding of the problem
Conflicting trade-offs
Solutions influence other sustainability issues
Resolution requires continuous efforts
Food waste is an *unstructured* problem
Food waste as a wicked problem
Food waste is a *cross-cutting* problem
Food waste is a *relentless* problem

Figure 2.5.1 Framework for Solving the Wicked Problem of Food Waste

Similar from the perspectives of the Framework that shown in Figure 6 by the research paper [15], which are the changing the behavior of actors and connecting actors and activities within systems. It focuses on the practices and motivations of different actors such as consumer, supermarket, and retail and how they can be influenced to adopt more sustainable habits and choices. Also, emphasizes the interdependencies and interactions among various actors and activities along the food supply chain, and how they can be coordinated and optimized to prevent and minimize food waste.

CHAPTER 3 Proposed Method/Approach

Application UI

The application UI helps users effectively manage and monitor their actions to reduce food waste. It is easy to use for users who may not be used to advanced technology. The application's main feature is to let users set their own goals and track their achievements. This can benefit users by saving money, saving resources, and helping the environment. The application does this by having a user-friendly and intuitive user interface. The application also focuses on making it accessible for older users, with clear and helpful buttons that have both icons and text. This way, users of any skill level can easily learn how to use the app and control their actions. The app also motivates users to achieve their goals and challenge themselves by using gamification techniques, such as points, and rewards. The application also gathers user feedback for usability testing to enhance the application's performance and user satisfaction.

Authentication and Profile Management

Some functionalities are required for the application such as login, registration, and edit profile information. These are some basic functions for user to create and manage their personal accounts, which can store their information such as email, username, points, and rank. Users are required to login before they access to the application if they have created an account before, otherwise they need to register a new account and login to the application. Moreover, provide Google account sign in is convenient for user, they do not require to enter the email and password or register a new account, they only need to click the Google button. For the edit profile function, users can change their username and profile picture.

Social Interaction

This application aims to create a vibrant community of people who are dedicated to reducing food waste and consuming sustainably. It empowers its users to make a positive impact by encouraging social interactions, collaborative projects, and engaging challenges. It also integrates a knowledge-sharing feature that allows users to learn from their own and others' experiences, as well as join in challenges. The application helps users to act, inspire behavior change, and support a global movement for a greener and more sustainable future. Users can understand the environmental consequences of their daily choices and actions better by using this dynamic knowledge source as a key learning tool. Through these intentional methodological approaches, the application seeks to nurture a deep respect for sustainable food practices among its users and promote a sense of community as they work together to create a more sustainable future.

Gamify Activity

This application combines the hidden object game concept with a mission to reduce food waste. In the game, users embark on a virtual supermarket environment to locate hidden expiration dates on various food products. The challenge lies in swiftly and accurately identifying food products approaching its expiration date. Points are earned based on the speed and precision of finding these soon-to-expire food products. Too add complexity, the users must click on food products to reveal its hidden expiration dates. Furthermore, social integration allows users to share their achievements to social media, and the leaderboard will show the points and rank of the users.

Market for food expire Soon

The application offers a unique feature that lets users buy products that are close to their expiry date at a lower price. This feature also gives users the option to either use the products themselves or donate them to the community, creating a sense of choice and social impact. By buying these expire soon products, users can reduce waste and help the environment.

The application also has a points system that rewards users for their purchases. Users can see their points and rank on a leaderboard, where they can compete and compare with other users in the community. The leaderboard also acknowledges and appreciates users who are active and involved in the platform.

Additionally, the points earned can be used to access special benefits or rewards such as get discounts on other purchases, motivating users to keep using the application and participate in both buying and donating almost expired products. This approach not only supports sustainability but also builds a lively and engaged user community.

3.1 System Requirement

3.1.1 Hardware

The hardware that included in this project is laptop and android mobile phone. The laptop is used to develop the food waste gamified mobile application such as design the user interface (UI) of the application, functions and features of the application required. The android mobile phone is used to install and test the developed application on current android version.

Table 1 Specification of Laptop

Description	Specifications
Model	MSI GF63 Thin 9SC
Processor	Intel Core i5-9300H @2.4GHz
Operating System	Windows 10
Graphic	NVIDIA GeForce GTX 1650 4GB GDDR5
Memory	16GB DDR4 RAM
Storage	756GB SSD

Table 2 Specification of Mobile Phone

Description	Specifications
Model	Mi 10T Pro
Processor	Octa-core Max 2.84GHz
Operating System	Android 12
Graphic	Adreno 650
Memory	8 GB RAM
Storage	256 GB

3.1.2 Software

The application uses Flutter, an open-source UI toolkit from Google, and Dart, a flexible programming language that supports object-oriented concepts. These tools help developers create a smooth and attractive user interface and a robust and reliable app logic and features. Flutter has an algorithm that enables the development of gamified mobile apps. Dart also has various algorithms for common tasks like sorting, searching, graphing, and more. Firebase is used to provide some features such as real-time database, authentication, and storage. This help developers build, scale, and maintain mobile applications. For the game development, Unity is the tool that used to develop the game for the application and the Firebase real-time database feature to save the user's data from the game and using the Visual Studio 2022 for create and edit the game script.

Table 3 Software

Software	Description
Android Studio (Environment)	Android studio is an open-source and cost-free software for creating android applications. It is also the official Integrated Development Environment (IDE) that is built on IntelliJ IDEA [16].
Flutter (Development tool)	Flutter is a free UI software development kit from Google. It allows developers to create applications that work on different platforms with one codebase for web browsers, Fuchsia, Android, iOS, Linux, macOS, and Windows[17].
Dart (Programming Language)	Dart is a language designed for creating speedy apps on any device. It aims to provide the most efficient programming language for developing applications that work on multiple platforms, along

	with a versatile execution runtime platform for app frameworks[18].
Firebase (backend app development platform)	Firebase is a platform that helps developers create, manage, and improve web and mobile applications with a range of tools and services. It gives developers access to features such as real-time databases, authentication, hosting, storage, and machine learning options. It was founded in 2011 and bought by Google in 2014.
Unity (Game Engine)	Unity is a game engine that works on multiple platforms, created by Unity Technologies in 2005. It allows developers to make both 2D and 3D games and other interactive experiences. Unity can run on more than 20 different platforms, but the most common ones are PC, Android, and iOS[19].
Visual Studio 2022 (IDE)	Visual Studio stands as a robust suite for developers, offering a unified platform to manage the full development lifecycle. This extensive IDE enables you to author, refine, troubleshoot, and compile code, culminating in the deployment of your application[20].

3.2 System Architecture Diagram

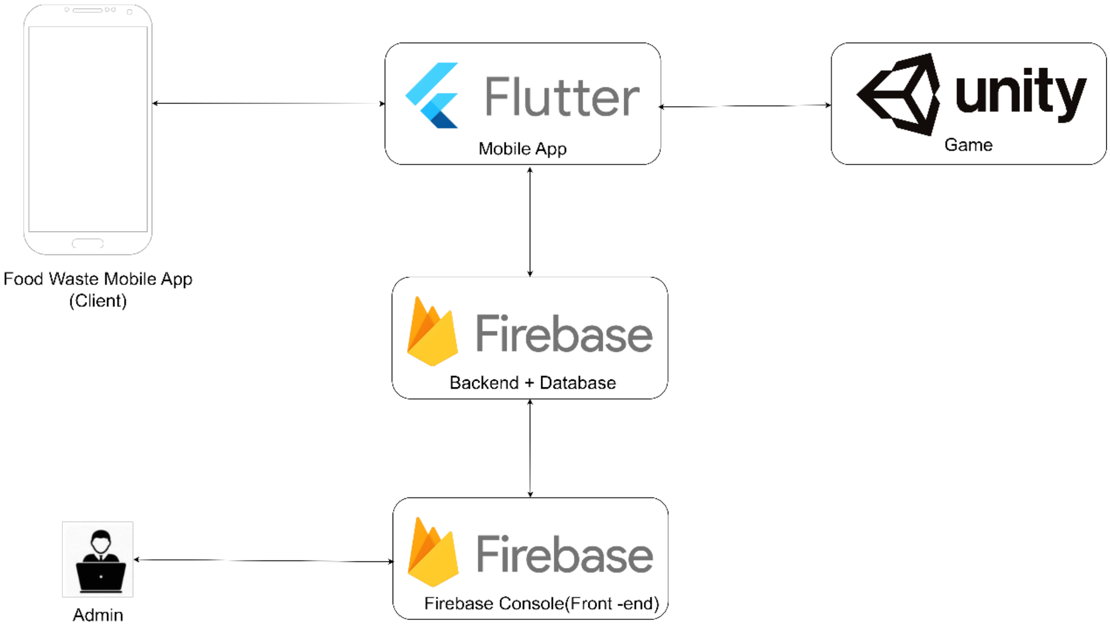


Figure 3.2.1 Application Architecture

For the architecture of the application, it is using the Flutter SDK to design and develop the UI and functionalities of the application. Flutter is an open-source UI SDK, it is one codebase so it can be developing cross platform application. Develop the game with Unity game engine and integrate the game into the Flutter application. Other than that, using Firebase as backend and database to manage, store and retrieve the data of the application such as username, rank, points, and email, it is free to used and easy to use, beneficial to beginner developer. Admin can be controlling the users account and the data store in database through the Firebase console.

3.1.3 Functional Requirements

These are the requirements that specify what the food waste gamified mobile application should do, such as:

- **User registration and profile management**: Users should be able to create and manage their personal accounts, store their information such as username and point and they can edit their personal information.
- **Gamify activity**: Users should be able to participate in a game that challenges them to find food products that are close to their expiry date in a virtual supermarket environment. Beside, for the game challenge will get points and this point will be involved in the leaderboard and ranking users based on the points.
- **Social interaction**: Users should be able to interact with other users in the platform share their achievements, share or write a post about the food waste and comment on the post.
- **Market for buy or donate food expire soon**: Users should be able to make a purchase or food products donation that are near their expiry date at a lower price to earn points and rewards.

3.1.4 Non-Functional Requirements

These are the requirements that specify how the food waste gamified mobile application should perform, such as:

- **Usability**: The application should have a user-friendly and intuitive user interface, provide clear and helpful feedback.
- **Reliability**: The application should function correctly and consistently, handle errors and exceptions gracefully, and recover from failures quickly.
- **Security**: The application should protect the users' data and privacy, prevent unauthorized access and manipulation.
- **Scalability**: The application should be able to handle increasing numbers of users and transactions and adapt to changing demands and requirements.

3.3 System Design Diagram

3.3.1 Use Case Diagram

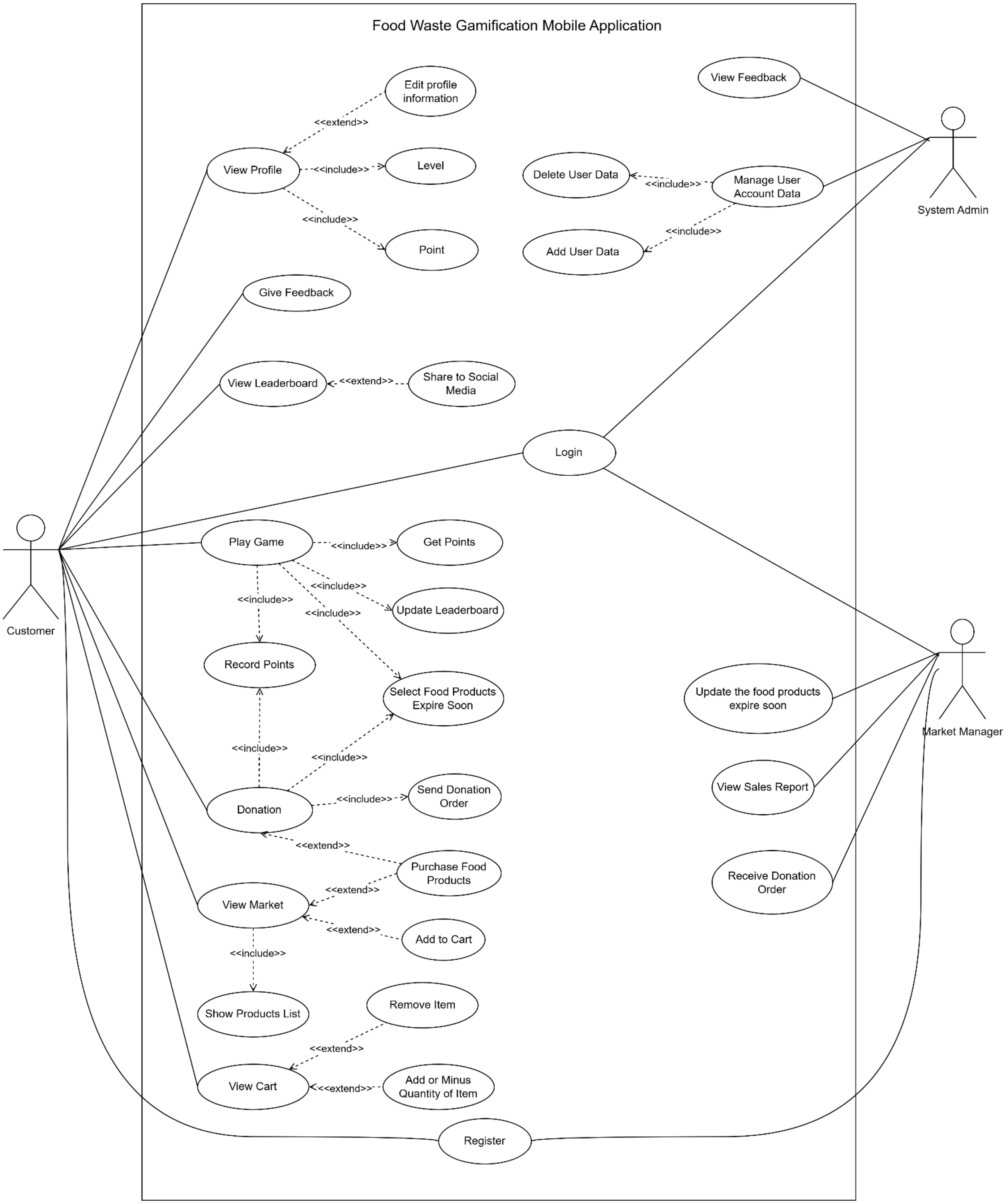


Figure 3.3.1 Use Case Diagram

3.3.2 Use Case Description

Login

Use Case Name: Login	ID: 1	Importance Level: High
Primary Actor: Customer, Admin, Market Manager	Use Case Type: detail, essential	
Stakeholders and interests: Customer – Want to access the application and use the features of the application. Admin – Want to login to as an admin in the application. Market Manager – Want to login as a market manager in the application.		
Brief Description: This use case describes the different roles login to the application.		
Trigger: Customer/Admin/Market Manager want to login to the application. Type: External		
Relationships: Association: Customer, Admin, Market Manager Include: Authentication Extend: Show Error Message Generalization:		
Normal Flow of Events: 1. The application displays the login page with email and password text fields. 2. The customer/admin/market manager fills the email and password. 3. The customer/admin/market manager click the “Login” button. 4. The application will authenticate the user credential. 5. The application will direct user to the home page.		
SubFlows:		
Alternate/Exceptional Flows: 4a. The email and password invalid, the users need to fill the email and password again.		

Register

<table>
<tr><td colspan="2">Use Case Name: Register</td><td>ID: 2</td><td>Importance Level: High</td></tr>
<tr><td colspan="3">Primary Actor: Customer, Market Manager</td><td>Use Case Type: detail, essential</td></tr>
<tr><td colspan="4">Stakeholders and interests:
Customer – Want to register an account for login to the application.
Market Manager – Want to register a manager account for login to the application.</td></tr>
<tr><td colspan="4">Brief Description: This use case describes the process of registration of an account.</td></tr>
<tr><td colspan="4">Trigger: Customers/Market Manager need an account to login to the application.
Type: External</td></tr>
<tr><td colspan="4">Relationships:
Association: Customer, Market Manager
Include:
Extend:
Generalization:</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The application displays the login page.
2. The users click the “register account”.
3. The application direct to registration page with email, password, confirm password text fields.
4. The users fill the email and password.
5. The users choose to register as different roles.
If users choose to register as customer,
the S-1: customer registration subflow is performed.
If users choose to register as market manager,
the S-2: manager registration subflow is performed.
6. The application validates the email and password.
7. The application directs the users to login page.</td></tr>
<tr><td colspan="4">SubFlows:
S-1: Customer Registration
1. The user chooses the role as customer.
2. The user needs to fill up their email and password for registration.
S-2: Manager Registration</td></tr>
</table>

1. The user chooses the role as market manager. 2. The user needs to fill up their email, password, and market detail for registration.
Alternate/Exceptional Flows:

View Profile

Use Case Name: View Profile	ID: 3	Importance Level: High
Primary Actor: Customer	Use Case Type: detail, essential	
Stakeholders and interests: Customer – Wants to view profile details and edit profile information.		
Brief Description: This use case describes the process of check profile and edit profile information.		
Trigger: Customers click profile icon. Type: External		
Relationships: Association: Customer Include: Level, Point Extend: Edit profile information Generalization:		
Normal Flow of Events: 1. The customer selects the “Profile” on the menu bar. 2. The application displays the profile information of the customer. 3. The customer clicks the “Edit” button for change the information. 4. The customer clicks the “Save” button for save the changes.		
SubFlows:		
Alternate/Exceptional Flows:		

Play Game

<table>
<tr><td>Use Case Name: Play Game</td><td colspan="2">ID: 4</td><td>Importance Level:</td></tr>
<tr><td colspan="2">Primary Actor: Customer</td><td colspan="2">Use Case Type: detail, essential</td></tr>
<tr><td colspan="4">Stakeholders and interests:
Customer – Wants to play game and get points</td></tr>
<tr><td colspan="4">Brief Description: This use case describes the process of get points in game</td></tr>
<tr><td colspan="4">Trigger: Customers click the game icon.
Type: External</td></tr>
<tr><td colspan="4">Relationships:
Association: Customer
Include: Get Points, Update Leaderboard, Select Food Products Expire Soon
Extend:
Generalization:</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. Customers select the game icon on the menu bar.
2. Customers play the game with select the food products expire soon.
If the customer selects the food products expire soon,
the S-1: Get points subflow is performed.
If the customer selects the food products not nearly expire,
The S-2: Deduct points subflow is performed.
3. The application updates the points and experience points of customers and the leaderboard.</td></tr>
<tr><td colspan="4">SubFlows:
S-1: Get Points
1. Select the food products expire soon.
2. Get points and experience points.
S-2: Deduct Points
1. Select the food products not nearly expire.
2. Deduct points and no experience points.</td></tr>
<tr><td colspan="4">Alternate/Exceptional Flows:</td></tr>
</table>

View Leaderboard

Use Case Name: View Leaderboard	ID: 5	Importance Level: Medium
Primary Actor: Customer	Use Case Type: detail, essential	
Stakeholders and interests: Customer – Wants to view their rank in the leaderboard and share it to social media.		
Brief Description: This use case describes the process of view the leaderboard,		
Trigger: Customer clicks the leaderboard icon Type: External		
Relationships: Association: Customer Include: Extend: Share to social media Generalization:		
Normal Flow of Events: 1. Customers select the “Leaderboard” on the menu bar. 2. The application shows the rank, level, and points of customer. 3. Customers view their rank on leaderboard. If customer want to share their achievement on leaderboard, the S-1: share to social media subflow is perform.		
SubFlows: S-1: Share to social media. 1. Customers choose to share their information on leaderboard to social media.		
Alternate/Exceptional Flows:		

Donation

Use Case Name: Donation	ID: 6	Importance Level: Medium
Primary Actor: Customer		Use Case Type: detail, essential
Stakeholders and interests: Customer – Wants to donate the food products that expire soon.		
Brief Description: This use case describes the process of food products donation		
Trigger: Customers click the donation option on the home page. Type: External		
Relationships: Association: Customer Include: Record Points, Select Food Products Expire Soon, Send Donation Order Extend: Purchase Food Products Generalization:		
Normal Flow of Events: 1. Customers select the donation option on the home page. 2. Customers select the food products that nearly expire. 3. Customers choose the charity to donate to. 4. Customers click “Confirm” button. 5. The application direct to purchase page. If customers want to use points to redeem discount when purchase, the S-1: redeem discount subflow is performed. 6. Customers confirm and purchase the food products. 7. Customers get points. 8. The application updates the points and leaderboard.		
SubFlows: S-1: Redeem Discount 1. The customer chooses to use the points to get discount on purchase the food products		
Alternate/Exceptional Flows:		

View Market

<table>
<tr><td>Use Case Name: View Market</td><td>ID: 7</td><td>Importance Level: Medium</td></tr>
<tr><td colspan="2">Primary Actor: Customer</td><td>Use Case Type: detail, essential</td></tr>
<tr><td colspan="3">Stakeholders and interests:
Customer – Wants to buy some food products on the market</td></tr>
<tr><td colspan="3">Brief Description: This use case describes the process of buy products on the market.</td></tr>
<tr><td colspan="3">Trigger: Customers click the market option on the home page
Type: External</td></tr>
<tr><td colspan="3">Relationships:
Association: Customer
Include: Show Products List
Extend: Purchase Food Products, Add to Cart
Generalization:</td></tr>
<tr><td colspan="3">Normal Flow of Events:
1. Customers select the market option on home page.
2. The application shows the products list.
3. Customers select the food products.
4. Customers add food products to cart.
If customers want to increase the quantity of the products,
the S-1: add product subflow is performed.
If customers want to remove the products,
the S-2: remove product subflow is performed.
5. The application calculates the total amount.
If customers want to use points to redeem discount when purchase,
the S-3: redeem discount subflow is performed.
6. Customers click “Confirm” button in cart page.
7. The application direct to purchase page.
8. Customers confirm and purchase the food products.</td></tr>
<tr><td colspan="3">SubFlows:
S-1: Add product.
1. The customer selects the food product in the cart.</td></tr>
</table>

2. Customers click the “+” button to increase the quantity of the food product. S-2: Remove Product 1. The customer selects the food product in the cart. 2. Customer clicks the “-” button to decrease the quantity or remove the food product. S-3: Redeem Discount 1. The customer chooses to use the points to get discount on purchase the food products.
Alternate/Exceptional Flows:

Give Feedback

Use Case Name: Give Feedback	ID: 8	Importance Level: High
Primary Actor: Customer	Use Case Type: detail, essential	
Stakeholders and interests: Customer – Wants to provide some feedback to admin about the problem of the application		
Brief Description: This use case describes the process of feedback from customer to admin		
Trigger: Customers select to feedback option on home page. Type: External		
Relationships: Association: Customer Include: Extend: Generalization:		
Normal Flow of Events: 1. Customers select the feedback option on home page. 2. The application direct to feedback page with a textbox. 3. Customers write comment or the problem of the application. 4. Customers click “Send Feedback” button. 5. The application sends the feedback to admin.		
SubFlows:		
Alternate/Exceptional Flows:		

View Cart

Use Case Name: View Cart	ID: 9	Importance Level: Medium
Primary Actor: Customer		Use Case Type: detail, essential

Stakeholders and interests:

Customer – Wants to view the product added to the cart or edit the product in the cart.

Brief Description: This use case describes the cart view to users.

Trigger: Customers select the market and inside the market have cart icon on top right, click the cart icon will navigate to cart view.

Type: External

Relationships:

Association: Customer

Include:

Extend: Remove Item, Add or Minus Quantity of Item

Generalization:

Normal Flow of Events:

1. Customers select the market option on home page.
2. The application navigates to product list page.
3. Customers click the cart icon on top right to cart page.

   If customers want to increase the quantity of the products,
   the S-1: add product subflow is performed.

   If customers want to minus the products,
   the S-2: remove product subflow is performed.
4. Customers can remove the item from cart.

SubFlows:

S-1: Add product.

3. The customer selects the food product in the cart.
4. Customers click the "+" button to increase the quantity of the food product.

S-2: Minus Product

3. The customer selects the food product in the cart.
4. Customer clicks the "-" button to decrease the quantity the food product.

Alternate/Exceptional Flows:

View Feedback

Use Case Name: Collect Feedback	ID: 10	Importance Level: High
Primary Actor: Admin	Use Case Type: detail, essential	
Stakeholders and interests: Admin – Wants to view the feedback from users and analysis the feedback for improve the application.		
Brief Description: This use case describes the process of view feedback and analysis feedback.		
Trigger: Admin select the “View Feedback” on the home page. Type: External		
Relationships: Association: Admin Include: Extend: Generalization:		
Normal Flow of Events: 1. Admin select the feedback on the homepage. 2. The application direct to feedback page. 3. The application shows the feedback from the users. 4. Admin collect the feedback. 5. Admin review and categorize the feedback.		
SubFlows:		
Alternate/Exceptional Flows:		

Manage User Account

Use Case Name: Manage User Account	ID: 11	Importance Level: High
Primary Actor: Admin	Use Case Type: detail, essential	
Stakeholders and interests: Admin – Wants to manage the user account		
Brief Description: This use case describes the process of manage the user account.		
Trigger: Admin click the user statistic option. Type: External		
Relationships: Association: Admin Include: Add User Data, Delete User Data Extend: Generalization:		
Normal Flow of Events: 1. Admin login with admin account. 2. The application direct to admin panel page. 3. Admin click on the user statistic option. 4. The application shows the users information list. If admin want to add new user, the S-1: add new user subflow is performed. If admin want to remove existing user, the S-2: remove user subflow is performed. 5. The application updates the user information list.		
SubFlows: S-1: Add New User 1. Admin click the “Create User” button. 2. The application direct to add user page with user email text field, password text field, and role selection. 3. Admin click the “Save” button. S-2: Remove User 1. Admin select the existing user account in the list. 2. Admin click the “remove” button. 3. The application pop-up a message “Confirm to delete?”.		

4. Admin click confirm.
Alternate/Exceptional Flows:

Update Food Products

Use Case Name: Update Food Products Expire Soon	ID: 12	Importance Level: High
Primary Actor: Market Manager		Use Case Type: detail, essential
Stakeholders and interests: Market Manager – Wants to update the food products expire soon to the market.		
Brief Description: This use case describes the process of updates the food product that expire soon to market.		
Trigger: Market Manager click the "Products" on the menu bar. Type: External		
Relationships: Association: Market Manager Include: Extend: Generalization:		
Normal Flow of Events: 1. Market Manager select the "Products" on the menu bar. 2. The application shows the products list. 3. Market Manager click the edit icon. If Market Manager want to add new food products nearly expire, the S-1: Add new food products subflow is performed. If Market Manager want to remove the food products that expired, The S-2: Remove expired food products subflow is performed. 4. The application updates the food products list.		
SubFlows: S-1: Add New Food Products		

1. Market Manager select add button. 2. Market Manager upload the picture and information of food products. 3. Market Manager click save button. S-2: Remove Expired Food Products 1. Market Manager select the food products that expired. 2. Market Manager click the remove button. 3. Market Manager click the save button.
Alternate/Exceptional Flows:

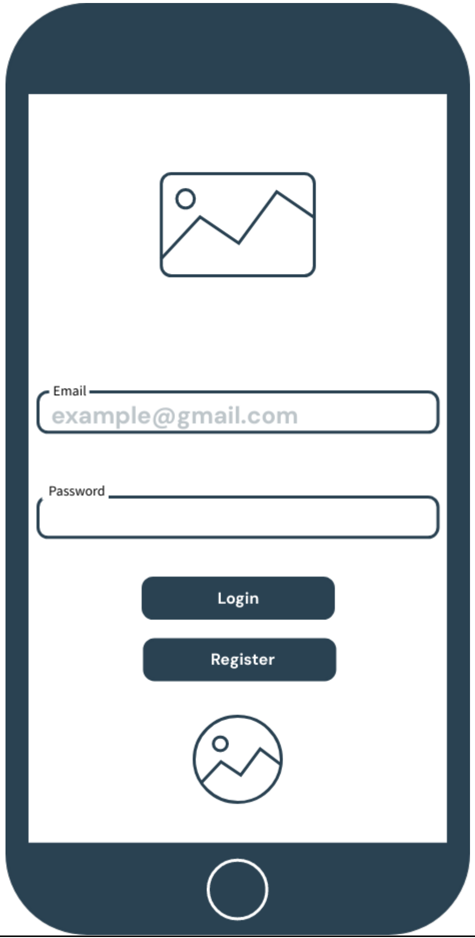
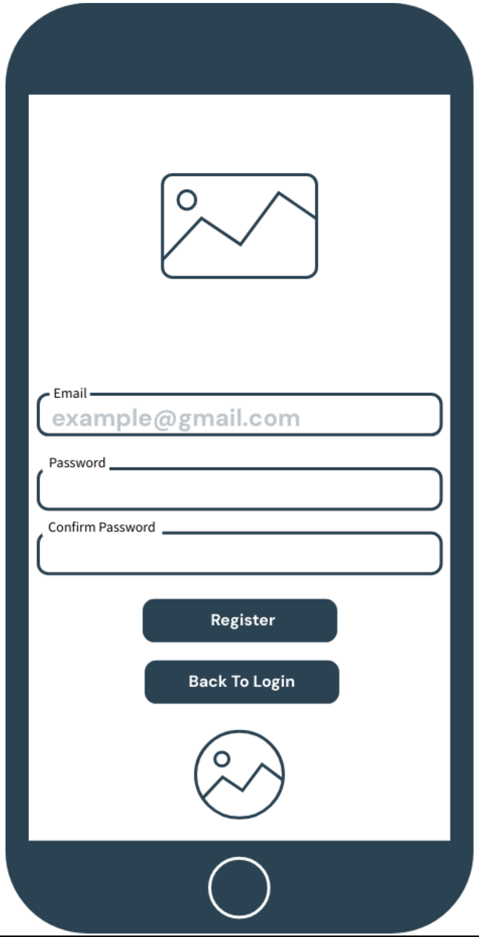
View Sales Report

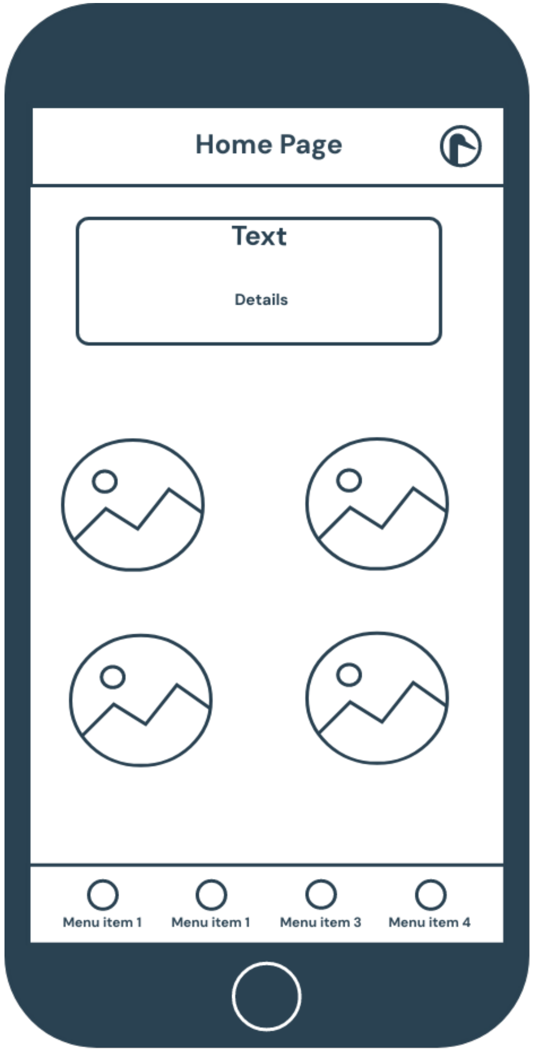
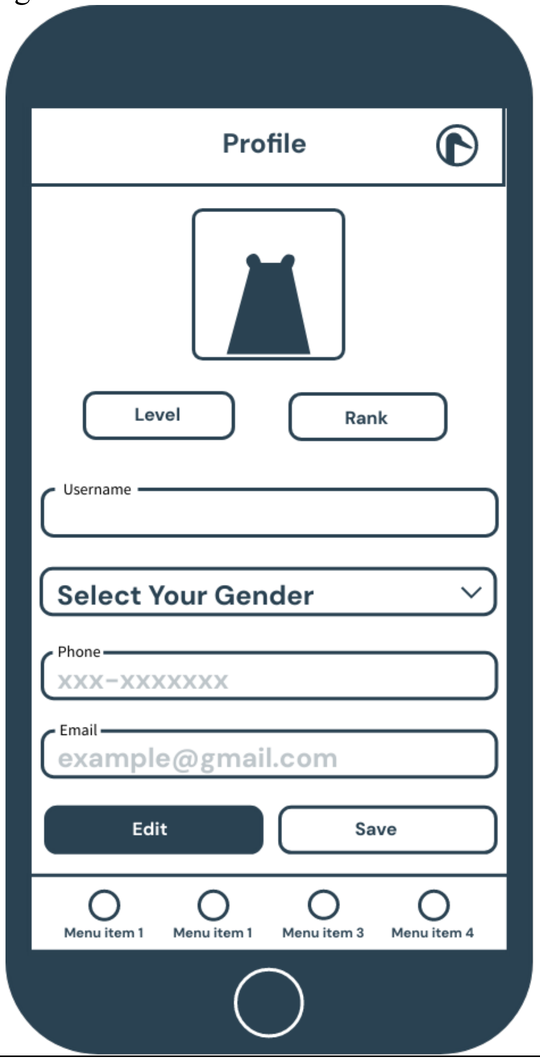
Use Case Name: View Sales Report	ID: 13	Importance Level: Medium
Primary Actor: Market Manager		Use Case Type: detail, essential
Stakeholders and interests: Market Manager – Wants to view the sales report of the food products that expire soon.		
Brief Description: This use case describes the process of market manager view the sales report of expire soon food products.		
Trigger: Market Manager click the “Sales Report” on the menu bar. Type: External		
Relationships: Association: Market Manager Include: Extend: Generalization:		
Normal Flow of Events: 1. Market Manager select the “Sales Report” on the menu bar. 2. The application shows the graph and detail of the sales of the food products.		
SubFlows:		
Alternate/Exceptional Flows:		

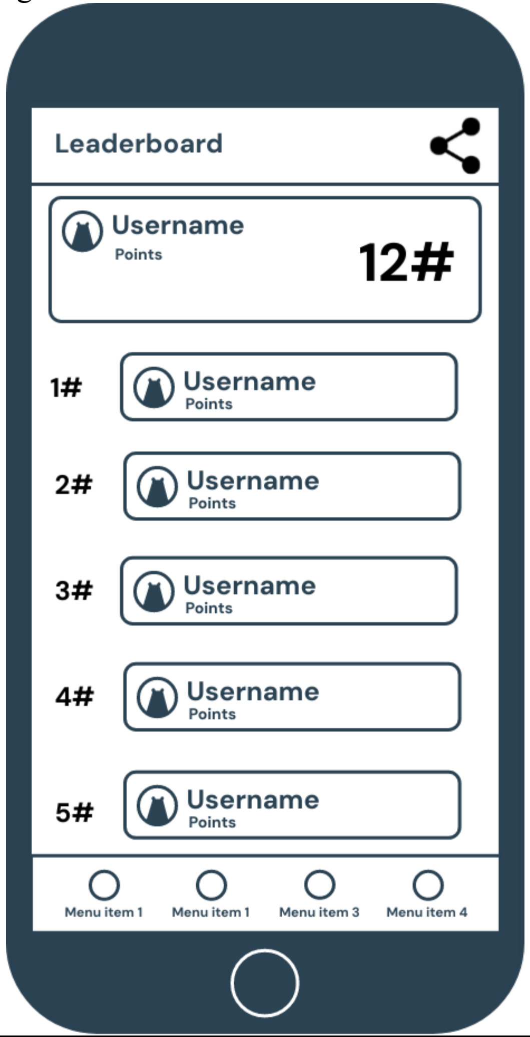
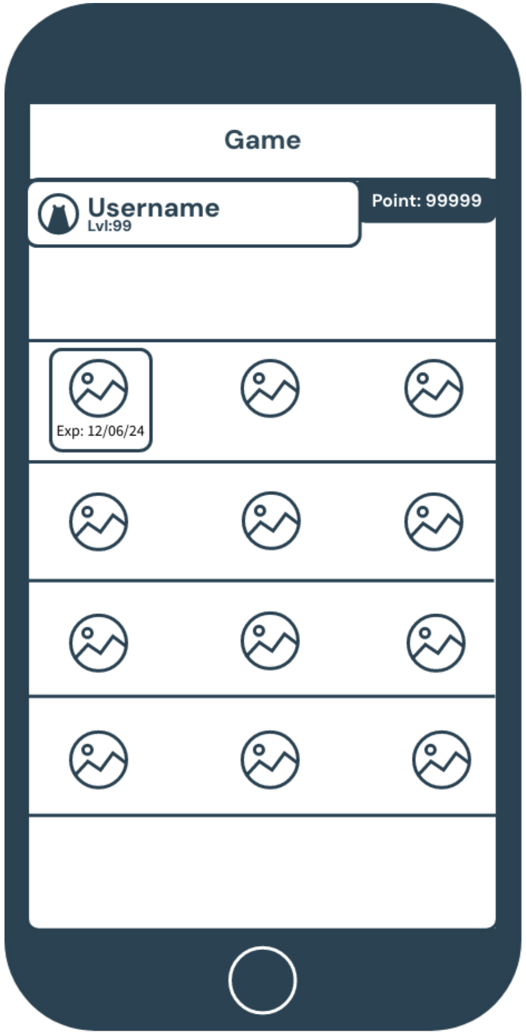
Receive Donation Order

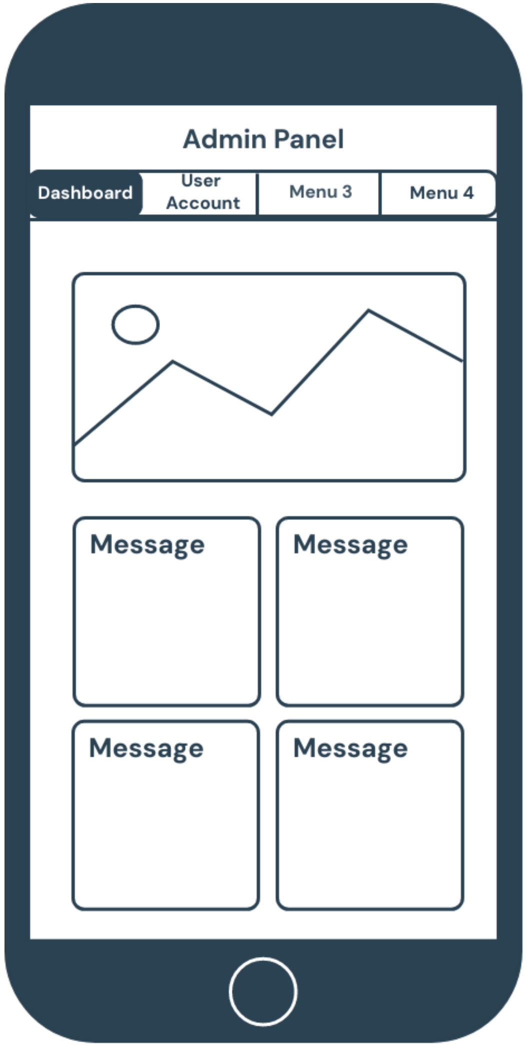
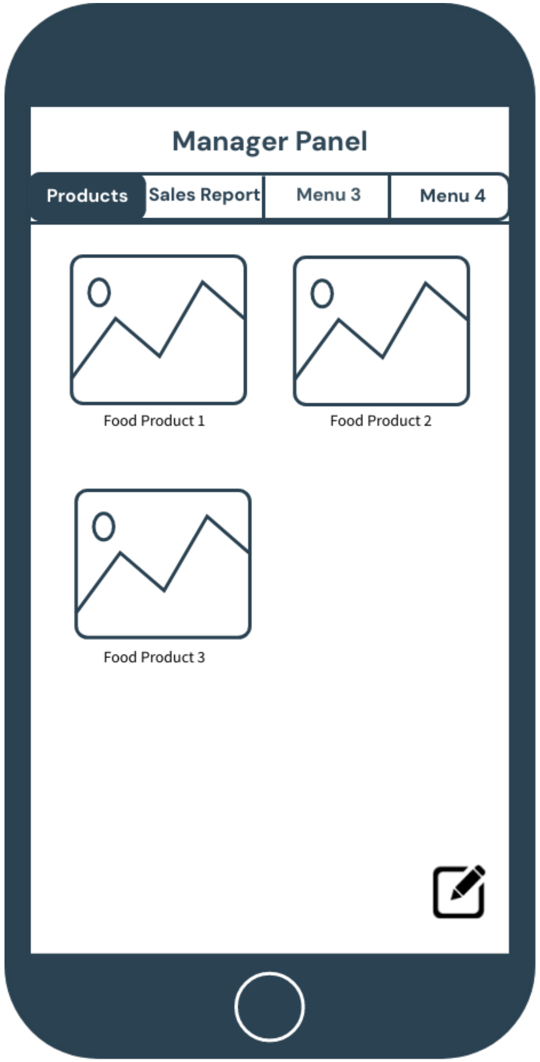
Use Case Name: Receive Donation Order	ID: 14	Importance Level: Medium
Primary Actor: Market Manager	Use Case Type: detail, essential	
Stakeholders and interests: Market Manager – Wants to processing donation order from the customer,		
Brief Description: This use case describes the process of donation order processing from customer.		
Trigger: Market Manager select the “Donation Order” on the menu bar Type: External		
Relationships: Association: Market Manager Include: Extend: Generalization:		
Normal Flow of Events: 1. Market Manager select the “Donation Order” on menu bar. 2. The application shows the donation order from the customer. 3. The application categorizes the order based on the charity selected by customer. 4. Market Manager confirm the donation order.		
SubFlows:		
Alternate/Exceptional Flows:		

3.3.3 Mid-Fidelity Wireframe

Login Page	Sign Up Page
	 Email example@gmail.com Password Confirm Password Register Back To Login

Home Page	User Profile Page
 Home Page Text Details Menu item 1 Menu item 1 Menu item 3 Menu item 4	 Profile Level Rank Username Select Your Gender Phone xxx-xxxxxxx Email example@gmail.com Edit Save Menu item 1 Menu item 1 Menu item 3 Menu item 4

Leaderboard Page	Game Page
	

Admin Page	Market Manager Page
 Admin Panel Dashboard User Account Menu 3 Menu 4 Message Message Message Message	 Manager Panel Products Sales Report Menu 3 Menu 4 Food Product 1 Food Product 2 Food Product 3

3.4 Usability Design Rules

There had Schneiderman’s 8 golden rules, which are Consistency, Shortcuts, Informative feedback, Dialogue, Error handling, Permit reversal of actions, support internal locus of control, and reduce short-term memory load. But in this application, not all rules are implemented, such as permit reversal of actions and support internal locus of control.

Layout and Navigation: The application should have a clear and consistent layout and navigation that follows the platform conventions and user expectations. This means that the application should use familiar and intuitive elements, such as menus, buttons, tabs, and icons, to organize and present the information and functions of the application. The application should also provide clear and consistent labels, headings, and instructions to guide the user through the application.

Feedback and Confirmation: The application should provide feedback and confirmation for user actions, such as logging the application, completing challenges, earning points, and redeeming rewards. This means that the application should use visual and auditory cues, such as sounds, animations, and pop-ups, to acknowledge and inform the user about the results and consequences of their actions. The application should also provide positive and constructive feedback, such as praise, encouragement, and suggestions, to motivate and support the user in their food waste reduction journey.

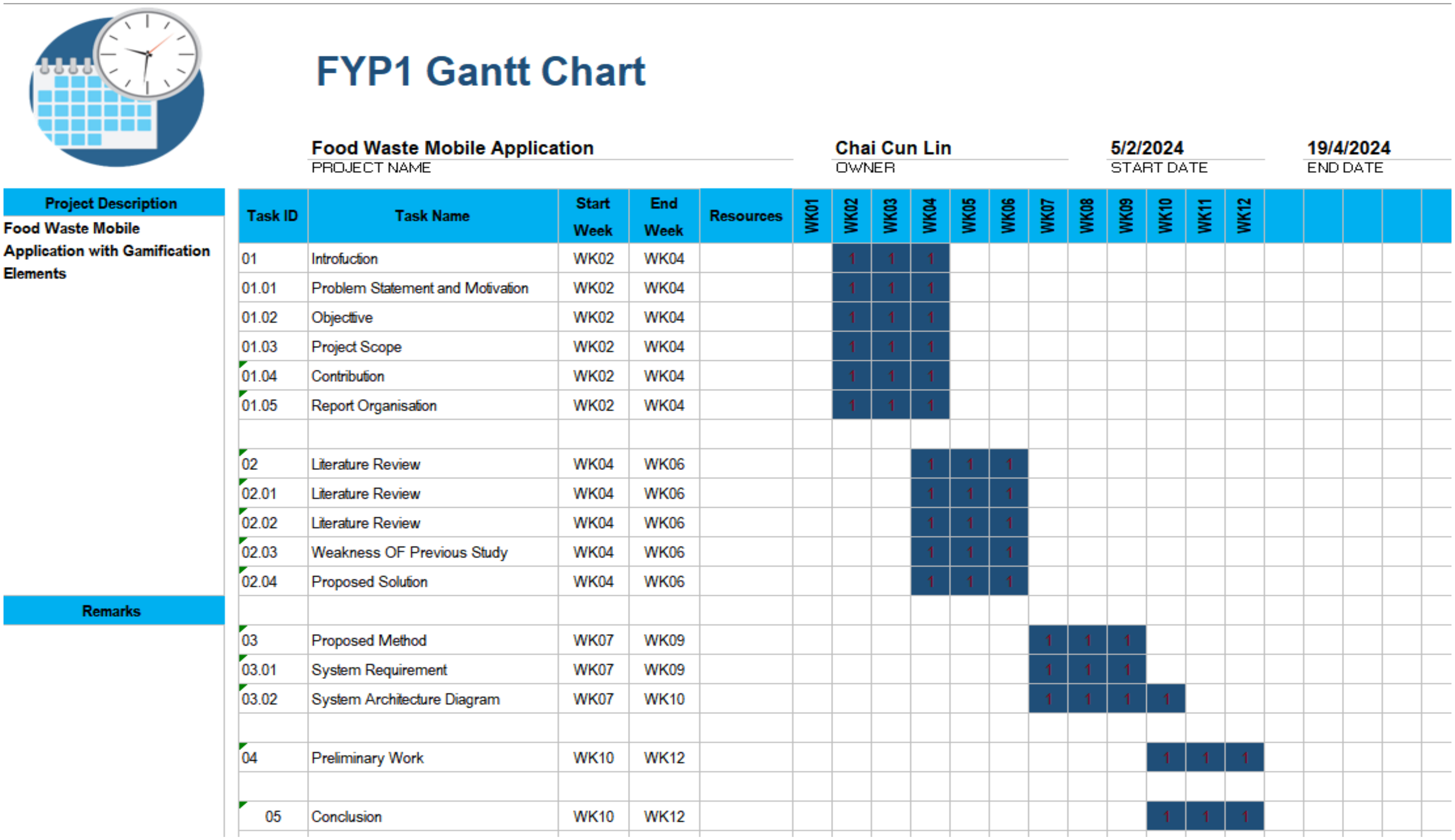
Gamification Elements: The application should use gamification elements, such as points, levels, and leaderboards, to motivate and engage users to reduce food waste and raise awareness of food waste management. This means that the application should use game mechanics, such as goals, challenges, rewards, and competition, to create a fun and immersive experience for the user. The application should also use game dynamics, such as progress, feedback, and social comparison, to create a sense of achievement, mastery, and belonging for the user.

Visual and Textual Cues: The application should use visual and textual cues, such as colors, icons, labels, and tooltips, to communicate information and instructions to users in an intuitive and accessible way. This means that the application should use appropriate and consistent symbols, images, and words to represent the information and functions of the application. The application should also use simple and concise language, and avoid technical terms, to explain the information and instructions to the user.

Personalization and Customization: The application should use personalization and customization features, such as user profiles, avatars, and notifications, to enhance user satisfaction and loyalty. This means that the application should allow the user to create and modify their own identity in the application, such as changing their name, and picture. The application should also use notifications and reminders, such as push notifications to keep the user updated and engaged with the application.

Social Interaction Features: The application should use social interaction features, such as sharing, and group challenges, to foster a sense of community and collaboration among users. This means that the application should enable the user to share and compare their food waste game achievements and experiences with other users, such as posting and sharing their achievement and tips. The application should also enable the user to participate and cooperate in group challenges, such as joining and creating teams, and competing.

3.5 Gantt Chart



FYP1 Gantt Chart

Food Waste Mobile Application
PROJECT NAME

Chai Cun Lin
OWNER

5/2/2024
START DATE

19/4/2024
END DATE

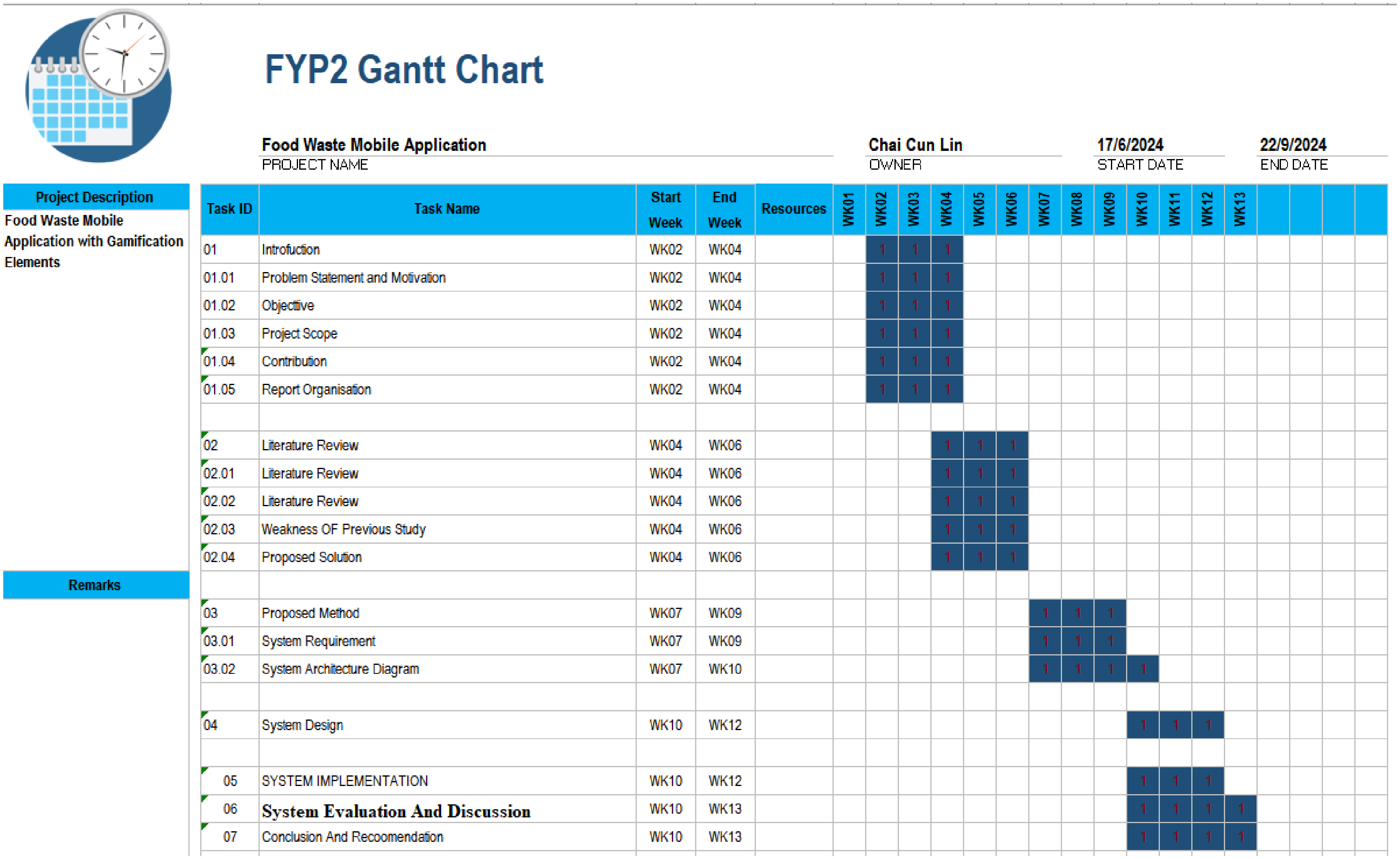
Project Description

Food Waste Mobile Application with Gamification Elements

Remarks

Task ID	Task Name	Start Week	End Week	Resources	WK01	WK02	WK03	WK04	WK05	WK06	WK07	WK08	WK09	WK10	WK11	WK12
01	Introfuction	WK02	WK04			1	1	1								
01.01	Problem Statement and Motivation	WK02	WK04			1	1	1								
01.02	Objective	WK02	WK04			1	1	1								
01.03	Project Scope	WK02	WK04			1	1	1								
01.04	Contribution	WK02	WK04			1	1	1								
01.05	Report Organisation	WK02	WK04			1	1	1								
02	Literature Review	WK04	WK06					1	1	1						
02.01	Literature Review	WK04	WK06					1	1	1						
02.02	Literature Review	WK04	WK06					1	1	1						
02.03	Weakness OF Previous Study	WK04	WK06					1	1	1						
02.04	Proposed Solution	WK04	WK06					1	1	1						
03	Proposed Method	WK07	WK09								1	1	1			
03.01	System Requirement	WK07	WK09								1	1	1			
03.02	System Architecture Diagram	WK07	WK10								1	1	1	1		
04	Preliminary Work	WK10	WK12											1	1	1
05	Conclusion	WK10	WK12											1	1	1

Figure 3.5.1 FYP 1 Gantt Chart



FYP2 Gantt Chart

Food Waste Mobile Application
PROJECT NAME

Chai Cun Lin
OWNER

17/6/2024
START DATE

22/9/2024
END DATE

Project Description

Food Waste Mobile Application with Gamification Elements

Remarks

Task ID	Task Name	Start Week	End Week	Resources	WK01	WK02	WK03	WK04	WK05	WK06	WK07	WK08	WK09	WK10	WK11	WK12	WK13				
01	Introfuction	WK02	WK04			1	1	1													
01.01	Problem Statement and Motivation	WK02	WK04			1	1	1													
01.02	Objective	WK02	WK04			1	1	1													
01.03	Project Scope	WK02	WK04			1	1	1													
01.04	Contribution	WK02	WK04			1	1	1													
01.05	Report Organisation	WK02	WK04			1	1	1													
02	Literature Review	WK04	WK06					1	1	1											
02.01	Literature Review	WK04	WK06					1	1	1											
02.02	Literature Review	WK04	WK06					1	1	1											
02.03	Weakness OF Previous Study	WK04	WK06					1	1	1											
02.04	Proposed Solution	WK04	WK06					1	1	1											
03	Proposed Method	WK07	WK09								1	1	1								
03.01	System Requirement	WK07	WK09								1	1	1								
03.02	System Architecture Diagram	WK07	WK10								1	1	1	1							
04	System Design	WK10	WK12											1	1	1					
05	SYSTEM IMPLEMENTATION	WK10	WK12											1	1	1					
06	**System Evaluation And Discussion**	WK10	WK13											1	1	1	1				
07	Conclusion And Recoomendation	WK10	WK13											1	1	1	1				

Figure 3.5.2 FYP 2 Gantt Chart

CHAPTER 4 System Design

4.1 Block Diagram

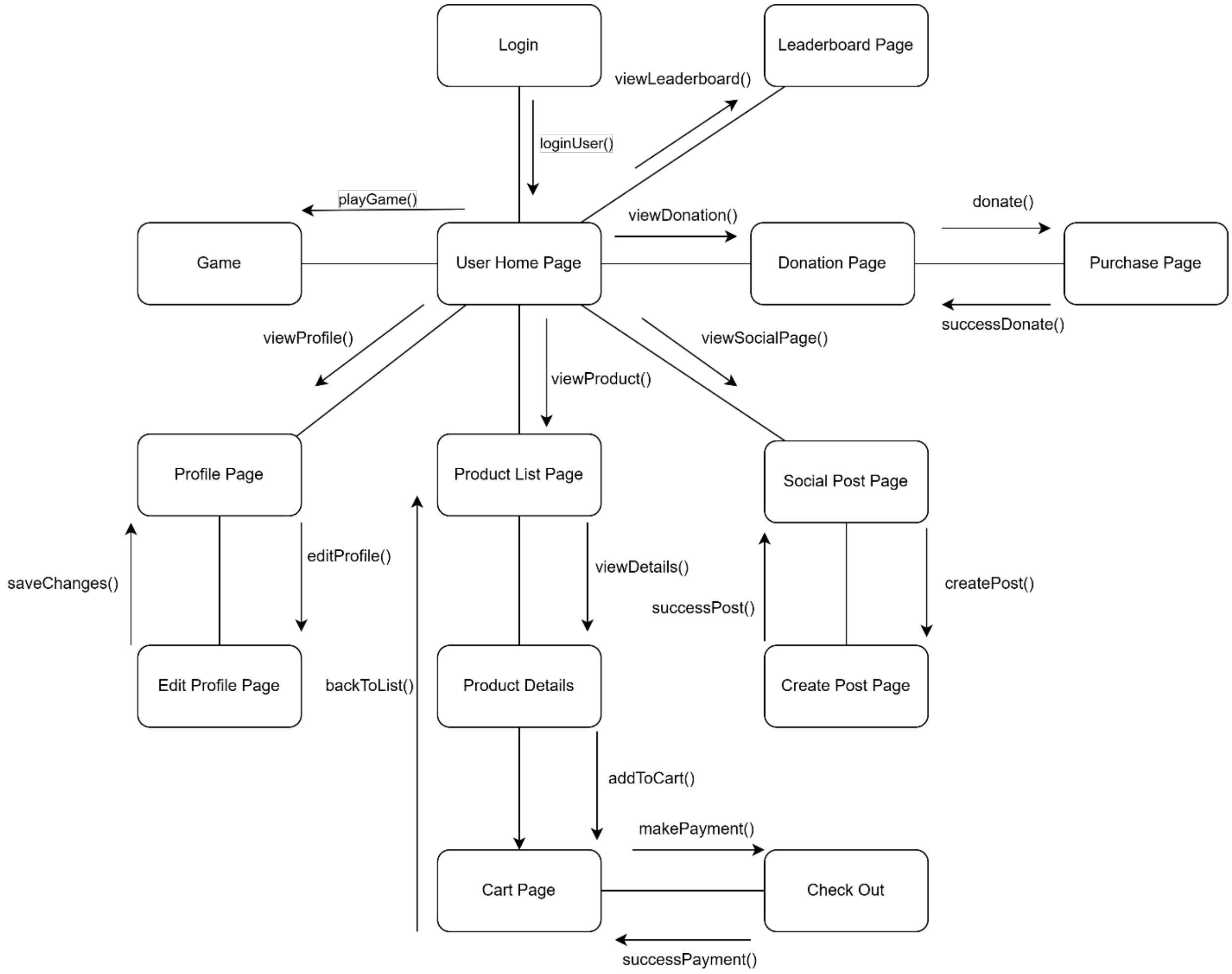


Figure 4.1.1 User Block Diagram

The Figure 4.1.1 above show the block diagram of the food waste mobile application, the navigation flow of the application from the perspective of the user. Users must first log in through the Login Page to access the system. Upon successful login, they are directed to the User Home Page, which serves as the central hub for accessing various functionalities within the application.

From the User Home Page, users can explore different sections of the app. They can choose to play games, manage their personal profiles by navigating to the Profile Page, or view and edit their profile information on the Edit Profile Page. Additionally, they have the option to check their rankings on the Leaderboard Page.

For users interested in making purchases, the application allows them to browse products through the Product List Page. By selecting a specific product, users can view detailed information, including descriptions and pricing, on the Product Details page. If they decide to purchase an item, they can add it to their shopping cart, where they can review and manage their selected products by adjusting quantities or removing items if necessary. When ready to make a purchase, users proceed to the Check Out page to finalize their transaction.

The application also enables users to interact socially through the Social Post Page, where they can create and share posts with others. Additionally, there is a Donation Page for users who wish to make donations, and a Purchase Page to complete the donation process.

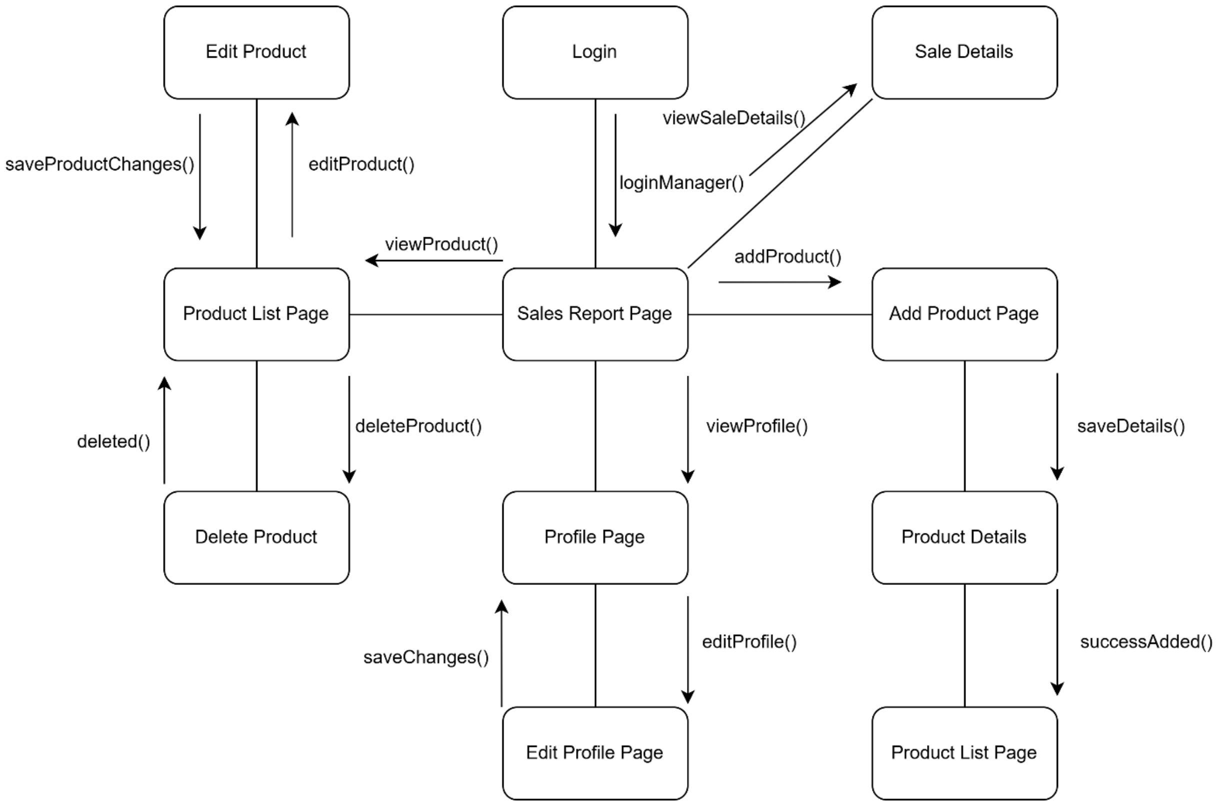


Figure 4.1.2 Manager Block Diagram

The Figure 4.1.2 above show the block diagram of the food waste mobile application from the perspective of the manager outlines key functionalities available to manage the products efficiently. After logging into the application via the Login page, the manager gains access to various control points across the platform.

The manager's role involves oversight and management of product-related activities. They can add new products to the system through the Add Product Page and review the list of available products via the Product List Page. The manager also has the ability to modify existing products by accessing the Edit Product and can delete products if necessary.

In terms of sales management, the manager can track sales performance through the Sales Report Page, which aggregates data to help in monitoring overall sales. Additionally, detailed information about individual sales is available on the Sale Details, providing insights into specific transactions.

The manager's personal profile can also be managed through the Profile Page, where they can view and update their profile information using the Edit Profile.

This diagram illustrates the manager's role in overseeing product inventory, monitoring sales performance, and managing personal account settings, ensuring efficient operation.

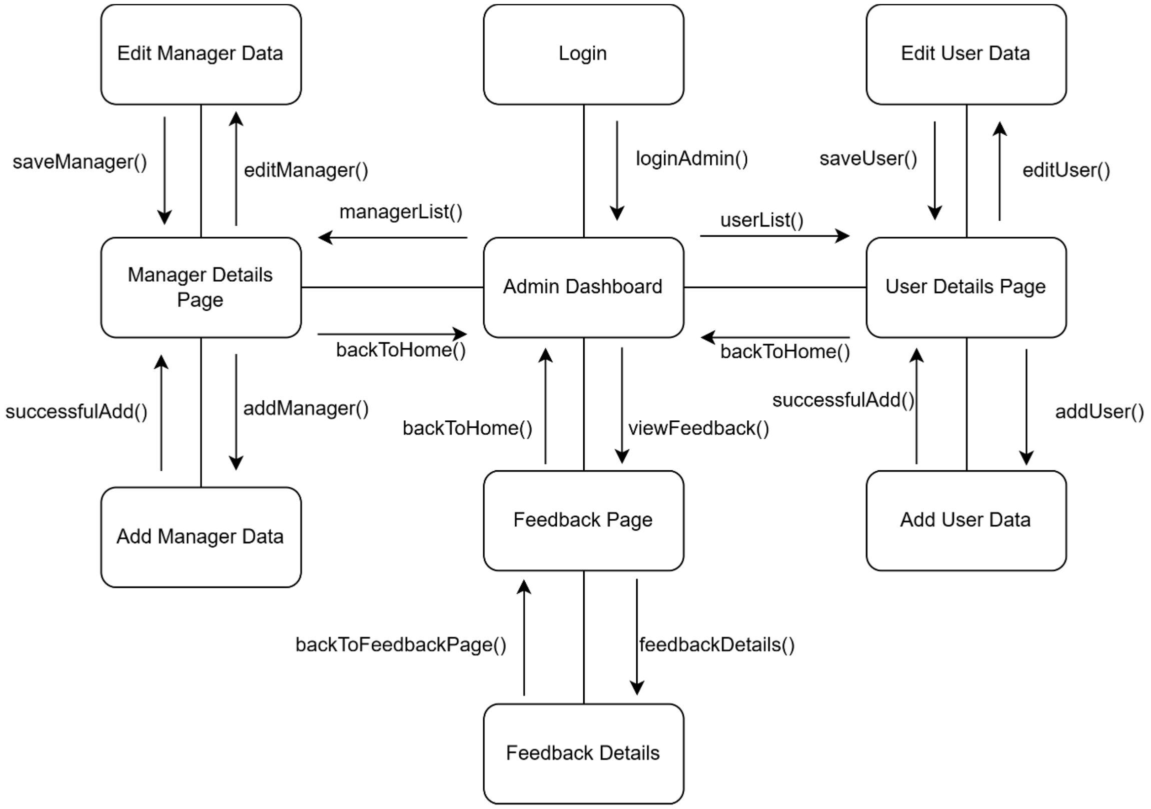


Figure 4.1.3 Admin Block Diagram

The Figure 4.1.3 above show the block diagram of the food waste mobile application from the perspective of the admin, outlines the core functionalities available to an admin for managing the users and managers effectively. After logging into the application via the Login page, the admin gains access to the Admin Dashboard, a central hub for navigating through various administrative functions.

The admin can manage both users and managers, with options to view details, add, and edit data for each. Specifically, the User Details Page and Manager Details Page allow the admin to view existing user and manager information, while the Add User Data and Add Manager Data facilitate adding new users and managers data into the database. Similarly, the Edit User Data and Edit Manager Data provide the admin with the ability to update or correct existing records.

Additionally, the Feedback Page allows the admin to oversee feedback submitted by users or managers, and the Feedback Details page gives more specific insights into individual feedback entries, enabling detailed responses or follow-ups as necessary.

4.2 Flowchart

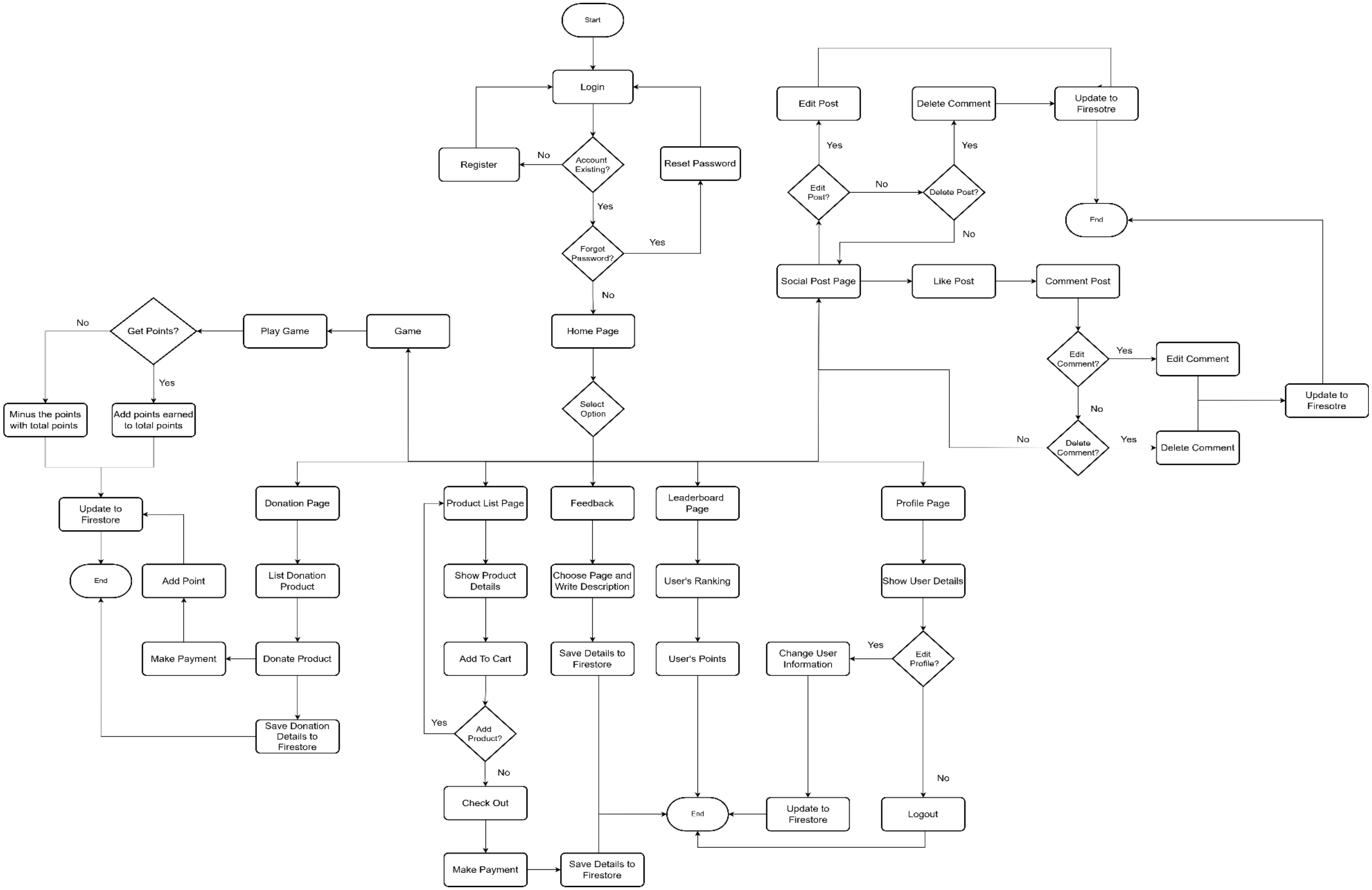


Figure 4.2.1 User Flowchart

The Figure 4.2.1 above is the flowchart from the user's perspective outlines the various actions they can take when interacting with the application, starting with the login process. If a user has an existing account, they can proceed by logging in, or if they forget their password, they are redirected to the reset password flow. For new users, there is a registration option to create an account.

Once logged in, the user arrives at the Home Page, where they are presented with different options. They can engage in social interactions, such as liking, commenting, and editing posts on the Social Post Page. If a user chooses to interact with a post, they are given the ability to edit or delete their comments, with the option to update the changes to Firestore for storage.

In the gamification aspect, users can play a game to earn points. Based on the outcome, points are either added or subtracted from their total points. This total is updated to Firestore for persistent tracking of the user’s progress and ranking on the Leaderboard Page.

For users interested in making donations, there is a Donation Page where they can list products for donation, view details of available donation items, and proceed to donate by saving the details to Firestore. Similarly, users can browse through a Product List Page, view product details, add items to their cart, and proceed to checkout, completing the purchase and saving the transaction record to Firestore.

Users can also manage their profile information on the Profile Page, where they can view and edit their personal details. After making changes, the profile data is updated in Firestore. Finally, when the user is done, they have the option to log out, securely ending the session.

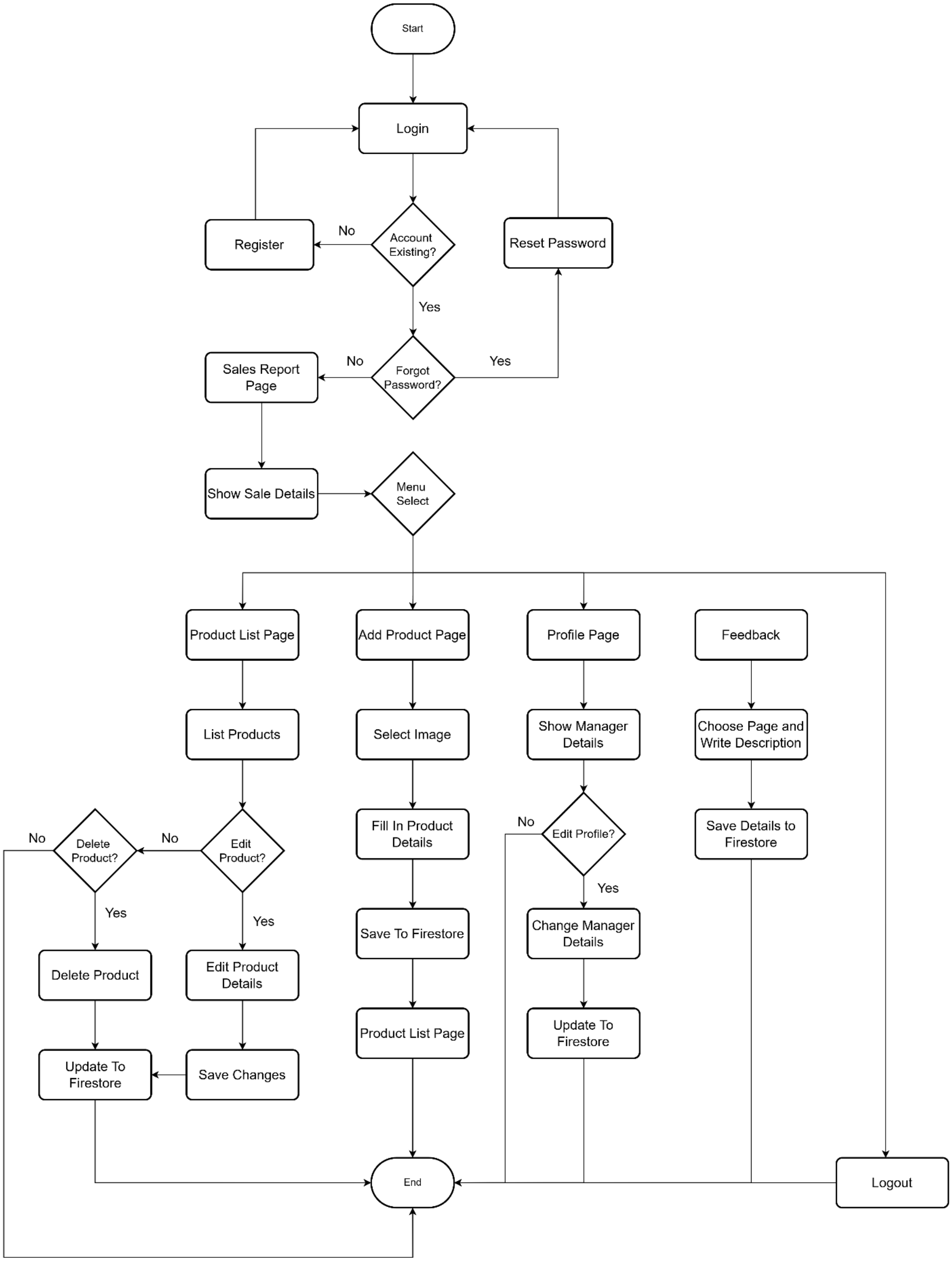


Figure 4.2.2 Manager Flowchart

The Figure 4.2.2 above show the flowchart provided from the manager's perspective outlines the various actions they can take when interacting with the application, beginning with the login process. If the manager has an existing account, they can proceed by logging in. If they forget their password, they can access the reset password flow. For new managers, a registration option is available to create a new account.

Once logged in, the manager arrives at Sales Report page. They can view sales data through the Sales Report Page, which allows them to access detailed information about sales activities via the click the bar of the bar chart based on month.

Managers also have full control over product management. By accessing the Product List Page, they can view all available products, and further actions include the ability to either delete or edit product details. If a product is deleted, the changes are directly updated in Firestore. If they choose to edit, they can modify the product details, save those changes, and ensure the updates are reflected in Firestore. Additionally, managers can add new products by selecting the Add Product Page, uploading images, filling in product details, and saving the new product information to Firestore.

Profile management is another key feature available to managers. By navigating to the Profile Page, they can view and update their personal information. If edits are made, the updated information is saved and stored in Firestore.

Additionally, the manager can leave feedback by choosing a relevant page and writing a description, with the details being saved to Firestore. When all tasks are completed, the manager has the option to securely log out, ending their session.

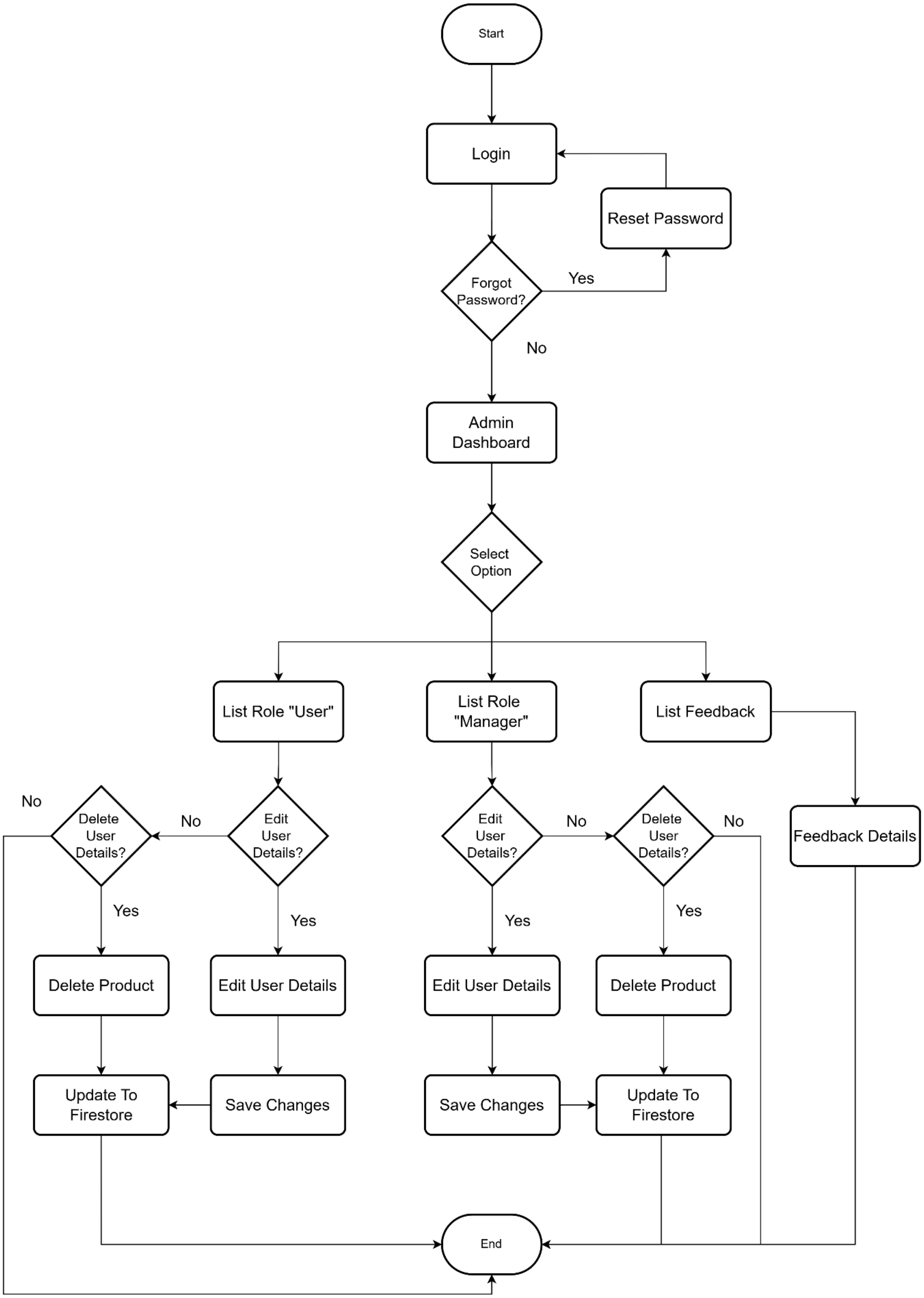


Figure 4.2.3 Admin Flowchart

The Figure 4.2.3 show the flowchart of the admin's perspective and outlines the various actions they can take when interacting with the application, beginning with the login process. If the admin has an existing account, they can proceed by logging in. In case they forget their password, they are redirected to the reset password flow.

Once logged in, the admin is taken to the Admin Dashboard, where they are presented with several management options. The admin can select different roles and manage user data. For example, they can choose to manage either Users or Managers by accessing the list of Role "User" or "Manager" sections.

In each role management section, the admin can either delete or edit user’s and manager’s details. If a user’s details are deleted, the information is updated in Firestore. Similarly, if the admin chooses to edit manager’s details, the modified information is saved to Firestore, ensuring that the system stays updated with the latest data.

The admin can also manage feedback from users and managers by selecting the Feedback option. Here, they can review Feedback Details, and any updates or changes from users or managers to feedback are saved in Firestore for storage.

Finally, after completing their tasks, the admin has the option to log out, securely ending their session and leaving the system in a maintained state.

4.3 Firestore Database Model

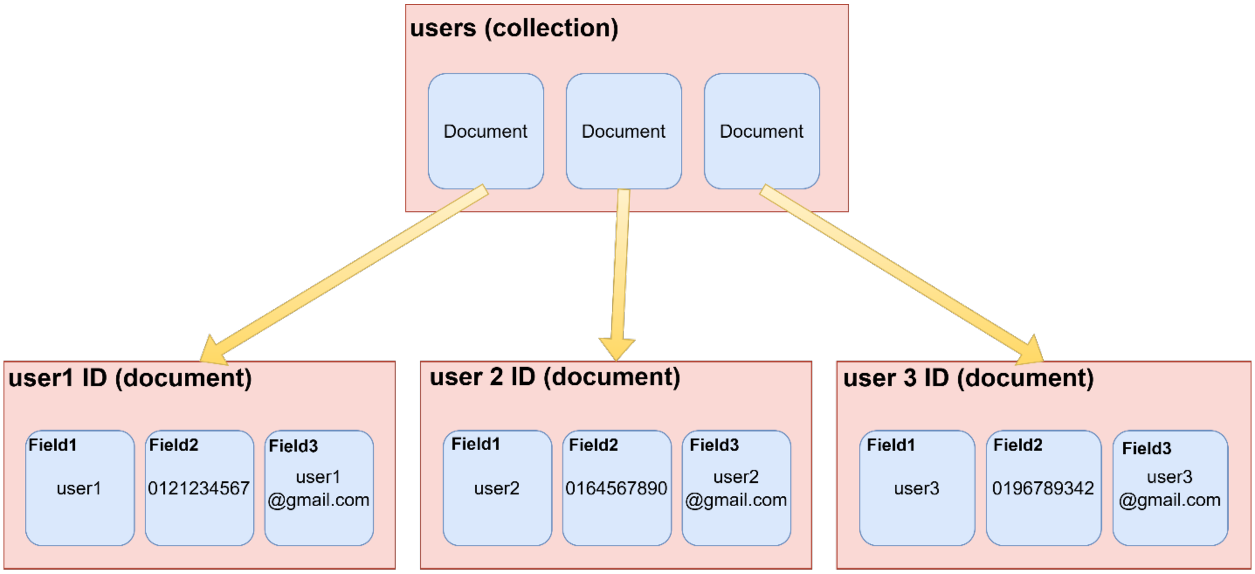


Figure 4.3.1 Firestore Database Model Example

Firestore is a NoSQL, document-oriented database that organizes data into collections and documents, as Figure 16 above. In this model, the top-level container is a collection named "users". Each collection groups related documents, which in this case represent individual users. Within the "users" collection, there are three documents, each identified by a unique user ID. These documents are akin to records in a traditional database.

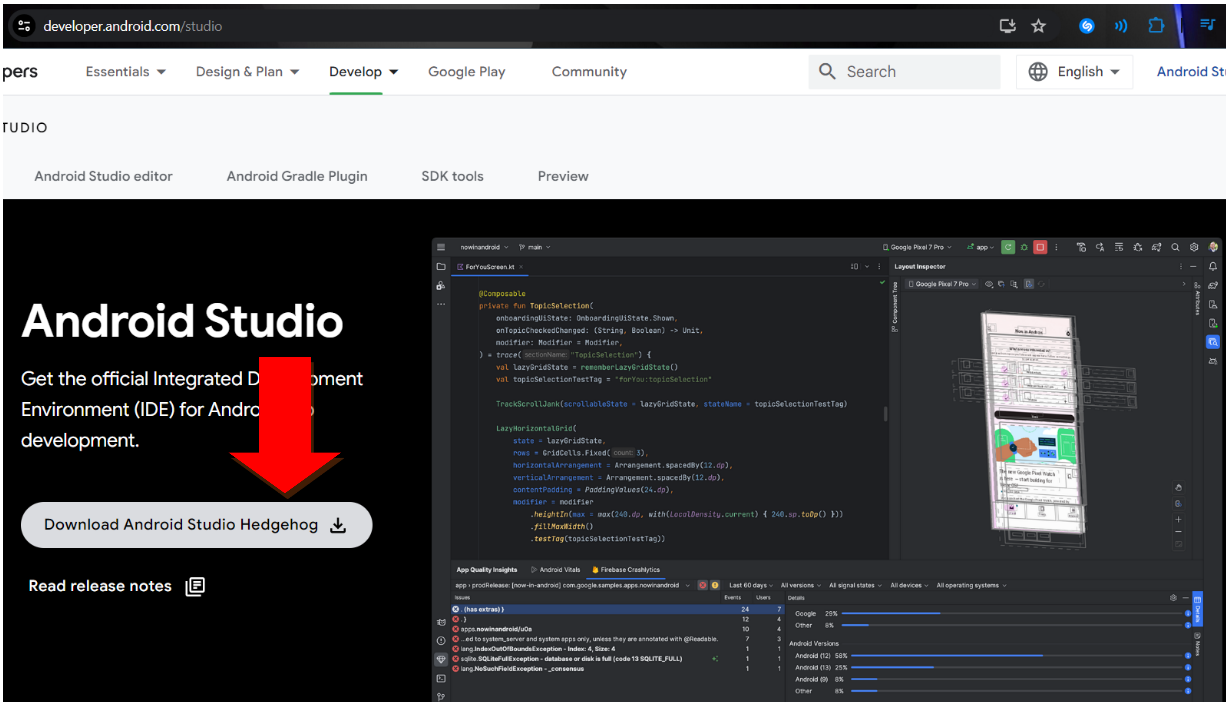
Each document contains several fields that store different types of data. For example, in the first document labelled "user 1 ID", there are three fields: Field1 contains a numerical value Field2 contains a text value, and Field3 contains an email address. This structure allows for a flexible schema where each document can have different fields or even different data types for the same field.

The hierarchical nature of Firestore, where collections contain documents and documents contain fields, enables efficient data organization and scalability. This model is particularly useful for applications requiring real-time updates and offline capabilities, making it ideal for mobile and web applications. The flexibility and scalability of Firestore's database model allow developers to easily manage and store large amounts of data while adapting to changing requirements.

CHAPTER 5 System Implementation

5.1 Software Setup

Android Studio Installation



Android Studio

Get the official Integrated D[...]ment Environment (IDE) for Andro[...] development.

```
@Composable
private fun TopicSelection(
    onboardingUiState: OnboardingUiState.Shown,
    onTopicCheckedChanged: (String, Boolean) -> Unit,
    modifier: Modifier = Modifier,
) = trace( sectionName: "TopicSelection") {
    val lazyGridState = rememberLazyGridState()
    val topicSelectionTestTag = "forYou:topicSelection"

    TrackScrollJank(scrollableState = lazyGridState, stateName = topicSelectionTestTag)

    LazyHorizontalGrid(
        state = lazyGridState,
        rows = GridCells.Fixed( count: 3),
        horizontalArrangement = Arrangement.spacedBy(12.dp),
        verticalArrangement = Arrangement.spacedBy(12.dp),
        contentPadding = PaddingValues(24.dp),
        modifier = modifier
            .heightIn(max = max(240.dp, with(LocalDensity.current) { 240.sp.toDp() }))
            .fillMaxWidth()
            .testTag(topicSelectionTestTag))
```

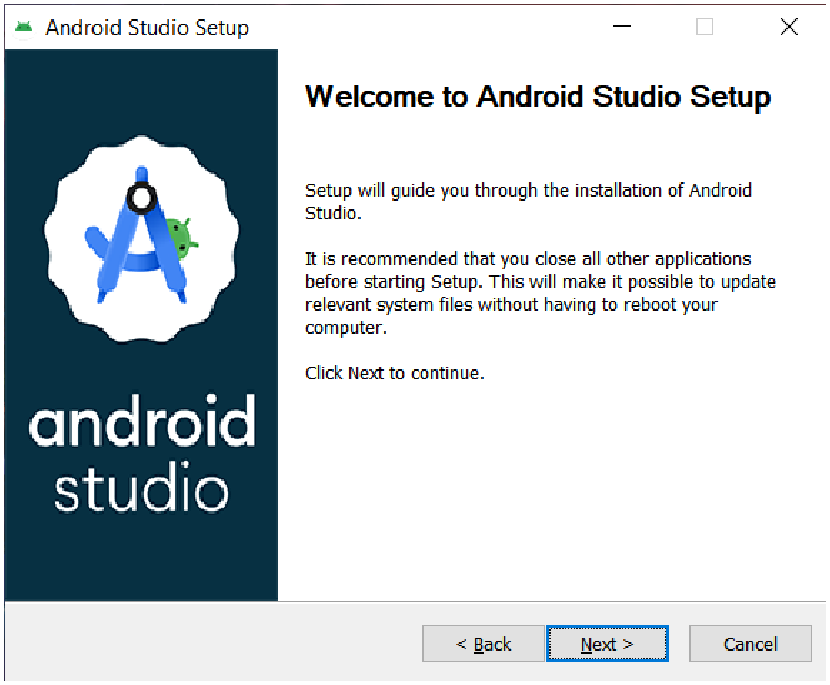
app › prodRelease: [now-in-android] com.google.samples.apps.nowinandroid

Issues	Events	Users
. (has extras) }	24	7
.}	12	4
apps.nowinandroid/u0a	10	4
...ed to system_server and system apps only, unless they are annotated with @Readable.	7	3
lang.IndexOutOfBoundsException - Index: 4, Size: 4	1	1
sqlite.SQLiteFullException - database or disk is full (code 13 SQLITE_FULL)	1	1
lang.NoSuchFieldException - _consensus	1	1

Details	
Google	29%
Other	8%
Android Versions	
Android (12)	58%
Android (13)	25%
Android (9)	8%
Other	8%

Figure 5.1.1 Android Studio Webpage

Click the "Download Android Studio Hedgehog" button to download the Android Studio installer.



Android Studio Setup

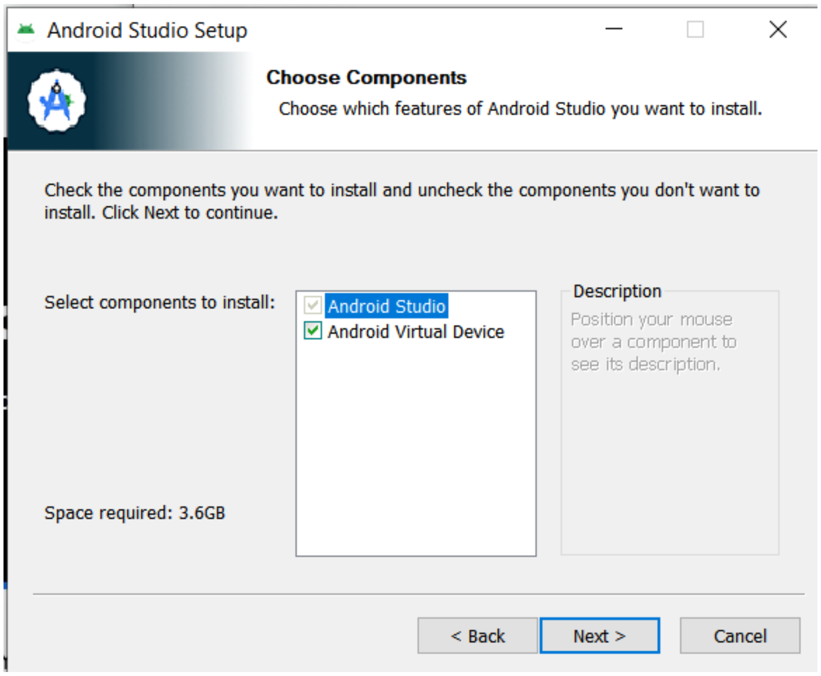
Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 5.1.2 Android Studio Setup 1



Android Studio Setup

Choose Components

Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

☑ Android Studio

☑ Android Virtual Device

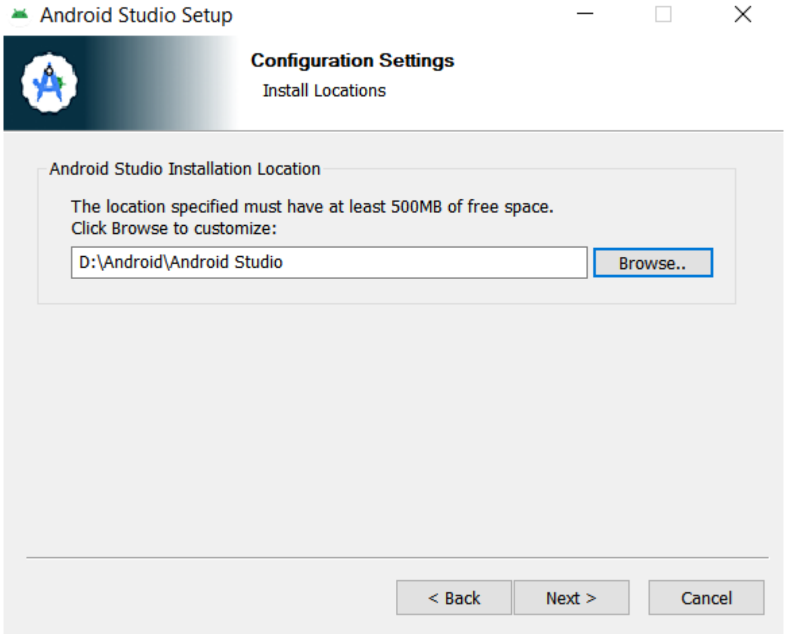
Space required: 3.6GB

Description

Position your mouse over a component to see its description.

Figure 5.1.3 Android Studio Setup 2

After done downloading the Android Studio installer, click and run the installer, it will pop up the installer wizard as shown in Figure 9 and 10. Click Next and make sure tick the "Android Virtual Device".



Android Studio Setup

Configuration Settings

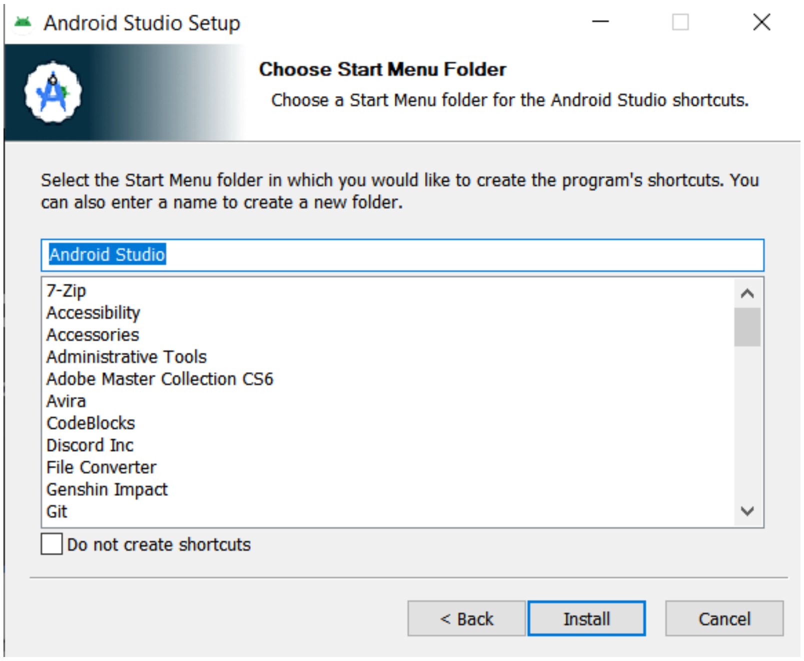
Install Locations

Android Studio Installation Location

The location specified must have at least 500MB of free space.
Click Browse to customize:

D:\Android\Android Studio

Figure 5.1.4 Android Studio installation location selection



Android Studio Setup

Choose Start Menu Folder

Choose a Start Menu folder for the Android Studio shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

Android Studio

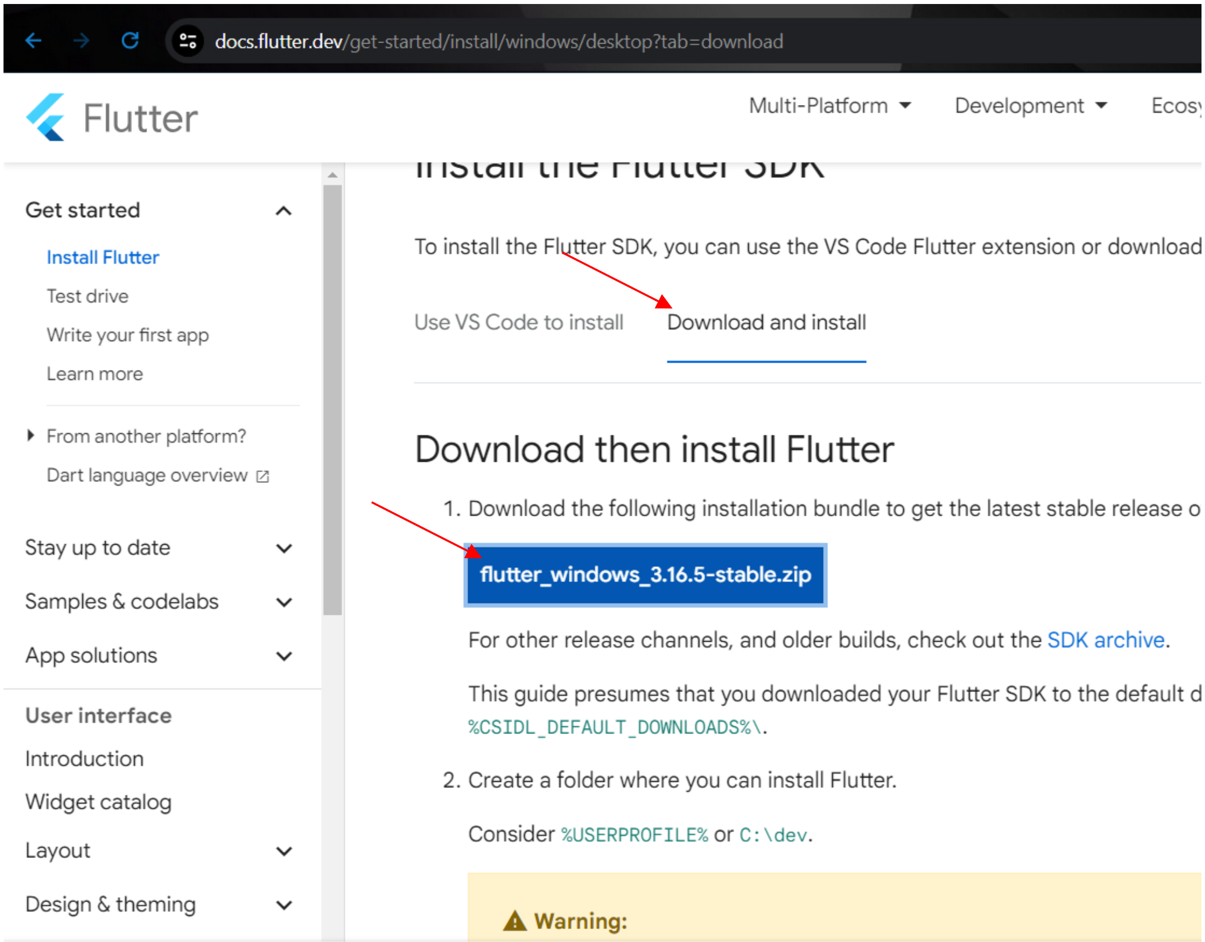
7-Zip
Accessibility
Accessories
Administrative Tools
Adobe Master Collection CS6
Avira
CodeBlocks
Discord Inc
File Converter
Genshin Impact
Git

□ Do not create shortcuts

Figure 5.1.5 Create shortcut

Select the location for install Android Studio and create a shortcut in the Start Menu folder.

Flutter Installation



docs.flutter.dev/get-started/install/windows/desktop?tab=download

Install the Flutter SDK

To install the Flutter SDK, you can use the VS Code Flutter extension or download

Download then install Flutter

1. Download the following installation bundle to get the latest stable release o

   For other release channels, and older builds, check out the SDK archive.

   This guide presumes that you downloaded your Flutter SDK to the default d `%CSIDL_DEFAULT_DOWNLOADS%\`.

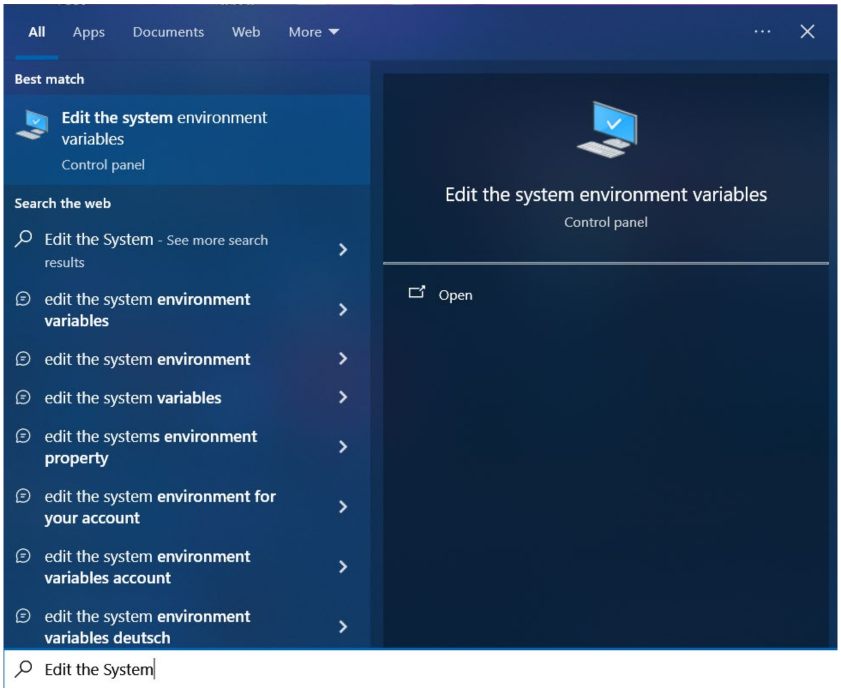
2. Create a folder where you can install Flutter.

   Consider `%USERPROFILE%` or `C:\dev`.

   ⚠ Warning:

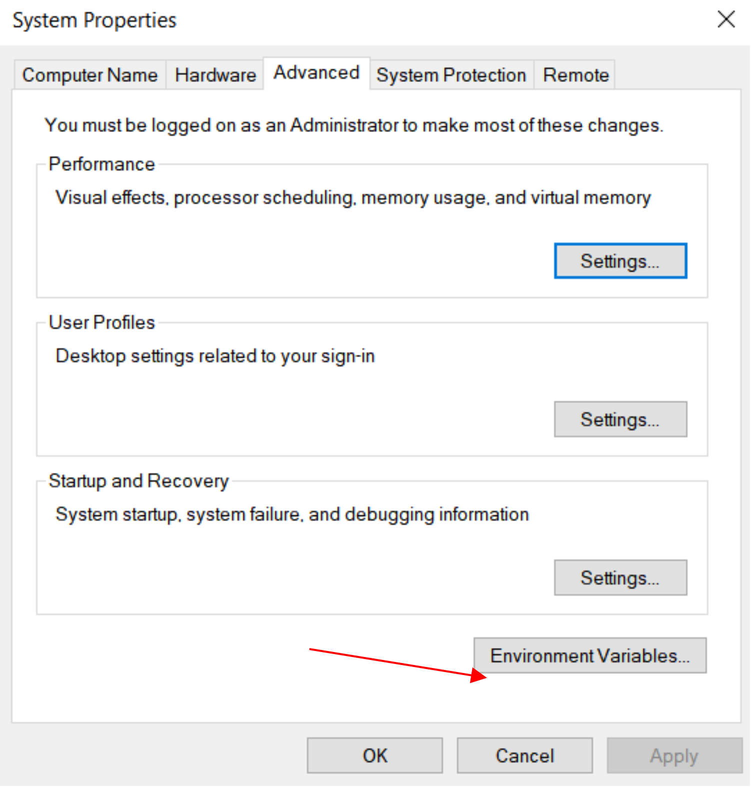
Figure 5.1.6 Flutter Webpage

First, click the "Download and install" to download the Flutter SDK, and click the "flutter_windows_3.16.5-stable.zip" to download the Flutter SDK zip file. Extract the zip in any directory except the directory that requires elevated privileges.



All
Apps
Documents
Web
More
Best match
Edit the system environment variables
Control panel
Search the web
Edit the System - See more search results
edit the system environment variables
edit the system environment
edit the system variables
edit the systems environment property
edit the system environment for your account
edit the system environment variables account
edit the system environment variables deutsch
Edit the system environment variables
Control panel
Open
Edit the System

Figure 5.1.7 Search Edit the system environment variables



System Properties

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

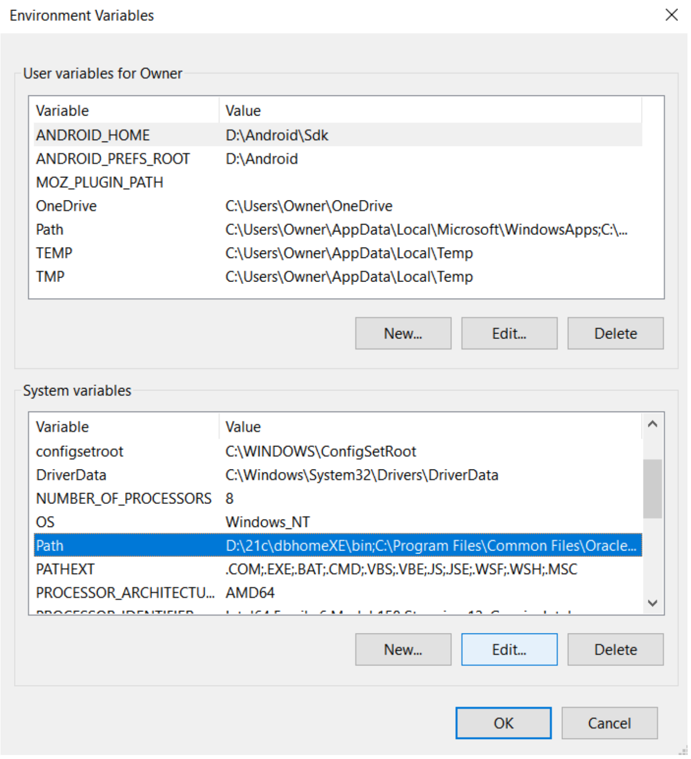
Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

Figure 5.1.8 System properties "Advanced" tab

After done extracting the zip file, go to search the Edit the system environment variables and it will show the system properties as shown in Figure 15, after that click the “Environment Variables”.

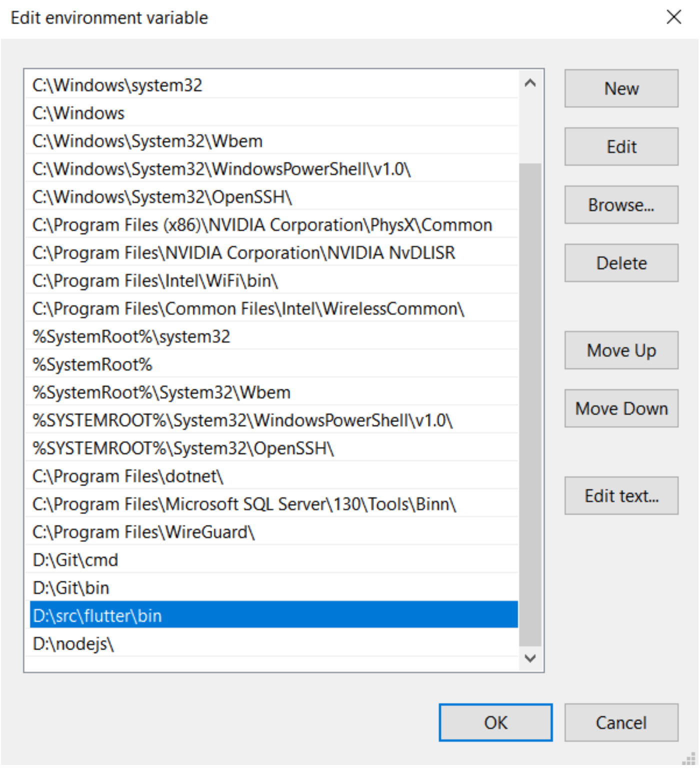


Variable	Value
ANDROID_HOME	D:\Android\Sdk
ANDROID_PREFS_ROOT	D:\Android
MOZ_PLUGIN_PATH	
OneDrive	C:\Users\Owner\OneDrive
Path	C:\Users\Owner\AppData\Local\Microsoft\WindowsApps;C:\...
TEMP	C:\Users\Owner\AppData\Local\Temp
TMP	C:\Users\Owner\AppData\Local\Temp

Variable	Value
configsetroot	C:\WINDOWS\ConfigSetRoot
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	D:\21c\dbhomeXE\bin;C:\Program Files\Common Files\Oracle...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTU...	AMD64

Figure 5.1.9 Environment Variables

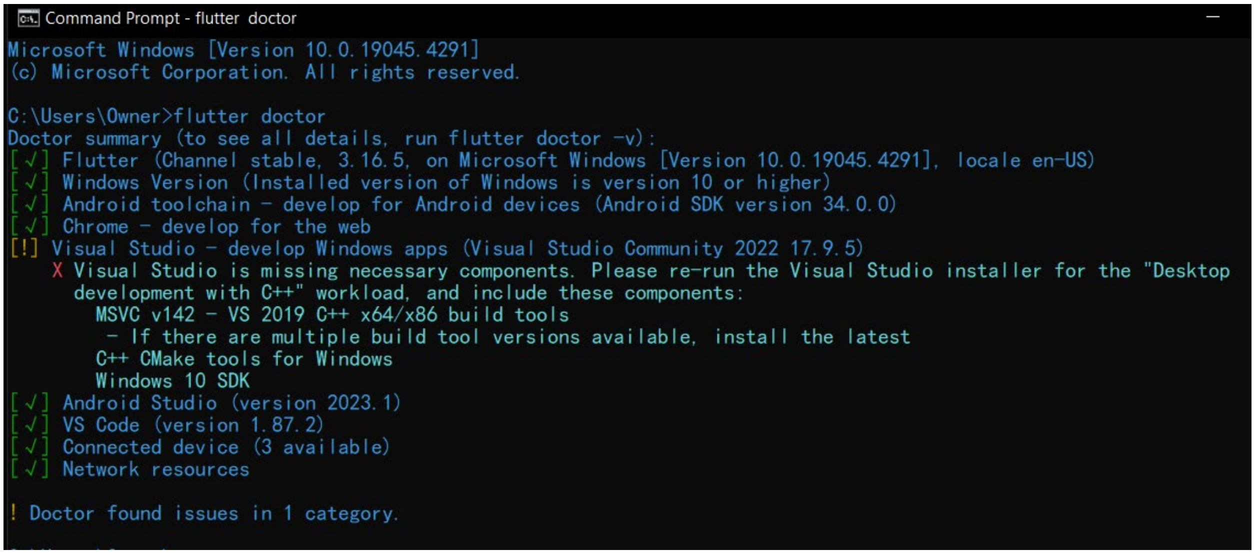
Select the Path variable and click Edit.



```
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Windows\System32\OpenSSH\
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
C:\Program Files\NVIDIA Corporation\NVIDIA NvDLISR
C:\Program Files\Intel\WiFi\bin\
C:\Program Files\Common Files\Intel\WirelessCommon\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\dotnet\
C:\Program Files\Microsoft SQL Server\130\Tools\Binn\
C:\Program Files\WireGuard\
D:\Git\cmd
D:\Git\bin
D:\src\flutter\bin
D:\nodejs\
```

Figure 5.1.10 Edit environment variable

Add the path of where the extracted Flutter folder by click the “Browse” and add the bin folder in the flutter folder.



```
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Owner>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.16.5, on Microsoft Windows [Version 10.0.19045.4291], locale en-US)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[√] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.9.5)
    X Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop
      development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
         - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[√] Android Studio (version 2023.1)
[√] VS Code (version 1.87.2)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 1 category.
```

Figure 5.1.11 Command Prompt

Open the CMD and type command “flutter doctor” for checking the Flutter install successfully.

Unity Installation

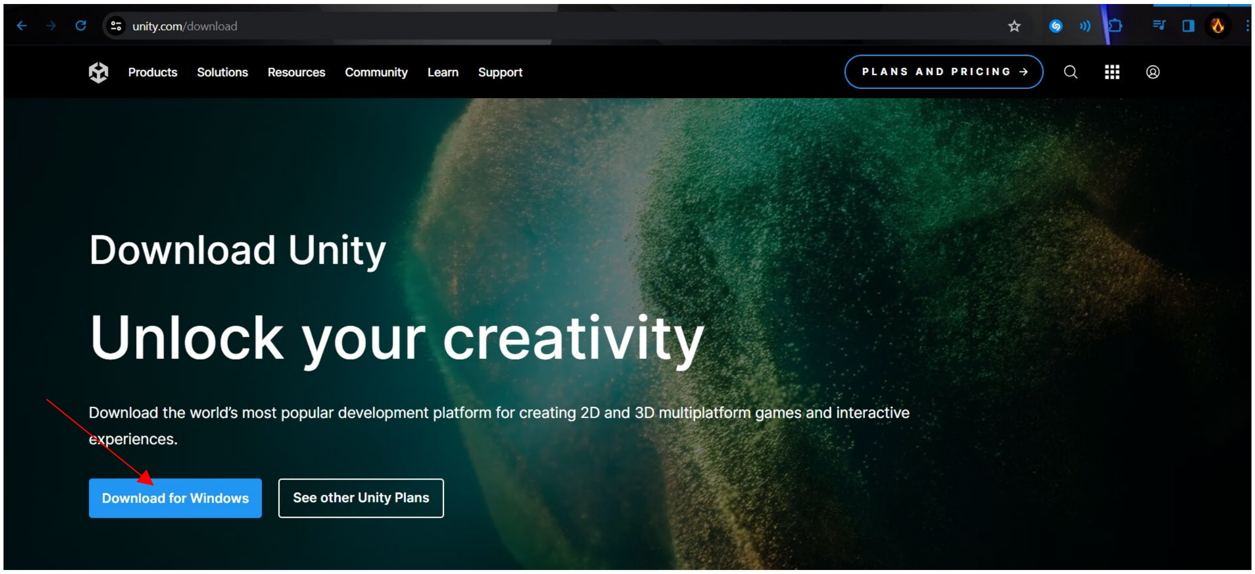
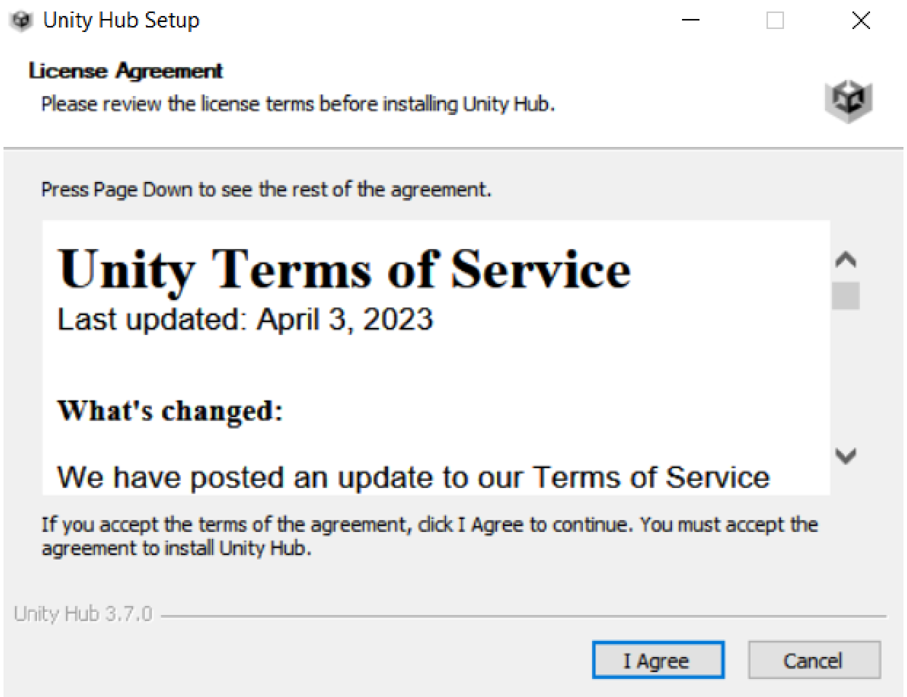


Figure 5.1.12 Unity Hub Download Webpage

First, visit the Unity Hub download website and click the "Download for Windows" to download the installer of Unity Hub.


Unity Hub Setup

License Agreement

Please review the license terms before installing Unity Hub.

Press Page Down to see the rest of the agreement.

Unity Terms of Service

Last updated: April 3, 2023

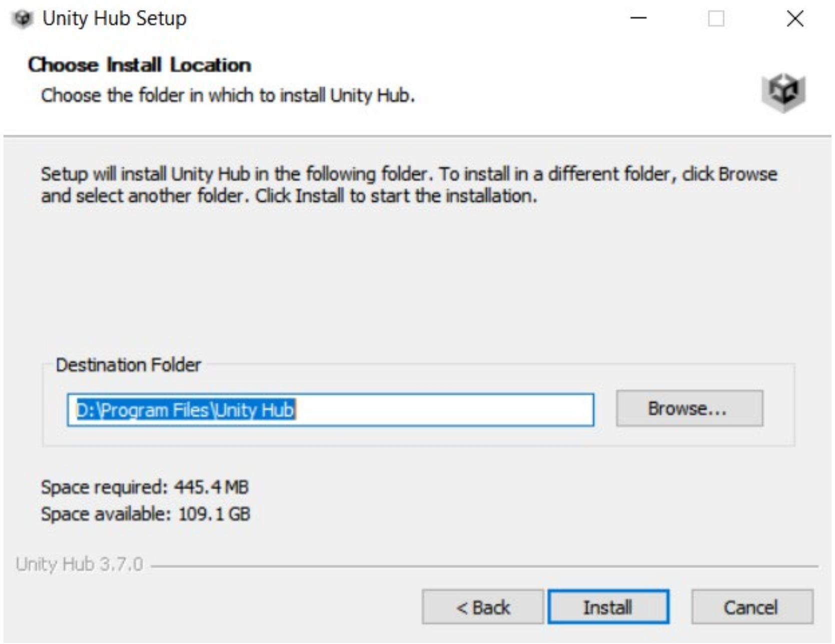
What's changed:

We have posted an update to our Terms of Service

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Unity Hub.

Unity Hub 3.7.0

Figure 5.1.13 Unity Setup 1



Choose Install Location

Choose the folder in which to install Unity Hub.

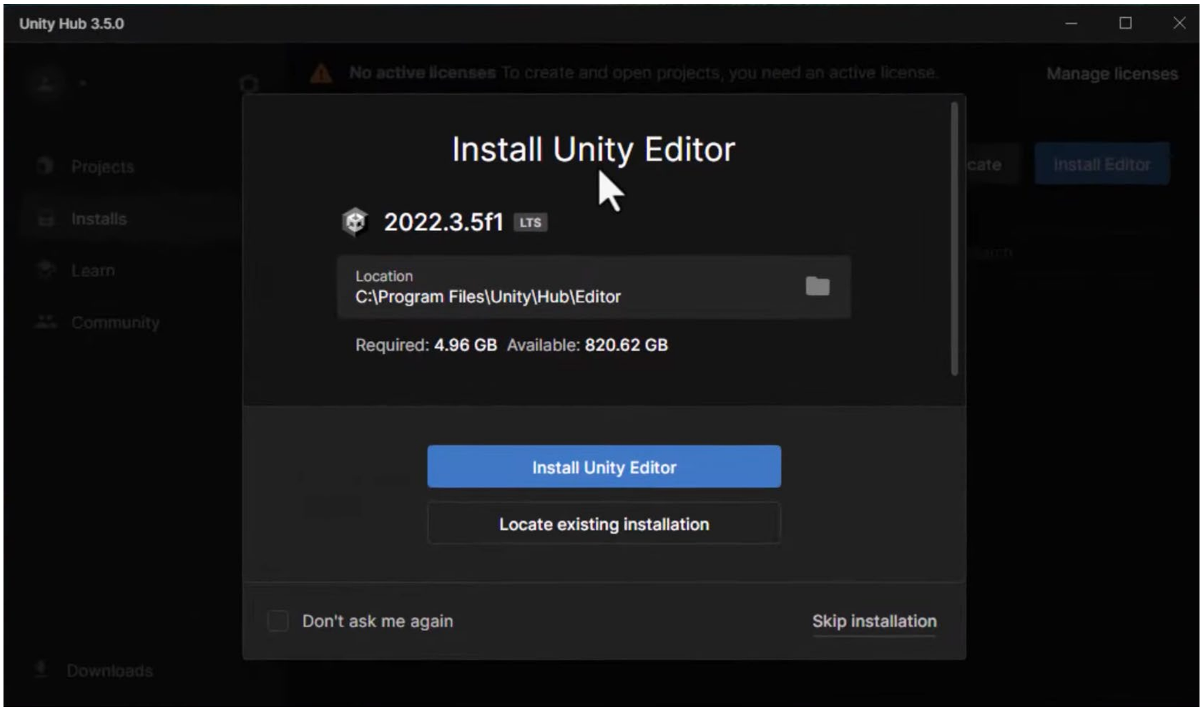
Setup will install Unity Hub in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: D:\Program Files\Unity Hub

Space required: 445.4 MB
Space available: 109.1 GB

Figure 5.1.14 Unity Hub Setup 2

After done downloading the Unity Hub installer, open the installer will popup the wizard that shows the Unity terms of service, click agree to next wizard which is select the location for install the Unity Hub.



No active licenses To create and open projects, you need an active license.

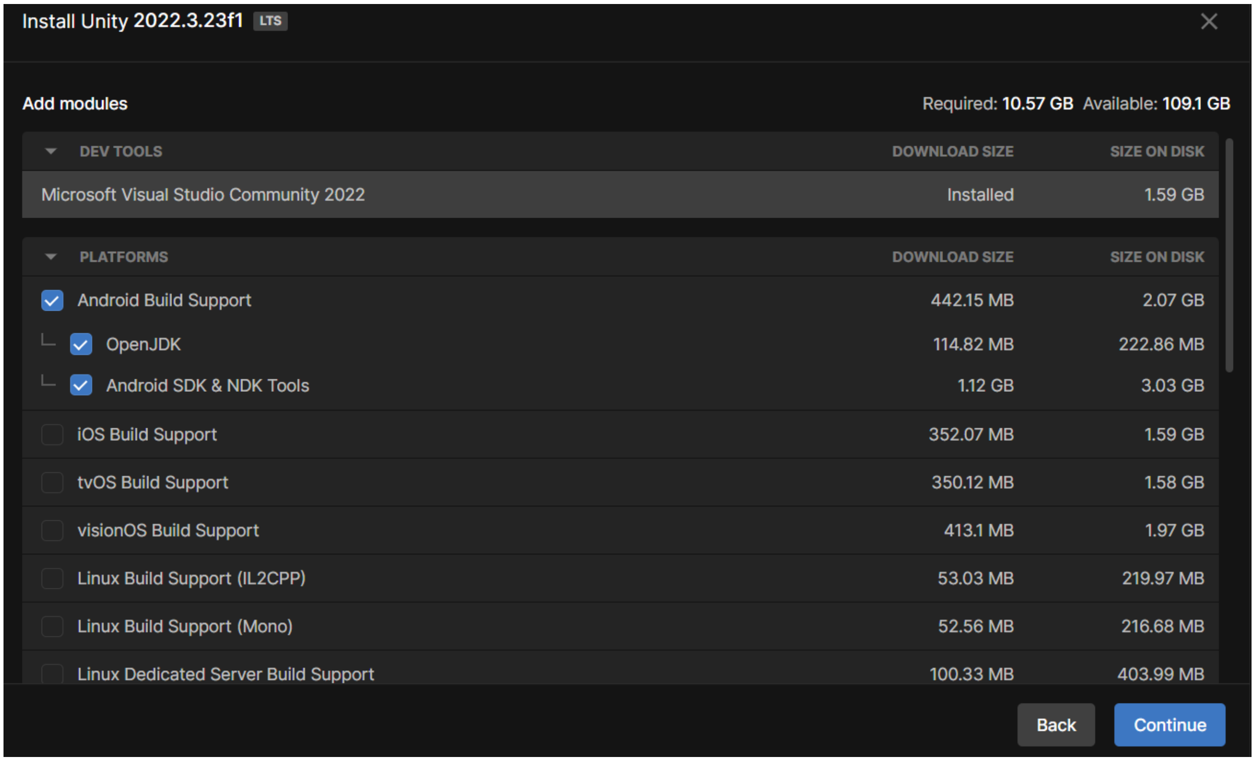
Install Unity Editor

2022.3.5f1 LTS

Location: C:\Program Files\Unity\Hub\Editor

Required: 4.96 GB Available: 820.62 GB

Figure 5.1.15 Unity Hub Installs Page



Install Unity 2022.3.23f1 LTS

Add modules

Required: **10.57 GB** Available: **109.1 GB**

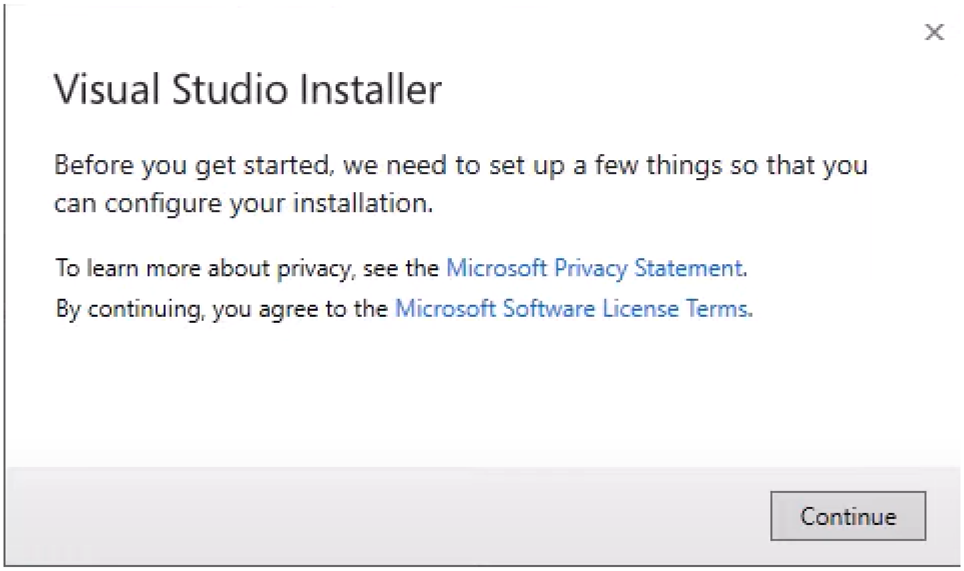
DEV TOOLS	DOWNLOAD SIZE	SIZE ON DISK
Microsoft Visual Studio Community 2022	Installed	1.59 GB

PLATFORMS	DOWNLOAD SIZE	SIZE ON DISK
☑ Android Build Support	442.15 MB	2.07 GB
└ ☑ OpenJDK	114.82 MB	222.86 MB
└ ☑ Android SDK & NDK Tools	1.12 GB	3.03 GB
□ iOS Build Support	352.07 MB	1.59 GB
□ tvOS Build Support	350.12 MB	1.58 GB
□ visionOS Build Support	413.1 MB	1.97 GB
□ Linux Build Support (IL2CPP)	53.03 MB	219.97 MB
□ Linux Build Support (Mono)	52.56 MB	216.68 MB
□ Linux Dedicated Server Build Support	100.33 MB	403.99 MB

Figure 5.1.16 Editor Configuration

After done the installation, open the Unity Hub and it will show a recommend Editor version, click Install Unity Editor, select the Dev tools and Platforms and click continue to start installing.

Visual Studio 2022 Installation



Visual Studio Installer

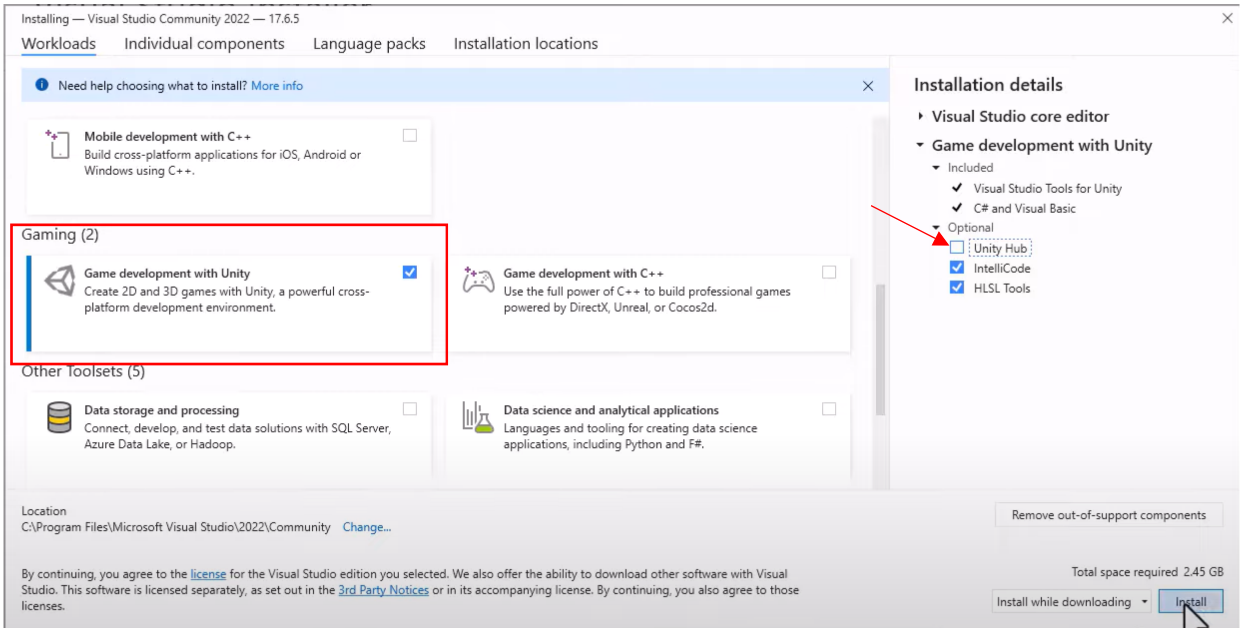
Before you get started, we need to set up a few things so that you can configure your installation.

To learn more about privacy, see the Microsoft Privacy Statement.

By continuing, you agree to the Microsoft Software License Terms.

Figure 5.1.17 Visual Studio Wizard

After downloading all the package that selected in the Unity Editor installation, it has a popup from Visual Studio Installer, and click Continue.



Installing — Visual Studio Community 2022 — 17.6.5

Workloads | Individual components | Language packs | Installation locations

Need help choosing what to install? More info

□ Mobile development with C++
Build cross-platform applications for iOS, Android or Windows using C++.

Gaming (2)

☑ Game development with Unity
Create 2D and 3D games with Unity, a powerful cross-platform development environment.

□ Game development with C++
Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Other Toolsets (5)

□ Data storage and processing
Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.

□ Data science and analytical applications
Languages and tooling for creating data science applications, including Python and F#.

Installation details

- Visual Studio core editor
- Game development with Unity
  - Included
    - ✓ Visual Studio Tools for Unity
    - ✓ C# and Visual Basic
  - Optional
    - □ Unity Hub
    - ☑ IntelliCode
    - ☑ HLSL Tools

Location
C:\Program Files\Microsoft Visual Studio\2022\Community Change...

Remove out-of-support components

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

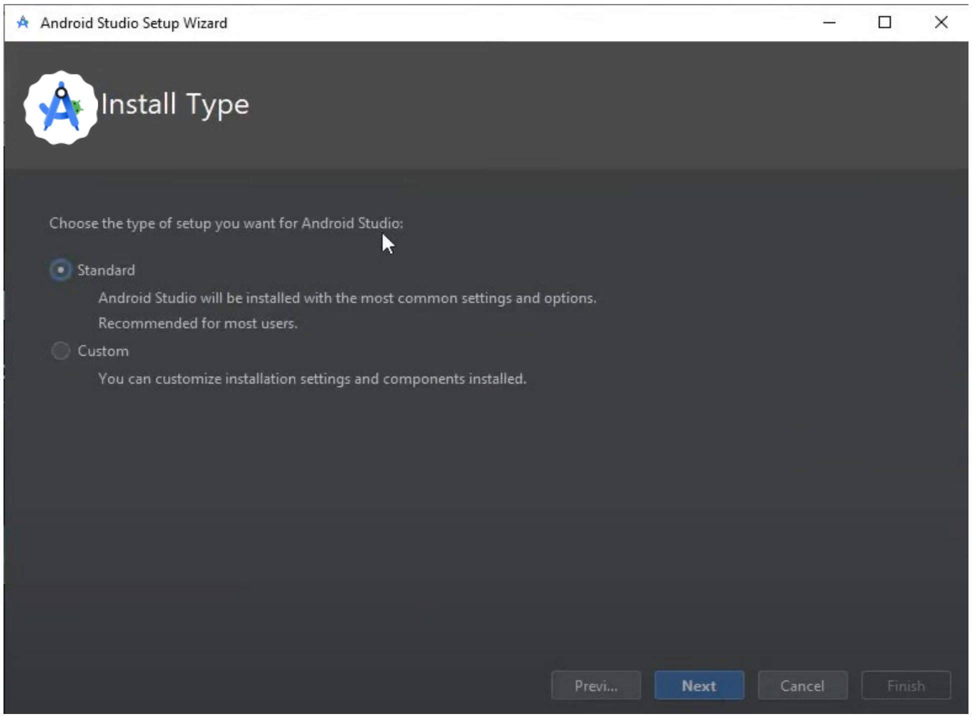
Total space required 2.45 GB

Figure 5.1.18 Visual Studio Workloads

Choose the Game Development with Unity and uncheck the Unity Hub in the installation details, after that click the Install button to start installing.

5.2 Settings and Configuration

Android Studio Configuration



Android Studio Setup Wizard

Install Type

Choose the type of setup you want for Android Studio:

● Standard

Android Studio will be installed with the most common settings and options.

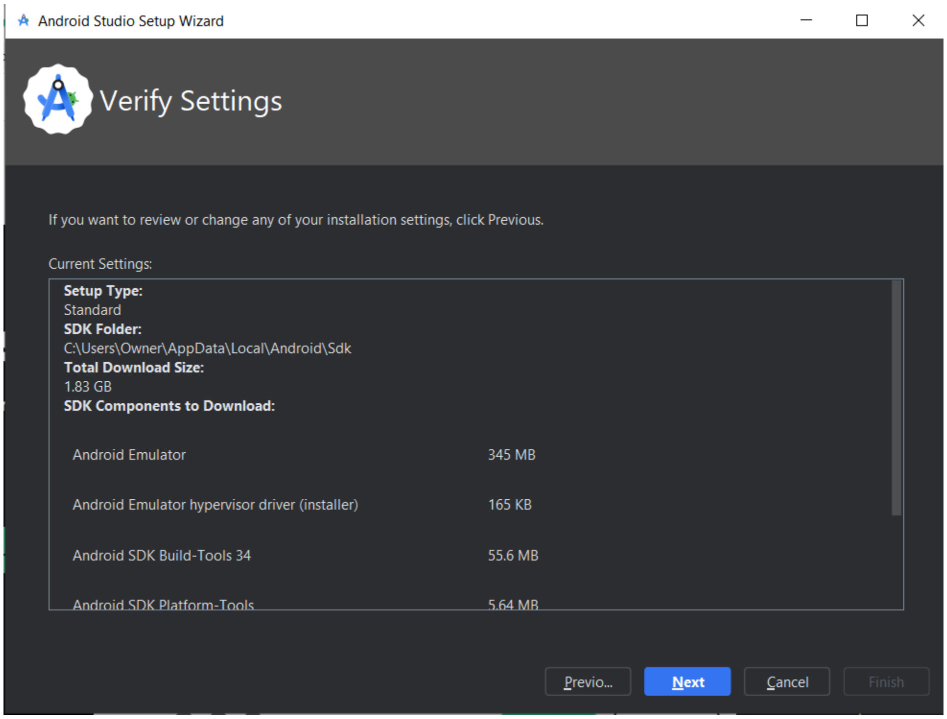
Recommended for most users.

○ Custom

You can customize installation settings and components installed.

Figure 5.2.1 Android Studio Install Type

Choose the Standard type of setup for Android Studio.



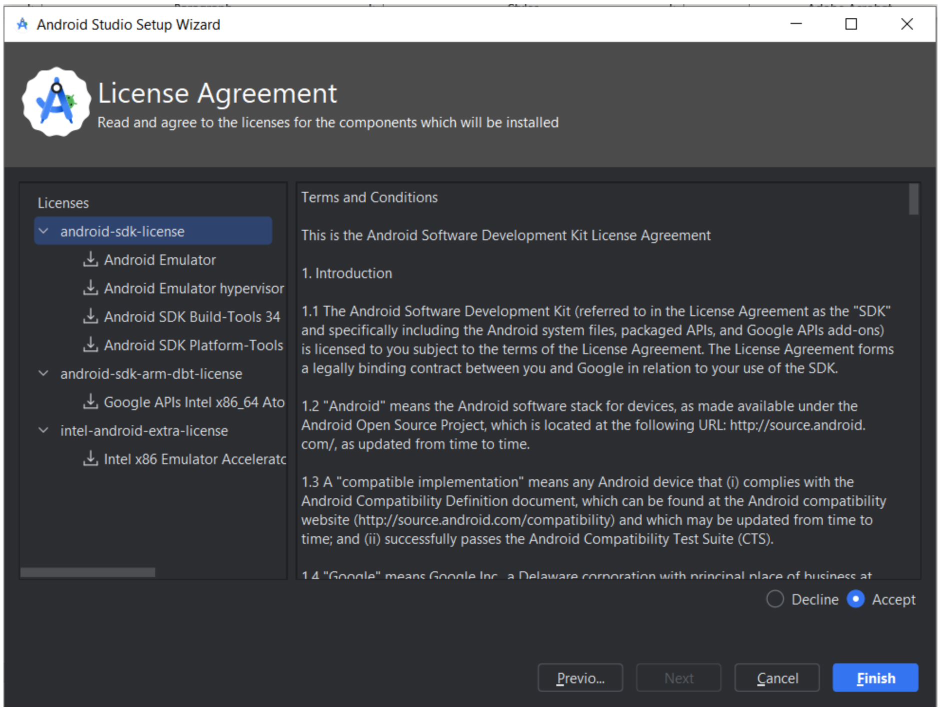
If you want to review or change any of your installation settings, click Previous.

Current Settings:

Setup Type:
Standard
SDK Folder:
C:\Users\Owner\AppData\Local\Android\Sdk
Total Download Size:
1.83 GB
SDK Components to Download:

Android Emulator	345 MB
Android Emulator hypervisor driver (installer)	165 KB
Android SDK Build-Tools 34	55.6 MB
Android SDK Platform-Tools	5.64 MB

Figure 5.2.2 Verify Setting


Android Studio Setup Wizard
License Agreement
Read and agree to the licenses for the components which will be installed
Licenses
android-sdk-license
Android Emulator
Android Emulator hypervisor
Android SDK Build-Tools 34
Android SDK Platform-Tools
android-sdk-arm-dbt-license
Google APIs Intel x86_64 Ato
intel-android-extra-license
Intel x86 Emulator Acceleratc
○ Decline ● Accept
Previo...
Next
Cancel
Finish

Terms and Conditions

This is the Android Software Development Kit License Agreement

1. Introduction

1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

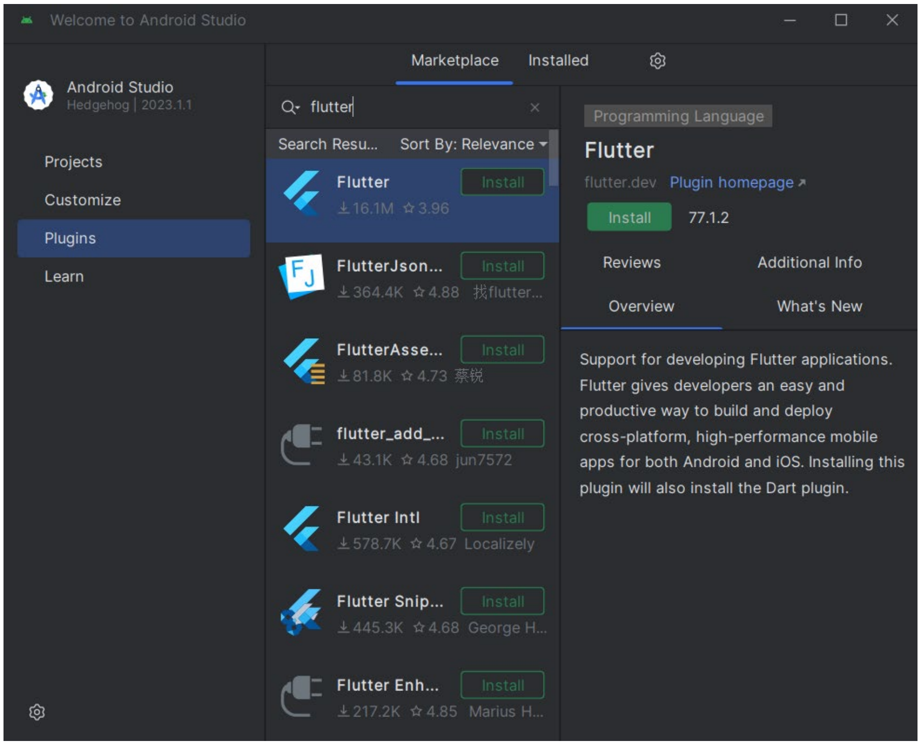
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android. com/, as updated from time to time.

1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).

1.4 "Google" means Google Inc., a Delaware corporation with principal place of business at

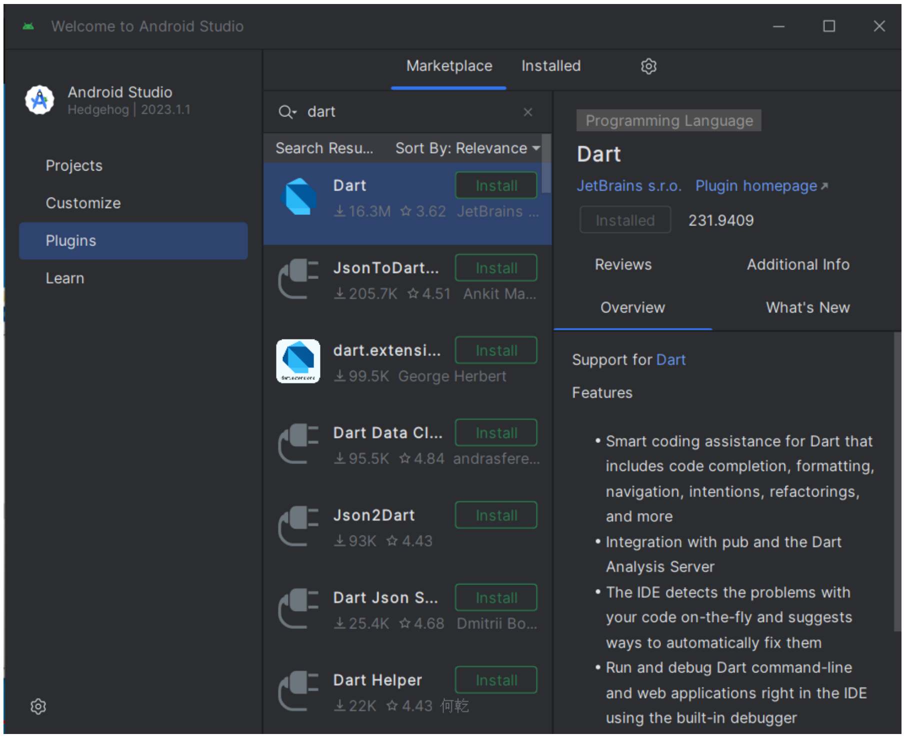
Figure 5.2.3 License Agreement

Check and Accept the License Agreement, after that click Finish to install the Licenses.



Support for developing Flutter applications. Flutter gives developers an easy and productive way to build and deploy cross-platform, high-performance mobile apps for both Android and iOS. Installing this plugin will also install the Dart plugin.

Figure 5.2.4 Flutter Plugin Installation



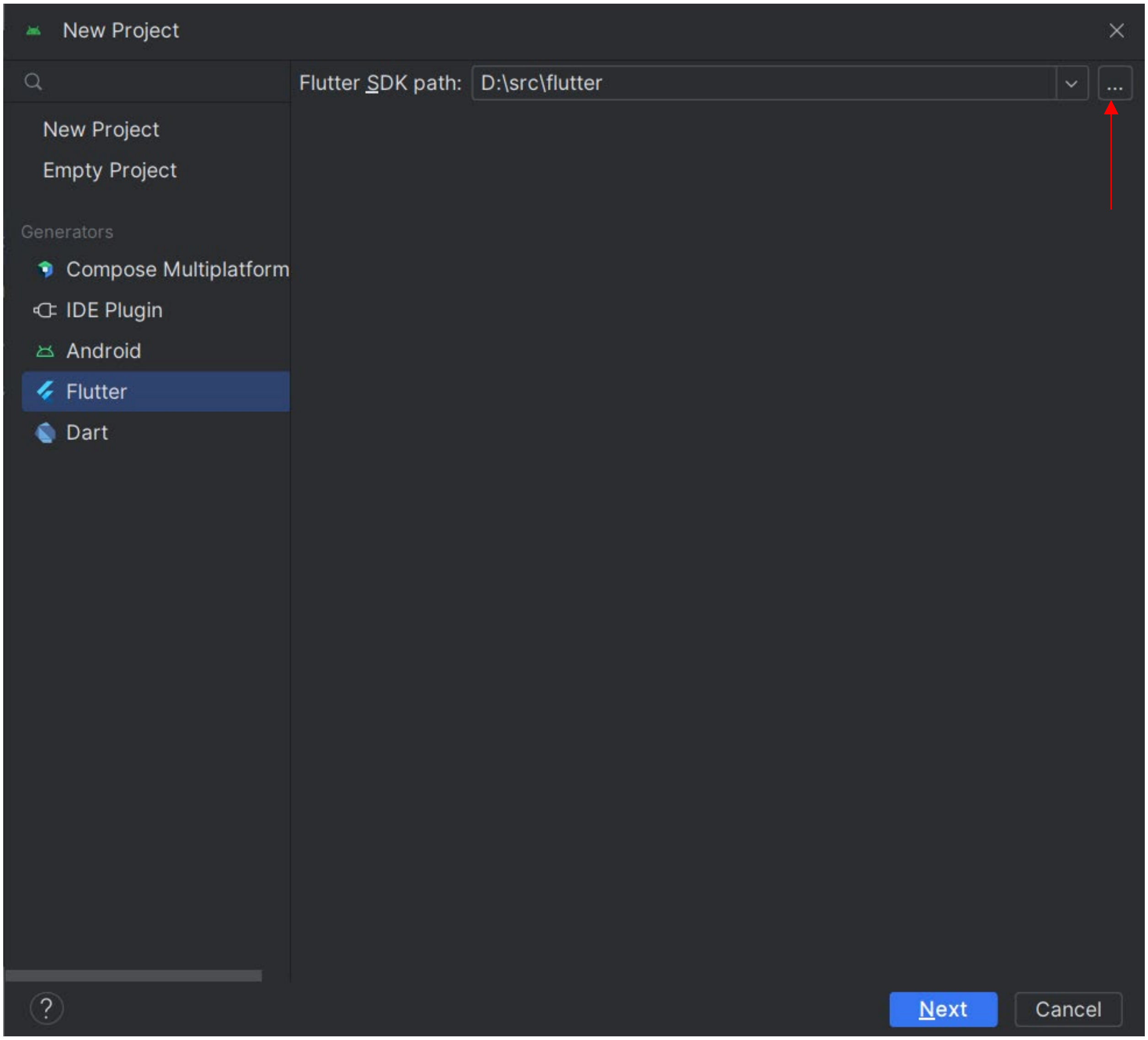
Support for Dart

Features

- Smart coding assistance for Dart that includes code completion, formatting, navigation, intentions, refactorings, and more
- Integration with pub and the Dart Analysis Server
- The IDE detects the problems with your code on-the-fly and suggests ways to automatically fix them
- Run and debug Dart command-line and web applications right in the IDE using the built-in debugger

Figure 5.2.5 Dart Installation

In the Marketplace of Android Studio, search the Flutter and Dart plugins and click Install to install these two plugins, these are the main plugins to develop the application.

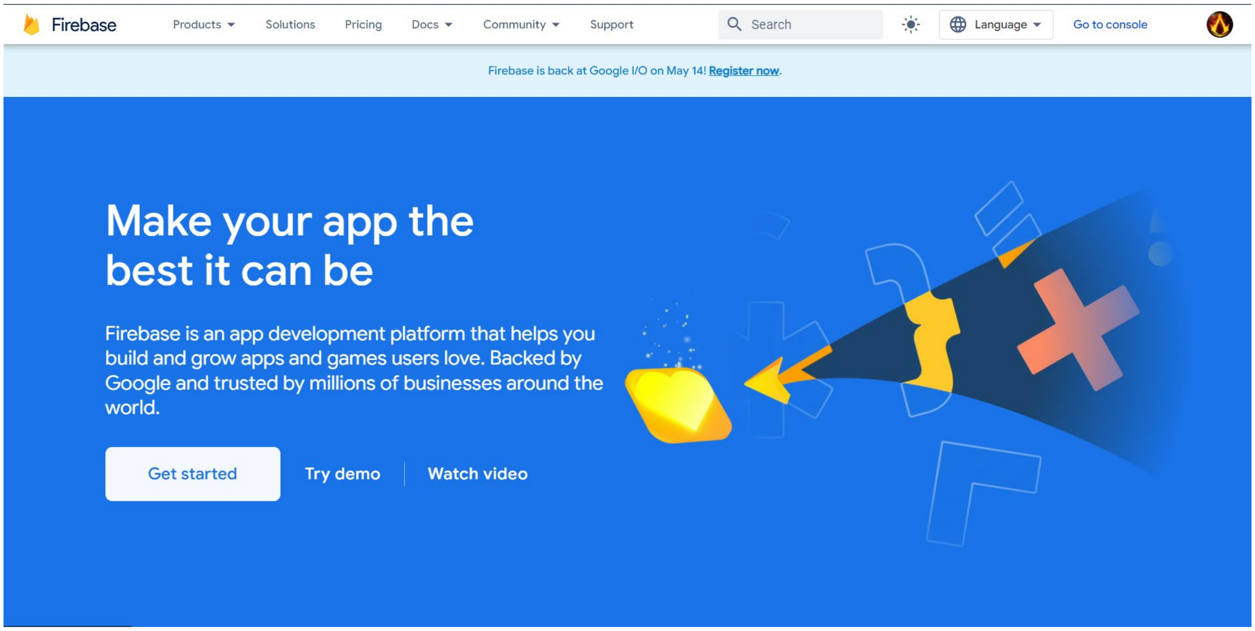


Flutter SDK path: D:\src\flutter

Figure 5.2.6 New Flutter Project Setting

Before creating a new Flutter project, it requires to select the Flutter SDK path, click the 3 dots for browse and select the Flutter SDK path.

Firebase Configuration

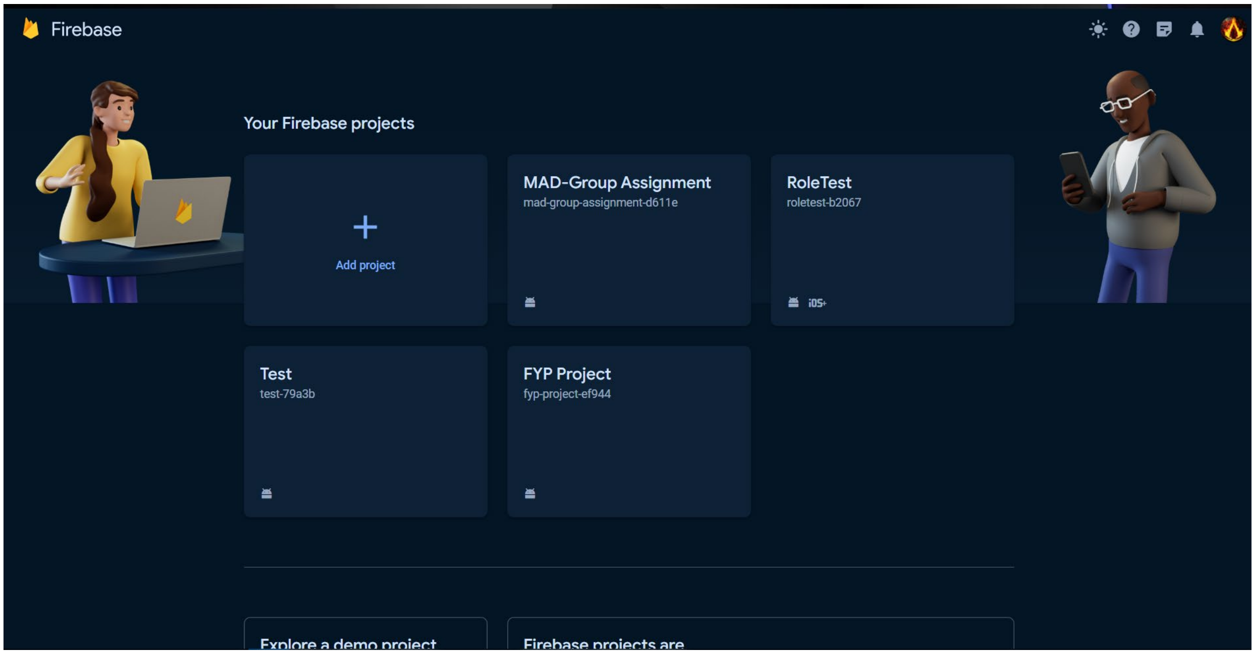


Firebase is back at Google I/O on May 14! Register now.

Make your app the best it can be

Firebase is an app development platform that helps you build and grow apps and games users love. Backed by Google and trusted by millions of businesses around the world.

Figure 5.2.7 Firebase Website



Your Firebase projects

MAD-Group Assignment
mad-group-assignment-d611e

RoleTest
roletest-b2067

Test
test-79a3b

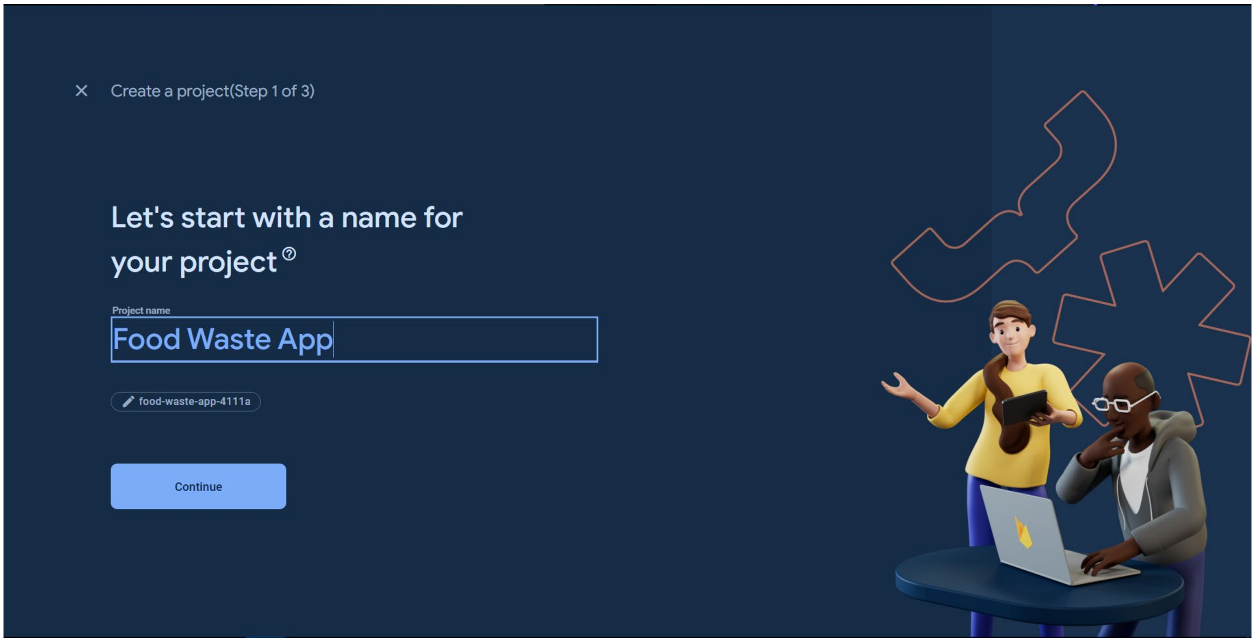
FYP Project
fyp-project-ef944

Explore a demo project

Firebase projects are

Figure 5.2.8 Firebase Console

Visit the Firebase website and click on the Get Started, login with account and access to Firebase console. Click on Add Project to create a new Firebase project for Flutter application.



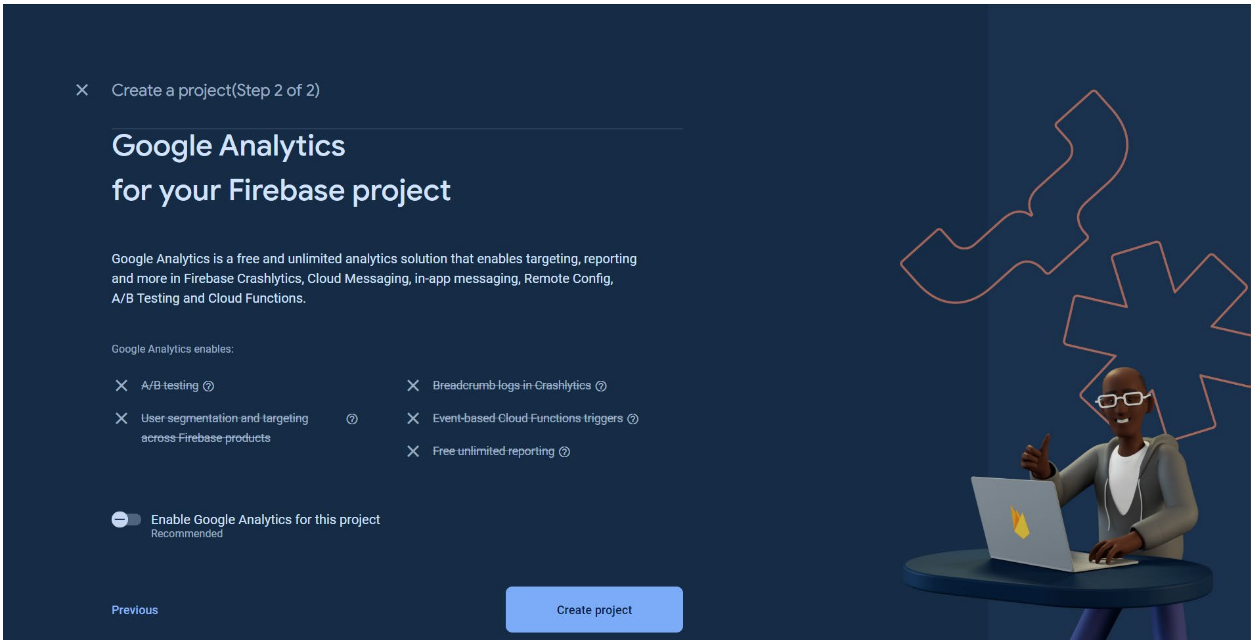
× Create a project(Step 1 of 3)

Let's start with a name for your project

Project name: Food Waste App

food-waste-app-4111a

Figure 5.2.9 Enter Project's Name



× Create a project(Step 2 of 2)

Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting and more in Firebase Crashlytics, Cloud Messaging, in-app messaging, Remote Config, A/B Testing and Cloud Functions.

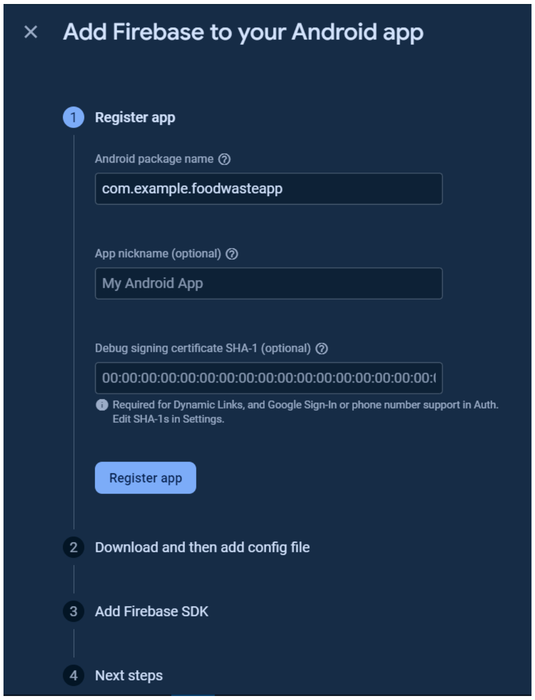
Google Analytics enables:

- × ~~A/B testing~~
- × ~~User segmentation and targeting across Firebase products~~
- × ~~Breadcrumb logs in Crashlytics~~
- × ~~Event-based Cloud Functions triggers~~
- × ~~Free unlimited reporting~~

○ Enable Google Analytics for this project
Recommended

Figure 5.2.10 Google Analytics

Enter a name for the project and disable the Google Analytics, because there is no need to use it in the project.



Add Firebase to your Android app

1 Register app

Android package name: com.example.foodwasteapp

App nickname (optional): My Android App

Debug signing certificate SHA-1 (optional): 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0

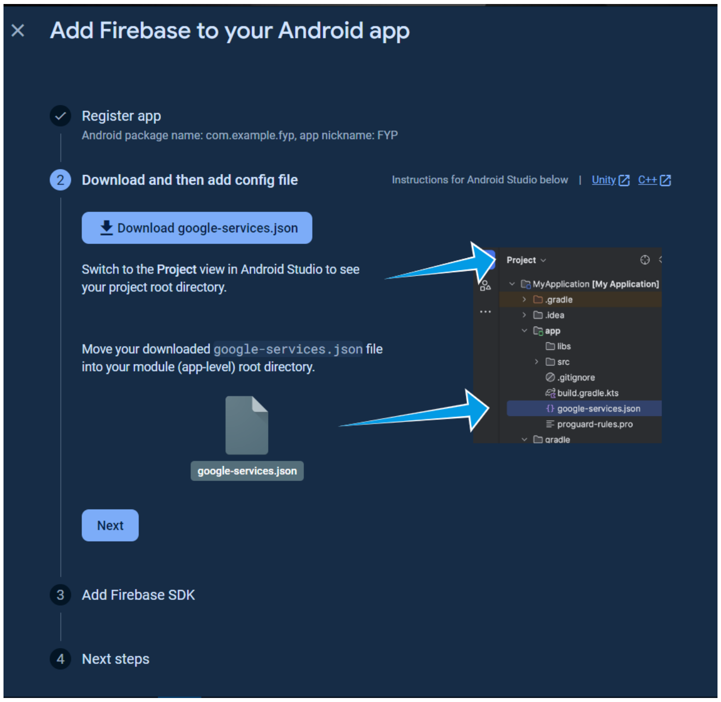
Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

2 Download and then add config file

3 Add Firebase SDK

4 Next steps

Figure 5.2.11 Add Firebase to Application



Add Firebase to your Android app

✓ Register app

Android package name: com.example.fyp, app nickname: FYP

2 Download and then add config file

Instructions for Android Studio below | Unity C++

Switch to the **Project** view in Android Studio to see your project root directory.

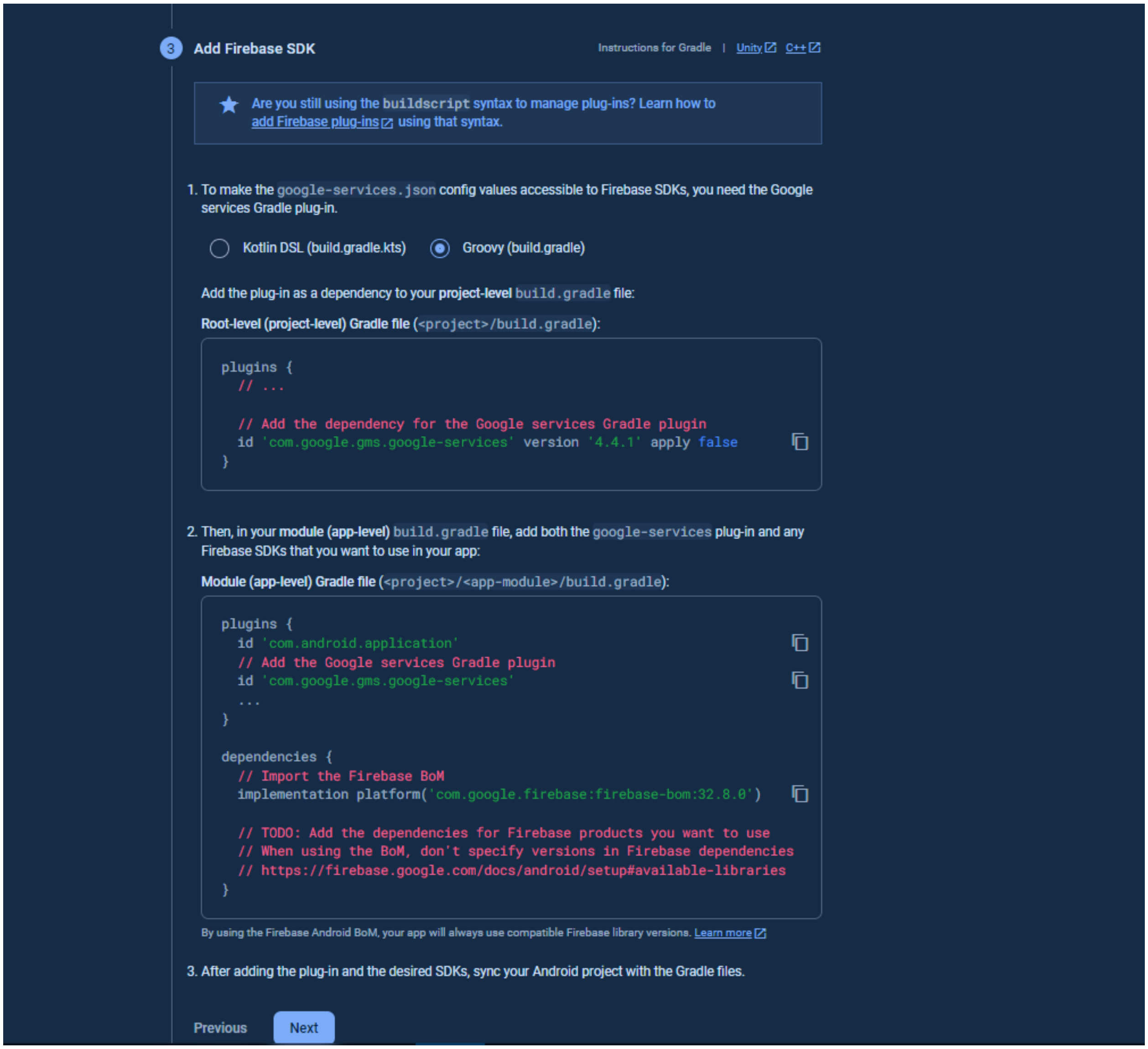
Move your downloaded `google-services.json` file into your module (app-level) root directory.

3 Add Firebase SDK

4 Next steps

Figure 5.2.12 Configure File

Download the google-services.json file and add it to Flutter project, module(app-level) root directory.



Are you still using the `buildscript` syntax to manage plug-ins? Learn how to add Firebase plug-ins using that syntax.

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plug-in.

○ Kotlin DSL (build.gradle.kts) ● Groovy (build.gradle)

Add the plug-in as a dependency to your **project-level** `build.gradle` file:

Root-level (project-level) Gradle file (`<project>/build.gradle`):

```
plugins {
  // ...

  // Add the dependency for the Google services Gradle plugin
  id 'com.google.gms.google-services' version '4.4.1' apply false
}
```

2. Then, in your **module (app-level)** `build.gradle` file, add both the `google-services` plug-in and any Firebase SDKs that you want to use in your app:

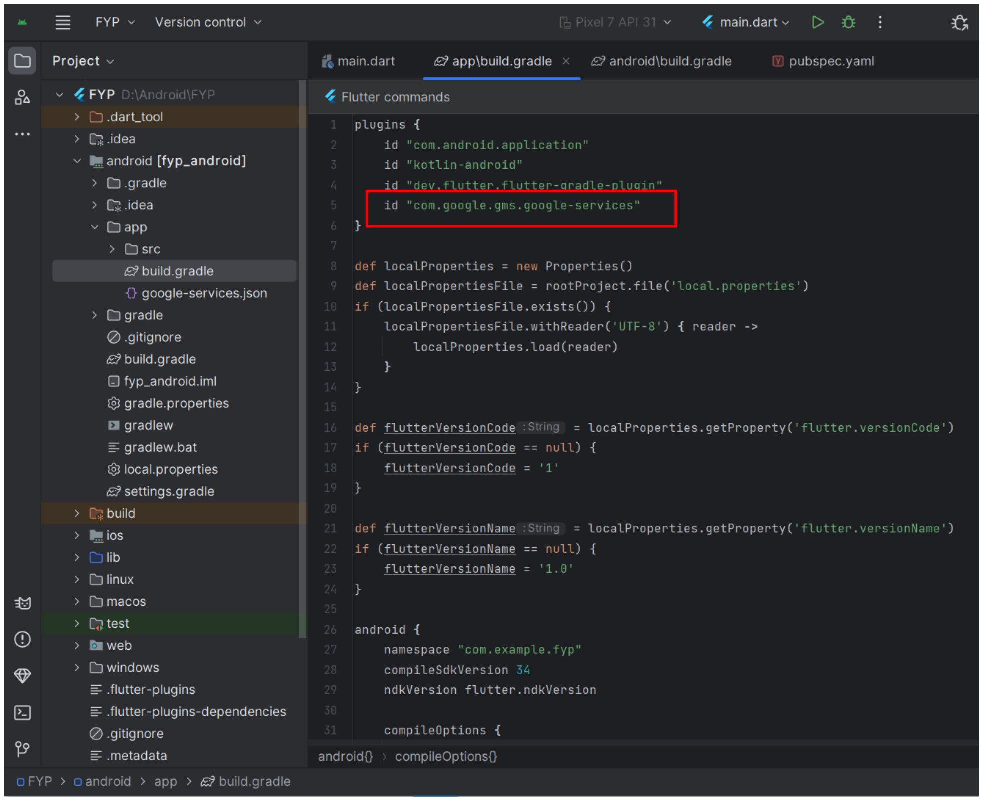
Module (app-level) Gradle file (`<project>/<app-module>/build.gradle`):

```
plugins {
  id 'com.android.application'
  // Add the Google services Gradle plugin
  id 'com.google.gms.google-services'
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:32.8.0')

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

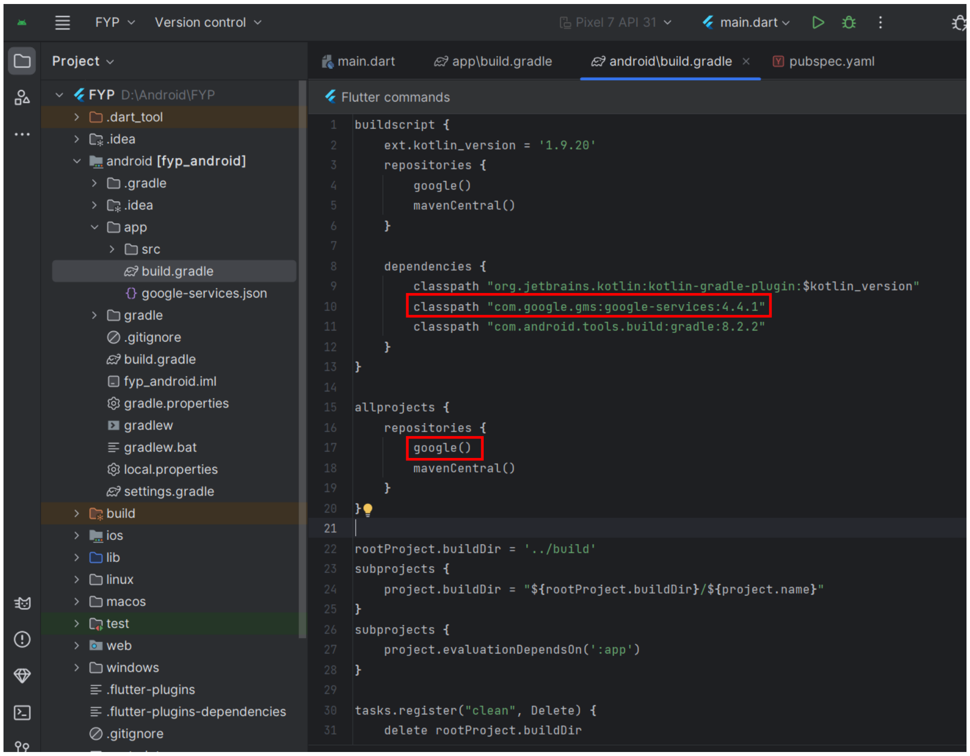
3. After adding the plug-in and the desired SDKs, sync your Android project with the Gradle files.

Figure 5.2.13 Firebase SDK



```
plugins {
    id "com.android.application"
    id "kotlin-android"
    id "dev.flutter.flutter-gradle-plugin"
    id "com.google.gms.google-services"
}

def localProperties = new Properties()
def localPropertiesFile = rootProject.file('local.properties')
if (localPropertiesFile.exists()) {
    localPropertiesFile.withReader('UTF-8') { reader ->
        localProperties.load(reader)
    }
}

def flutterVersionCode :String = localProperties.getProperty('flutter.versionCode')
if (flutterVersionCode == null) {
    flutterVersionCode = '1'
}

def flutterVersionName :String = localProperties.getProperty('flutter.versionName')
if (flutterVersionName == null) {
    flutterVersionName = '1.0'
}

android {
    namespace "com.example.fyp"
    compileSdkVersion 34
    ndkVersion flutter.ndkVersion

    compileOptions {
```

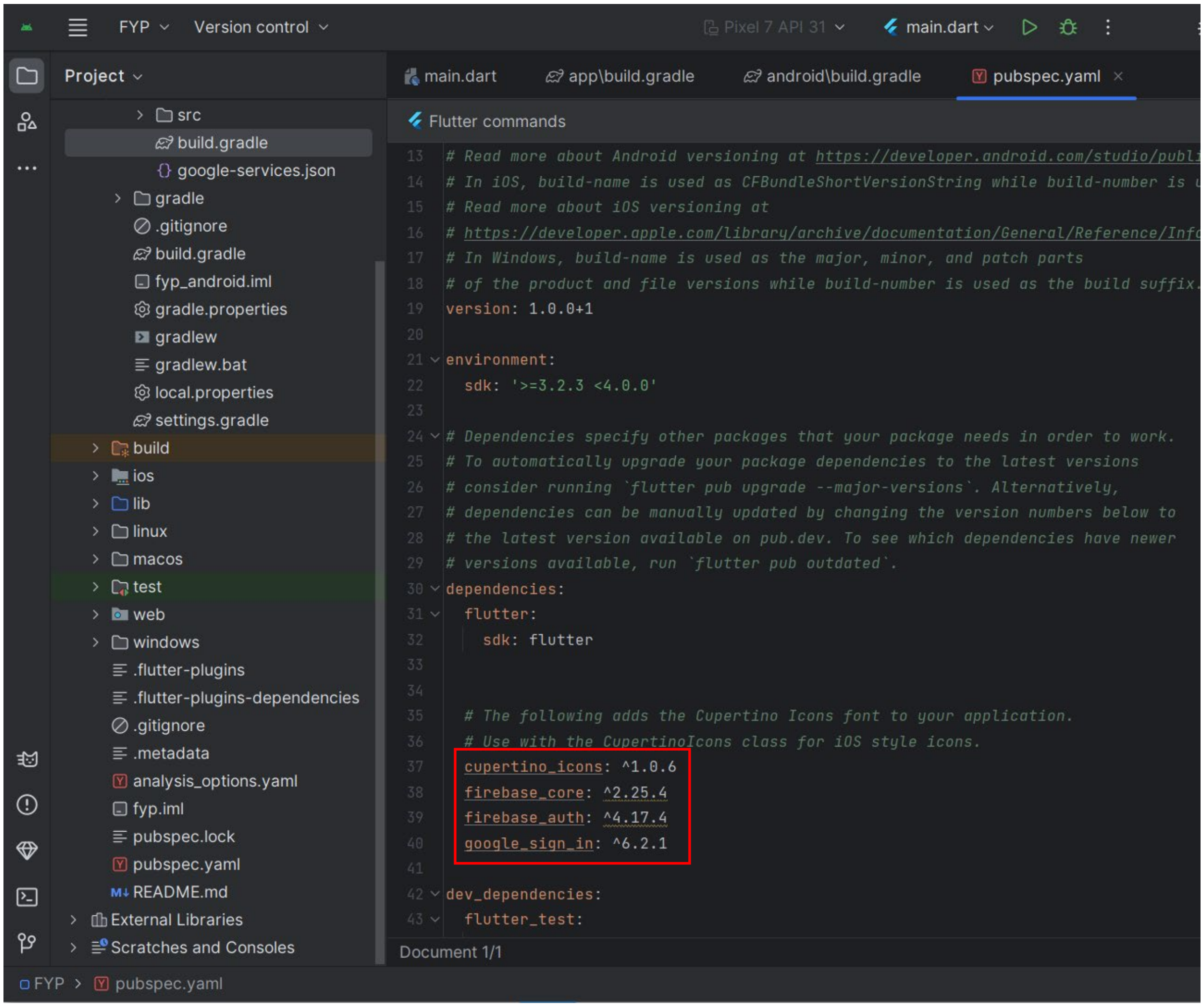
Figure 5.2.14 Module build.gradle



```
buildscript {
    ext.kotlin_version = '1.9.20'
    repositories {
        google()
        mavenCentral()
    }

    dependencies {
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
        classpath "com.google.gms:google-services:4.4.1"
        classpath "com.android.tools.build:gradle:8.2.2"
    }
}

allprojects {
    repositories {
        google()
        mavenCentral()
    }
}

rootProject.buildDir = '../build'
subprojects {
    project.buildDir = "${rootProject.buildDir}/${project.name}"
}
subprojects {
    project.evaluationDependsOn(':app')
}

tasks.register("clean", Delete) {
    delete rootProject.buildDir
```

Figure 5.2.15 Project-level build_gradle

Add the Firebase SDK into Flutter application. Add plugin as dependency to project-level build_gradle file, also add the google-services plugin to module build_gradle file.



```yaml
# Read more about Android versioning at https://developer.android.com/studio/publi
# In iOS, build-name is used as CFBundleShortVersionString while build-number is u
# Read more about iOS versioning at
# https://developer.apple.com/library/archive/documentation/General/Reference/Inf
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build suffix.
version: 1.0.0+1

environment:
  sdk: '>=3.2.3 <4.0.0'

# Dependencies specify other packages that your package needs in order to work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.6
  firebase_core: ^2.25.4
  firebase_auth: ^4.17.4
  google_sign_in: ^6.2.1

dev_dependencies:
  flutter_test:
```

Figure 5.2.16 Pubspec.yaml file

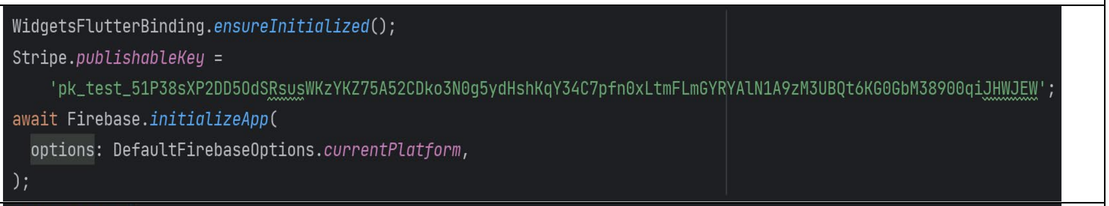
Also, add plugins to the project package's pubspec.yaml file as a library.

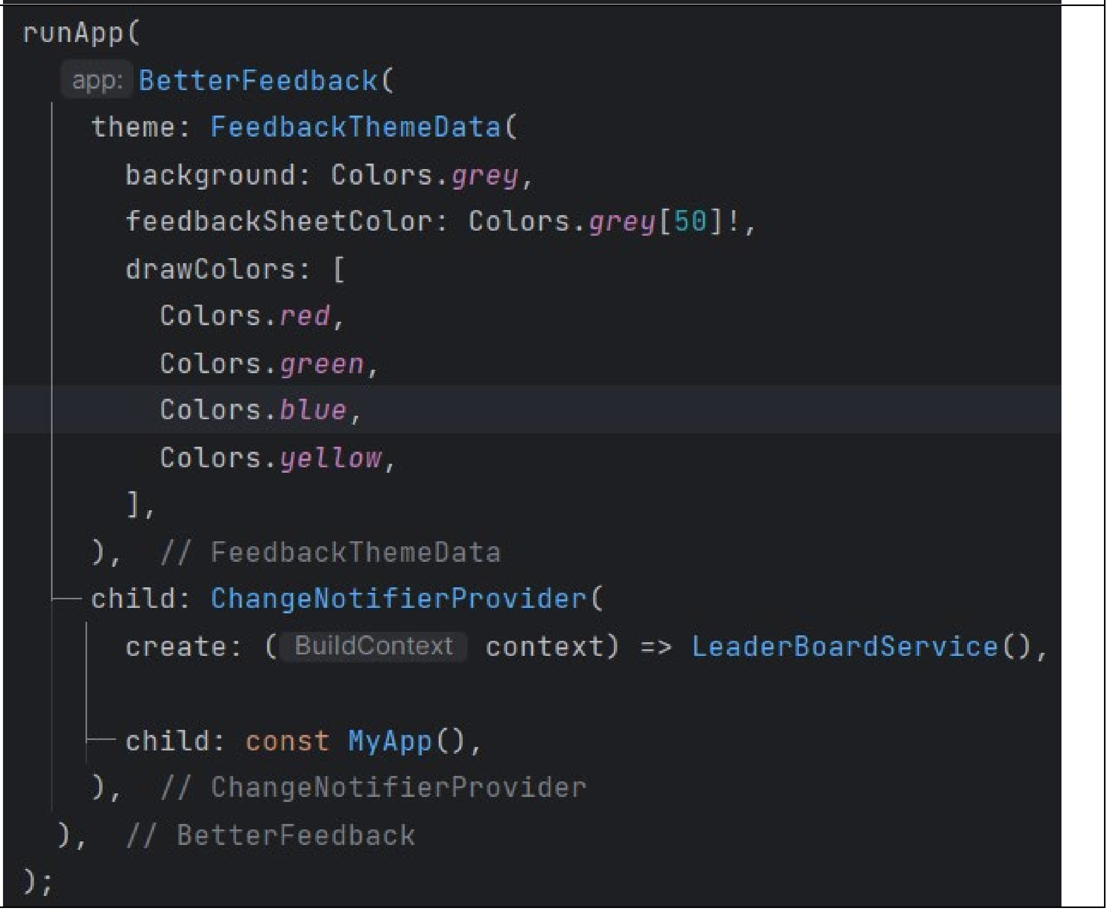
5.3 Coding and UI

5.3.1 User Pages Code

main.dart

Lines	Code

12-16	
	`WidgetsFlutterBinding.ensureInitialized();`
	`Stripe.publishableKey =`
	`    'pk_test_51P38sXP2DD50dSRsusWKzYKZ75A52CDko3N0g5ydHshKqY34C7pfn0xLtmFLmGYRYAlN1A9zM3UBQt6KG0GbM38900qiJHWJEW';`
	`await Firebase.initializeApp(`
	`  options: DefaultFirebaseOptions.currentPlatform,`
	`);`

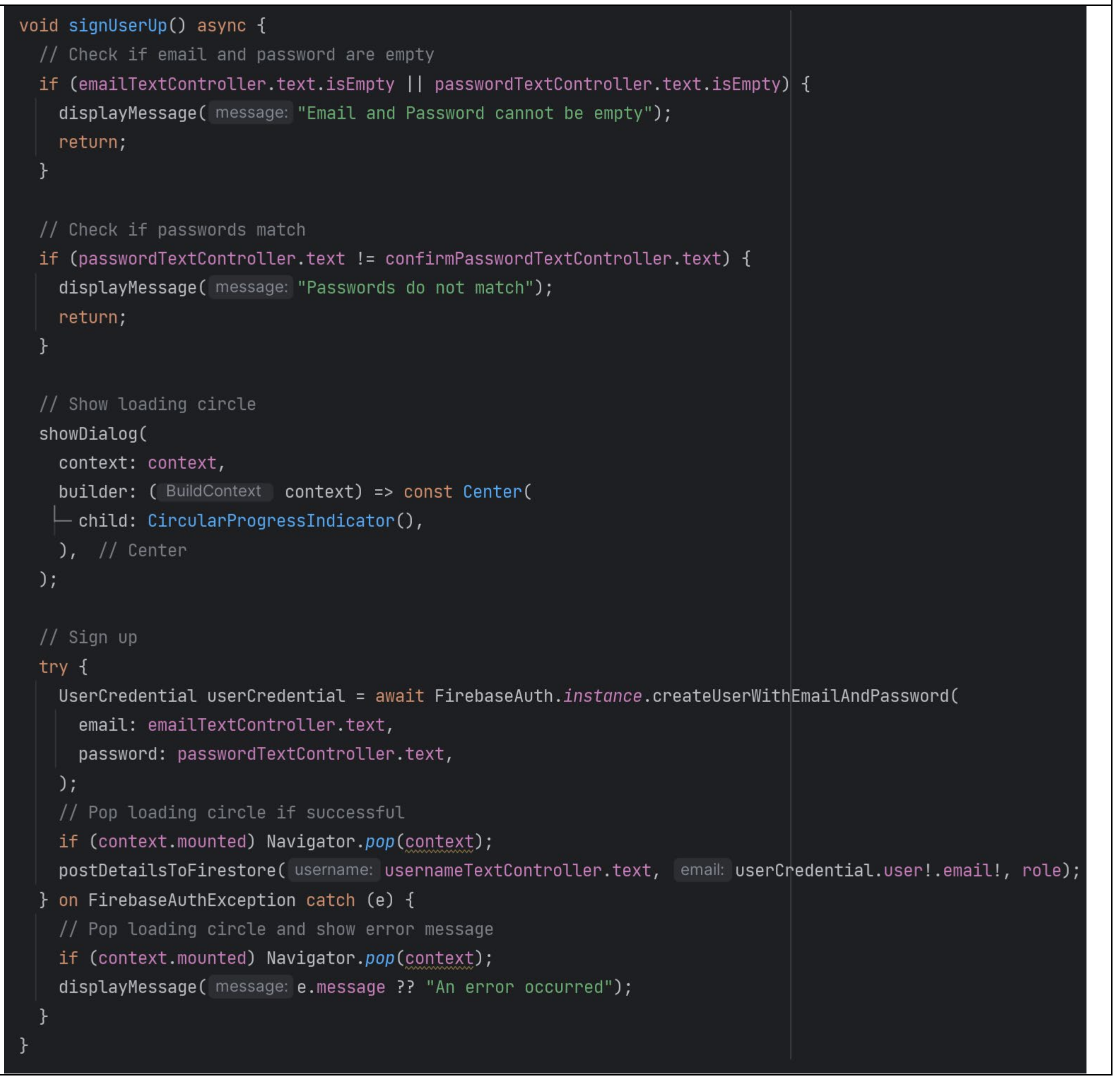
18-36	
	`runApp(`
	`  app: BetterFeedback(`
	`    theme: FeedbackThemeData(`
	`      background: Colors.grey,`
	`      feedbackSheetColor: Colors.grey[50]!,`
	`      drawColors: [`
	`        Colors.red,`
	`        Colors.green,`
	`        Colors.blue,`
	`        Colors.yellow,`
	`      ],`
	`    ), // FeedbackThemeData`
	`    child: ChangeNotifierProvider(`
	`      create: (BuildContext context) => LeaderBoardService(),`
	
	`      child: const MyApp(),`
	`    ), // ChangeNotifierProvider`
	`  ), // BetterFeedback`
	`);`

The line 15-16 is to make sure the Firebase initialize when the application running. Essentially, this line of code is used to set up Firebase in your app with the appropriate settings for whichever platform the app is currently running on. It’s a crucial step to ensure that Firebase services work correctly in your application.

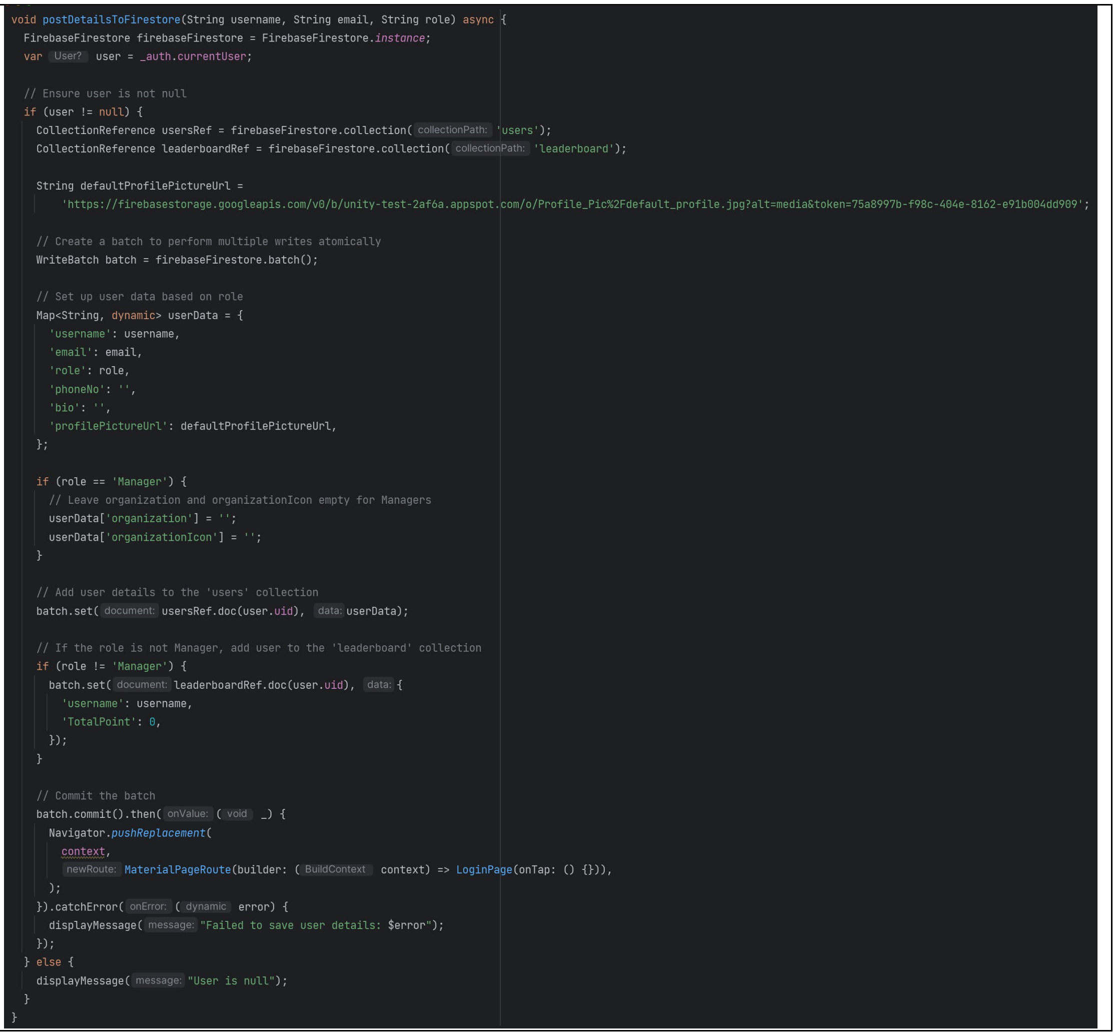
The line 18-36 code snippet show that the app wraps inside a feedback widget called **BetterFeedback**, which allows users to give feedback by drawing on the screen and submitting comments. The feedback interface is customized with a grey background, a light grey feedback sheet, and options for drawing in red, green, blue, and yellow.

Additionally, initializing the application with **ChangeNotifierProvider** at the top of the widget tree, providing an instance of **LeaderBoardService** method to any widgets that might need it. This setup is to manage the state related to leaderboard feature.

register_page.dart

Lines	Code
32-67	

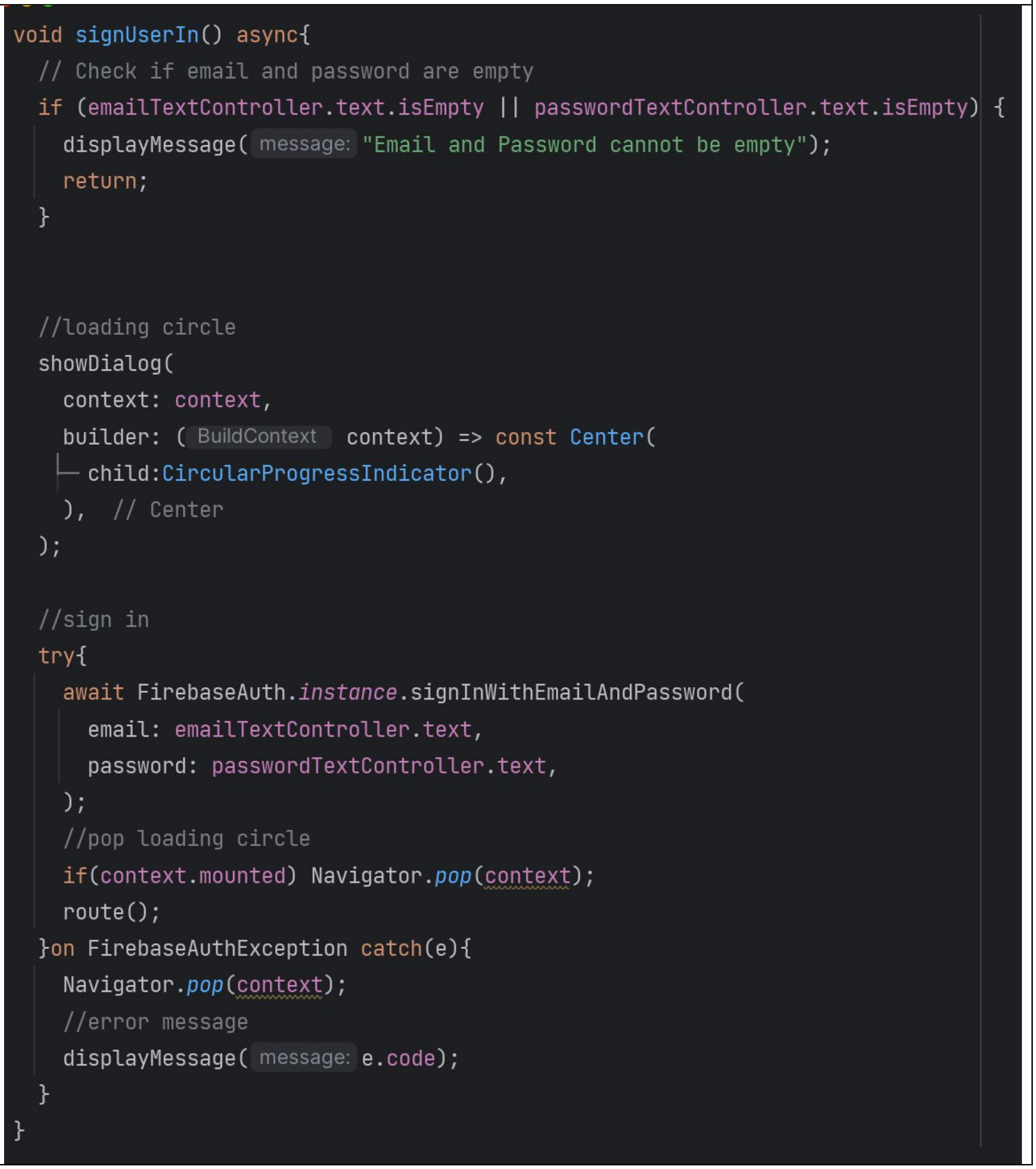
```
void signUserUp() async {
  // Check if email and password are empty
  if (emailTextController.text.isEmpty || passwordTextController.text.isEmpty) {
    displayMessage( message: "Email and Password cannot be empty");
    return;
  }

  // Check if passwords match
  if (passwordTextController.text != confirmPasswordTextController.text) {
    displayMessage( message: "Passwords do not match");
    return;
  }

  // Show loading circle
  showDialog(
    context: context,
    builder: ( BuildContext  context) => const Center(
      child: CircularProgressIndicator(),
    ),  // Center
  );

  // Sign up
  try {
    UserCredential userCredential = await FirebaseAuth.instance.createUserWithEmailAndPassword(
      email: emailTextController.text,
      password: passwordTextController.text,
    );
    // Pop loading circle if successful
    if (context.mounted) Navigator.pop(context);
    postDetailsToFirestore( username: usernameTextController.text,  email: userCredential.user!.email!, role);
  } on FirebaseAuthException catch (e) {
    // Pop loading circle and show error message
    if (context.mounted) Navigator.pop(context);
    displayMessage( message: e.message ?? "An error occurred");
  }
}
```

79-133	

```dart
void postDetailsToFirestore(String username, String email, String role) async {
  FirebaseFirestore firebaseFirestore = FirebaseFirestore.instance;
  var User? user = _auth.currentUser;

  // Ensure user is not null
  if (user != null) {
    CollectionReference usersRef = firebaseFirestore.collection( collectionPath: 'users');
    CollectionReference leaderboardRef = firebaseFirestore.collection( collectionPath: 'leaderboard');

    String defaultProfilePictureUrl =
        'https://firebasestorage.googleapis.com/v0/b/unity-test-2af6a.appspot.com/o/Profile_Pic%2Fdefault_profile.jpg?alt=media&token=75a8997b-f98c-404e-8162-e91b004dd909';

    // Create a batch to perform multiple writes atomically
    WriteBatch batch = firebaseFirestore.batch();

    // Set up user data based on role
    Map<String, dynamic> userData = {
      'username': username,
      'email': email,
      'role': role,
      'phoneNo': '',
      'bio': '',
      'profilePictureUrl': defaultProfilePictureUrl,
    };

    if (role == 'Manager') {
      // Leave organization and organizationIcon empty for Managers
      userData['organization'] = '';
      userData['organizationIcon'] = '';
    }

    // Add user details to the 'users' collection
    batch.set( document: usersRef.doc(user.uid),  data: userData);

    // If the role is not Manager, add user to the 'leaderboard' collection
    if (role != 'Manager') {
      batch.set( document: leaderboardRef.doc(user.uid),  data: {
        'username': username,
        'TotalPoint': 0,
      });
    }

    // Commit the batch
    batch.commit().then( onValue: ( void  _) {
      Navigator.pushReplacement(
        context,
        newRoute: MaterialPageRoute(builder: ( BuildContext  context) => LoginPage(onTap: () {})),
      );
    }).catchError( onError: ( dynamic  error) {
      displayMessage( message: "Failed to save user details: $error");
    });
  } else {
    displayMessage( message: "User is null");
  }
}
```

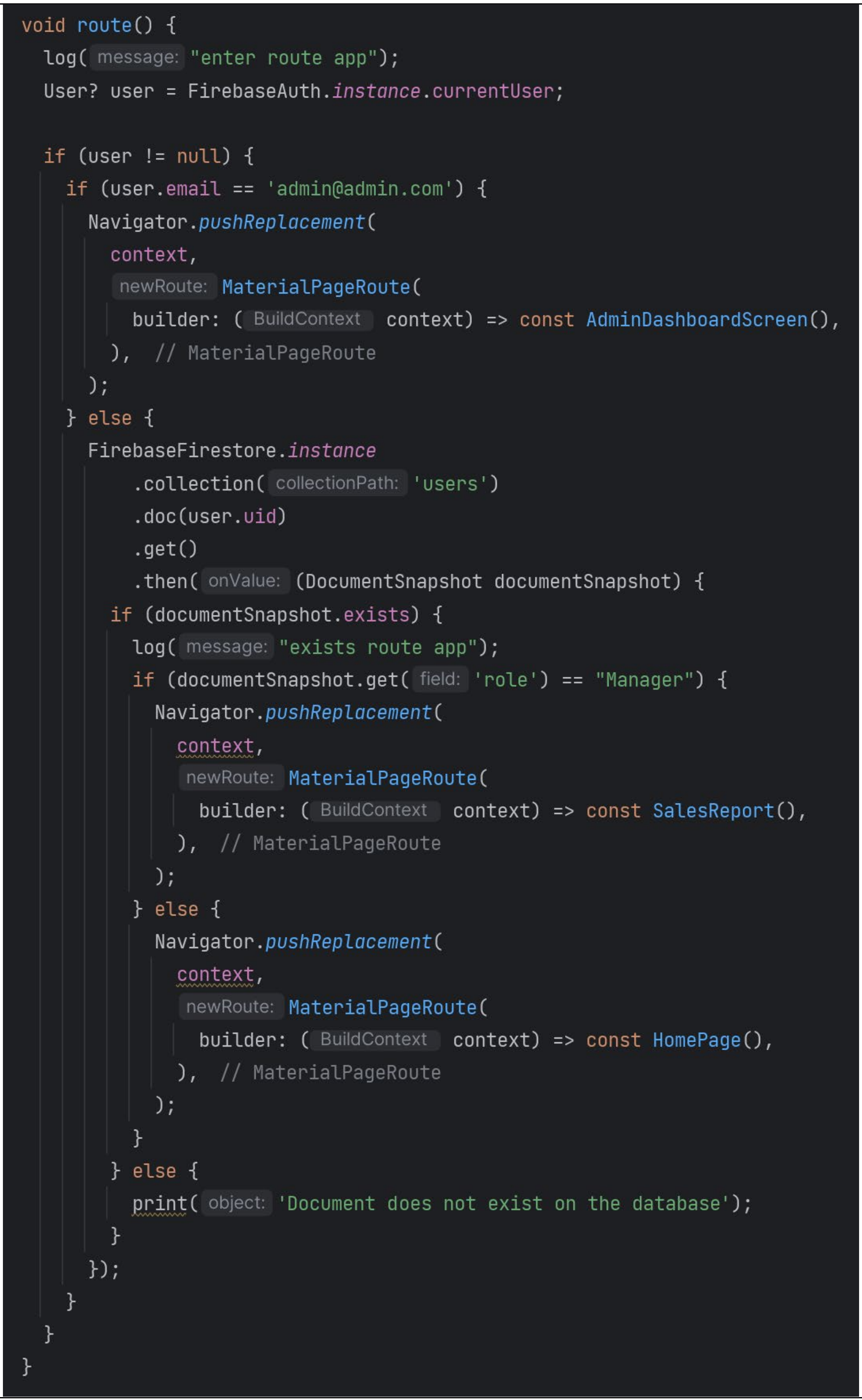
The RegisterPage class in register_page.dart allows users to sign up for an account in the app using Firebase Authentication, there a sample interface show in Figure 5.3.1. It provides a registration form where users input a username, email, password, and confirm their password. The form also includes a dropdown menu to select the user's role, either “User” or ‘Manager.” The app checks that the email and password fields are not empty and that the passwords match. After submitting the form, the user's data is saved to Firebase Firestore, where their details such as username, email, role, and profile picture URL are stored. If the user is not a manager, they are also added to the app's leaderboard. The page also includes a Google sign-in option and a button to navigate to the login page if the user already has an account.

login_page.dart

Lines	Code
27-57	

```
void signUserIn() async{
  // Check if email and password are empty
  if (emailTextController.text.isEmpty || passwordTextController.text.isEmpty) {
    displayMessage( message: "Email and Password cannot be empty");
    return;
  }


  //loading circle
  showDialog(
    context: context,
    builder: ( BuildContext  context) => const Center(
    ├─ child:CircularProgressIndicator(),
    ),  // Center
  );

  //sign in
  try{
    await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: emailTextController.text,
      password: passwordTextController.text,
    );
    //pop loading circle
    if(context.mounted) Navigator.pop(context);
    route();
  }on FirebaseAuthException catch(e){
    Navigator.pop(context);
    //error message
    displayMessage( message: e.code);
  }
}
```

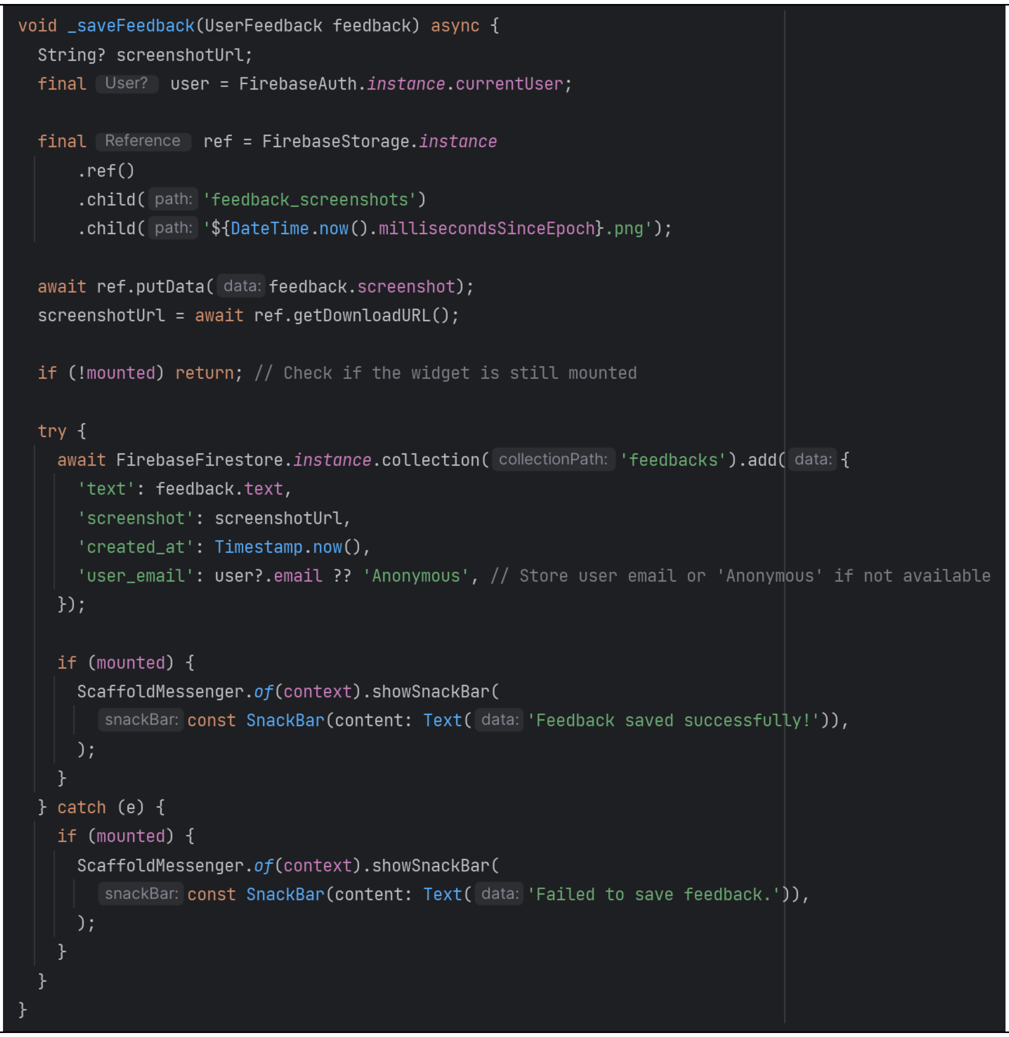
59-100



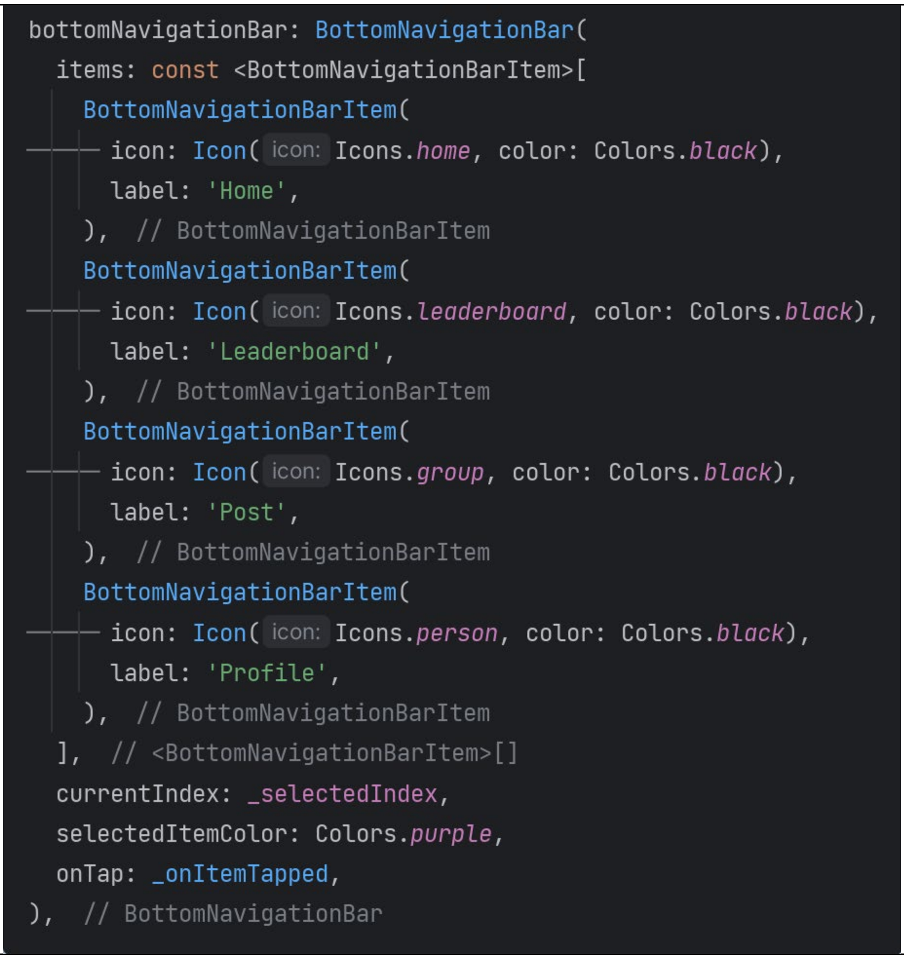
```
void route() {
  log( message: "enter route app");
  User? user = FirebaseAuth.instance.currentUser;

  if (user != null) {
    if (user.email == 'admin@admin.com') {
      Navigator.pushReplacement(
        context,
        newRoute: MaterialPageRoute(
          builder: ( BuildContext  context) => const AdminDashboardScreen(),
        ),  // MaterialPageRoute
      );
    } else {
      FirebaseFirestore.instance
          .collection( collectionPath: 'users')
          .doc(user.uid)
          .get()
          .then( onValue: (DocumentSnapshot documentSnapshot) {
        if (documentSnapshot.exists) {
          log( message: "exists route app");
          if (documentSnapshot.get( field: 'role') == "Manager") {
            Navigator.pushReplacement(
              context,
              newRoute: MaterialPageRoute(
                builder: ( BuildContext  context) => const SalesReport(),
              ),  // MaterialPageRoute
            );
          } else {
            Navigator.pushReplacement(
              context,
              newRoute: MaterialPageRoute(
                builder: ( BuildContext  context) => const HomePage(),
              ),  // MaterialPageRoute
            );
          }
        } else {
          print( object: 'Document does not exist on the database');
        }
      });
    }
  }
}
```

The login_page handles the user authentication process using Firebase, the sample interface of login page show in Figure 5.3.2. The main functionality allows users to log in using their email and password or with Google Sign-In via Firebase Authentication. It also includes error handling for incorrect login attempts, such as displaying an error message if the email or password is empty. Upon successful login, the app routes the user to different screens based on their role, either an Admin Dashboard, Sales Report page for Managers, or the Home Page for general users, as determined by the data stored in Firestore. The UI also provides options for users to navigate to the password reset page or registration page if they do not have an account. Overall, this class manages the login flow and directs users to the appropriate content based on their credentials and role.

HomePage.dart

Lines	Code
47-81	

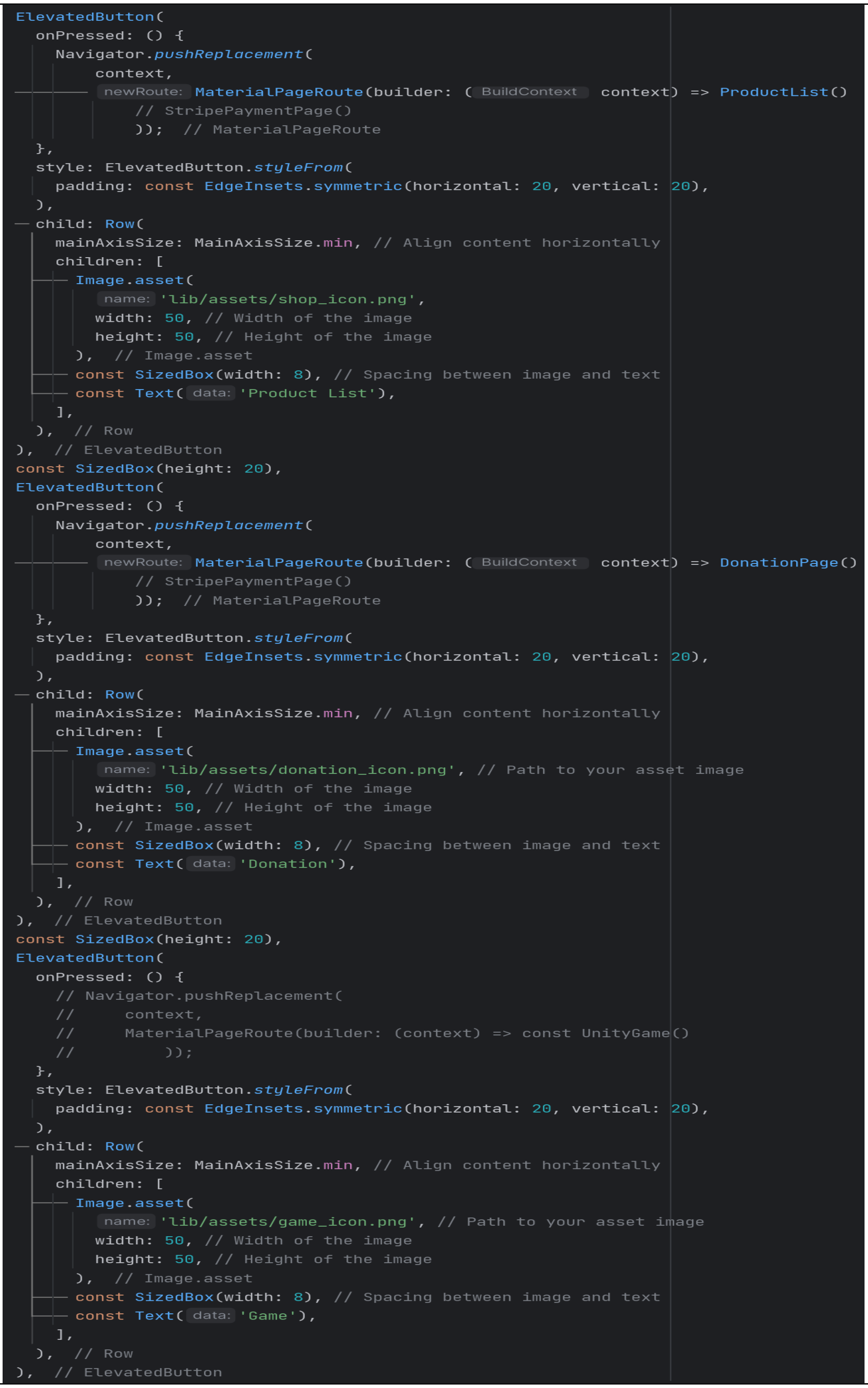
```
void _saveFeedback(UserFeedback feedback) async {
  String? screenshotUrl;
  final User? user = FirebaseAuth.instance.currentUser;

  final Reference ref = FirebaseStorage.instance
      .ref()
      .child( path: 'feedback_screenshots')
      .child( path: '${DateTime.now().millisecondsSinceEpoch}.png');

  await ref.putData( data: feedback.screenshot);
  screenshotUrl = await ref.getDownloadURL();

  if (!mounted) return; // Check if the widget is still mounted

  try {
    await FirebaseFirestore.instance.collection( collectionPath: 'feedbacks').add( data: {
      'text': feedback.text,
      'screenshot': screenshotUrl,
      'created_at': Timestamp.now(),
      'user_email': user?.email ?? 'Anonymous', // Store user email or 'Anonymous' if not available
    });

    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
         snackBar: const SnackBar(content: Text( data: 'Feedback saved successfully!')),
      );
    }
  } catch (e) {
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
         snackBar: const SnackBar(content: Text( data: 'Failed to save feedback.')),
      );
    }
  }
}
```

129-151	

```
bottomNavigationBar: BottomNavigationBar(
  items: const <BottomNavigationBarItem>[
    BottomNavigationBarItem(
      icon: Icon( icon: Icons.home, color: Colors.black),
      label: 'Home',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon( icon: Icons.leaderboard, color: Colors.black),
      label: 'Leaderboard',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon( icon: Icons.group, color: Colors.black),
      label: 'Post',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon( icon: Icons.person, color: Colors.black),
      label: 'Profile',
    ), // BottomNavigationBarItem
  ], // <BottomNavigationBarItem>[]
  currentIndex: _selectedIndex,
  selectedItemColor: Colors.purple,
  onTap: _onItemTapped,
), // BottomNavigationBar
```

In line 47-81 of HomePage.dart, **saveFeedback** function is responsible for saving user feedback to Firebase Firestore and uploading a screenshot to Firebase Storage. First, it checks if there's a current user logged in via Firebase Authentication. It then creates a reference in Firebase Storage to save the feedback screenshot, with the filename based on the current timestamp. The screenshot is uploaded, and the function retrieves the URL where the screenshot is stored. Once the screenshot is uploaded, the feedback text, screenshot URL, the timestamp when the feedback was created, and the user's email are added to the Firestore database under a collection named “feedbacks”. If everything works successfully, a success message "Feedback saved successfully!" is shown to the user. If something goes wrong during the saving process, an error message “Failed to save feedback." is displayed. The function also checks if the widget is still active “mounted” before showing messages to ensure proper UI behavior. There a sample interface shows in Figure 5.3.3 for home page and Figure 5.3.4 for feedback view.

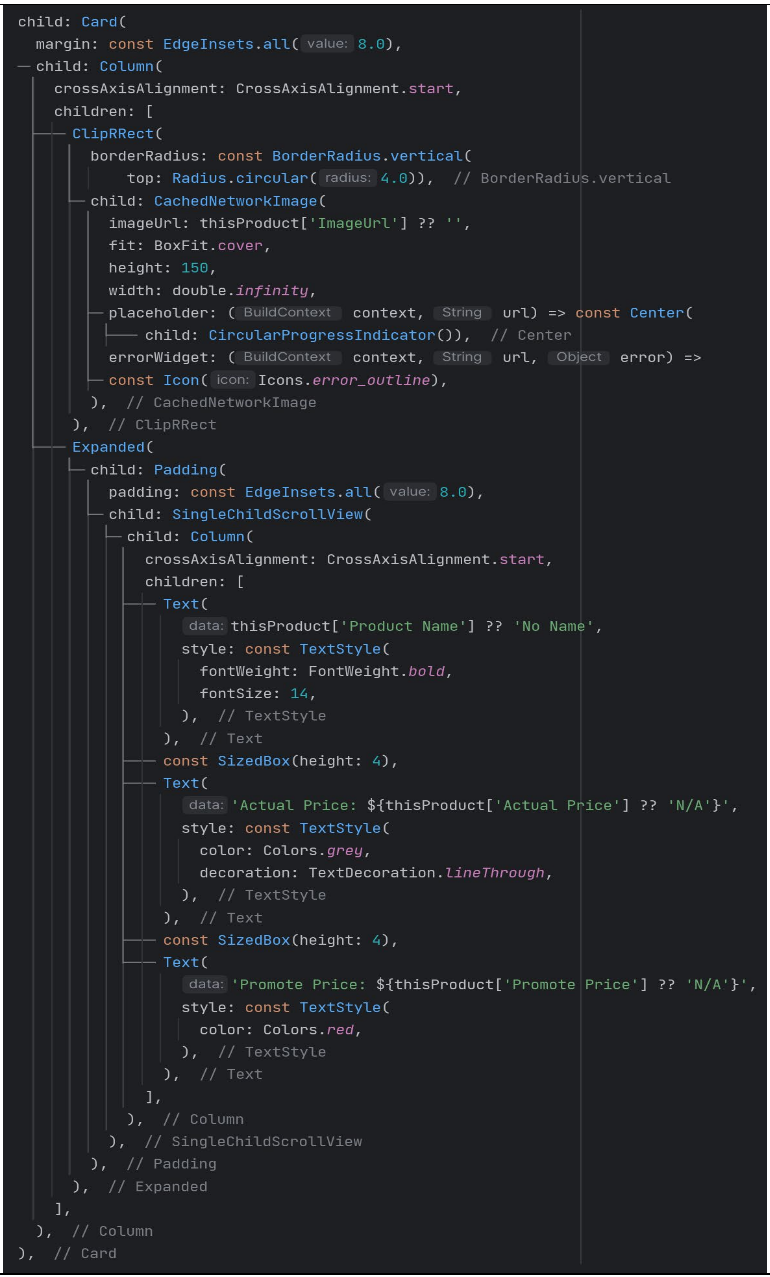
HomeButton.dart

Lines	Code
78-142	

```
ElevatedButton(
  onPressed: () {
    Navigator.pushReplacement(
        context,
        newRoute: MaterialPageRoute(builder: ( BuildContext  context) => ProductList()
          // StripePaymentPage()
          ));  // MaterialPageRoute
  },
  style: ElevatedButton.styleFrom(
    padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 20),
  ),
  child: Row(
    mainAxisSize: MainAxisSize.min, // Align content horizontally
    children: [
      Image.asset(
        name: 'lib/assets/shop_icon.png',
        width: 50, // Width of the image
        height: 50, // Height of the image
      ),  // Image.asset
      const SizedBox(width: 8), // Spacing between image and text
      const Text( data: 'Product List'),
    ],
  ),  // Row
),  // ElevatedButton
const SizedBox(height: 20),
ElevatedButton(
  onPressed: () {
    Navigator.pushReplacement(
        context,
        newRoute: MaterialPageRoute(builder: ( BuildContext  context) => DonationPage()
          // StripePaymentPage()
          ));  // MaterialPageRoute
  },
  style: ElevatedButton.styleFrom(
    padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 20),
  ),
  child: Row(
    mainAxisSize: MainAxisSize.min, // Align content horizontally
    children: [
      Image.asset(
        name: 'lib/assets/donation_icon.png', // Path to your asset image
        width: 50, // Width of the image
        height: 50, // Height of the image
      ),  // Image.asset
      const SizedBox(width: 8), // Spacing between image and text
      const Text( data: 'Donation'),
    ],
  ),  // Row
),  // ElevatedButton
const SizedBox(height: 20),
ElevatedButton(
  onPressed: () {
    // Navigator.pushReplacement(
    //     context,
    //     MaterialPageRoute(builder: (context) => const UnityGame()
    //         ));
  },
  style: ElevatedButton.styleFrom(
    padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 20),
  ),
  child: Row(
    mainAxisSize: MainAxisSize.min, // Align content horizontally
    children: [
      Image.asset(
        name: 'lib/assets/game_icon.png', // Path to your asset image
        width: 50, // Width of the image
        height: 50, // Height of the image
      ),  // Image.asset
      const SizedBox(width: 8), // Spacing between image and text
      const Text( data: 'Game'),
    ],
  ),  // Row
),  // ElevatedButton
```

In line 78-142 of HomeButton.dart, this code consists of three **ElevatedButton** widgets that provide navigation to different screens when clicked. Each button is styled with padding and presents both an image and text side by side, using a “Row” layout to align them horizontally. The first button, when clicked, navigates the user to the **ProductList** screen. The second button navigates to the **DonationPage** screen. The third button is intended to lead to the **UnityGame** page.

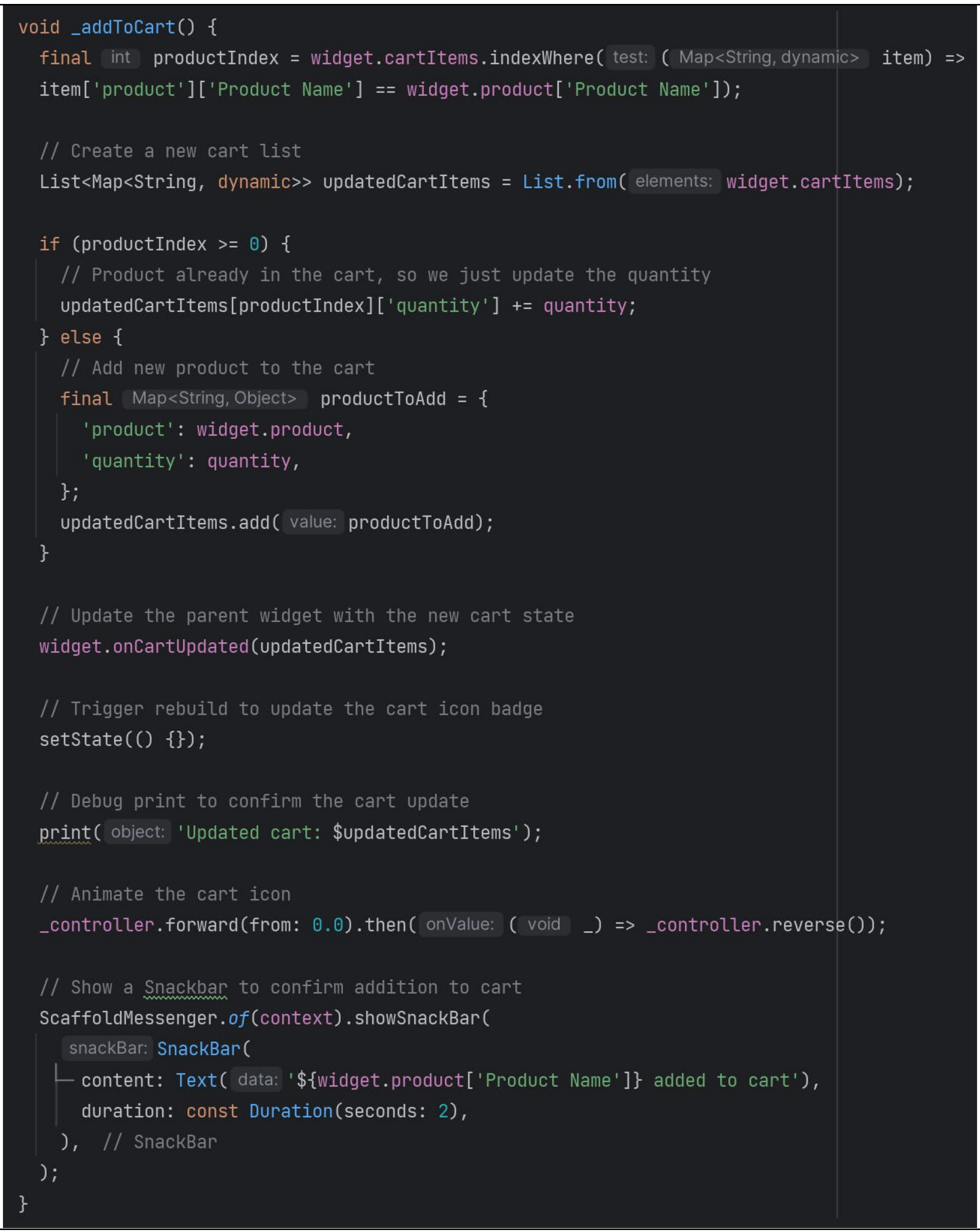
Product_list.dart

Lines	Code
108- 163	

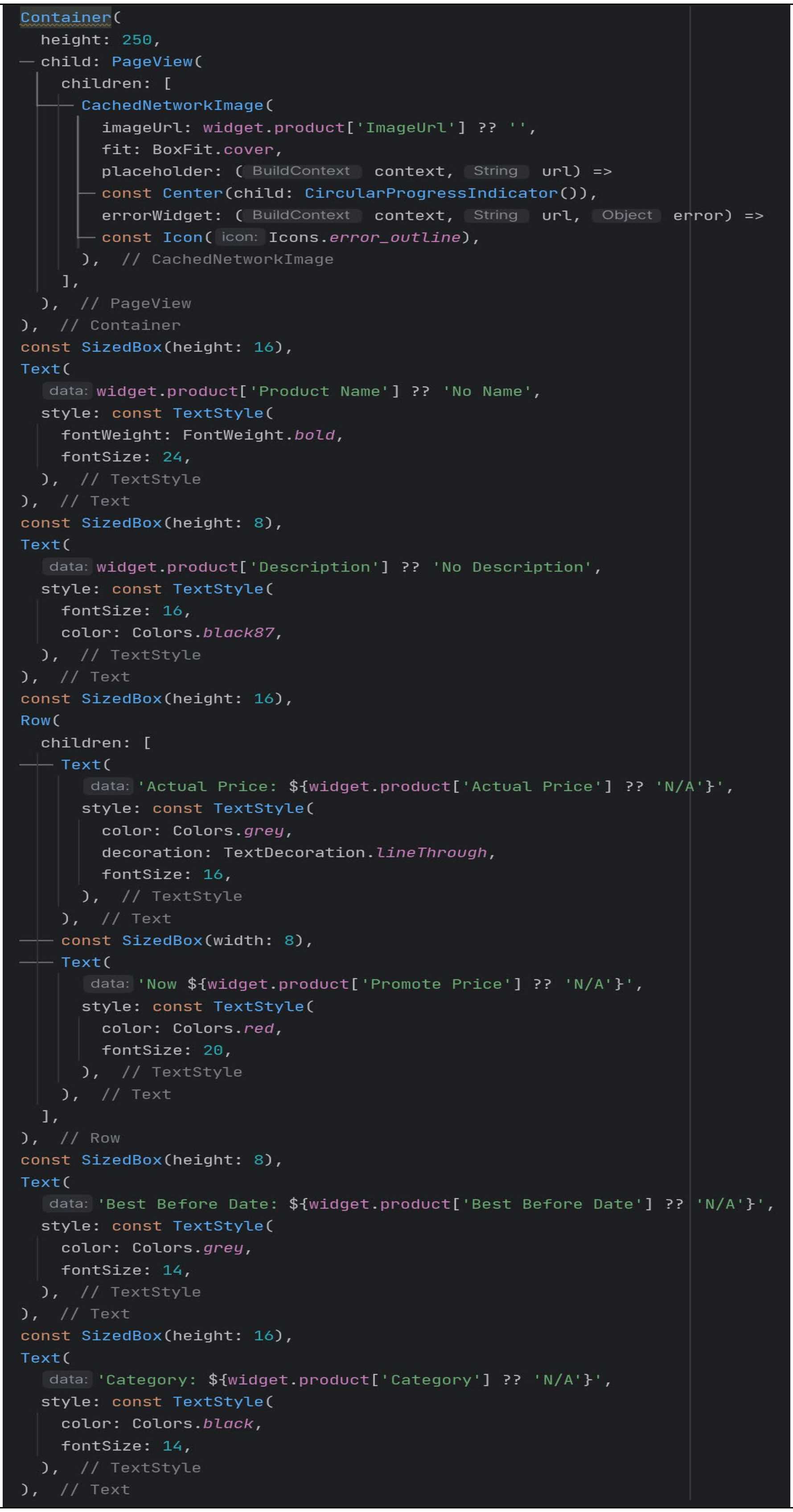
```
child: Card(
  margin: const EdgeInsets.all( value: 8.0),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      ClipRRect(
        borderRadius: const BorderRadius.vertical(
            top: Radius.circular( radius: 4.0)),  // BorderRadius.vertical
        child: CachedNetworkImage(
          imageUrl: thisProduct['ImageUrl'] ?? '',
          fit: BoxFit.cover,
          height: 150,
          width: double.infinity,
          placeholder: ( BuildContext  context,  String  url) => const Center(
              child: CircularProgressIndicator()),  // Center
          errorWidget: ( BuildContext  context,  String  url,  Object  error) =>
          const Icon( icon: Icons.error_outline),
        ),  // CachedNetworkImage
      ),  // ClipRRect
      Expanded(
        child: Padding(
          padding: const EdgeInsets.all( value: 8.0),
          child: SingleChildScrollView(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                   data: thisProduct['Product Name'] ?? 'No Name',
                  style: const TextStyle(
                    fontWeight: FontWeight.bold,
                    fontSize: 14,
                  ),  // TextStyle
                ),  // Text
                const SizedBox(height: 4),
                Text(
                   data: 'Actual Price: ${thisProduct['Actual Price'] ?? 'N/A'}',
                  style: const TextStyle(
                    color: Colors.grey,
                    decoration: TextDecoration.lineThrough,
                  ),  // TextStyle
                ),  // Text
                const SizedBox(height: 4),
                Text(
                   data: 'Promote Price: ${thisProduct['Promote Price'] ?? 'N/A'}',
                  style: const TextStyle(
                    color: Colors.red,
                  ),  // TextStyle
                ),  // Text
              ],
            ),  // Column
          ),  // SingleChildScrollView
        ),  // Padding
      ),  // Expanded
    ],
  ),  // Column
),  // Card
```

This code creates a card that shows product details, sample interface shows in Figure 5.3.5. At the top, there's an image of the product loaded from a URL, and it shows a loading spinner while the image is loading or an error icon if it fails. Below the image, the product's name, original price will cross out, and discounted price in red colour are displayed. The layout ensures the image is at the top, with the text details below, and everything is styled to look neat and easy to read. The content can also scroll if there's too much text.

Product_Details.dart

Lines	Code
43-81	

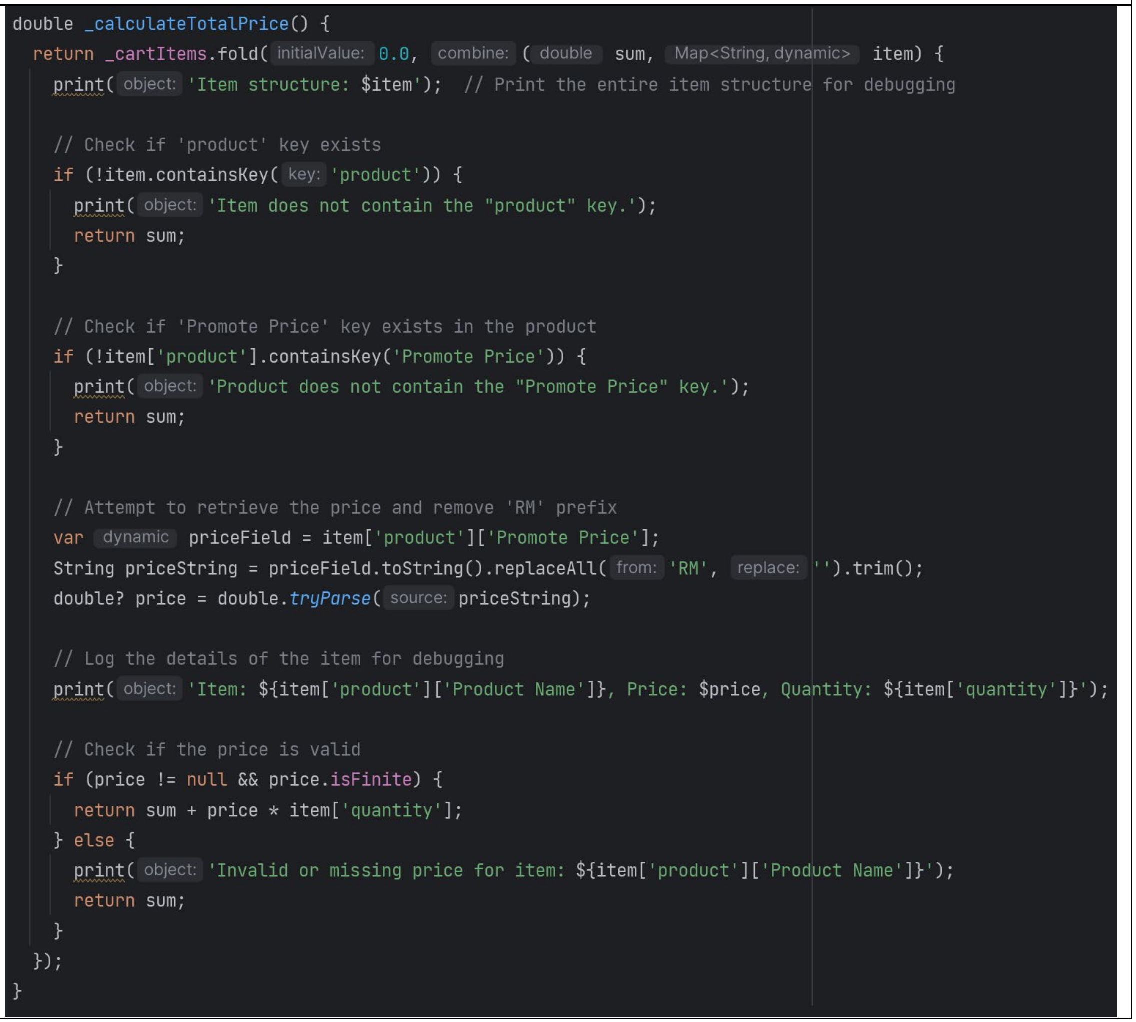
```dart
void _addToCart() {
  final int productIndex = widget.cartItems.indexWhere(test: (Map<String, dynamic> item) =>
  item['product']['Product Name'] == widget.product['Product Name']);

  // Create a new cart list
  List<Map<String, dynamic>> updatedCartItems = List.from(elements: widget.cartItems);

  if (productIndex >= 0) {
    // Product already in the cart, so we just update the quantity
    updatedCartItems[productIndex]['quantity'] += quantity;
  } else {
    // Add new product to the cart
    final Map<String, Object> productToAdd = {
      'product': widget.product,
      'quantity': quantity,
    };
    updatedCartItems.add(value: productToAdd);
  }

  // Update the parent widget with the new cart state
  widget.onCartUpdated(updatedCartItems);

  // Trigger rebuild to update the cart icon badge
  setState(() {});

  // Debug print to confirm the cart update
  print(object: 'Updated cart: $updatedCartItems');

  // Animate the cart icon
  _controller.forward(from: 0.0).then(onValue: (void _) => _controller.reverse());

  // Show a Snackbar to confirm addition to cart
  ScaffoldMessenger.of(context).showSnackBar(
    snackBar: SnackBar(
      content: Text(data: '${widget.product['Product Name']} added to cart'),
      duration: const Duration(seconds: 2),
    ), // SnackBar
  );
}
```

127- 194	

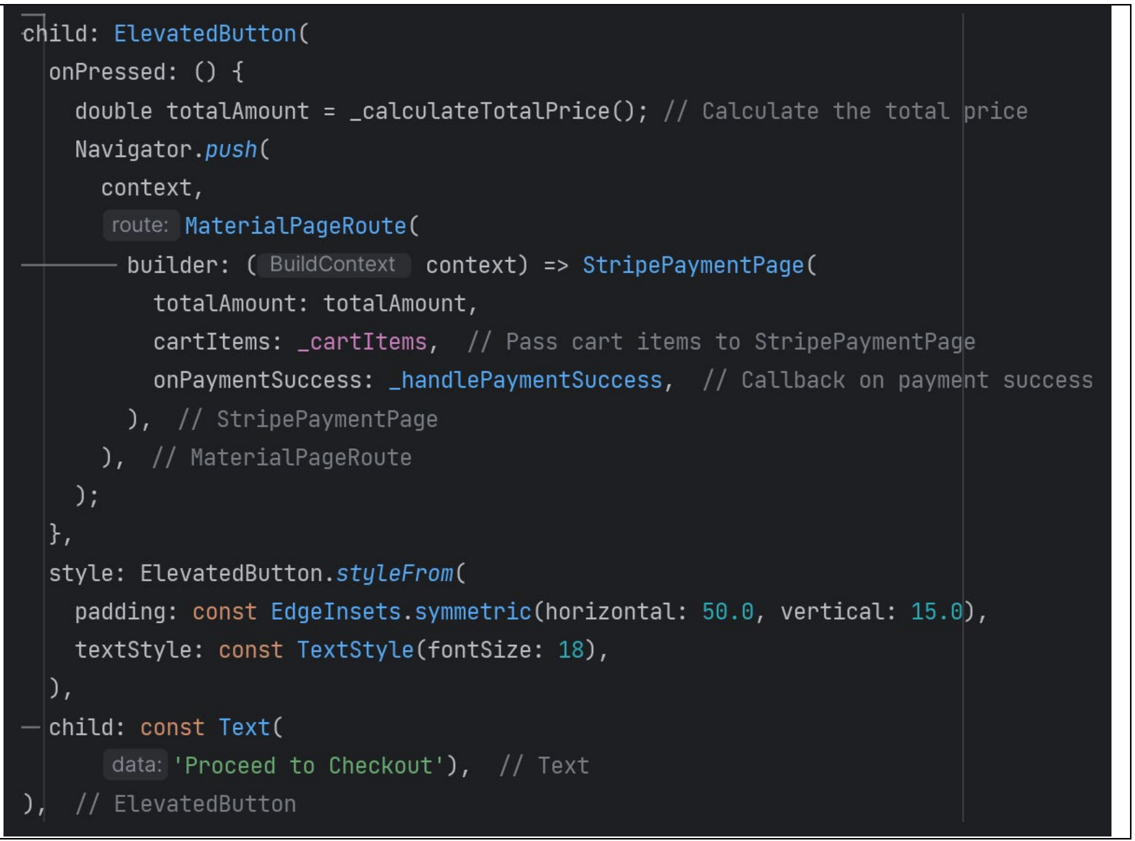
```
Container(
  height: 250,
  child: PageView(
    children: [
      CachedNetworkImage(
        imageUrl: widget.product['ImageUrl'] ?? '',
        fit: BoxFit.cover,
        placeholder: ( BuildContext  context,  String  url) =>
        const Center(child: CircularProgressIndicator()),
        errorWidget: ( BuildContext  context,  String  url,  Object  error) =>
        const Icon( icon: Icons.error_outline),
      ),  // CachedNetworkImage
    ],
  ),  // PageView
),  // Container
const SizedBox(height: 16),
Text(
   data: widget.product['Product Name'] ?? 'No Name',
  style: const TextStyle(
    fontWeight: FontWeight.bold,
    fontSize: 24,
  ),  // TextStyle
),  // Text
const SizedBox(height: 8),
Text(
   data: widget.product['Description'] ?? 'No Description',
  style: const TextStyle(
    fontSize: 16,
    color: Colors.black87,
  ),  // TextStyle
),  // Text
const SizedBox(height: 16),
Row(
  children: [
    Text(
       data: 'Actual Price: ${widget.product['Actual Price'] ?? 'N/A'}',
      style: const TextStyle(
        color: Colors.grey,
        decoration: TextDecoration.lineThrough,
        fontSize: 16,
      ),  // TextStyle
    ),  // Text
    const SizedBox(width: 8),
    Text(
       data: 'Now ${widget.product['Promote Price'] ?? 'N/A'}',
      style: const TextStyle(
        color: Colors.red,
        fontSize: 20,
      ),  // TextStyle
    ),  // Text
  ],
),  // Row
const SizedBox(height: 8),
Text(
   data: 'Best Before Date: ${widget.product['Best Before Date'] ?? 'N/A'}',
  style: const TextStyle(
    color: Colors.grey,
    fontSize: 14,
  ),  // TextStyle
),  // Text
const SizedBox(height: 16),
Text(
   data: 'Category: ${widget.product['Category'] ?? 'N/A'}',
  style: const TextStyle(
    color: Colors.black,
    fontSize: 14,
  ),  // TextStyle
),  // Text
```

The ProductDetailPage class displays detailed information about a specific product, including its name, description, price, best before date, and category, there a sample interface shows in Figure 5.3.6. It allows users to view an image of the product and select the quantity they wish to purchase. Users can adjust the quantity using “+” and “-“buttons and then add the product to their cart by pressing the “Add to Cart” button. The cart is updated and shows a badge on the shopping cart icon indicating the total number of items in the cart. A Snackbar is displayed to confirm that the product has been added to the cart, and the shopping cart icon briefly animates to highlight the change. The page also provides a button to navigate to the cart page, where users can review their selected items. This setup ensures a smooth shopping experience, allowing users to manage their cart while viewing individual product details.

Cart_Page.dart

Lines	Code
28-60	

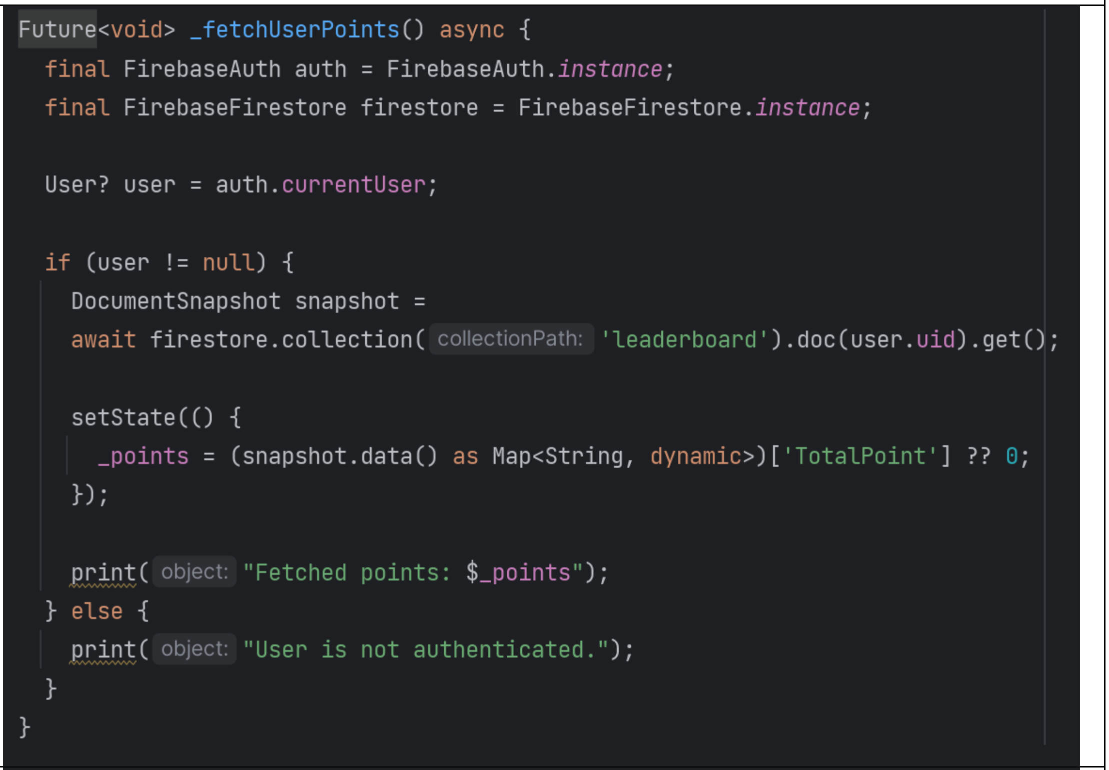
```
double _calculateTotalPrice() {
  return _cartItems.fold( initialValue: 0.0,  combine: ( double  sum,  Map<String, dynamic>  item) {
    print( object: 'Item structure: $item');  // Print the entire item structure for debugging

    // Check if 'product' key exists
    if (!item.containsKey( key: 'product')) {
      print( object: 'Item does not contain the "product" key.');
      return sum;
    }

    // Check if 'Promote Price' key exists in the product
    if (!item['product'].containsKey('Promote Price')) {
      print( object: 'Product does not contain the "Promote Price" key.');
      return sum;
    }

    // Attempt to retrieve the price and remove 'RM' prefix
    var  dynamic  priceField = item['product']['Promote Price'];
    String priceString = priceField.toString().replaceAll( from: 'RM',  replace: '').trim();
    double? price = double.tryParse( source: priceString);

    // Log the details of the item for debugging
    print( object: 'Item: ${item['product']['Product Name']}, Price: $price, Quantity: ${item['quantity']}');

    // Check if the price is valid
    if (price != null && price.isFinite) {
      return sum + price * item['quantity'];
    } else {
      print( object: 'Invalid or missing price for item: ${item['product']['Product Name']}');
      return sum;
    }
  });
}
```

132- 152	

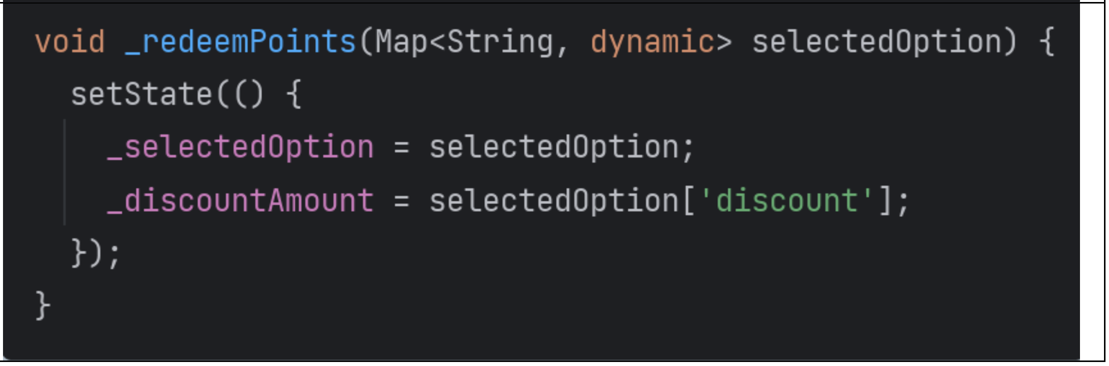
```
child: ElevatedButton(
  onPressed: () {
    double totalAmount = _calculateTotalPrice(); // Calculate the total price
    Navigator.push(
      context,
      route: MaterialPageRoute(
        builder: ( BuildContext  context) => StripePaymentPage(
          totalAmount: totalAmount,
          cartItems: _cartItems,  // Pass cart items to StripePaymentPage
          onPaymentSuccess: _handlePaymentSuccess,  // Callback on payment success
        ),  // StripePaymentPage
      ),  // MaterialPageRoute
    );
  },
  style: ElevatedButton.styleFrom(
    padding: const EdgeInsets.symmetric(horizontal: 50.0, vertical: 15.0),
    textStyle: const TextStyle(fontSize: 18),
  ),
  child: const Text(
      data: 'Proceed to Checkout'),  // Text
),  // ElevatedButton
```

The AddToCartPage class provides a user interface for viewing and managing items in the shopping cart, a sample interface shows in Figure 5.3.7. It displays a list of products the user has added to their cart, showing each item's image, name, price, and quantity. Users can adjust the quantity of each item or remove items from the cart entirely. The total price of the items in the cart is calculated dynamically. The “Proceed to Checkout’ button at the bottom takes the user to the payment gateway when clicked, passing the cart items and total amount for payment. If the payment is successful, the cart is cleared, and the parent widget is updated with the new cart state.

payment_gateway.dart

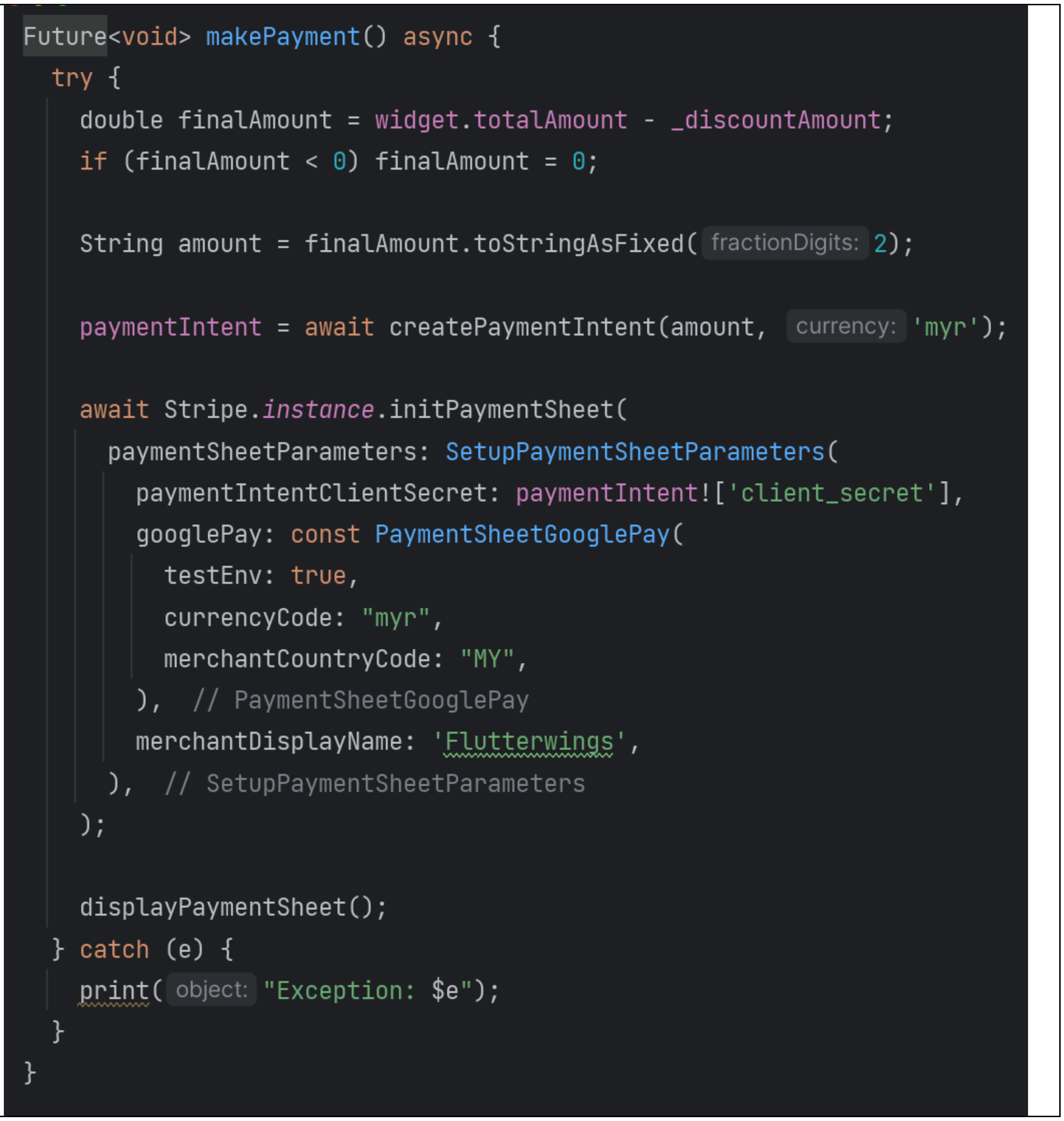
<table>
<tr><th>Lines</th><th>Code</th></tr>
<tr><td>45-63</td><td>


```
Future<void> _fetchUserPoints() async {
  final FirebaseAuth auth = FirebaseAuth.instance;
  final FirebaseFirestore firestore = FirebaseFirestore.instance;

  User? user = auth.currentUser;

  if (user != null) {
    DocumentSnapshot snapshot =
    await firestore.collection( collectionPath: 'leaderboard').doc(user.uid).get();

    setState(() {
      _points = (snapshot.data() as Map<String, dynamic>)['TotalPoint'] ?? 0;
    });

    print( object: "Fetched points: $_points");
  } else {
    print( object: "User is not authenticated.");
  }
}
```

</td></tr>
<tr><td>65-70</td><td>


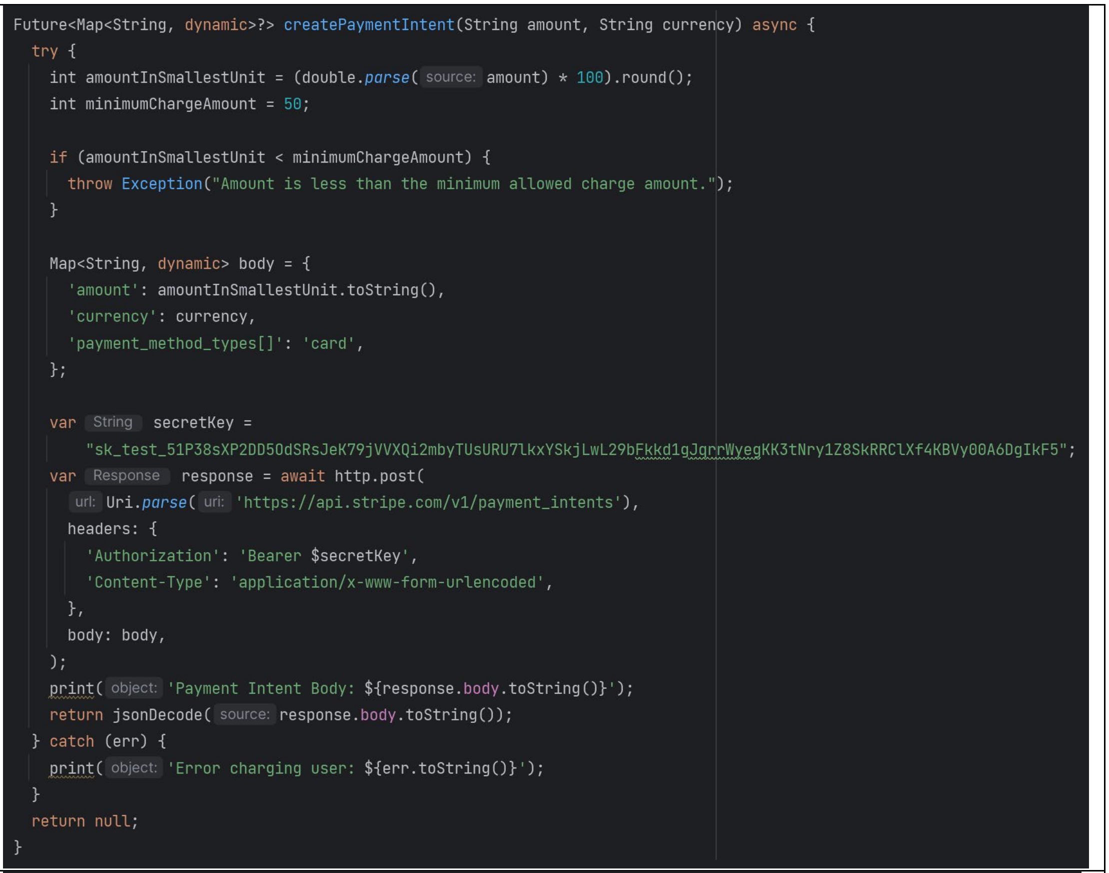
```
void _redeemPoints(Map<String, dynamic> selectedOption) {
  setState(() {
    _selectedOption = selectedOption;
    _discountAmount = selectedOption['discount'];
  });
}
```

</td></tr>
</table>

72-97	

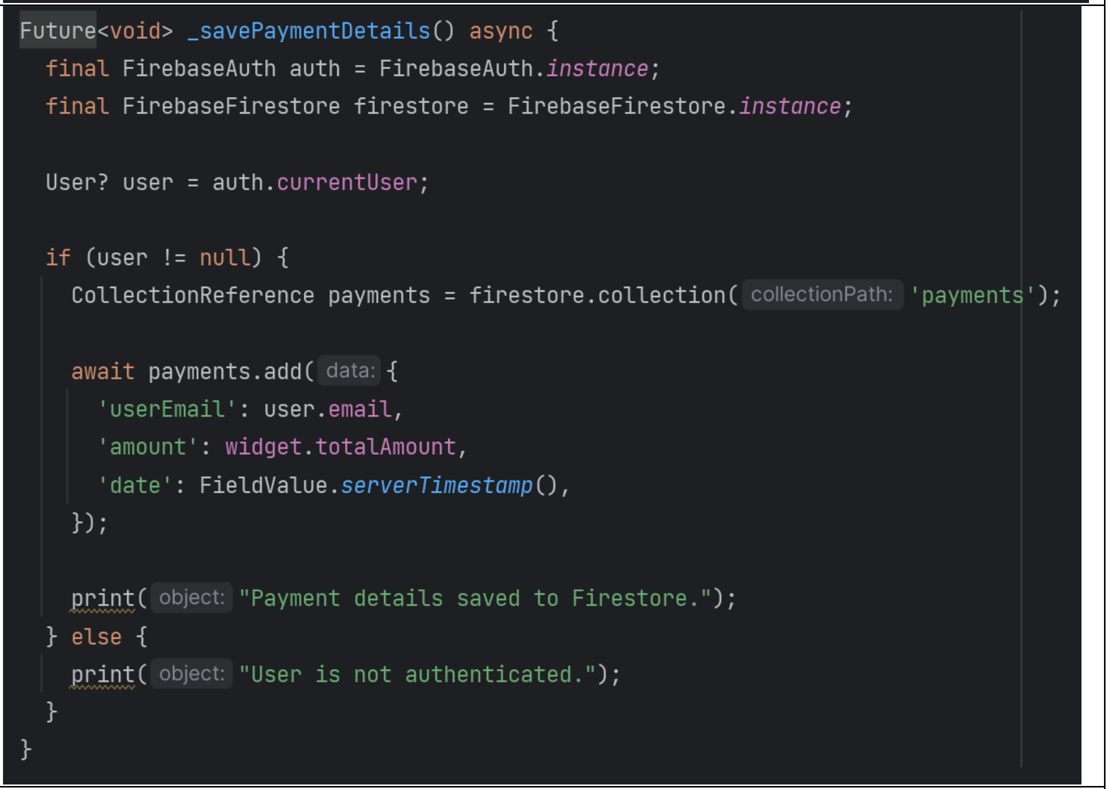
```
Future<void> makePayment() async {
  try {
    double finalAmount = widget.totalAmount - _discountAmount;
    if (finalAmount < 0) finalAmount = 0;

    String amount = finalAmount.toStringAsFixed( fractionDigits: 2);

    paymentIntent = await createPaymentIntent(amount,  currency: 'myr');

    await Stripe.instance.initPaymentSheet(
      paymentSheetParameters: SetupPaymentSheetParameters(
        paymentIntentClientSecret: paymentIntent!['client_secret'],
        googlePay: const PaymentSheetGooglePay(
          testEnv: true,
          currencyCode: "myr",
          merchantCountryCode: "MY",
        ),  // PaymentSheetGooglePay
        merchantDisplayName: 'Flutterwings',
      ),  // SetupPaymentSheetParameters
    );

    displayPaymentSheet();
  } catch (e) {
    print( object: "Exception: $e");
  }
}
```

<table>
<tr>
<td>120-
151</td>
<td>



```
Future<Map<String, dynamic>?> createPaymentIntent(String amount, String currency) async {
  try {
    int amountInSmallestUnit = (double.parse( source: amount) * 100).round();
    int minimumChargeAmount = 50;

    if (amountInSmallestUnit < minimumChargeAmount) {
      throw Exception("Amount is less than the minimum allowed charge amount.");
    }

    Map<String, dynamic> body = {
      'amount': amountInSmallestUnit.toString(),
      'currency': currency,
      'payment_method_types[]': 'card',
    };

    var String secretKey =
        "sk_test_51P38sXP2DD5OdSRsJeK79jVVXQi2mbyTUsURU7lkxYSkjLwL29bFkkd1gJqrrWyegKK3tNry1Z8SkRRClXf4KBVy00A6DgIkF5";
    var Response response = await http.post(
      url: Uri.parse( uri: 'https://api.stripe.com/v1/payment_intents'),
      headers: {
        'Authorization': 'Bearer $secretKey',
        'Content-Type': 'application/x-www-form-urlencoded',
      },
      body: body,
    );
    print( object: 'Payment Intent Body: ${response.body.toString()}');
    return jsonDecode( source: response.body.toString());
  } catch (err) {
    print( object: 'Error charging user: ${err.toString()}');
  }
  return null;
}
```

</td>
</tr>
<tr>
<td>153-
172</td>
<td>



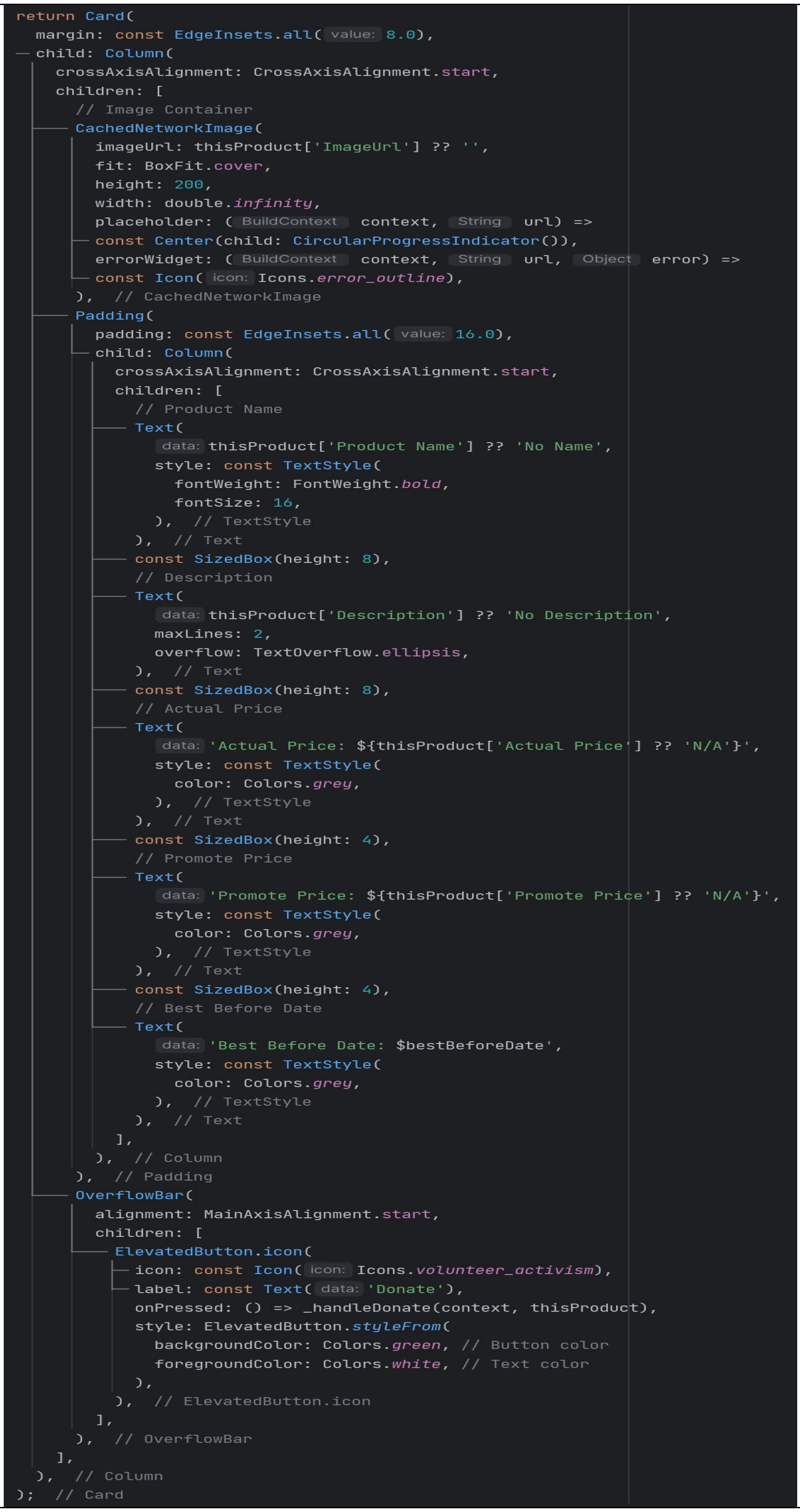
```
Future<void> _savePaymentDetails() async {
  final FirebaseAuth auth = FirebaseAuth.instance;
  final FirebaseFirestore firestore = FirebaseFirestore.instance;

  User? user = auth.currentUser;

  if (user != null) {
    CollectionReference payments = firestore.collection( collectionPath: 'payments');

    await payments.add( data: {
      'userEmail': user.email,
      'amount': widget.totalAmount,
      'date': FieldValue.serverTimestamp(),
    });

    print( object: "Payment details saved to Firestore.");
  } else {
    print( object: "User is not authenticated.");
  }
}
```

</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</table>

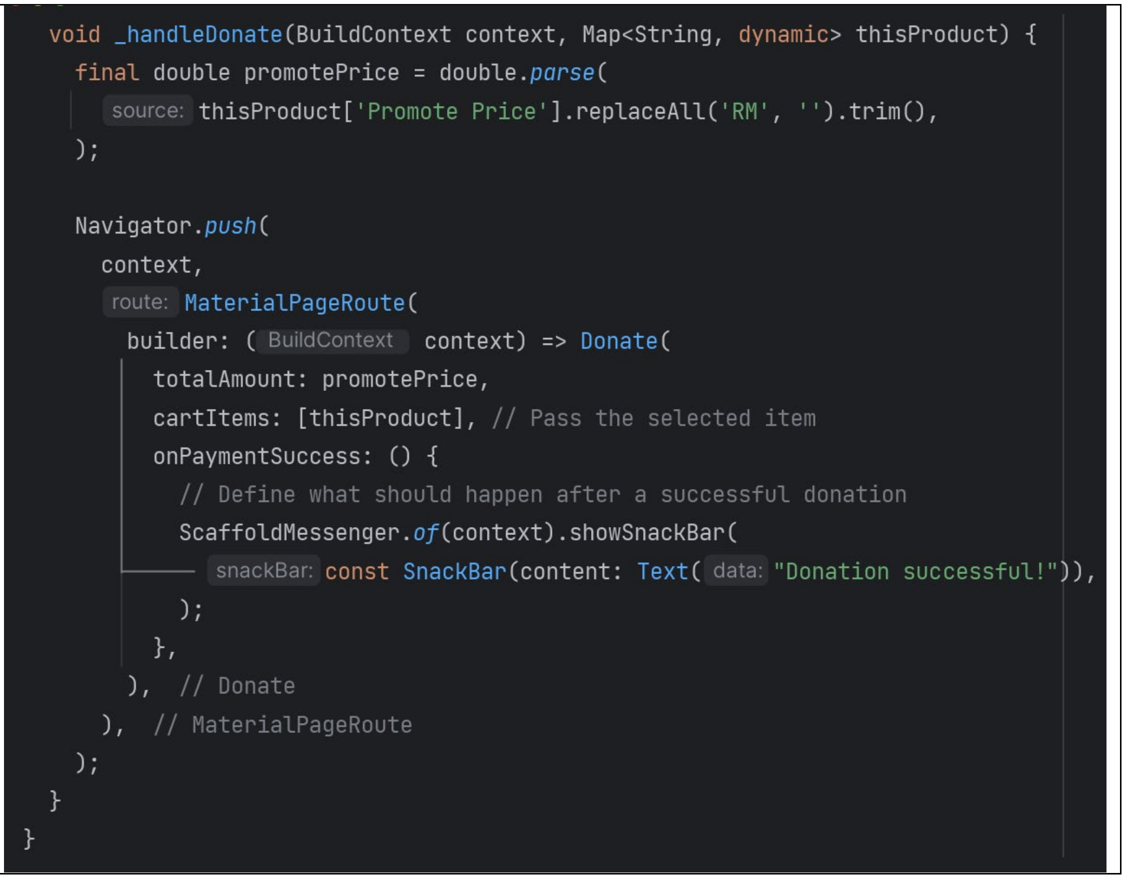
The **StripePaymentPage** is to manage the payment process using Stripe, while also offering users the ability to apply discounts based on user’s points, sample interface shows in. The page displays the user's cart items, showing each product's name, price, quantity, and subtotal. It also retrieves the user's available points from Firestore and presents various discount options depending on the points earned. Users can select a discount, which is applied to reduce the total amount, and they can also cancel the discount if needed. The total price is recalculated based on the applied discount, and the updated amount is shown.

When the user clicks the "Proceed to Payment" button, the **makePayment** function initiates the payment process by creating a payment intent with Stripe. The function presents a payment sheet where the user can complete the payment, and once the transaction is successful, the payment details are saved to Firestore, and a success message is shown. In case of failure or cancellation, appropriate error messages are displayed. Overall, this page efficiently handles the entire payment workflow, including discount application and payment confirmation using Stripe. There some sample interfaces shows in Figure 5.3.8, Figure 5.3.9 and Figure 5.3.10.

Donate_Page.dart

Lines	Code
57- 116	

```
return Card(
  margin: const EdgeInsets.all( value: 8.0),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      // Image Container
      CachedNetworkImage(
        imageUrl: thisProduct['ImageUrl'] ?? '',
        fit: BoxFit.cover,
        height: 200,
        width: double.infinity,
        placeholder: ( BuildContext  context,  String  url) =>
        const Center(child: CircularProgressIndicator()),
        errorWidget: ( BuildContext  context,  String  url,  Object  error) =>
        const Icon( icon: Icons.error_outline),
      ),  // CachedNetworkImage
      Padding(
        padding: const EdgeInsets.all( value: 16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Product Name
            Text(
              data: thisProduct['Product Name'] ?? 'No Name',
              style: const TextStyle(
                fontWeight: FontWeight.bold,
                fontSize: 16,
              ),  // TextStyle
            ),  // Text
            const SizedBox(height: 8),
            // Description
            Text(
              data: thisProduct['Description'] ?? 'No Description',
              maxLines: 2,
              overflow: TextOverflow.ellipsis,
            ),  // Text
            const SizedBox(height: 8),
            // Actual Price
            Text(
              data: 'Actual Price: ${thisProduct['Actual Price'] ?? 'N/A'}',
              style: const TextStyle(
                color: Colors.grey,
              ),  // TextStyle
            ),  // Text
            const SizedBox(height: 4),
            // Promote Price
            Text(
              data: 'Promote Price: ${thisProduct['Promote Price'] ?? 'N/A'}',
              style: const TextStyle(
                color: Colors.grey,
              ),  // TextStyle
            ),  // Text
            const SizedBox(height: 4),
            // Best Before Date
            Text(
              data: 'Best Before Date: $bestBeforeDate',
              style: const TextStyle(
                color: Colors.grey,
              ),  // TextStyle
            ),  // Text
          ],
        ),  // Column
      ),  // Padding
      OverflowBar(
        alignment: MainAxisAlignment.start,
        children: [
          ElevatedButton.icon(
            icon: const Icon( icon: Icons.volunteer_activism),
            label: const Text( data: 'Donate'),
            onPressed: () => _handleDonate(context, thisProduct),
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.green, // Button color
              foregroundColor: Colors.white, // Text color
            ),
          ),  // ElevatedButton.icon
        ],
      ),  // OverflowBar
    ],
  ),  // Column
);  // Card
```

145-167	

```
  void _handleDonate(BuildContext context, Map<String, dynamic> thisProduct) {
    final double promotePrice = double.parse(
      source: thisProduct['Promote Price'].replaceAll('RM', '').trim(),
    );

    Navigator.push(
      context,
      route: MaterialPageRoute(
        builder: ( BuildContext context) => Donate(
          totalAmount: promotePrice,
          cartItems: [thisProduct], // Pass the selected item
          onPaymentSuccess: () {
            // Define what should happen after a successful donation
            ScaffoldMessenger.of(context).showSnackBar(
              snackBar: const SnackBar(content: Text( data: "Donation successful!")),
            );
          },
        ),  // Donate
      ),  // MaterialPageRoute
    );
  }
}
```

The DonationPage class displays a list of products available for donation, sourced from Firebase Firestore, a sample interface shows in Figure5.3.11. Each product card shows an image, product name, description, actual price, promotion price, and best-before date. The user can browse through the list and choose to donate by pressing a “Donate” button, which triggers a navigation to the ‘Donate’ page. The donation amount is set based on the promotion price of the selected product, and upon successful donation, a success message is displayed via a Snackbar. The page is dynamic, fetching the product list in real-time from Firestore, and the UI is built with ‘StreamBuilder’ to reflect any changes to the product data immediately.

LeaderBoardPage.dart

Line	Code

33-37	Expanded(
	    child: Column(
	      children: [
	        LeaderboardScorePanel(),
	        LeaderboardPanel()

LeaderBoardService.dart

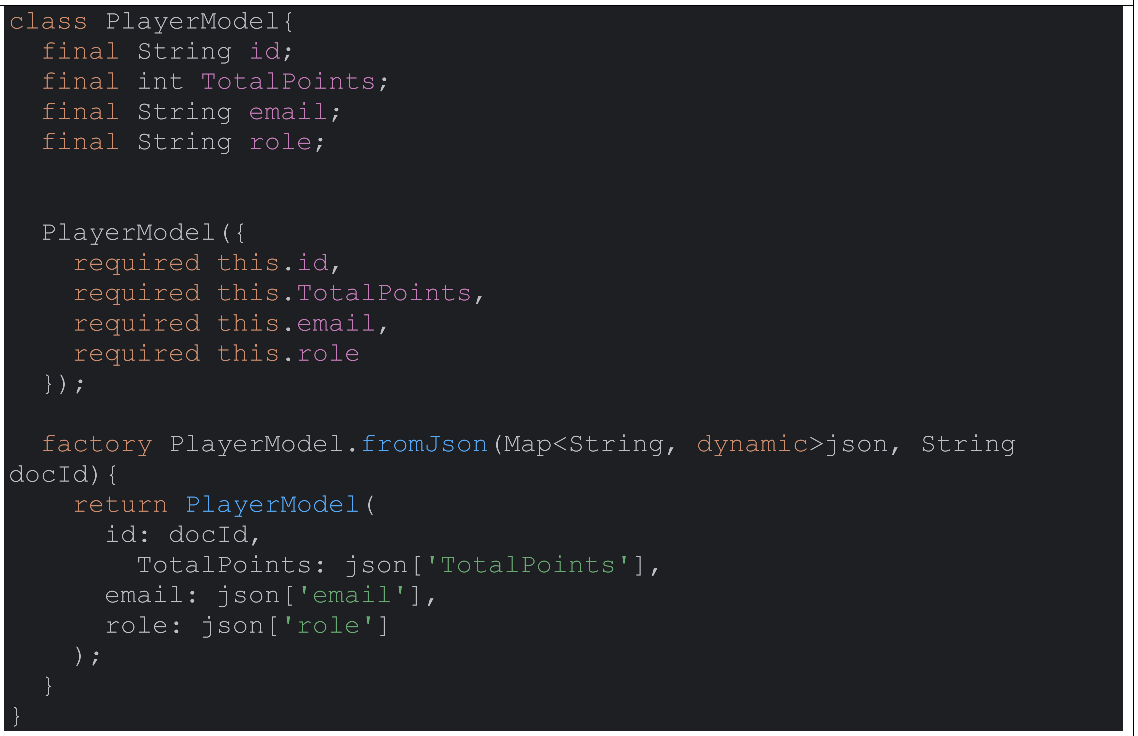
Line	Code

9-29	FirebaseFirestore firestore = FirebaseFirestore.*instance*;
	List<PlayerModel> players = [];
	
	Future<List<PlayerModel>> getPlayers(){
	  Completer<List<PlayerModel>>playersCompleter = Completer();
	
	  if(players.isEmpty){
	    firestore.collection('users').get().then((playerDocs){
	      players =
	playerDocs.docs.map((p)=>PlayerModel.fromJson(p.data(),p.id)).toList();
	
	      Future.delayed(const Duration(seconds: 2), (){
	        playersCompleter.complete(players);
	      });
	    });
	  }
	  else{
	    playersCompleter.complete(players);
	  }
	
	  return playersCompleter.future;
	}

LeaderBoardList.dart

Line	Code
50-54	Expanded(flex: 2, child: Text(player.email, style: rowStyle,)), Expanded(child: Text("${player.TotalPoints}", style: rowStyle,)), Expanded(child: Text("${player.role}", style: rowStyle,)),

playerModel.dart

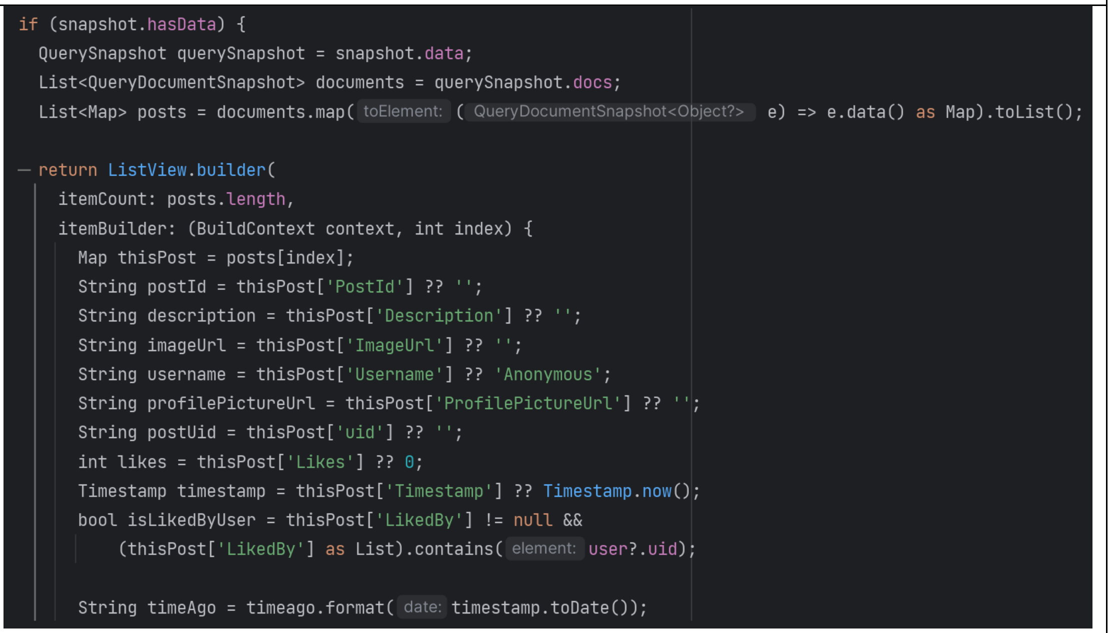
Line	Code
1-23	

```
class PlayerModel{
  final String id;
  final int TotalPoints;
  final String email;
  final String role;


  PlayerModel({
    required this.id,
    required this.TotalPoints,
    required this.email,
    required this.role
  });

  factory PlayerModel.fromJson(Map<String, dynamic>json, String
docId){
    return PlayerModel(
      id: docId,
        TotalPoints: json['TotalPoints'],
      email: json['email'],
      role: json['role']
    );
  }
}
```

The leaderboard system fetches and displays player scores using various components, there a sample interface shows in Figure 5.3.14. The ‘LeaderBoardService’ retrieves player data from Firestore, asynchronously fetching scores and simulating a loading delay. The main page ‘LeaderBoardPage.dart’ brings together components like ‘LeaderboardList’, which sorts players by points and displays their rank, username, and score in a scrollable, well-formatted list.

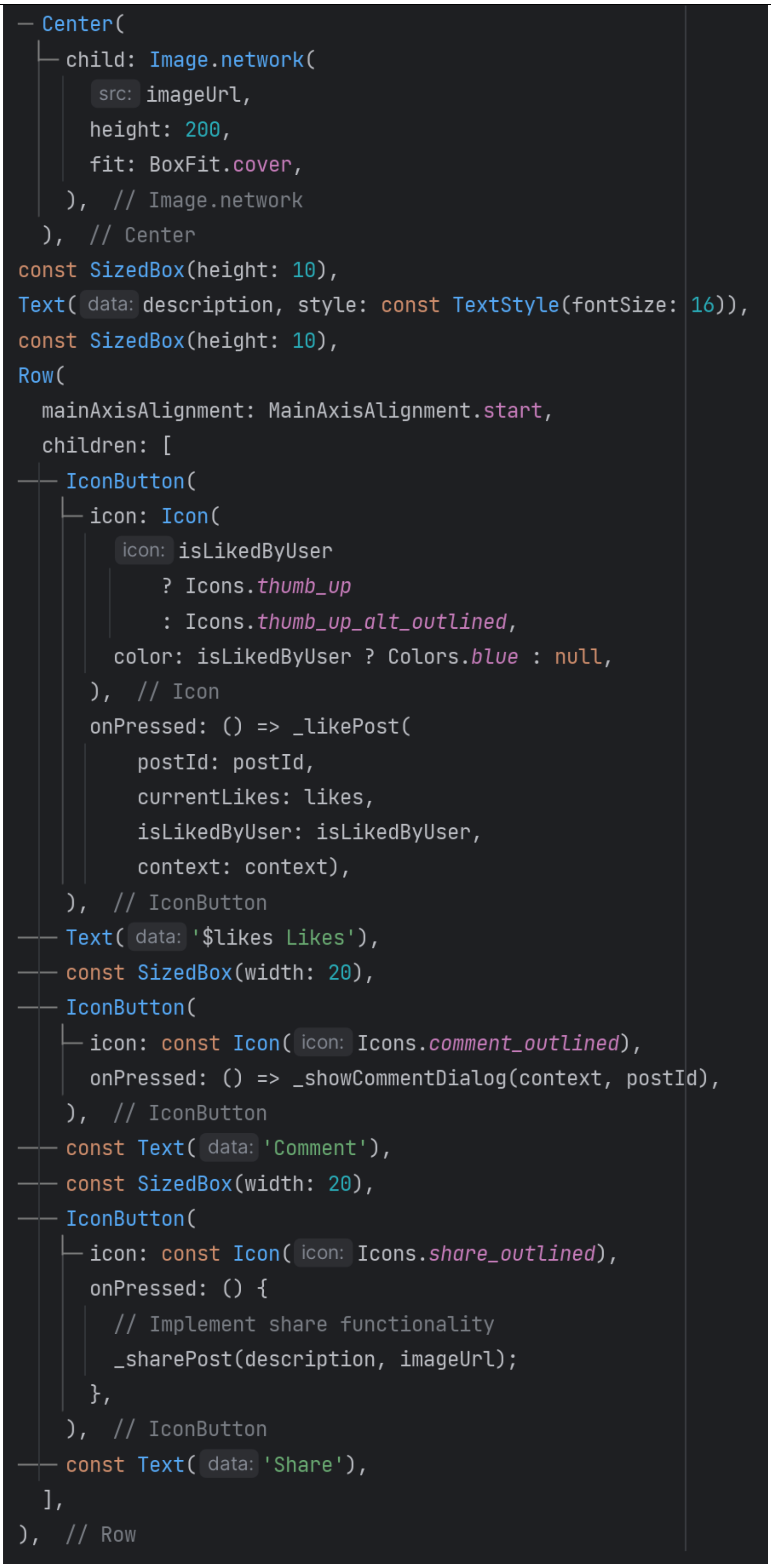
The ‘LeaderBoardLoading.dart’ component shows a loading indicator while data is being fetched and displays player details once loaded. Additionally, ‘LeaderBoardPanel’ and ‘LeaderBoardScorePanel’ handle the layout and display of individual scores, contributing to a structured and user-friendly leaderboard interface.

Post_List.dart

Lines	Code
47-67	

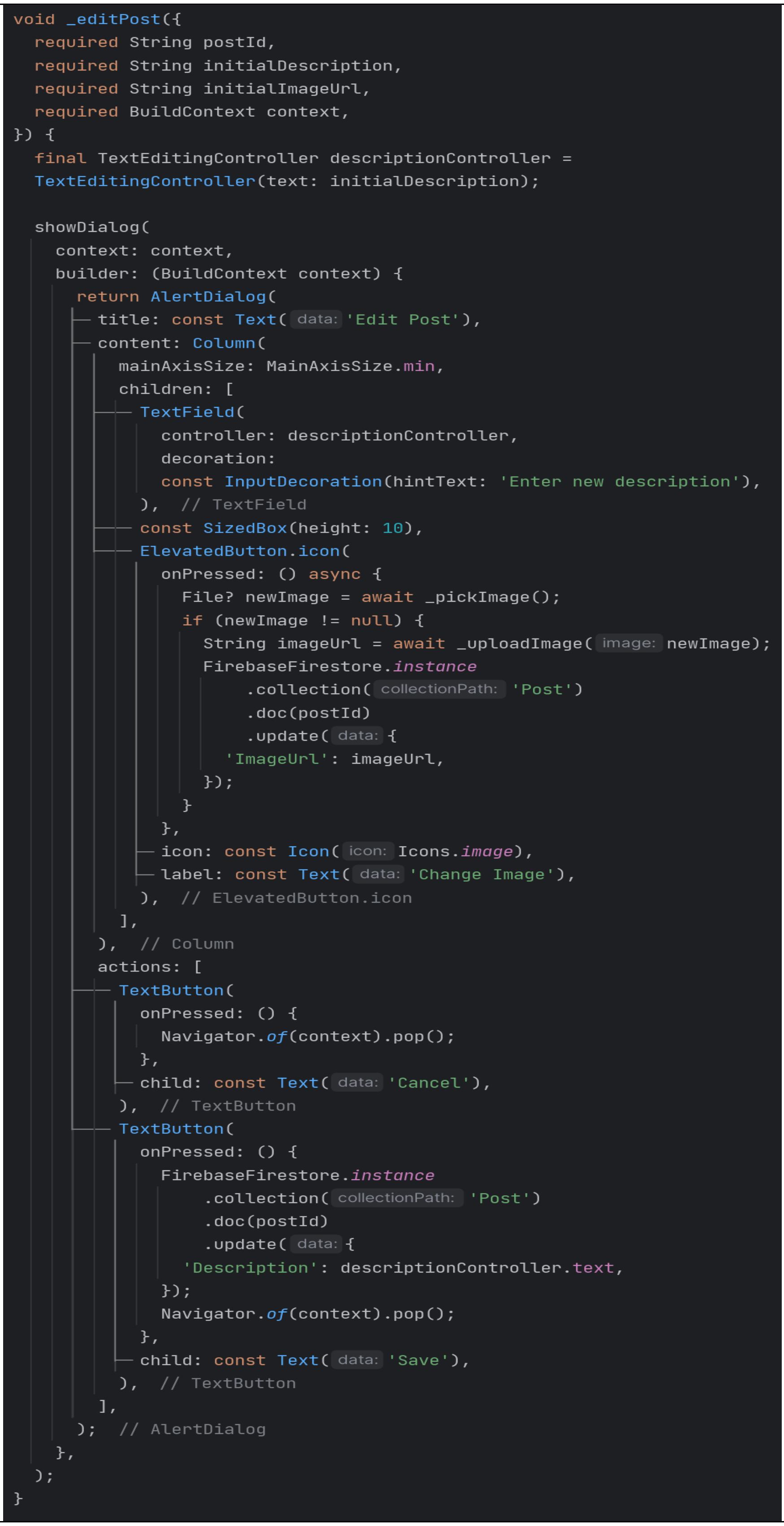
```
if (snapshot.hasData) {
  QuerySnapshot querySnapshot = snapshot.data;
  List<QueryDocumentSnapshot> documents = querySnapshot.docs;
  List<Map> posts = documents.map( toElement: ( QueryDocumentSnapshot<Object?>  e) => e.data() as Map).toList();

  return ListView.builder(
    itemCount: posts.length,
    itemBuilder: (BuildContext context, int index) {
      Map thisPost = posts[index];
      String postId = thisPost['PostId'] ?? '';
      String description = thisPost['Description'] ?? '';
      String imageUrl = thisPost['ImageUrl'] ?? '';
      String username = thisPost['Username'] ?? 'Anonymous';
      String profilePictureUrl = thisPost['ProfilePictureUrl'] ?? '';
      String postUid = thisPost['uid'] ?? '';
      int likes = thisPost['Likes'] ?? 0;
      Timestamp timestamp = thisPost['Timestamp'] ?? Timestamp.now();
      bool isLikedByUser = thisPost['LikedBy'] != null &&
          (thisPost['LikedBy'] as List).contains( element: user?.uid);

      String timeAgo = timeago.format( date: timestamp.toDate());
```

130-173



```
Center(
  child: Image.network(
    src: imageUrl,
    height: 200,
    fit: BoxFit.cover,
  ), // Image.network
), // Center
const SizedBox(height: 10),
Text( data: description, style: const TextStyle(fontSize: 16)),
const SizedBox(height: 10),
Row(
  mainAxisAlignment: MainAxisAlignment.start,
  children: [
    IconButton(
      icon: Icon(
        icon: isLikedByUser
            ? Icons.thumb_up
            : Icons.thumb_up_alt_outlined,
        color: isLikedByUser ? Colors.blue : null,
      ), // Icon
      onPressed: () => _likePost(
          postId: postId,
          currentLikes: likes,
          isLikedByUser: isLikedByUser,
          context: context),
    ), // IconButton
    Text( data: '$likes Likes'),
    const SizedBox(width: 20),
    IconButton(
      icon: const Icon( icon: Icons.comment_outlined),
      onPressed: () => _showCommentDialog(context, postId),
    ), // IconButton
    const Text( data: 'Comment'),
    const SizedBox(width: 20),
    IconButton(
      icon: const Icon( icon: Icons.share_outlined),
      onPressed: () {
        // Implement share functionality
        _sharePost(description, imageUrl);
      },
    ), // IconButton
    const Text( data: 'Share'),
  ],
), // Row
```

372-436

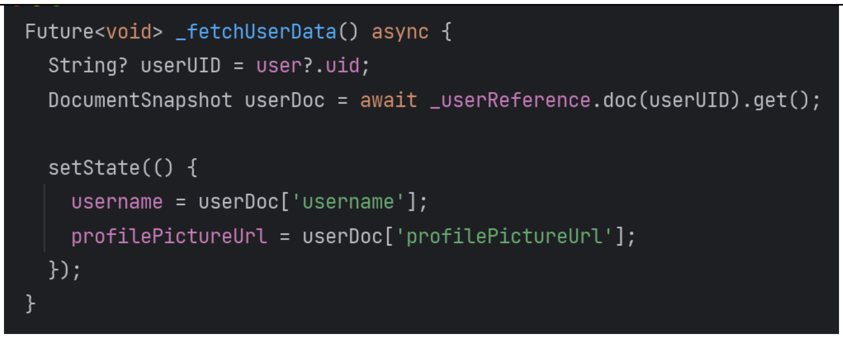


```
void _editPost({
  required String postId,
  required String initialDescription,
  required String initialImageUrl,
  required BuildContext context,
}) {
  final TextEditingController descriptionController =
  TextEditingController(text: initialDescription);

  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text( data: 'Edit Post'),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            TextField(
              controller: descriptionController,
              decoration:
              const InputDecoration(hintText: 'Enter new description'),
            ),  // TextField
            const SizedBox(height: 10),
            ElevatedButton.icon(
              onPressed: () async {
                File? newImage = await _pickImage();
                if (newImage != null) {
                  String imageUrl = await _uploadImage( image: newImage);
                  FirebaseFirestore.instance
                      .collection( collectionPath: 'Post')
                      .doc(postId)
                      .update( data: {
                    'ImageUrl': imageUrl,
                  });
                }
              },
              icon: const Icon( icon: Icons.image),
              label: const Text( data: 'Change Image'),
            ),  // ElevatedButton.icon
          ],
        ),  // Column
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: const Text( data: 'Cancel'),
          ),  // TextButton
          TextButton(
            onPressed: () {
              FirebaseFirestore.instance
                  .collection( collectionPath: 'Post')
                  .doc(postId)
                  .update( data: {
                'Description': descriptionController.text,
              });
              Navigator.of(context).pop();
            },
            child: const Text( data: 'Save'),
          ),  // TextButton
        ],
      );  // AlertDialog
    },
  );
}
```

This Post_List.dart displays a list of social media posts, allowing users to interact with them in various ways. It retrieves posts from a Firebase Firestore collection named "Post" and displays them using a “StreamBuilder”, which updates the UI in real-time as new posts are added or modified. Each post includes a username, profile picture, description, image, timestamp, and buttons for liking, commenting, and sharing. Users can like posts, add comments, or share the post content. If the user is the owner of a post, they can also edit or delete it. The page also allows users to create a new post using a floating action button, which opens a form to submit new content. There are some sample interfaces for edit comment and share post which are shows in Figure 5.3.15 and Figure 5.3.16.

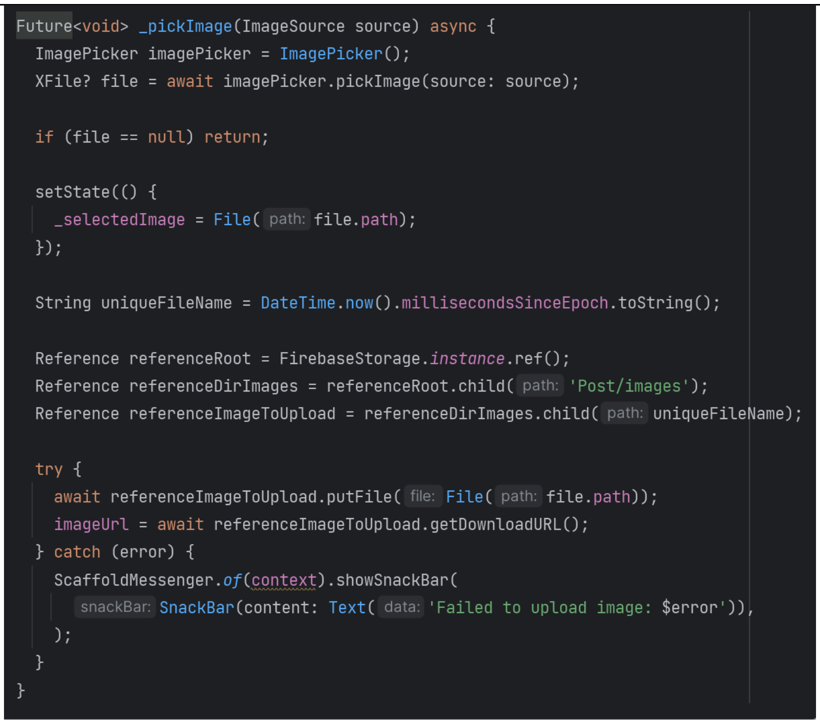
create_post.dart

<table>
<tr><th>Lines</th><th>Code</th></tr>
<tr><td>37-
45</td><td>



```
Future<void> _fetchUserData() async {
  String? userUID = user?.uid;
  DocumentSnapshot userDoc = await _userReference.doc(userUID).get();

  setState(() {
    username = userDoc['username'];
    profilePictureUrl = userDoc['profilePictureUrl'];
  });
}
```

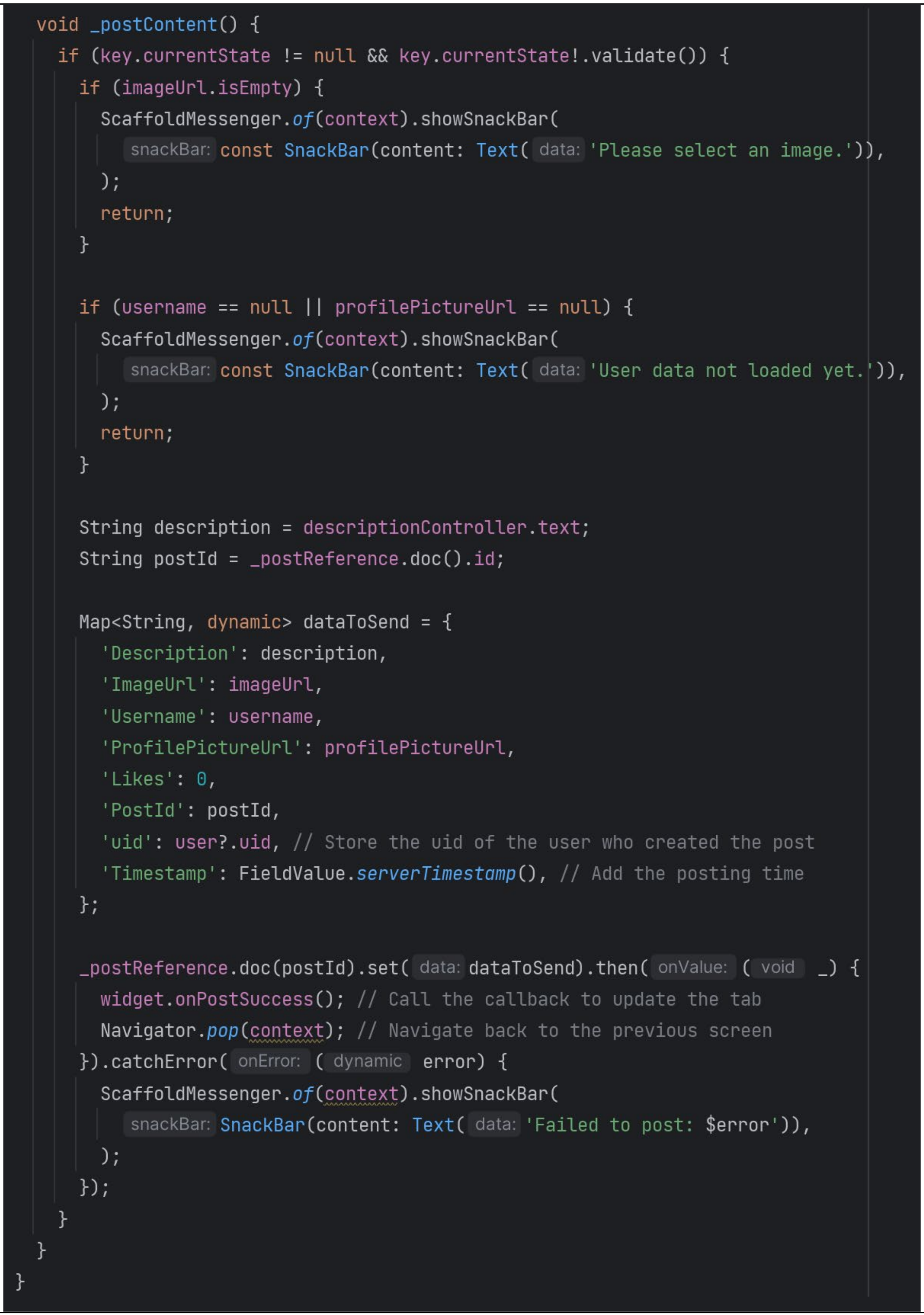
</td></tr>
<tr><td>173-
197</td><td>



```
Future<void> _pickImage(ImageSource source) async {
  ImagePicker imagePicker = ImagePicker();
  XFile? file = await imagePicker.pickImage(source: source);

  if (file == null) return;

  setState(() {
    _selectedImage = File( path: file.path);
  });

  String uniqueFileName = DateTime.now().millisecondsSinceEpoch.toString();

  Reference referenceRoot = FirebaseStorage.instance.ref();
  Reference referenceDirImages = referenceRoot.child( path: 'Post/images');
  Reference referenceImageToUpload = referenceDirImages.child( path: uniqueFileName);

  try {
    await referenceImageToUpload.putFile( file: File( path: file.path));
    imageUrl = await referenceImageToUpload.getDownloadURL();
  } catch (error) {
    ScaffoldMessenger.of(context).showSnackBar(
      snackBar: SnackBar(content: Text( data: 'Failed to upload image: $error')),
    );
  }
}
```

</td></tr>
</table>

199-239

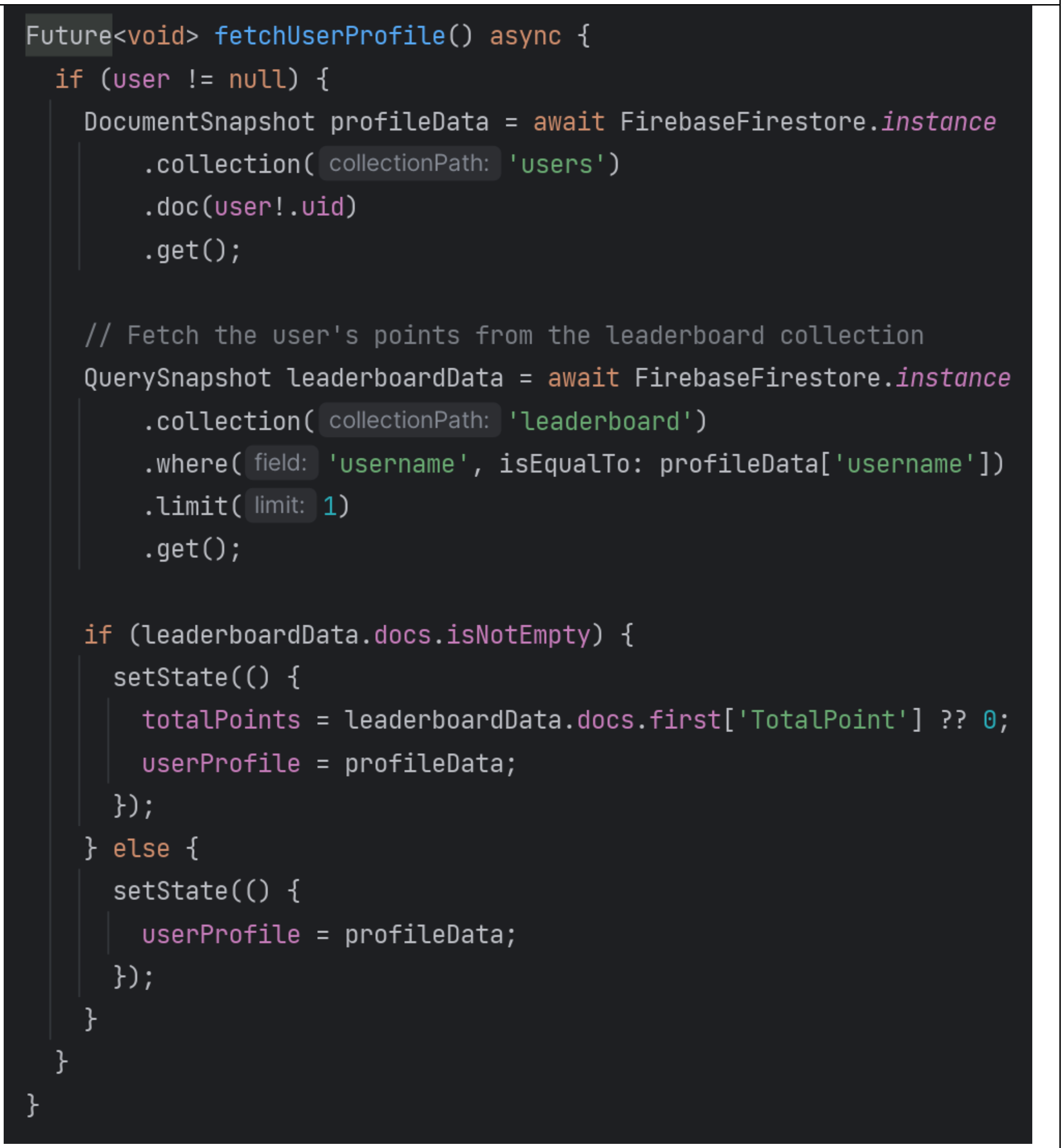


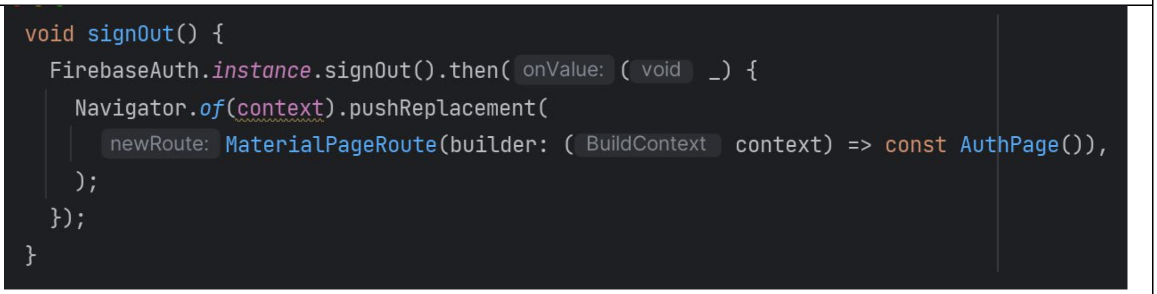
```
  void _postContent() {
    if (key.currentState != null && key.currentState!.validate()) {
      if (imageUrl.isEmpty) {
        ScaffoldMessenger.of(context).showSnackBar(
          snackBar: const SnackBar(content: Text( data: 'Please select an image.')),
        );
        return;
      }

      if (username == null || profilePictureUrl == null) {
        ScaffoldMessenger.of(context).showSnackBar(
          snackBar: const SnackBar(content: Text( data: 'User data not loaded yet.')),
        );
        return;
      }

      String description = descriptionController.text;
      String postId = _postReference.doc().id;

      Map<String, dynamic> dataToSend = {
        'Description': description,
        'ImageUrl': imageUrl,
        'Username': username,
        'ProfilePictureUrl': profilePictureUrl,
        'Likes': 0,
        'PostId': postId,
        'uid': user?.uid, // Store the uid of the user who created the post
        'Timestamp': FieldValue.serverTimestamp(), // Add the posting time
      };

      _postReference.doc(postId).set( data: dataToSend).then( onValue: ( void  _) {
        widget.onPostSuccess(); // Call the callback to update the tab
        Navigator.pop(context); // Navigate back to the previous screen
      }).catchError( onError: ( dynamic  error) {
        ScaffoldMessenger.of(context).showSnackBar(
          snackBar: SnackBar(content: Text( data: 'Failed to post: $error')),
        );
      });
    }
  }
}
```

The CreatePost class allows users to create a new post, upload an image, and write a description, a sample interface shows in Figure 5.3.17. When the page is loaded, it fetches the user's data from Firebase Firestore to display with the post. The user can select an image either from the gallery or by using the camera, which is then uploaded to Firebase Storage. The image's download URL is saved for the post. The user must also enter a description for the post, validated through a form.

Upon submitting the post, the content such as image URL, description, username, profile picture URL, and user ID is saved to Firestore in the 'Post' collection. If the post is successfully created, will navigates back to the previous screen and updates the relevant list or feed to show the new post. Also, ensures proper error handling, such as requiring both an image and a description before allowing the post to be submitted. This class integrates Firebase Authentication, Firestore, and Firebase Storage to handle user posts dynamically.

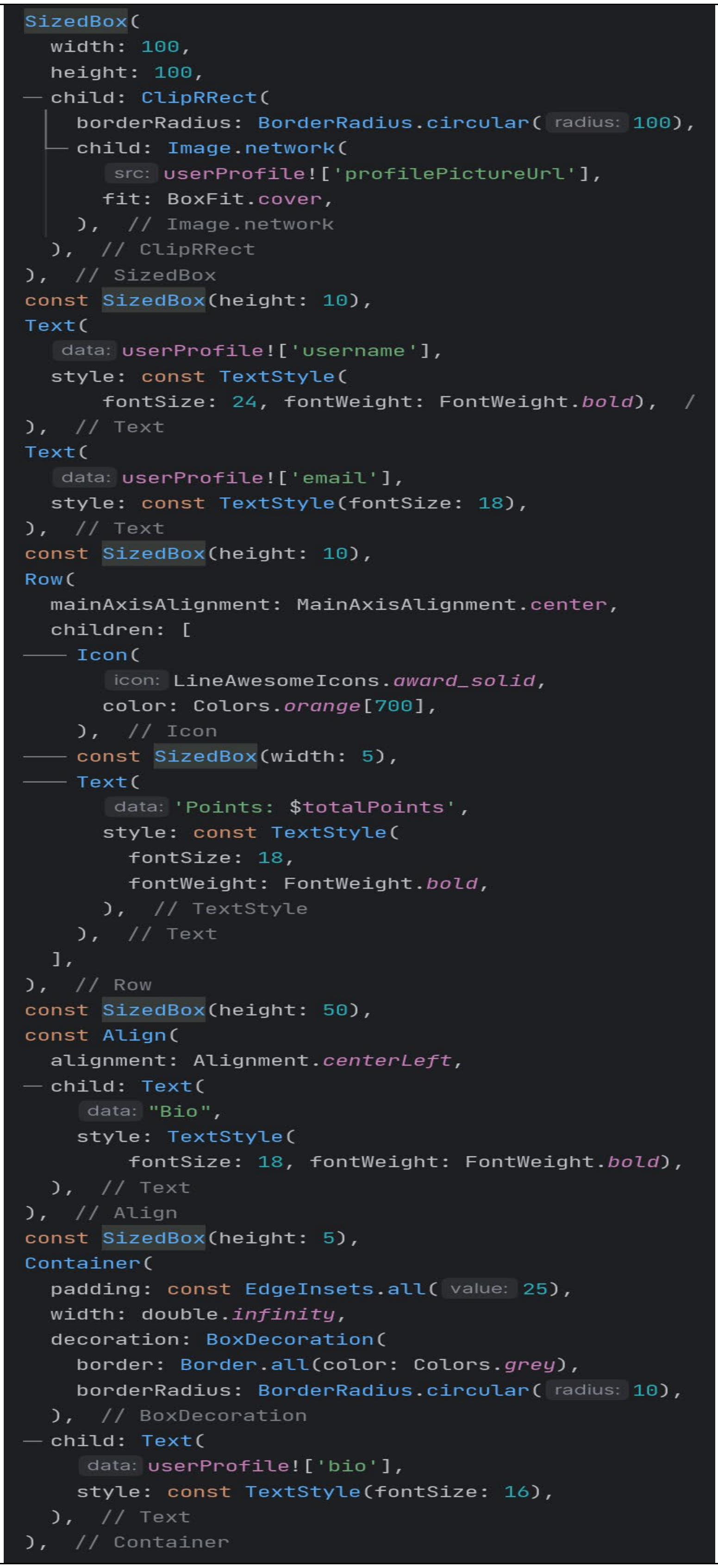
UserProfile.dart

<table>
<tr><th>Lines</th><th>Code</th></tr>
<tr><td>28-
53</td><td>


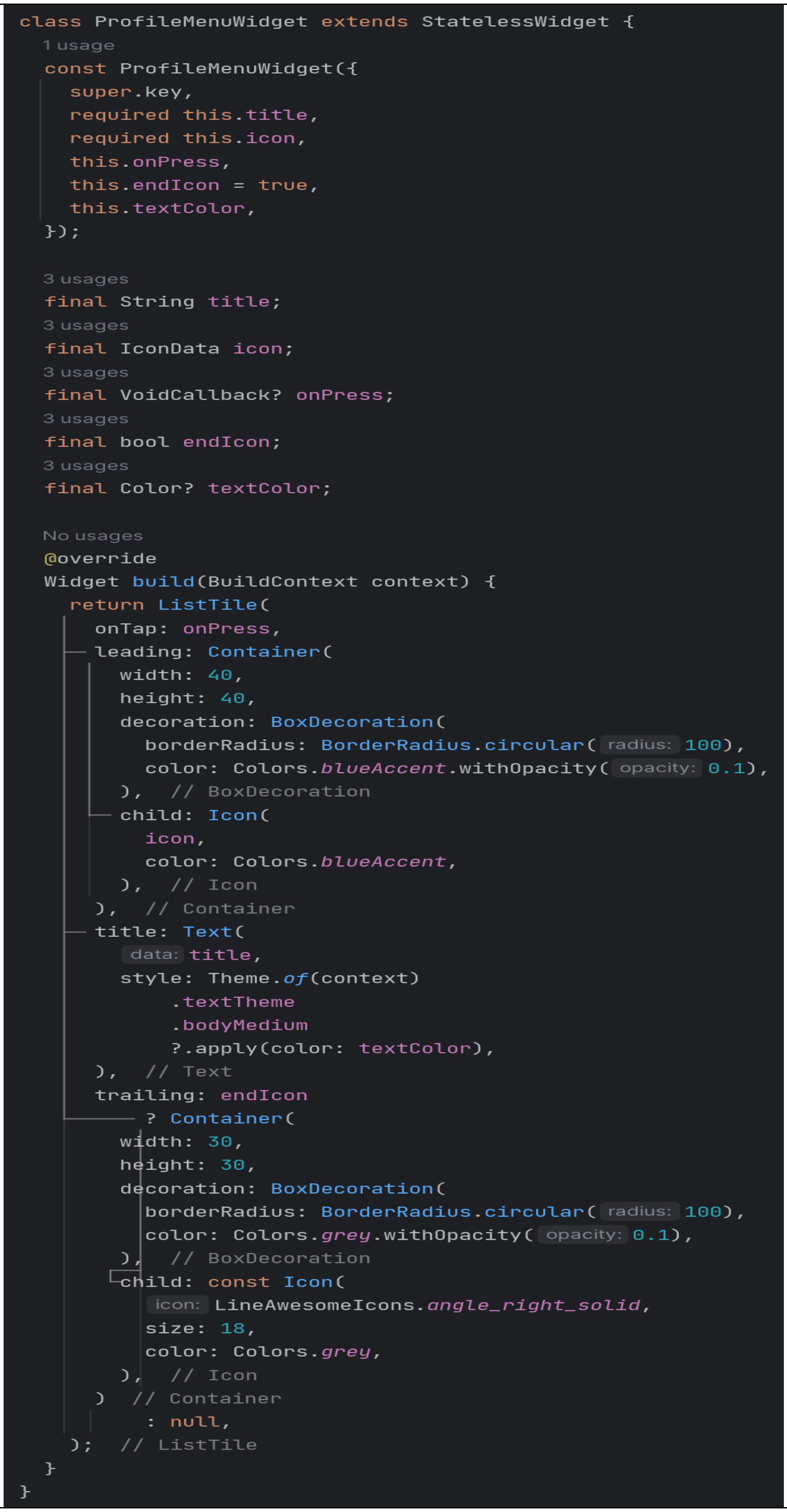
```
Future<void> fetchUserProfile() async {
  if (user != null) {
    DocumentSnapshot profileData = await FirebaseFirestore.instance
        .collection( collectionPath: 'users')
        .doc(user!.uid)
        .get();

    // Fetch the user's points from the leaderboard collection
    QuerySnapshot leaderboardData = await FirebaseFirestore.instance
        .collection( collectionPath: 'leaderboard')
        .where( field: 'username', isEqualTo: profileData['username'])
        .limit( limit: 1)
        .get();

    if (leaderboardData.docs.isNotEmpty) {
      setState(() {
        totalPoints = leaderboardData.docs.first['TotalPoint'] ?? 0;
        userProfile = profileData;
      });
    } else {
      setState(() {
        userProfile = profileData;
      });
    }
  }
}
```
</td></tr>
<tr><td>55-
61</td><td>


```
void signOut() {
  FirebaseAuth.instance.signOut().then( onValue: ( void  _) {
    Navigator.of(context).pushReplacement(
       newRoute: MaterialPageRoute(builder: ( BuildContext  context) => const AuthPage()),
    );
  });
}
```
</td></tr>
</table>

74-134



```
SizedBox(
  width: 100,
  height: 100,
  child: ClipRRect(
    borderRadius: BorderRadius.circular( radius: 100),
    child: Image.network(
       src: userProfile!['profilePictureUrl'],
      fit: BoxFit.cover,
    ), // Image.network
  ), // ClipRRect
), // SizedBox
const SizedBox(height: 10),
Text(
   data: userProfile!['username'],
  style: const TextStyle(
      fontSize: 24, fontWeight: FontWeight.bold),   /
), // Text
Text(
   data: userProfile!['email'],
  style: const TextStyle(fontSize: 18),
), // Text
const SizedBox(height: 10),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Icon(
       icon: LineAwesomeIcons.award_solid,
      color: Colors.orange[700],
    ), // Icon
    const SizedBox(width: 5),
    Text(
       data: 'Points: $totalPoints',
      style: const TextStyle(
        fontSize: 18,
        fontWeight: FontWeight.bold,
      ), // TextStyle
    ), // Text
  ],
), // Row
const SizedBox(height: 50),
const Align(
  alignment: Alignment.centerLeft,
  child: Text(
     data: "Bio",
    style: TextStyle(
        fontSize: 18, fontWeight: FontWeight.bold),
  ), // Text
), // Align
const SizedBox(height: 5),
Container(
  padding: const EdgeInsets.all( value: 25),
  width: double.infinity,
  decoration: BoxDecoration(
    border: Border.all(color: Colors.grey),
    borderRadius: BorderRadius.circular( radius: 10),
  ), // BoxDecoration
  child: Text(
     data: userProfile!['bio'],
    style: const TextStyle(fontSize: 16),
  ), // Text
), // Container
```

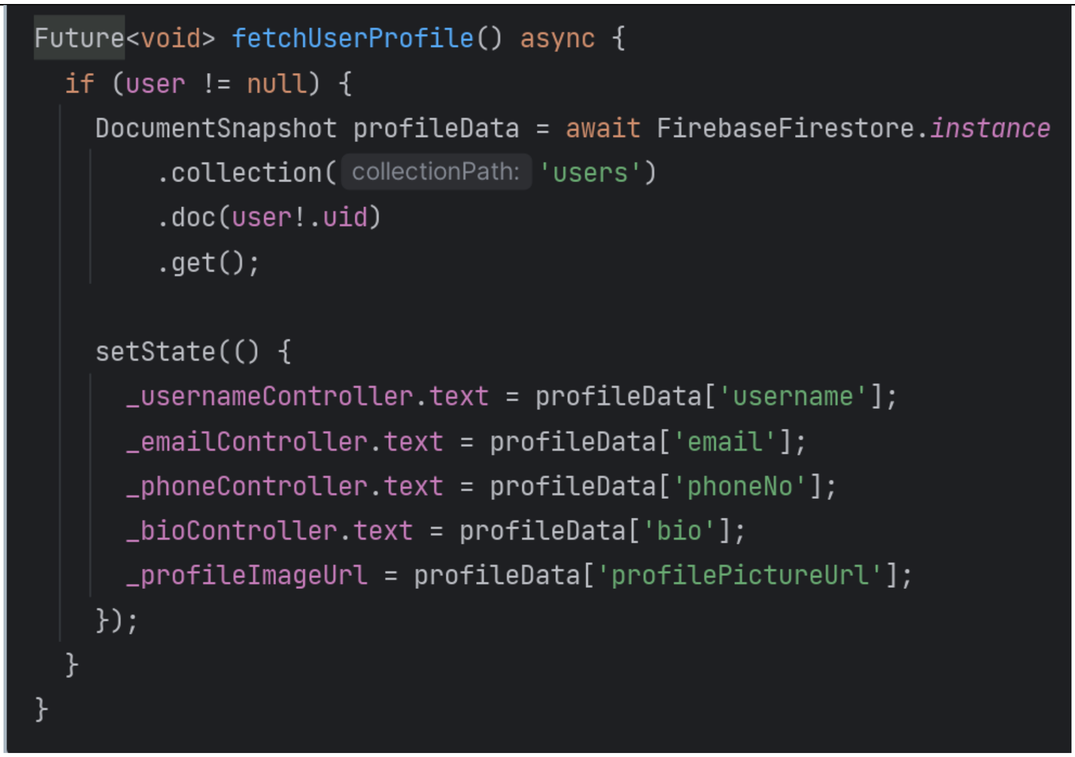
178- 234	

```
class ProfileMenuWidget extends StatelessWidget {
  1 usage
  const ProfileMenuWidget({
    super.key,
    required this.title,
    required this.icon,
    this.onPress,
    this.endIcon = true,
    this.textColor,
  });

  3 usages
  final String title;
  3 usages
  final IconData icon;
  3 usages
  final VoidCallback? onPress;
  3 usages
  final bool endIcon;
  3 usages
  final Color? textColor;

  No usages
  @override
  Widget build(BuildContext context) {
    return ListTile(
      onTap: onPress,
      leading: Container(
        width: 40,
        height: 40,
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular( radius: 100),
          color: Colors.blueAccent.withOpacity( opacity: 0.1),
        ),  // BoxDecoration
        child: Icon(
          icon,
          color: Colors.blueAccent,
        ),  // Icon
      ),  // Container
      title: Text(
        data: title,
        style: Theme.of(context)
            .textTheme
            .bodyMedium
            ?.apply(color: textColor),
      ),  // Text
      trailing: endIcon
          ? Container(
        width: 30,
        height: 30,
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular( radius: 100),
          color: Colors.grey.withOpacity( opacity: 0.1),
        ),  // BoxDecoration
        child: const Icon(
          icon: LineAwesomeIcons.angle_right_solid,
          size: 18,
          color: Colors.grey,
        ),  // Icon
      )  // Container
          : null,
    );  // ListTile
  }
}
```

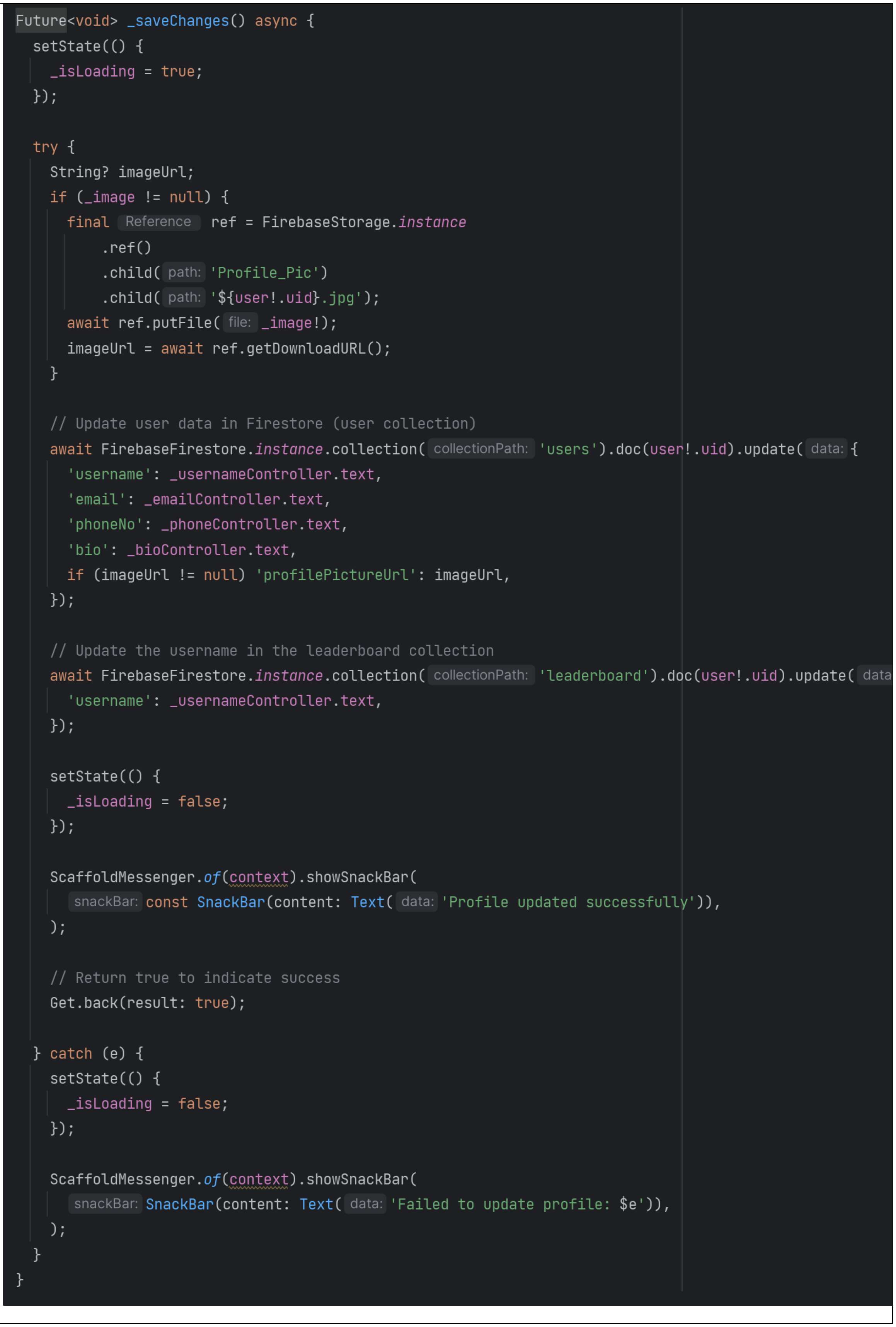
The UserProfilePage class allows users to view and manage their profile information, a sample interface shows in Figure 5.3.18. When the page is initialized, it fetches the user's data from Firestore, including their profile picture, username, email, bio, and points from the leaderboard. This data is displayed on the profile page, with options to edit the profile or log out. The profile page features the user's image, username, email, total points, and bio. The “Edit Profile” button allows users to navigate to an edit screen where they can update their profile information, and after returning, the updated data is re-fetched to reflect changes. A “Logout” button is also provided, which signs the user out of the app and navigates them back to the authentication page.

The ProfileMenuWidget class is a reusable widget that displays a menu option with an icon and text, such as the “Logout” option. It can trigger different actions based on user interaction. Overall, the page provides a structured interface for viewing and updating user profile details while integrating with Firebase for data management.

Edit_UserProfile.dart

Lines	Code
33- 48	

```
Future<void> fetchUserProfile() async {
  if (user != null) {
    DocumentSnapshot profileData = await FirebaseFirestore.instance
        .collection( collectionPath: 'users')
        .doc(user!.uid)
        .get();

    setState(() {
      _usernameController.text = profileData['username'];
      _emailController.text = profileData['email'];
      _phoneController.text = profileData['phoneNo'];
      _bioController.text = profileData['bio'];
      _profileImageUrl = profileData['profilePictureUrl'];
    });
  }
}
```

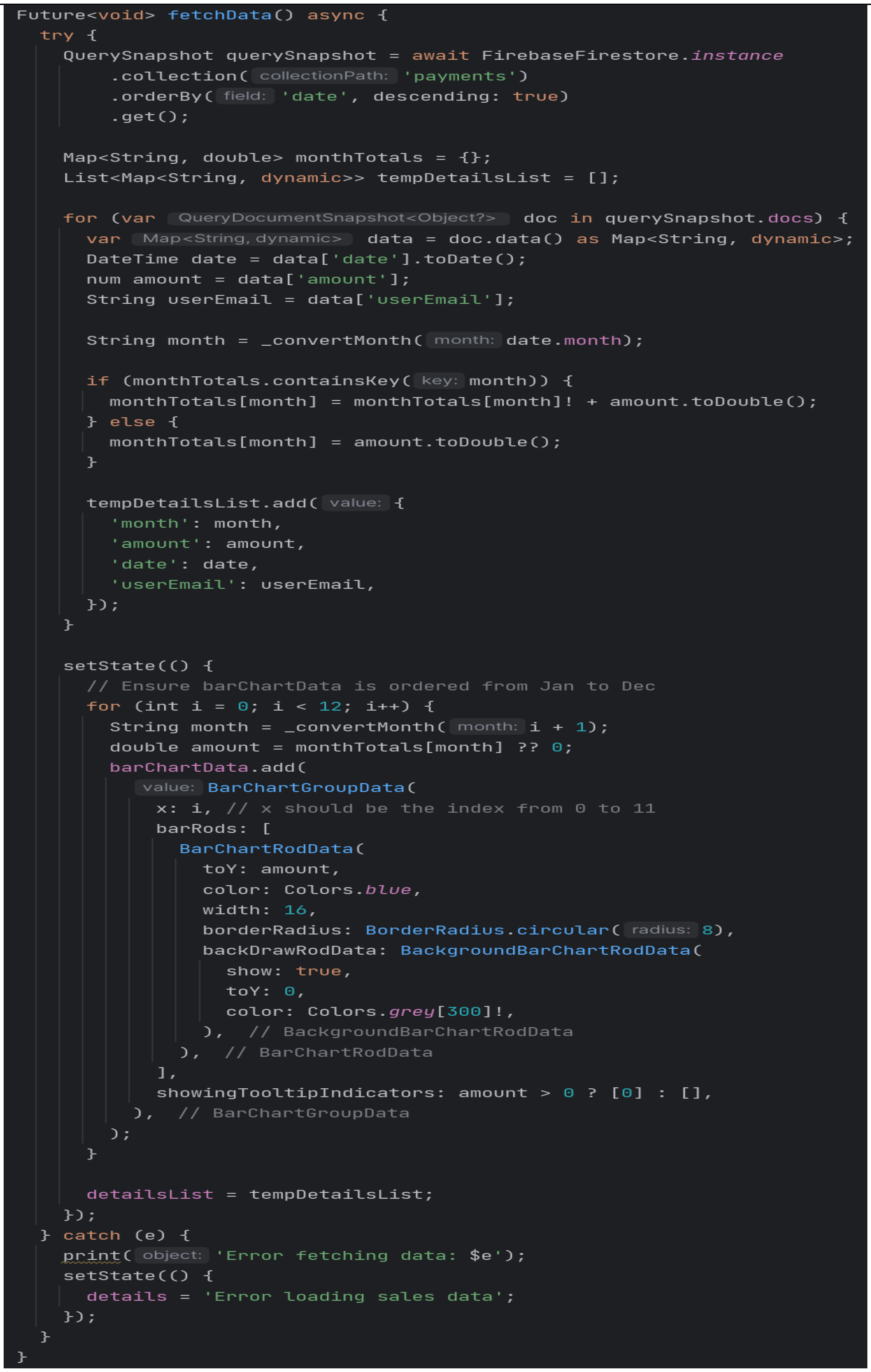
60- 110	

```
Future<void> _saveChanges() async {
  setState(() {
    _isLoading = true;
  });

  try {
    String? imageUrl;
    if (_image != null) {
      final Reference ref = FirebaseStorage.instance
          .ref()
          .child( path: 'Profile_Pic')
          .child( path: '${user!.uid}.jpg');
      await ref.putFile( file: _image!);
      imageUrl = await ref.getDownloadURL();
    }

    // Update user data in Firestore (user collection)
    await FirebaseFirestore.instance.collection( collectionPath: 'users').doc(user!.uid).update( data: {
      'username': _usernameController.text,
      'email': _emailController.text,
      'phoneNo': _phoneController.text,
      'bio': _bioController.text,
      if (imageUrl != null) 'profilePictureUrl': imageUrl,
    });

    // Update the username in the leaderboard collection
    await FirebaseFirestore.instance.collection( collectionPath: 'leaderboard').doc(user!.uid).update( data
      'username': _usernameController.text,
    });

    setState(() {
      _isLoading = false;
    });

    ScaffoldMessenger.of(context).showSnackBar(
       snackBar: const SnackBar(content: Text( data: 'Profile updated successfully')),
    );

    // Return true to indicate success
    Get.back(result: true);

  } catch (e) {
    setState(() {
      _isLoading = false;
    });

    ScaffoldMessenger.of(context).showSnackBar(
       snackBar: SnackBar(content: Text( data: 'Failed to update profile: $e')),
    );
  }
}
```

The EditUserProfile class allows users to edit their profile information, a sample interface shows in Figure 5.3.19. When the page loads, the user's current profile data such as username, email, phone number, bio, and profile picture is fetched from Firestore and displayed in form fields. The user can update these details and select a new profile picture from their device's gallery. Once the user submits the changes by pressing the “Save Changes” button, the updated data is saved back to Firestore. If a new profile picture is selected, it is uploaded to Firebase Storage, and its URL is updated in the user's profile. Additionally, the username is updated in both the ‘users’ and ‘leaderboard’ collections.

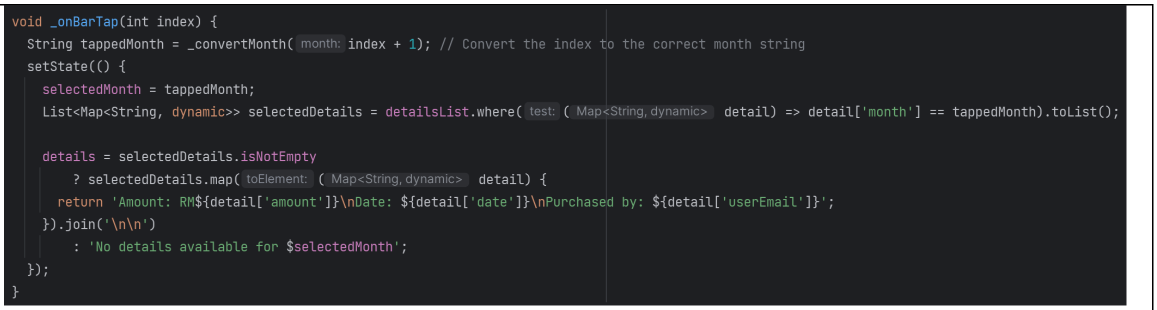
5.3.2 Manager Pages Code

Sales_Report.dart

Lines	Code
31- 97	

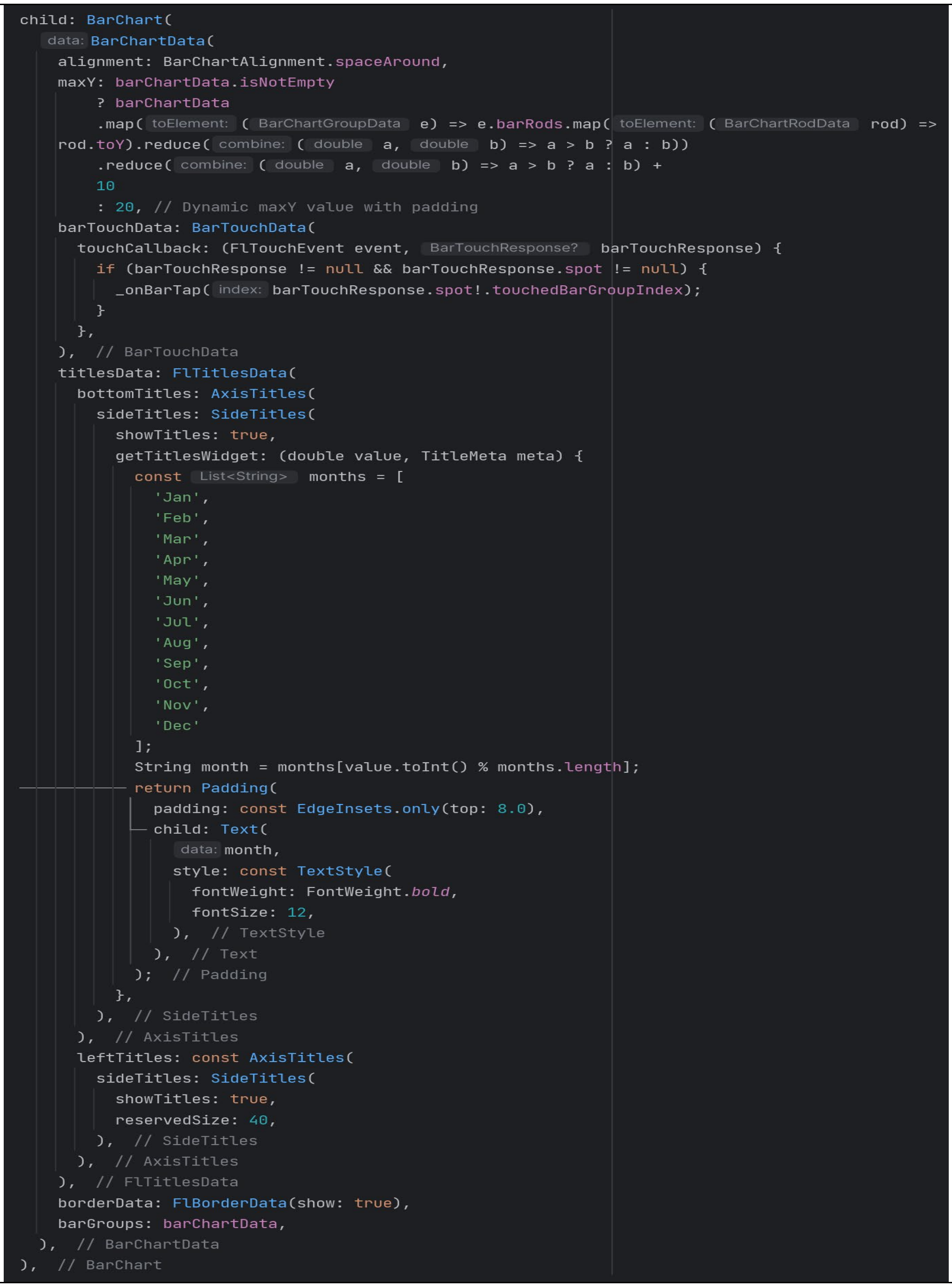
```
Future<void> fetchData() async {
  try {
    QuerySnapshot querySnapshot = await FirebaseFirestore.instance
        .collection( collectionPath: 'payments')
        .orderBy( field: 'date', descending: true)
        .get();

    Map<String, double> monthTotals = {};
    List<Map<String, dynamic>> tempDetailsList = [];

    for (var  QueryDocumentSnapshot<Object?>  doc in querySnapshot.docs) {
      var  Map<String, dynamic>  data = doc.data() as Map<String, dynamic>;
      DateTime date = data['date'].toDate();
      num amount = data['amount'];
      String userEmail = data['userEmail'];

      String month = _convertMonth( month: date.month);

      if (monthTotals.containsKey( key: month)) {
        monthTotals[month] = monthTotals[month]! + amount.toDouble();
      } else {
        monthTotals[month] = amount.toDouble();
      }

      tempDetailsList.add( value: {
        'month': month,
        'amount': amount,
        'date': date,
        'userEmail': userEmail,
      });
    }

    setState(() {
      // Ensure barChartData is ordered from Jan to Dec
      for (int i = 0; i < 12; i++) {
        String month = _convertMonth( month: i + 1);
        double amount = monthTotals[month] ?? 0;
        barChartData.add(
           value: BarChartGroupData(
            x: i, // x should be the index from 0 to 11
            barRods: [
              BarChartRodData(
                toY: amount,
                color: Colors.blue,
                width: 16,
                borderRadius: BorderRadius.circular( radius: 8),
                backDrawRodData: BackgroundBarChartRodData(
                  show: true,
                  toY: 0,
                  color: Colors.grey[300]!,
                ),  // BackgroundBarChartRodData
              ),  // BarChartRodData
            ],
            showingTooltipIndicators: amount > 0 ? [0] : [],
          ),  // BarChartGroupData
        );
      }

      detailsList = tempDetailsList;
    });
  } catch (e) {
    print( object: 'Error fetching data: $e');
    setState(() {
      details = 'Error loading sales data';
    });
  }
}
```

130-142



```
void _onBarTap(int index) {
  String tappedMonth = _convertMonth( month: index + 1); // Convert the index to the correct month string
  setState(() {
    selectedMonth = tappedMonth;
    List<Map<String, dynamic>> selectedDetails = detailsList.where( test: ( Map<String, dynamic>  detail) => detail['month'] == tappedMonth).toList();

    details = selectedDetails.isNotEmpty
        ? selectedDetails.map( toElement: ( Map<String, dynamic>  detail) {
      return 'Amount: RM${detail['amount']}\nDate: ${detail['date']}\nPurchased by: ${detail['userEmail']}';
    }).join('\n\n')
        : 'No details available for $selectedMonth';
  });
}
```

177-237

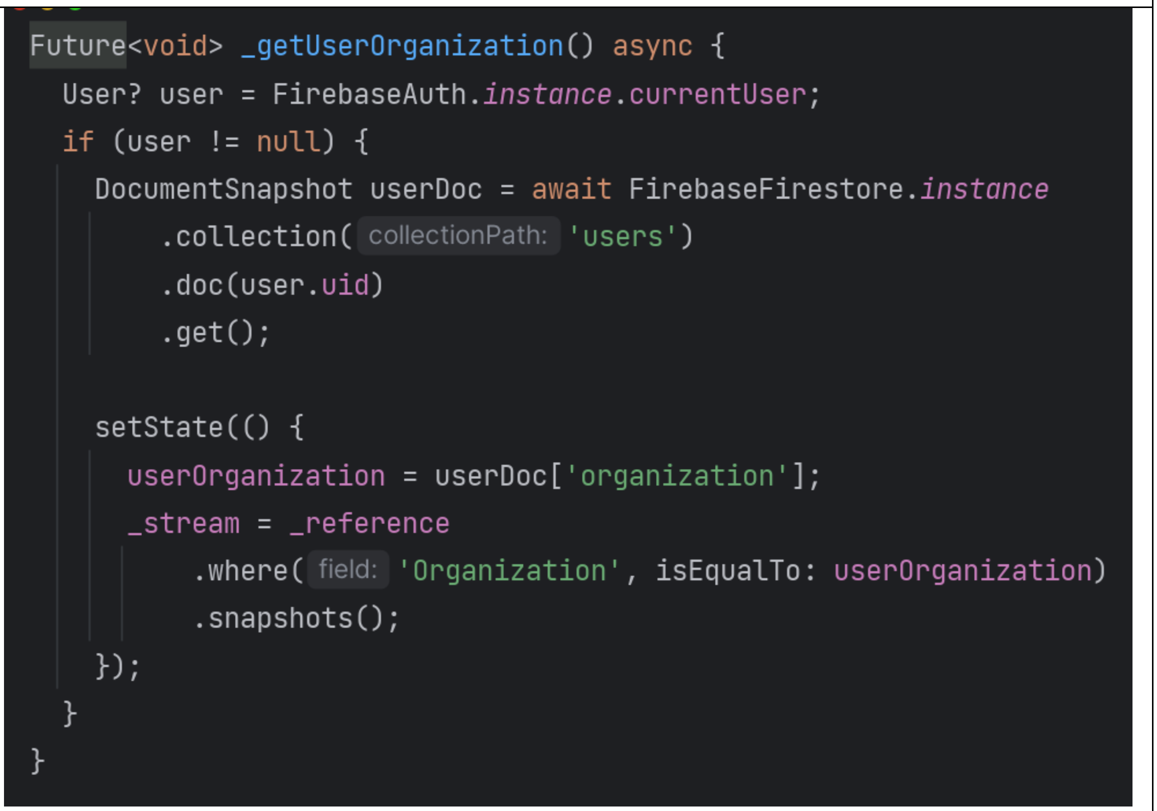


```
child: BarChart(
   data: BarChartData(
    alignment: BarChartAlignment.spaceAround,
    maxY: barChartData.isNotEmpty
        ? barChartData
        .map( toElement: ( BarChartGroupData  e) => e.barRods.map( toElement: ( BarChartRodData  rod) =>
    rod.toY).reduce( combine: ( double  a,  double  b) => a > b ? a : b))
        .reduce( combine: ( double  a,  double  b) => a > b ? a : b) +
        10
        : 20, // Dynamic maxY value with padding
    barTouchData: BarTouchData(
      touchCallback: (FlTouchEvent event,  BarTouchResponse?  barTouchResponse) {
        if (barTouchResponse != null && barTouchResponse.spot != null) {
          _onBarTap( index: barTouchResponse.spot!.touchedBarGroupIndex);
        }
      },
    ),  // BarTouchData
    titlesData: FlTitlesData(
      bottomTitles: AxisTitles(
        sideTitles: SideTitles(
          showTitles: true,
          getTitlesWidget: (double value, TitleMeta meta) {
            const  List<String>  months = [
              'Jan',
              'Feb',
              'Mar',
              'Apr',
              'May',
              'Jun',
              'Jul',
              'Aug',
              'Sep',
              'Oct',
              'Nov',
              'Dec'
            ];
            String month = months[value.toInt() % months.length];
            return Padding(
              padding: const EdgeInsets.only(top: 8.0),
              child: Text(
                 data: month,
                style: const TextStyle(
                  fontWeight: FontWeight.bold,
                  fontSize: 12,
                ),  // TextStyle
              ),  // Text
            );  // Padding
          },
        ),  // SideTitles
      ),  // AxisTitles
      leftTitles: const AxisTitles(
        sideTitles: SideTitles(
          showTitles: true,
          reservedSize: 40,
        ),  // SideTitles
      ),  // AxisTitles
    ),  // FlTitlesData
    borderData: FlBorderData(show: true),
    barGroups: barChartData,
  ),  // BarChartData
),  // BarChart
```

The SalesReport class displays a sales report with a bar chart showing the total sales for each month, a sample interface shows in Figure 5.3.20. When the page is initialized, it fetches data from a Firestore collection named ‘payments.’ The data is organized by month, and the total sales amount for each month is calculated and displayed in a bar chart. The **fetchData** function processes the payment data, grouping it by month and adding each month's total to a bar chart. The chart is interactive, allowing users to tap on a bar to see details of the sales for that specific month, such as the amount, date, and customer email. The page features a horizontal scrollable bar chart that dynamically adjusts its max Y-axis value based on the highest sales figure. Below the chart, users can view detailed sales information for the selected month. Additionally, the page includes a navigation drawer with options to view product lists, add products, view the manager's profile, and sign out, a sample interface shows in Figure 5.3.21. The sign-out function logs the user out and navigates back to the login page. Overall, this page provides a visual overview of monthly sales and detailed data for each transaction.

Product_List.dart

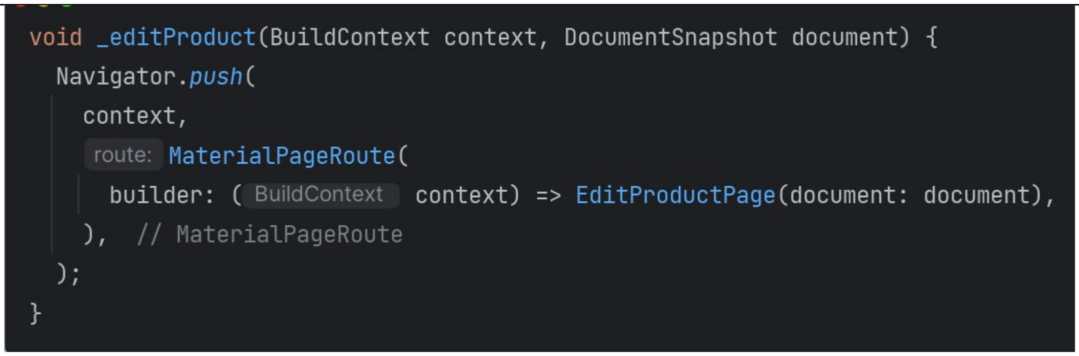
<table>
<tr><th>Lines</th><th>Code</th></tr>
<tr><td>32-
47</td><td>



```
Future<void> _getUserOrganization() async {
  User? user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    DocumentSnapshot userDoc = await FirebaseFirestore.instance
        .collection( collectionPath: 'users')
        .doc(user.uid)
        .get();

    setState(() {
      userOrganization = userDoc['organization'];
      _stream = _reference
          .where( field: 'Organization', isEqualTo: userOrganization)
          .snapshots();
    });
  }
}
```

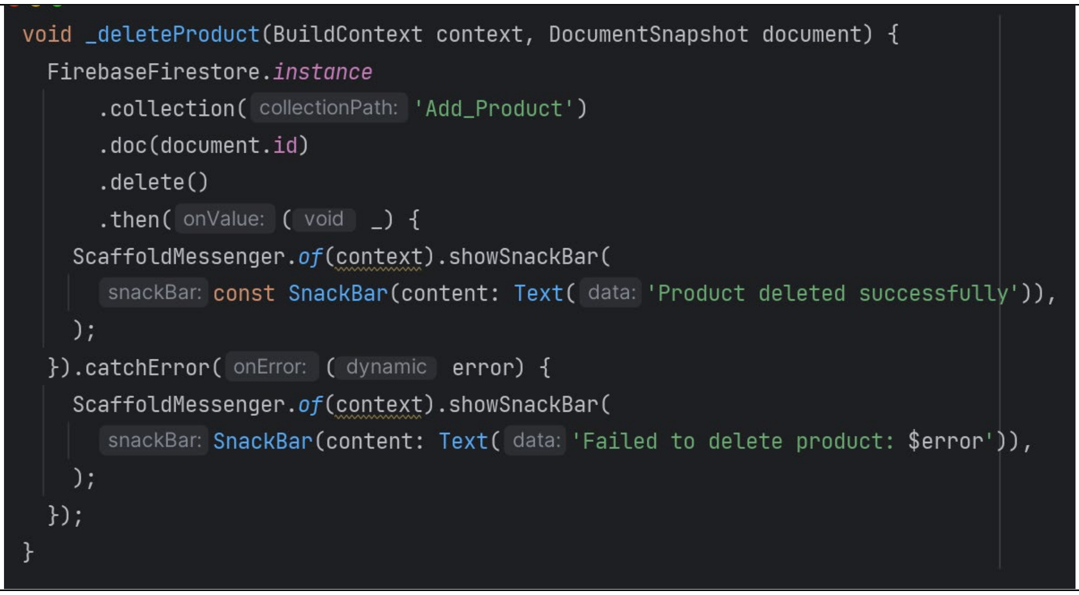
</td></tr>
</table>

<table>
<tr>
<td>81-
88</td>
<td>



```
void _editProduct(BuildContext context, DocumentSnapshot document) {
  Navigator.push(
    context,
    route: MaterialPageRoute(
      builder: ( BuildContext  context) => EditProductPage(document: document),
    ),  // MaterialPageRoute
  );
}
```

</td>
</tr>
<tr>
<td>90-
104</td>
<td>



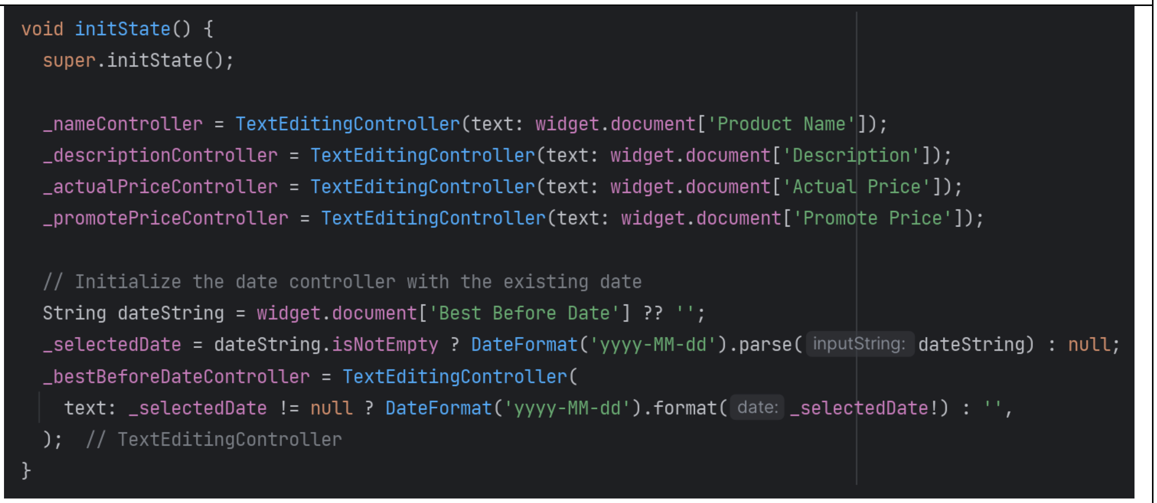
```
void _deleteProduct(BuildContext context, DocumentSnapshot document) {
  FirebaseFirestore.instance
      .collection( collectionPath: 'Add_Product')
      .doc(document.id)
      .delete()
      .then( onValue: ( void  _) {
    ScaffoldMessenger.of(context).showSnackBar(
       snackBar: const SnackBar(content: Text( data: 'Product deleted successfully')),
    );
  }).catchError( onError: ( dynamic  error) {
    ScaffoldMessenger.of(context).showSnackBar(
       snackBar: SnackBar(content: Text( data: 'Failed to delete product: $error')),
    );
  });
}
```

</td>
</tr>
</table>

The Product_List_Manager class display a list of products managed by a specific organization; a sample interface shows in Figure 5.3.22. When the page is initialized, it fetches the organization data for the current authenticated user from Firestore and retrieves the products associated with that organization. The products are displayed in a list, each showing details like product name, description, price, promotion price, and best-before date. Users can interact with the products by long-pressing on a product card, which opens a dialog with options to either edit or delete the product, a sample interface shows in Figure 5.3.23. If the user chooses to edit a product, they are navigated to an 'EditProductPage', and if they choose to delete, the product is removed from Firestore. The page also includes a drawer for navigation, allowing the user to access other parts of the app, such as adding new products, viewing sales reports, managing their profile, or signing out. It provides a smooth way for managers to manage the product list of their organization in real time, updating the view dynamically as changes occur in the Firestore database.

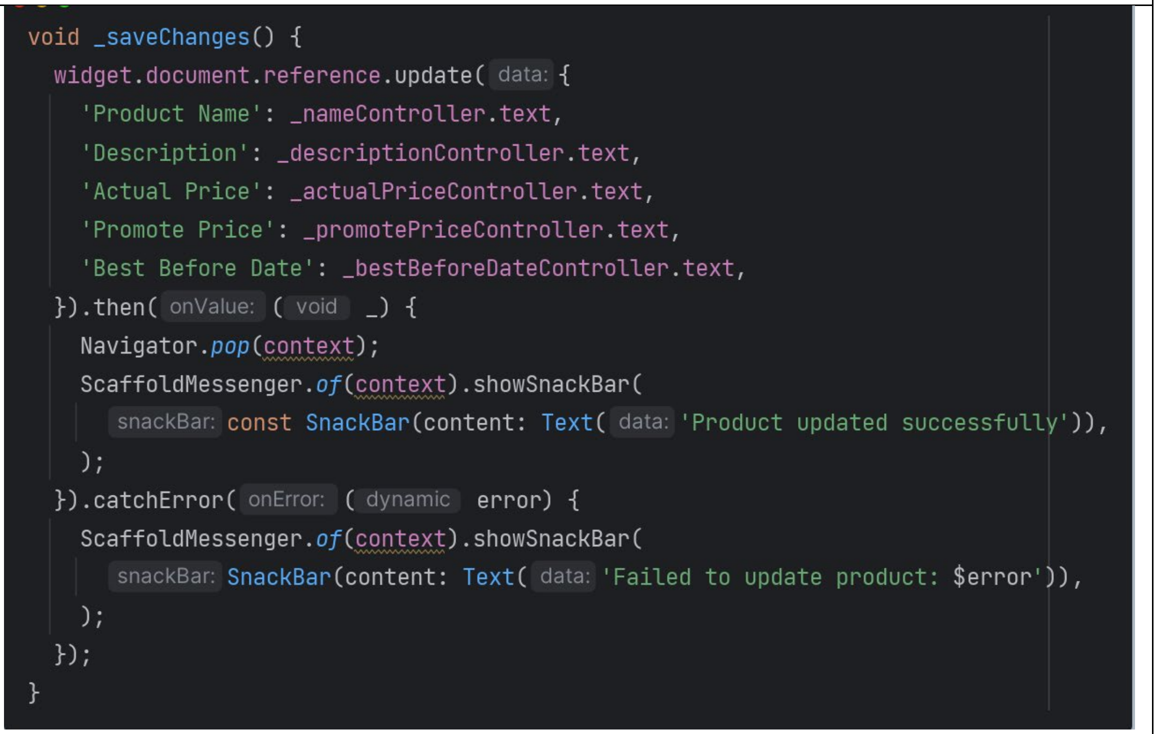
EditProduct.dart

<table>
<tr><th>Lines</th><th>Code</th></tr>
<tr><td>24-
37</td><td>



```
void initState() {
  super.initState();

  _nameController = TextEditingController(text: widget.document['Product Name']);
  _descriptionController = TextEditingController(text: widget.document['Description']);
  _actualPriceController = TextEditingController(text: widget.document['Actual Price']);
  _promotePriceController = TextEditingController(text: widget.document['Promote Price']);

  // Initialize the date controller with the existing date
  String dateString = widget.document['Best Before Date'] ?? '';
  _selectedDate = dateString.isNotEmpty ? DateFormat('yyyy-MM-dd').parse( inputString: dateString) : null;
  _bestBeforeDateController = TextEditingController(
    text: _selectedDate != null ? DateFormat('yyyy-MM-dd').format( date: _selectedDate!) : '',
  );  // TextEditingController
}
```

</td></tr>
<tr><td>55-
72</td><td>

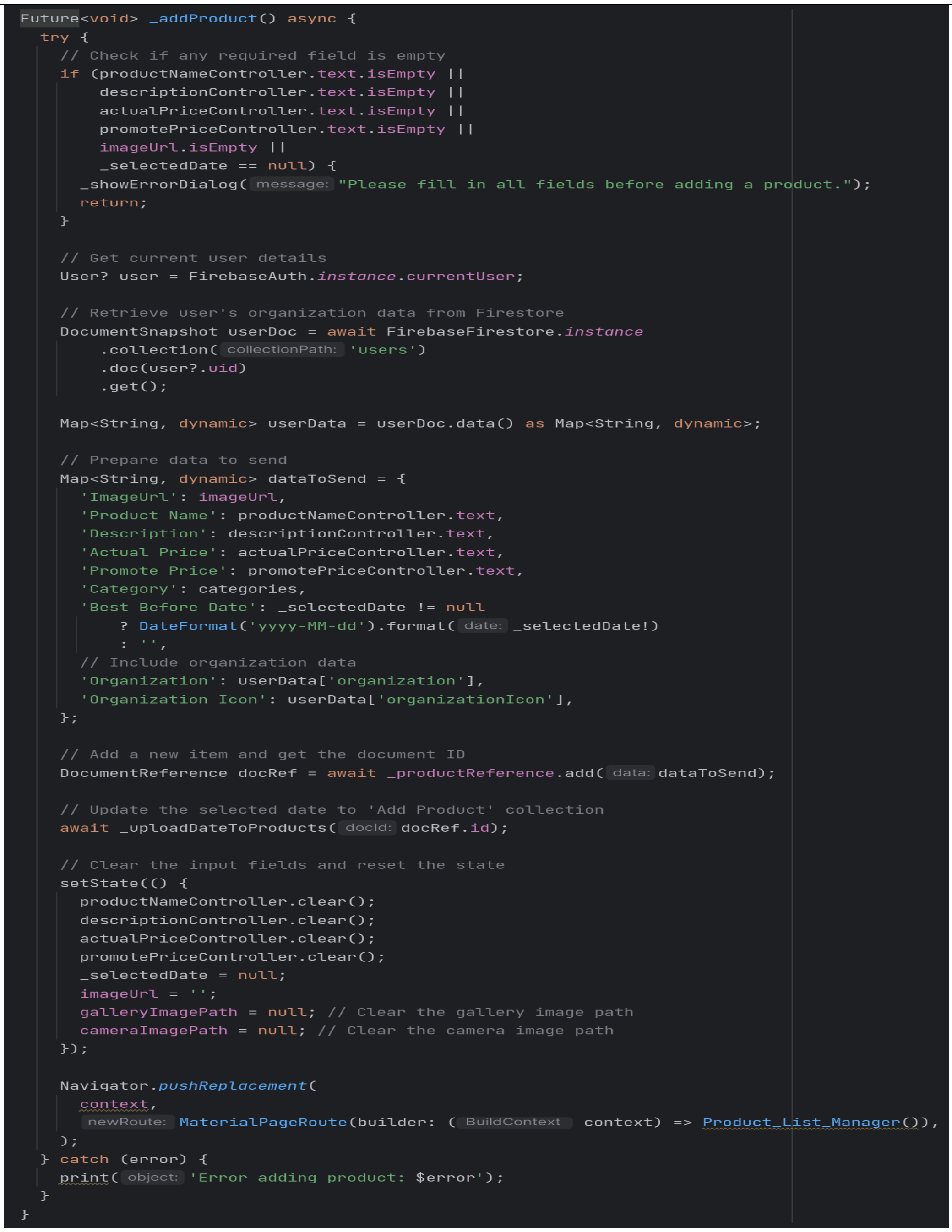


```
void _saveChanges() {
  widget.document.reference.update( data: {
    'Product Name': _nameController.text,
    'Description': _descriptionController.text,
    'Actual Price': _actualPriceController.text,
    'Promote Price': _promotePriceController.text,
    'Best Before Date': _bestBeforeDateController.text,
  }).then( onValue: ( void  _) {
    Navigator.pop(context);
    ScaffoldMessenger.of(context).showSnackBar(
      snackBar: const SnackBar(content: Text( data: 'Product updated successfully')),
    );
  }).catchError( onError: ( dynamic  error) {
    ScaffoldMessenger.of(context).showSnackBar(
      snackBar: SnackBar(content: Text( data: 'Failed to update product: $error')),
    );
  });
}
```

</td></tr>
</table>

The EditProductPage class allows users to edit the details of an existing product in a Flutter app, a sample interface shows in Figure 5.3.24. When the page is loaded, the current product details are pre-populated in text fields using data from the Firestore DocumentSnapshot. The user can modify the product's name, description, actual price, promotion price, and best-before date. The user can select a new date for the best-before date using a date picker, and the selected date is formatted and displayed in the text field. Upon pressing the “Save Changes” button, the updated information is saved back to Firestore. If the operation is successful, the user is taken back to the previous page with a success message. If an error occurs during the update, an error message is displayed using a ‘SnackBar’.

Add_Products.dart

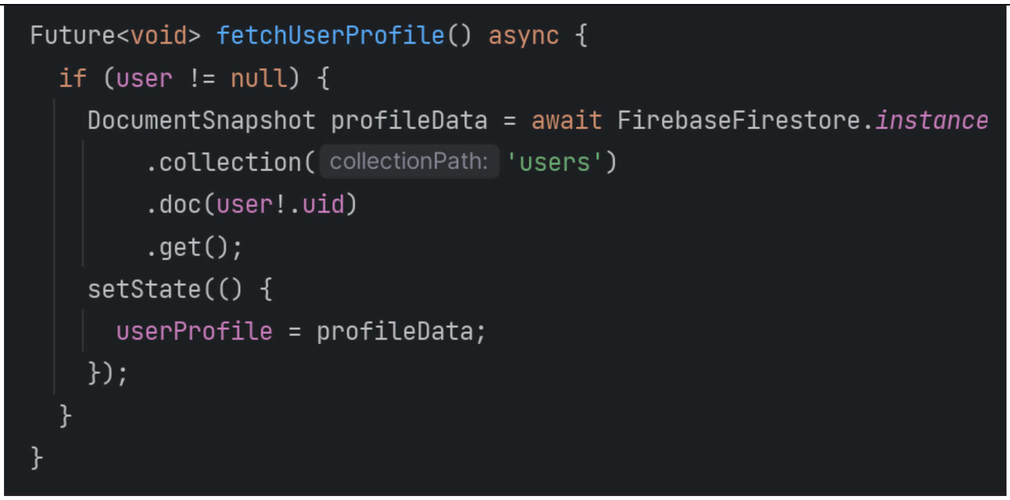
Lines	Code
93-177	

```dart
Future<void> _addProduct() async {
  try {
    // Check if any required field is empty
    if (productNameController.text.isEmpty ||
        descriptionController.text.isEmpty ||
        actualPriceController.text.isEmpty ||
        promotePriceController.text.isEmpty ||
        imageUrl.isEmpty ||
        _selectedDate == null) {
      _showErrorDialog( message: "Please fill in all fields before adding a product.");
      return;
    }

    // Get current user details
    User? user = FirebaseAuth.instance.currentUser;

    // Retrieve user's organization data from Firestore
    DocumentSnapshot userDoc = await FirebaseFirestore.instance
        .collection( collectionPath: 'users')
        .doc(user?.uid)
        .get();

    Map<String, dynamic> userData = userDoc.data() as Map<String, dynamic>;

    // Prepare data to send
    Map<String, dynamic> dataToSend = {
      'ImageUrl': imageUrl,
      'Product Name': productNameController.text,
      'Description': descriptionController.text,
      'Actual Price': actualPriceController.text,
      'Promote Price': promotePriceController.text,
      'Category': categories,
      'Best Before Date': _selectedDate != null
          ? DateFormat('yyyy-MM-dd').format( date: _selectedDate!)
          : '',
      // Include organization data
      'Organization': userData['organization'],
      'Organization Icon': userData['organizationIcon'],
    };

    // Add a new item and get the document ID
    DocumentReference docRef = await _productReference.add( data: dataToSend);

    // Update the selected date to 'Add_Product' collection
    await _uploadDateToProducts( docId: docRef.id);

    // Clear the input fields and reset the state
    setState(() {
      productNameController.clear();
      descriptionController.clear();
      actualPriceController.clear();
      promotePriceController.clear();
      _selectedDate = null;
      imageUrl = '';
      galleryImagePath = null; // Clear the gallery image path
      cameraImagePath = null; // Clear the camera image path
    });

    Navigator.pushReplacement(
      context,
       newRoute: MaterialPageRoute(builder: ( BuildContext  context) => Product_List_Manager()),
    );
  } catch (error) {
    print( object: 'Error adding product: $error');
  }
}
```

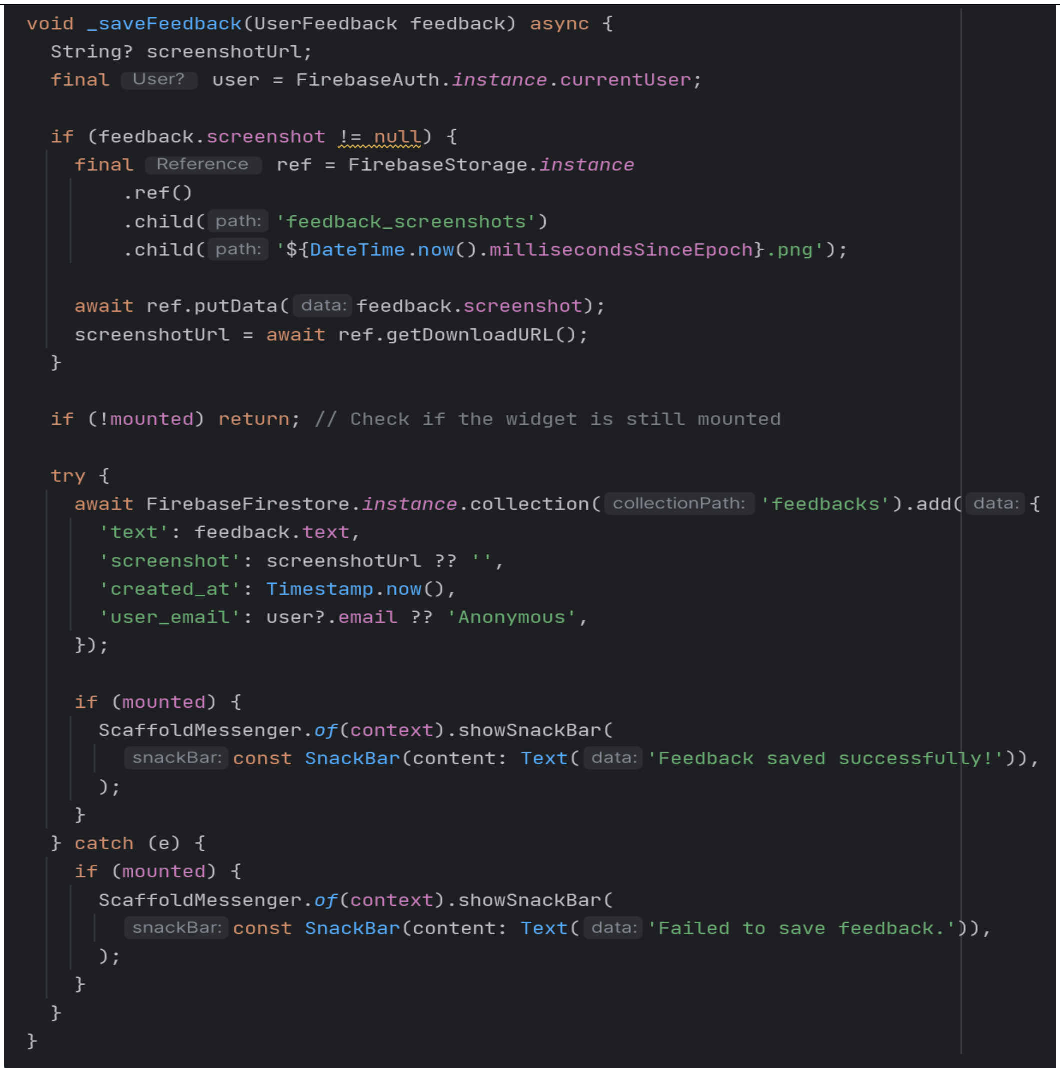
The AddProducts class provides an interface for users to add new products to a Firestore database, a sample interface shows in Figure 5.3.25. The page contains a form that includes fields for product name, description, actual price, promotion price, category selection, best-before date, and the option to upload images from the camera or gallery. The user can select an image for the product by either taking a picture using the camera or selecting one from the gallery. The selected image is then uploaded to Firebase Storage, and its URL is saved for later use in Firestore.

Additionally, the user must select a best-before date using a date picker, and the selected date is formatted and stored along with other product details.Once the user fills in all the fields and submits the form by pressing the "Add Product" button, the product details, along with the uploaded image and selected date, are saved in Firestore under the `Add_Product` collection. If any fields are missing, an error dialog is displayed prompting the user to fill in the required information. After successfully adding the product, the app navigates back to the product list page.

Manager_Profile.dart

Lines	Code
41- 51	

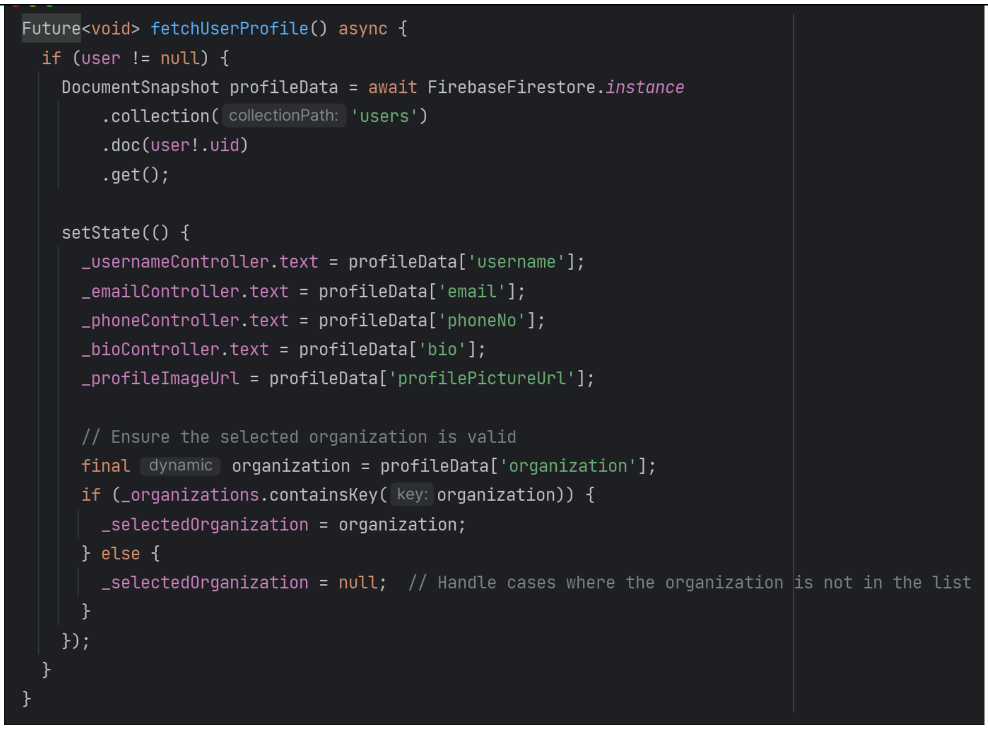
```dart
Future<void> fetchUserProfile() async {
  if (user != null) {
    DocumentSnapshot profileData = await FirebaseFirestore.instance
        .collection( collectionPath: 'users')
        .doc(user!.uid)
        .get();
    setState(() {
      userProfile = profileData;
    });
  }
}
```

53- 89	

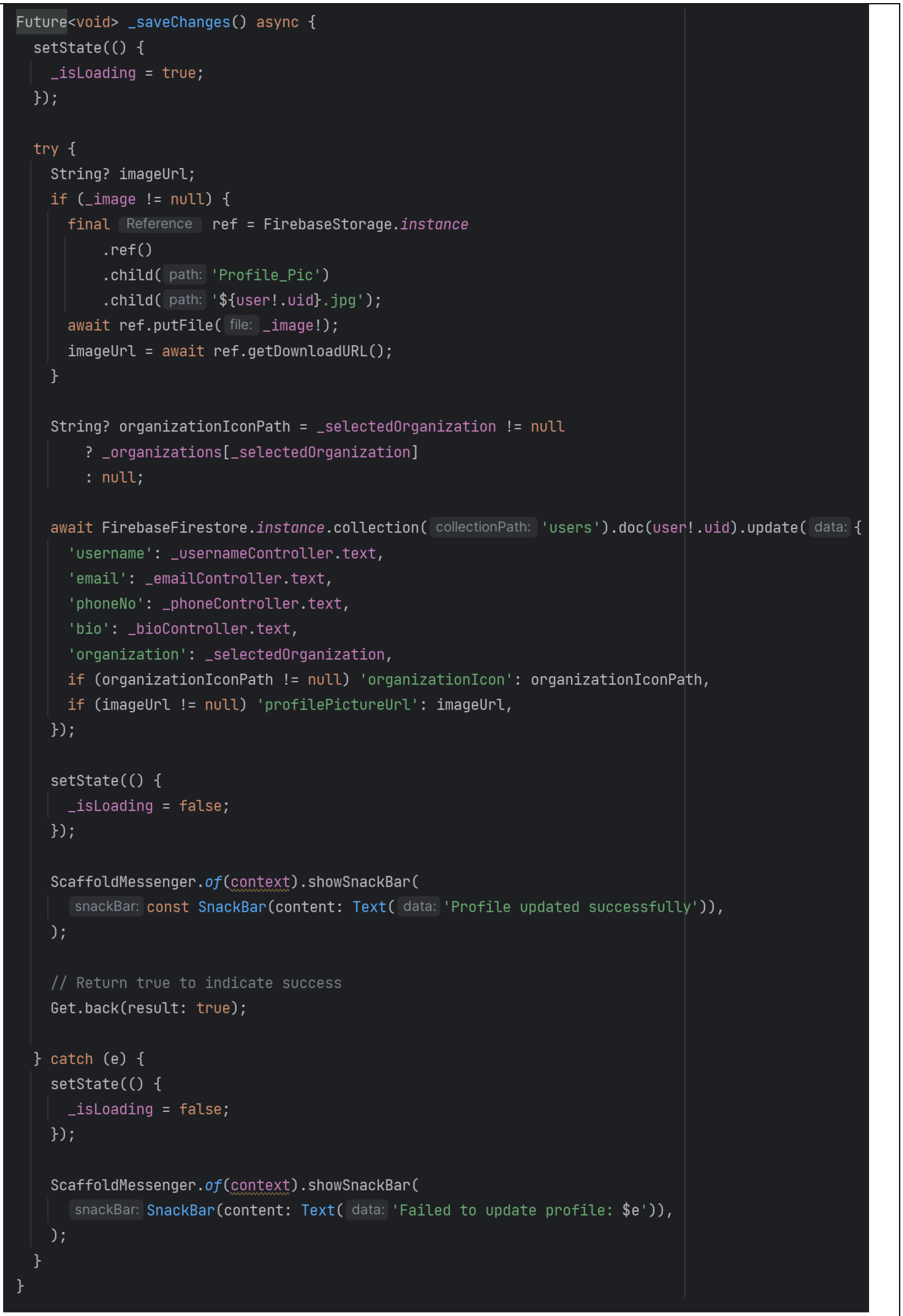
```
void _saveFeedback(UserFeedback feedback) async {
  String? screenshotUrl;
  final User? user = FirebaseAuth.instance.currentUser;

  if (feedback.screenshot != null) {
    final Reference ref = FirebaseStorage.instance
        .ref()
        .child( path: 'feedback_screenshots')
        .child( path: '${DateTime.now().millisecondsSinceEpoch}.png');

    await ref.putData( data: feedback.screenshot);
    screenshotUrl = await ref.getDownloadURL();
  }

  if (!mounted) return; // Check if the widget is still mounted

  try {
    await FirebaseFirestore.instance.collection( collectionPath: 'feedbacks').add( data: {
      'text': feedback.text,
      'screenshot': screenshotUrl ?? '',
      'created_at': Timestamp.now(),
      'user_email': user?.email ?? 'Anonymous',
    });

    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
         snackBar: const SnackBar(content: Text( data: 'Feedback saved successfully!')),
      );
    }
  } catch (e) {
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
         snackBar: const SnackBar(content: Text( data: 'Failed to save feedback.')),
      );
    }
  }
}
```

The ManagerProfile class displays the manager's profile information, including their profile picture, username, email, bio, and associated organization, there are some sample interfaces shows in Figure 5.3.26 and Figure 5.3.27. The app retrieves the manager's details from Firebase Firestore, and based on their organization, displays the corresponding logo. The app bar includes a feedback button that allows the user to submit feedback with an optional screenshot. This feedback is stored in Firestore. The **fetchUserProfile** function retrieves the manager's profile data from Firestore and updates the state with the profile information. The profile image and organization logo are displayed using network images and default placeholders in case of errors. The manager can also view and edit their bio, and an "Edit Profile" button allows them to navigate to an editing page for updating their profile. Additionally, the **_saveFeedback** function handles the feedback submission process, uploading the manager’s comments and screenshot to Firestore and Firebase Storage. If the feedback is submitted successfully, a confirmation message is displayed. The profile layout uses various widgets to organize and present the user's information clearly and consistently.

Edit_Manager_Profile.dart

Lines	Code
	

```
Future<void> fetchUserProfile() async {
  if (user != null) {
    DocumentSnapshot profileData = await FirebaseFirestore.instance
        .collection( collectionPath: 'users')
        .doc(user!.uid)
        .get();

    setState(() {
      _usernameController.text = profileData['username'];
      _emailController.text = profileData['email'];
      _phoneController.text = profileData['phoneNo'];
      _bioController.text = profileData['bio'];
      _profileImageUrl = profileData['profilePictureUrl'];

      // Ensure the selected organization is valid
      final  dynamic  organization = profileData['organization'];
      if (_organizations.containsKey( key: organization)) {
        _selectedOrganization = organization;
      } else {
        _selectedOrganization = null;  // Handle cases where the organization is not in the list
      }
    });
  }
}
```



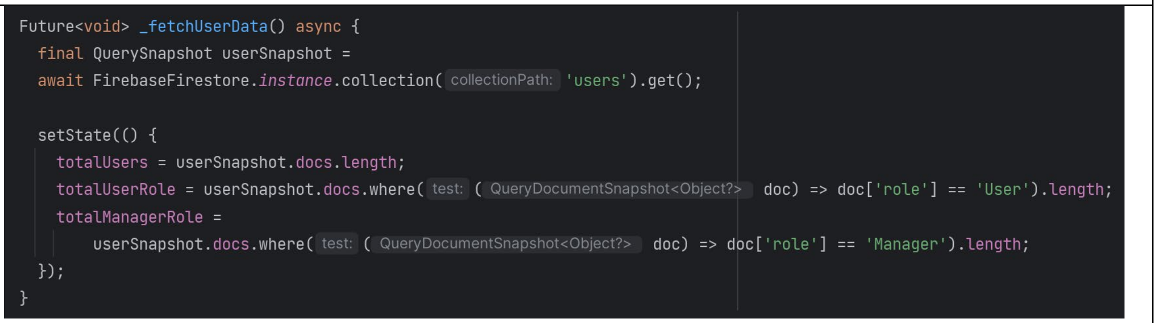
```
Future<void> _saveChanges() async {
  setState(() {
    _isLoading = true;
  });

  try {
    String? imageUrl;
    if (_image != null) {
      final Reference ref = FirebaseStorage.instance
          .ref()
          .child( path: 'Profile_Pic')
          .child( path: '${user!.uid}.jpg');
      await ref.putFile( file: _image!);
      imageUrl = await ref.getDownloadURL();
    }

    String? organizationIconPath = _selectedOrganization != null
        ? _organizations[_selectedOrganization]
        : null;

    await FirebaseFirestore.instance.collection( collectionPath: 'users').doc(user!.uid).update( data: {
      'username': _usernameController.text,
      'email': _emailController.text,
      'phoneNo': _phoneController.text,
      'bio': _bioController.text,
      'organization': _selectedOrganization,
      if (organizationIconPath != null) 'organizationIcon': organizationIconPath,
      if (imageUrl != null) 'profilePictureUrl': imageUrl,
    });

    setState(() {
      _isLoading = false;
    });

    ScaffoldMessenger.of(context).showSnackBar(
       snackBar: const SnackBar(content: Text( data: 'Profile updated successfully')),
    );

    // Return true to indicate success
    Get.back(result: true);

  } catch (e) {
    setState(() {
      _isLoading = false;
    });

    ScaffoldMessenger.of(context).showSnackBar(
       snackBar: SnackBar(content: Text( data: 'Failed to update profile: $e')),
    );
  }
}
```

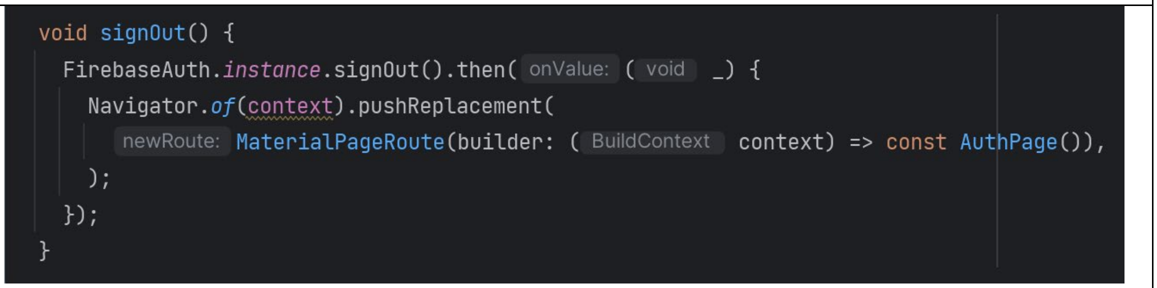
The EditManagerProfile class allows a manager to edit their profile information, including their username, email, phone number, bio, profile picture, and organization, a sample interface shows in Figure 5.3.27. It retrieves the manager's existing profile details from Firebase Firestore when the widget is initialized and displays them in respective text fields. The manager can modify their profile details, select a new profile picture, and choose their organization from a dropdown list. When the "Save Changes" button is clicked, the app uploads the new profile picture to Firebase Storage, retrieves the image URL, and updates the manager's details in Firestore, including their organization and its corresponding logo. If the profile update is successful, a success message is shown, and the manager is navigated back to the previous screen. If there is an error, an error message is displayed.

5.3.3 Admin Pages Code

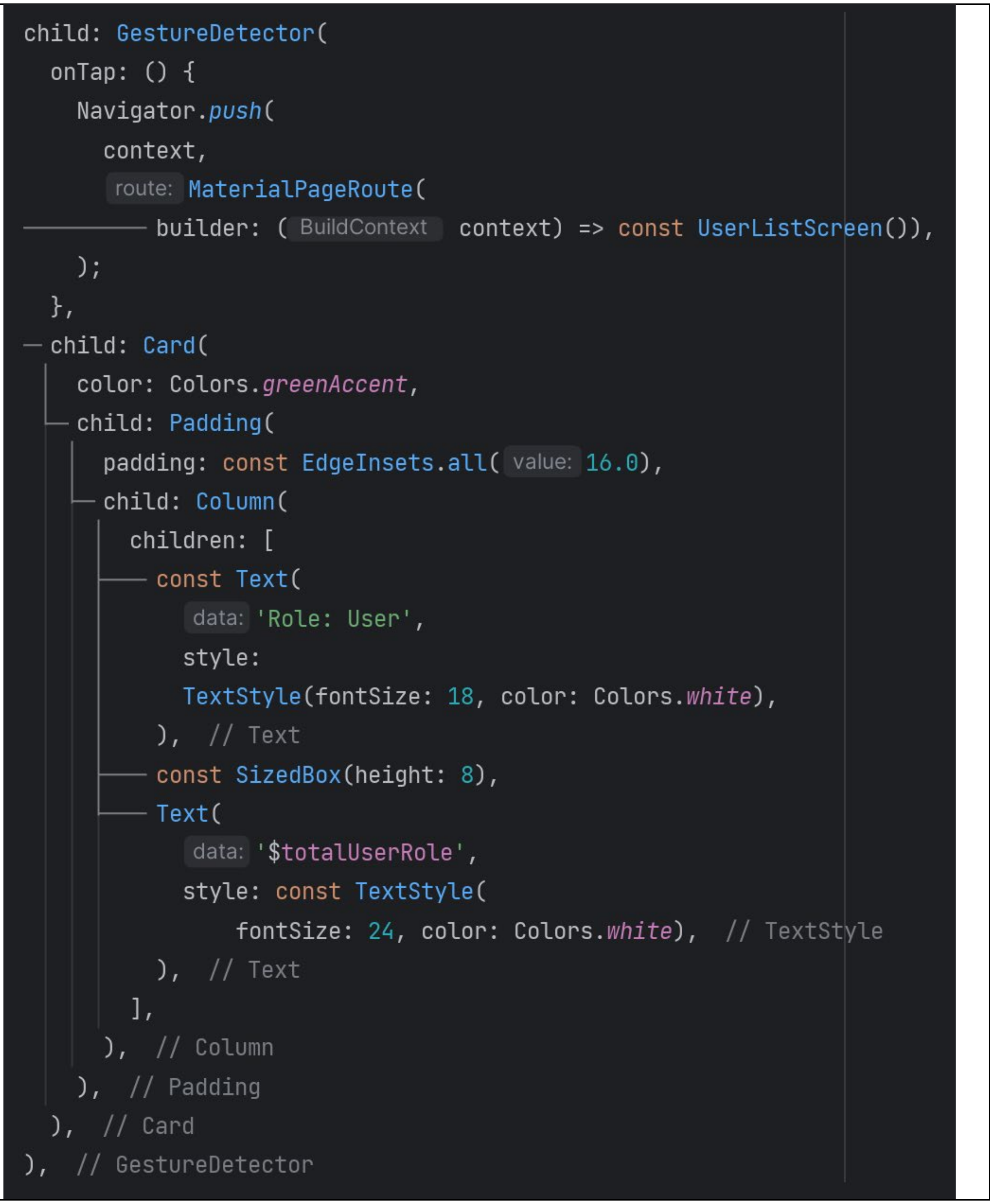
Admin_Dashboard.dart

Lines	Code

27-37	
	Future<void> _fetchUserData() async {
	final QuerySnapshot userSnapshot =
	await FirebaseFirestore.instance.collection(collectionPath: 'users').get();
	
	setState(() {
	totalUsers = userSnapshot.docs.length;
	totalUserRole = userSnapshot.docs.where(test: (QueryDocumentSnapshot<Object?> doc) => doc['role'] == 'User').length;
	totalManagerRole =
	userSnapshot.docs.where(test: (QueryDocumentSnapshot<Object?> doc) => doc['role'] == 'Manager').length;
	});
	}

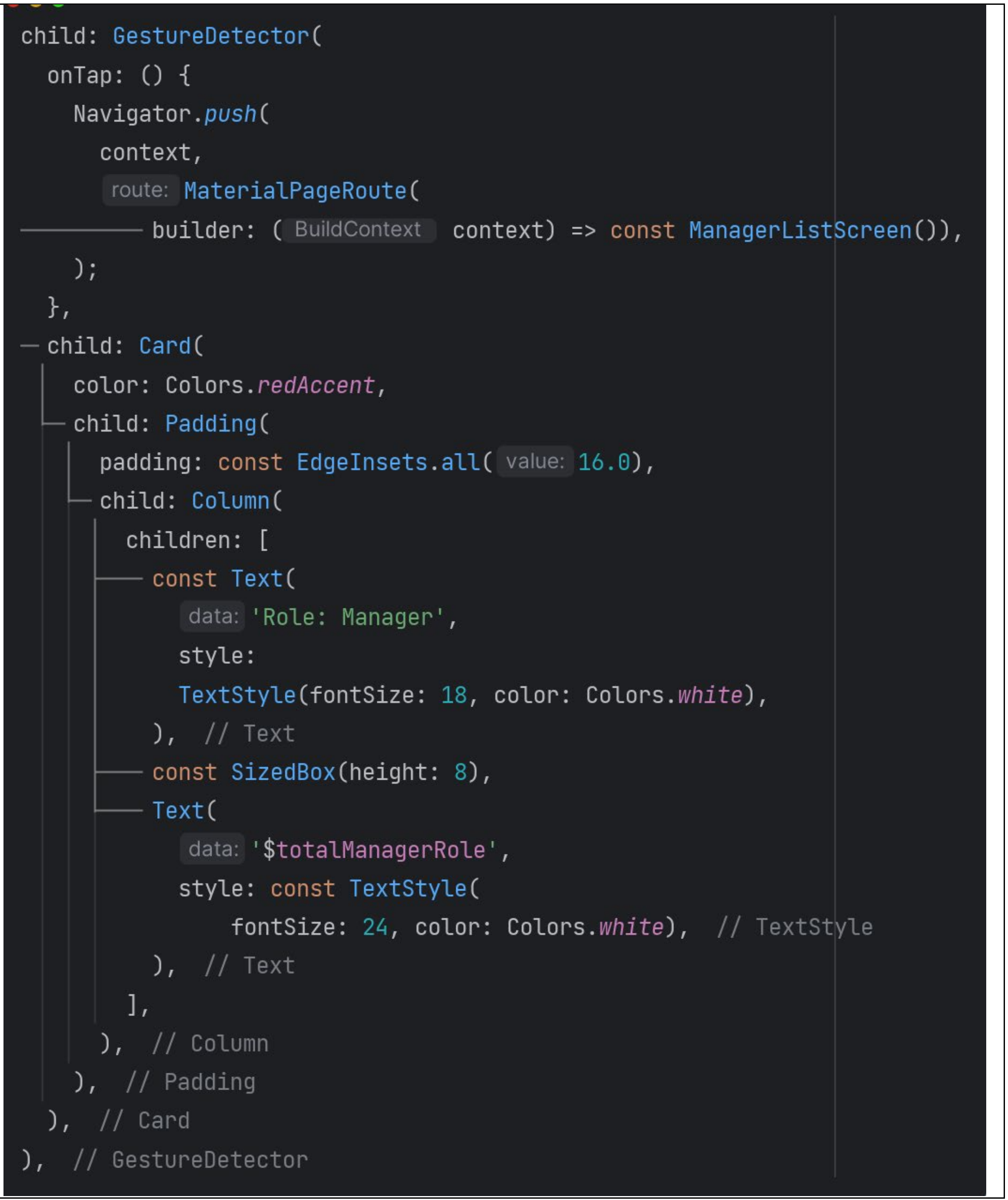
39-45	
	void signOut() {
	FirebaseAuth.instance.signOut().then(onValue: (void _) {
	Navigator.of(context).pushReplacement(
	newRoute: MaterialPageRoute(builder: (BuildContext context) => const AuthPage()),
	);
	});
	}

92-121

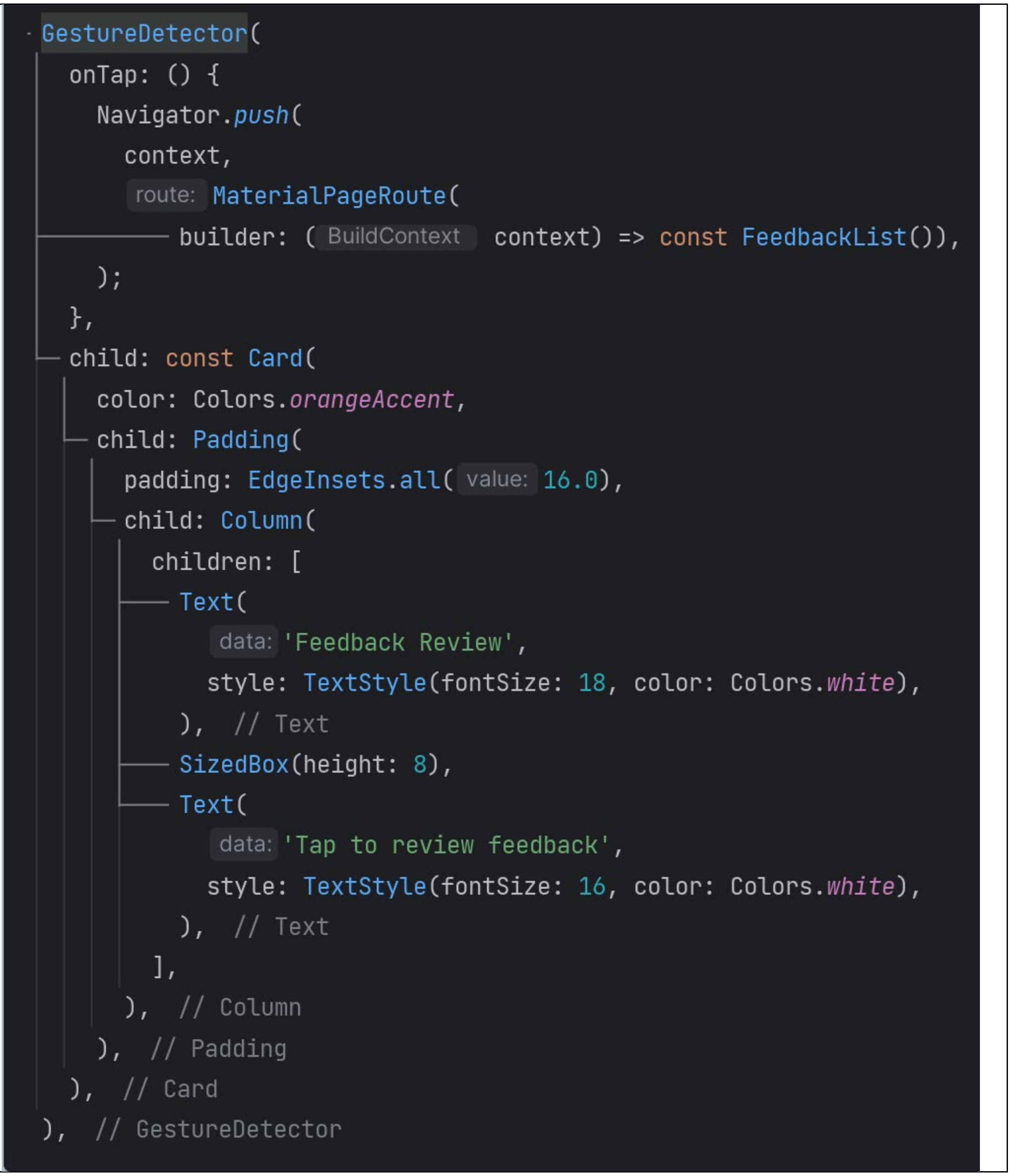


```
child: GestureDetector(
  onTap: () {
    Navigator.push(
      context,
      route: MaterialPageRoute(
        builder: ( BuildContext  context) => const UserListScreen()),
    );
  },
  child: Card(
    color: Colors.greenAccent,
    child: Padding(
      padding: const EdgeInsets.all( value: 16.0),
      child: Column(
        children: [
          const Text(
            data: 'Role: User',
            style:
            TextStyle(fontSize: 18, color: Colors.white),
          ),  // Text
          const SizedBox(height: 8),
          Text(
            data: '$totalUserRole',
            style: const TextStyle(
                fontSize: 24, color: Colors.white),  // TextStyle
          ),  // Text
        ],
      ),  // Column
    ),  // Padding
  ),  // Card
),  // GestureDetector
```

125-154



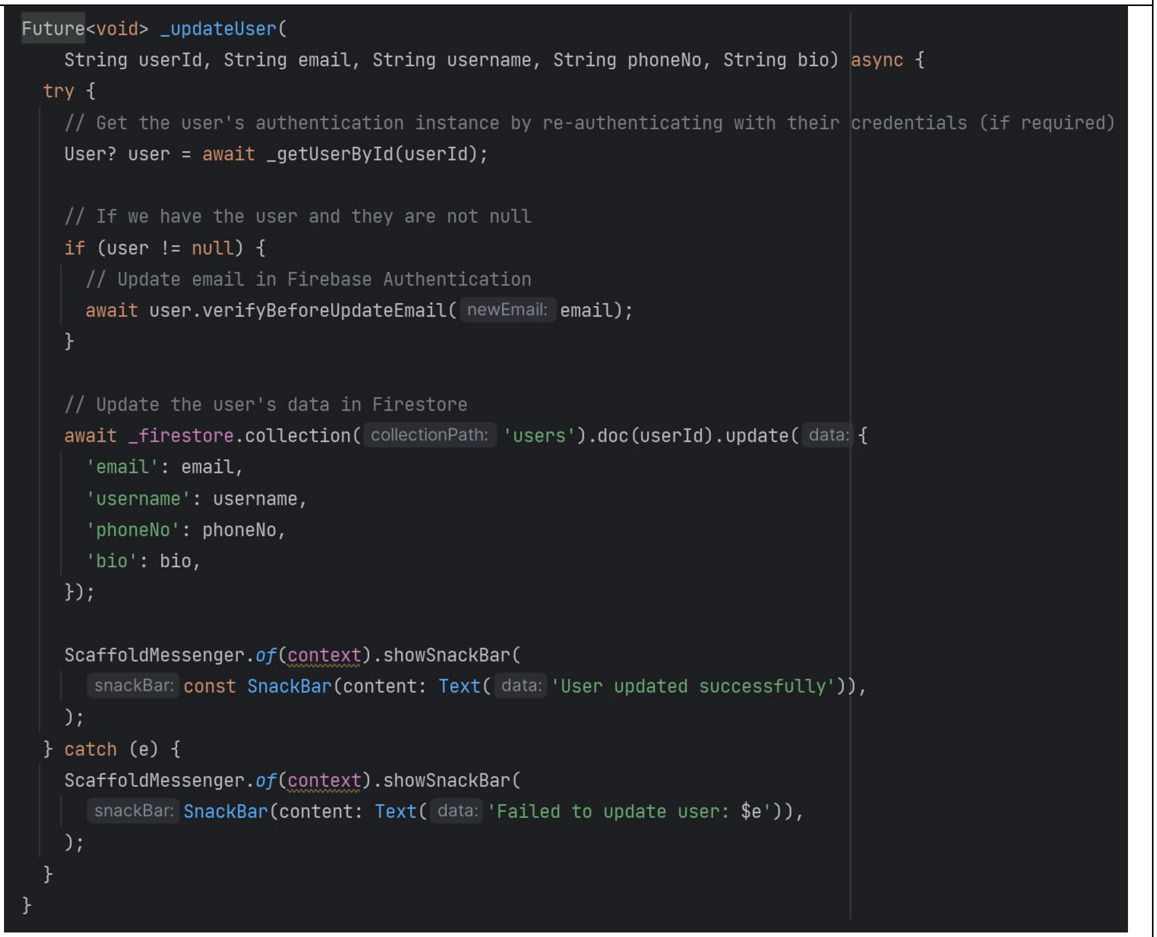
```
child: GestureDetector(
  onTap: () {
    Navigator.push(
      context,
      route: MaterialPageRoute(
          builder: ( BuildContext  context) => const ManagerListScreen()),
    );
  },
  child: Card(
    color: Colors.redAccent,
    child: Padding(
      padding: const EdgeInsets.all( value: 16.0),
      child: Column(
        children: [
          const Text(
            data: 'Role: Manager',
            style:
            TextStyle(fontSize: 18, color: Colors.white),
          ),  // Text
          const SizedBox(height: 8),
          Text(
            data: '$totalManagerRole',
            style: const TextStyle(
                fontSize: 24, color: Colors.white),  // TextStyle
          ),  // Text
        ],
      ),  // Column
    ),  // Padding
  ),  // Card
),  // GestureDetector
```

<table>
<tr>
<td>160-
187</td>
<td>

<pre>
GestureDetector(
  onTap: () {
    Navigator.push(
      context,
      route: MaterialPageRoute(
        builder: (BuildContext context) => const FeedbackList()),
    );
  },
  child: const Card(
    color: Colors.orangeAccent,
    child: Padding(
      padding: EdgeInsets.all(value: 16.0),
      child: Column(
        children: [
          Text(
            data: 'Feedback Review',
            style: TextStyle(fontSize: 18, color: Colors.white),
          ), // Text
          SizedBox(height: 8),
          Text(
            data: 'Tap to review feedback',
            style: TextStyle(fontSize: 16, color: Colors.white),
          ), // Text
        ],
      ), // Column
    ), // Padding
  ), // Card
), // GestureDetector
</pre>
</td>
</tr>
</table>

The AdminDashboardScreen class is a part of the admin interface designed to provide an overview of user statistics and manage the user base, a sample interface shows in Figure 5.3.28. Upon loading, the dashboard fetches data from Firestore to display the total number of users, as well as the breakdown of users by roles, specifically “User” and “Manager.” This is achieved by querying the ‘users’ collection and counting the documents based on the assigned role. The data is then displayed in distinct, visually appealing cards for easy navigation.

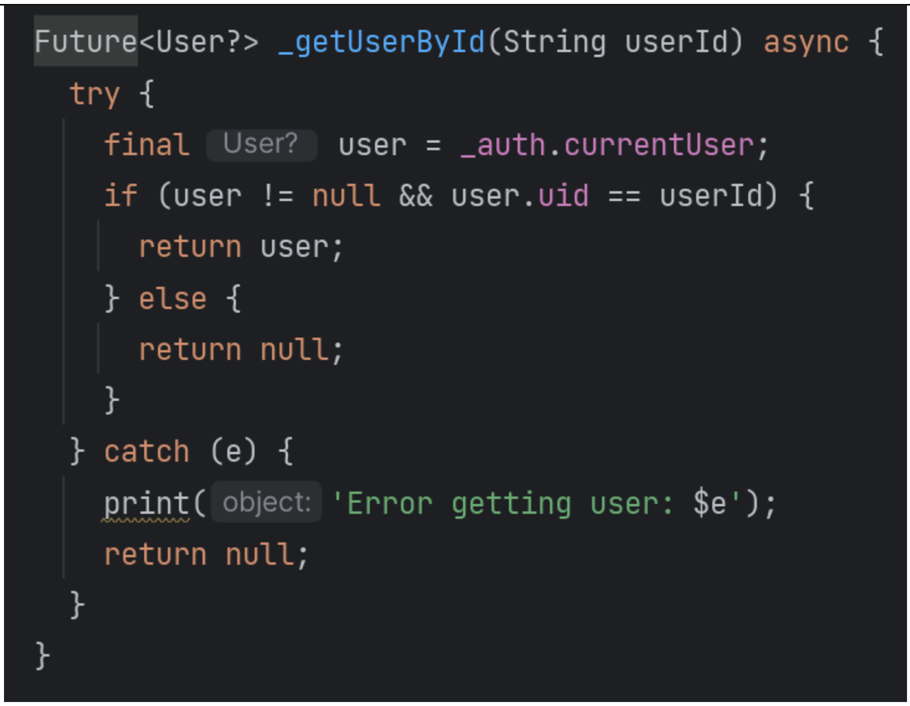
The dashboard layout consists of three primary sections. The first section displays the "Total Users" in the system, showing the overall number of registered users. The second section consists of two cards representing "Role-Based User Counts"—one for users with the "User" role and another for those with the "Manager" role. These cards not only display the counts but are also clickable, allowing the admin to navigate to the respective lists of users or managers for further management tasks like editing or viewing details. The third section is a card for "Feedback Review", which, when clicked, navigates the admin to a page where they can review feedback provided by users. Additionally, the dashboard includes a logout button in the app bar, allowing the admin to sign out of their session and return to the authentication page. This dashboard serves as a control panel for managing users and reviewing feedback, ensuring that the admin has quick access to essential statistics and user management tools.

Edit_User.dart

Lines	Code
18- 46	

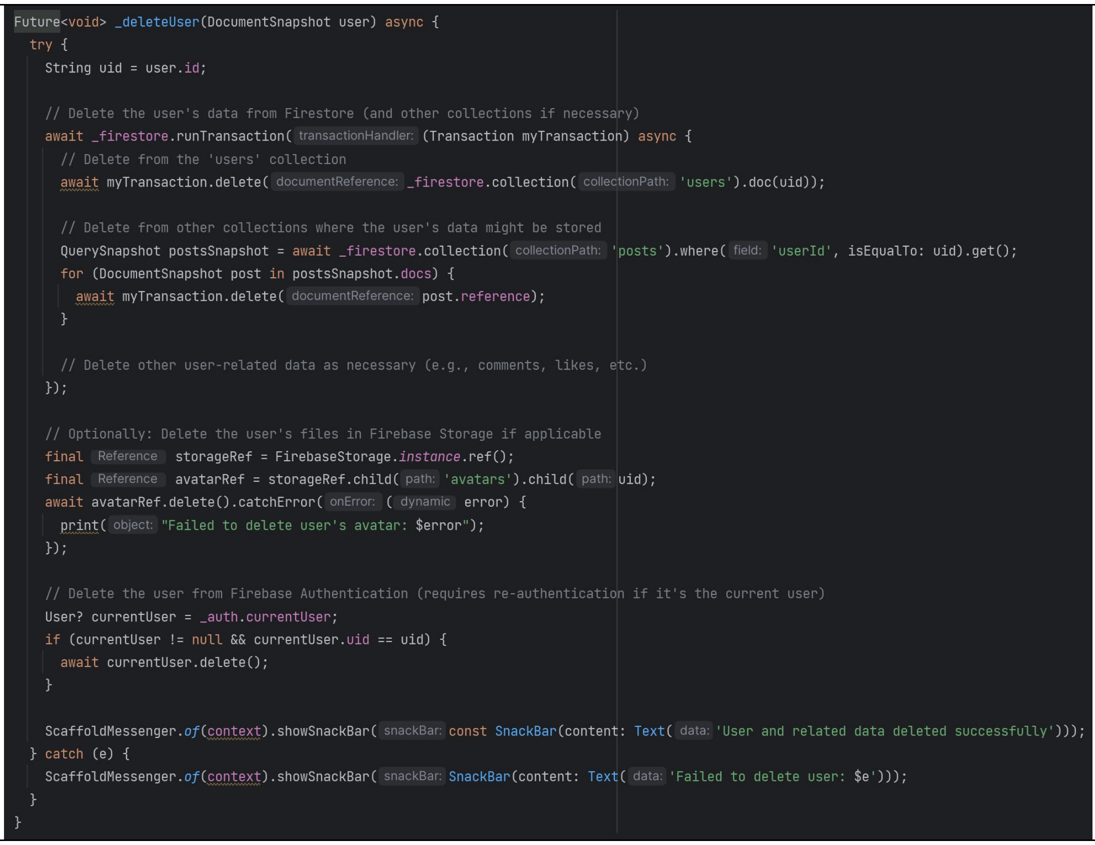
```dart
Future<void> _updateUser(
    String userId, String email, String username, String phoneNo, String bio) async {
  try {
    // Get the user's authentication instance by re-authenticating with their credentials (if required)
    User? user = await _getUserById(userId);

    // If we have the user and they are not null
    if (user != null) {
      // Update email in Firebase Authentication
      await user.verifyBeforeUpdateEmail( newEmail: email);
    }

    // Update the user's data in Firestore
    await _firestore.collection( collectionPath: 'users').doc(userId).update( data: {
      'email': email,
      'username': username,
      'phoneNo': phoneNo,
      'bio': bio,
    });

    ScaffoldMessenger.of(context).showSnackBar(
       snackBar: const SnackBar(content: Text( data: 'User updated successfully')),
    );
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
       snackBar: SnackBar(content: Text( data: 'Failed to update user: $e')),
    );
  }
}
```

49-61

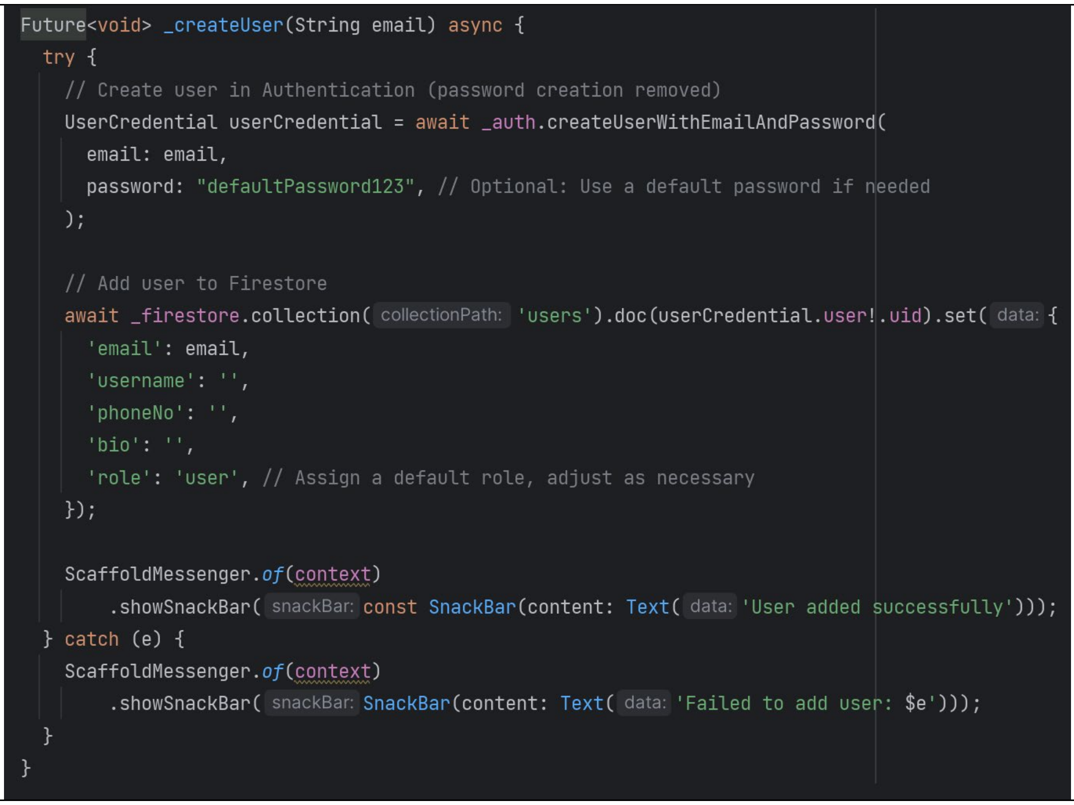


```
Future<User?> _getUserById(String userId) async {
  try {
    final User? user = _auth.currentUser;
    if (user != null && user.uid == userId) {
      return user;
    } else {
      return null;
    }
  } catch (e) {
    print( object: 'Error getting user: $e');
    return null;
  }
}
```

64-99

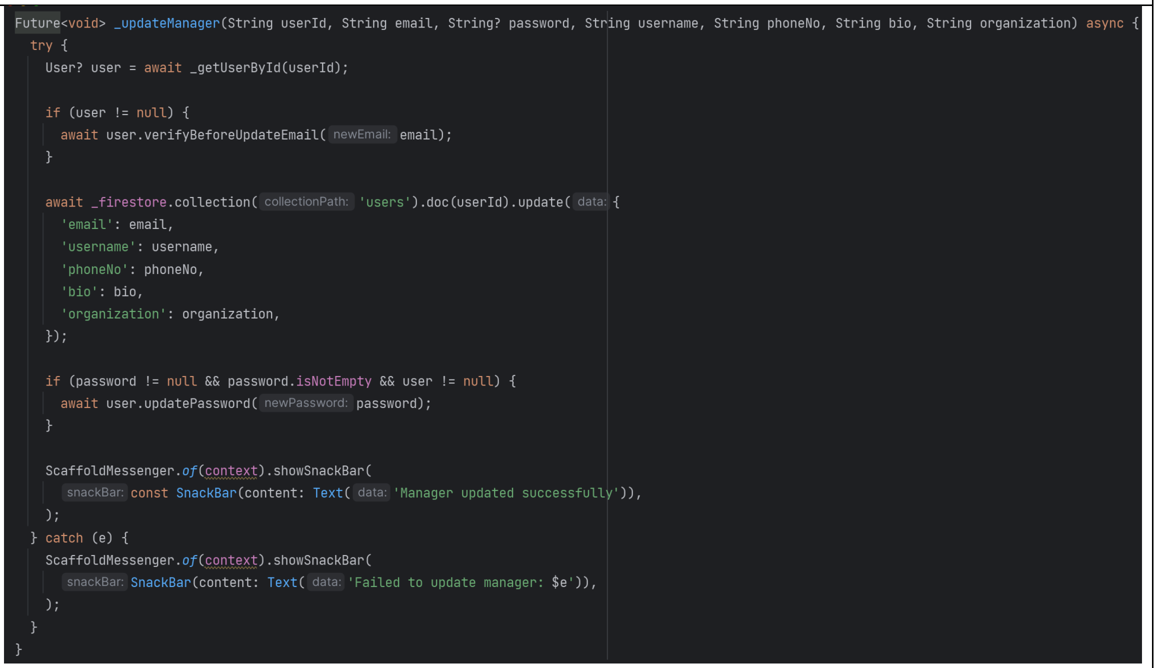


```
Future<void> _deleteUser(DocumentSnapshot user) async {
  try {
    String uid = user.id;

    // Delete the user's data from Firestore (and other collections if necessary)
    await _firestore.runTransaction( transactionHandler: (Transaction myTransaction) async {
      // Delete from the 'users' collection
      await myTransaction.delete( documentReference: _firestore.collection( collectionPath: 'users').doc(uid));

      // Delete from other collections where the user's data might be stored
      QuerySnapshot postsSnapshot = await _firestore.collection( collectionPath: 'posts').where( field: 'userId', isEqualTo: uid).get();
      for (DocumentSnapshot post in postsSnapshot.docs) {
        await myTransaction.delete( documentReference: post.reference);
      }

      // Delete other user-related data as necessary (e.g., comments, likes, etc.)
    });

    // Optionally: Delete the user's files in Firebase Storage if applicable
    final Reference  storageRef = FirebaseStorage.instance.ref();
    final Reference  avatarRef = storageRef.child( path: 'avatars').child( path: uid);
    await avatarRef.delete().catchError( onError: ( dynamic  error) {
      print( object: "Failed to delete user's avatar: $error");
    });

    // Delete the user from Firebase Authentication (requires re-authentication if it's the current user)
    User? currentUser = _auth.currentUser;
    if (currentUser != null && currentUser.uid == uid) {
      await currentUser.delete();
    }

    ScaffoldMessenger.of(context).showSnackBar( snackBar: const SnackBar(content: Text( data: 'User and related data deleted successfully')));
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar( snackBar: SnackBar(content: Text( data: 'Failed to delete user: $e')));
  }
}
```

102- 125	

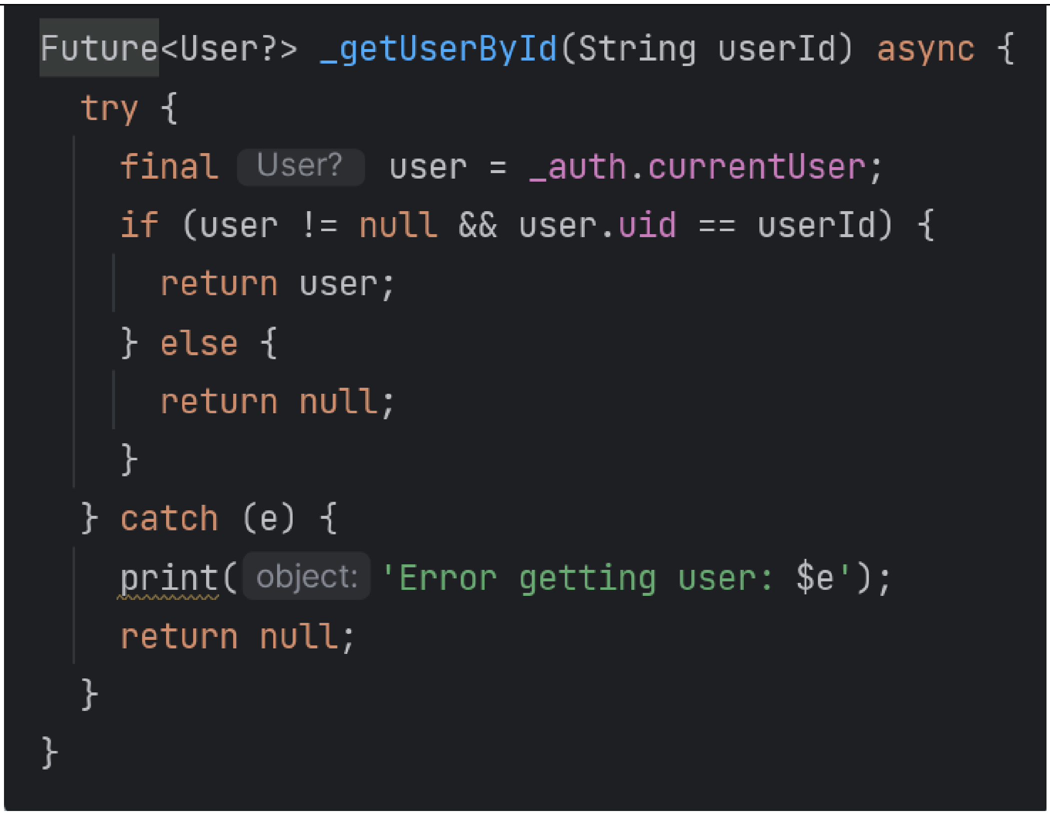
```
Future<void> _createUser(String email) async {
  try {
    // Create user in Authentication (password creation removed)
    UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
      email: email,
      password: "defaultPassword123", // Optional: Use a default password if needed
    );

    // Add user to Firestore
    await _firestore.collection( collectionPath: 'users').doc(userCredential.user!.uid).set( data: {
      'email': email,
      'username': '',
      'phoneNo': '',
      'bio': '',
      'role': 'user', // Assign a default role, adjust as necessary
    });

    ScaffoldMessenger.of(context)
        .showSnackBar( snackBar: const SnackBar(content: Text( data: 'User added successfully')));
  } catch (e) {
    ScaffoldMessenger.of(context)
        .showSnackBar( snackBar: SnackBar(content: Text( data: 'Failed to add user: $e')));
  }
}
```

Edit_Manager.dart

Line s	Code
17- 45	

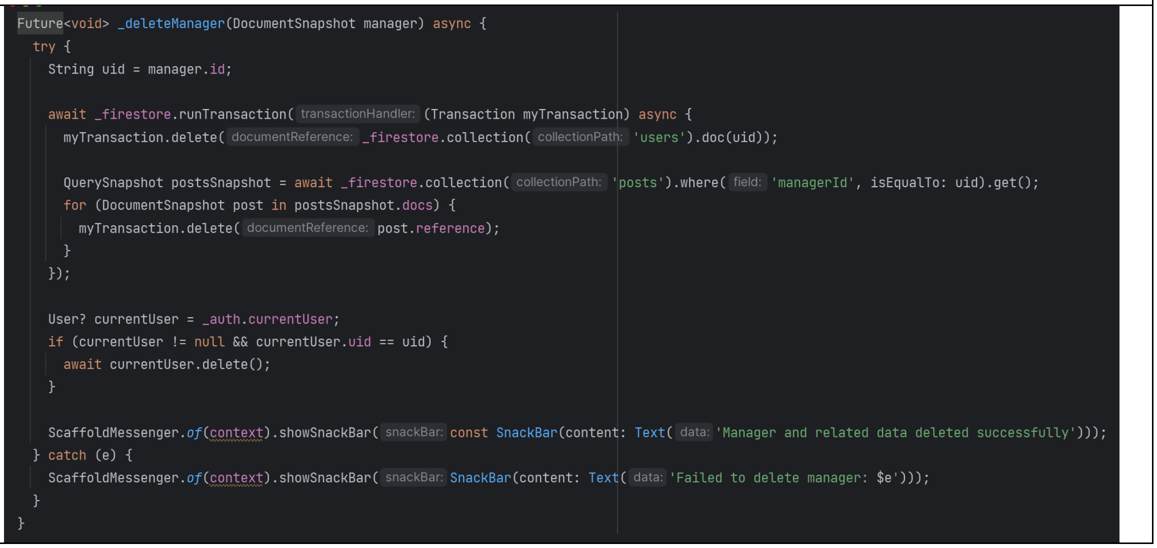
```
Future<void> _updateManager(String userId, String email, String? password, String username, String phoneNo, String bio, String organization) async {
  try {
    User? user = await _getUserById(userId);

    if (user != null) {
      await user.verifyBeforeUpdateEmail( newEmail: email);
    }

    await _firestore.collection( collectionPath: 'users').doc(userId).update( data: {
      'email': email,
      'username': username,
      'phoneNo': phoneNo,
      'bio': bio,
      'organization': organization,
    });

    if (password != null && password.isNotEmpty && user != null) {
      await user.updatePassword( newPassword: password);
    }

    ScaffoldMessenger.of(context).showSnackBar(
      snackBar: const SnackBar(content: Text( data: 'Manager updated successfully')),
    );
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      snackBar: SnackBar(content: Text( data: 'Failed to update manager: $e')),
    );
  }
}
```

<table>
<tr>
<td>47-
59</td>
<td>



```
Future<User?> _getUserById(String userId) async {
  try {
    final User? user = _auth.currentUser;
    if (user != null && user.uid == userId) {
      return user;
    } else {
      return null;
    }
  } catch (e) {
    print( object: 'Error getting user: $e');
    return null;
  }
}
```

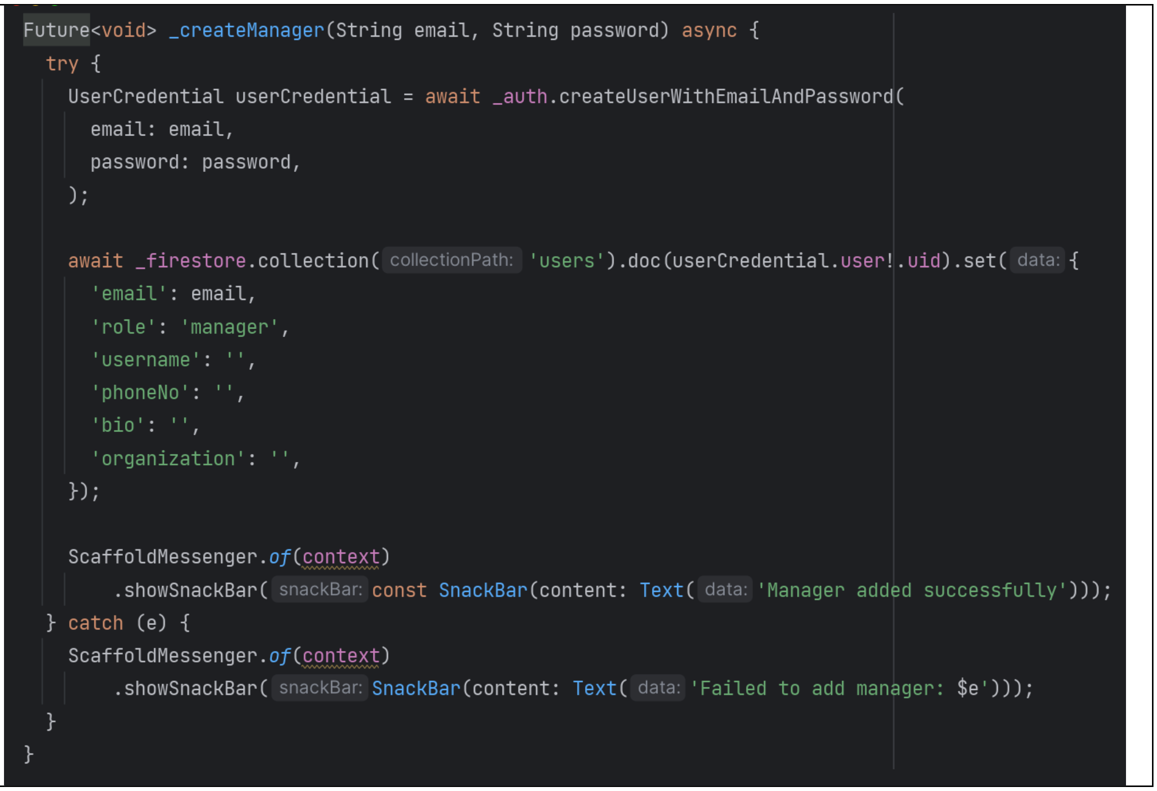
</td>
</tr>
<tr>
<td>61-
83</td>
<td>



```
Future<void> _deleteManager(DocumentSnapshot manager) async {
  try {
    String uid = manager.id;

    await _firestore.runTransaction( transactionHandler: (Transaction myTransaction) async {
      myTransaction.delete( documentReference: _firestore.collection( collectionPath: 'users').doc(uid));

      QuerySnapshot postsSnapshot = await _firestore.collection( collectionPath: 'posts').where( field: 'managerId', isEqualTo: uid).get();
      for (DocumentSnapshot post in postsSnapshot.docs) {
        myTransaction.delete( documentReference: post.reference);
      }
    });

    User? currentUser = _auth.currentUser;
    if (currentUser != null && currentUser.uid == uid) {
      await currentUser.delete();
    }

    ScaffoldMessenger.of(context).showSnackBar( snackBar: const SnackBar(content: Text( data: 'Manager and related data deleted successfully')));
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar( snackBar: SnackBar(content: Text( data: 'Failed to delete manager: $e')));
  }
}
```

</td>
</tr>
</table>

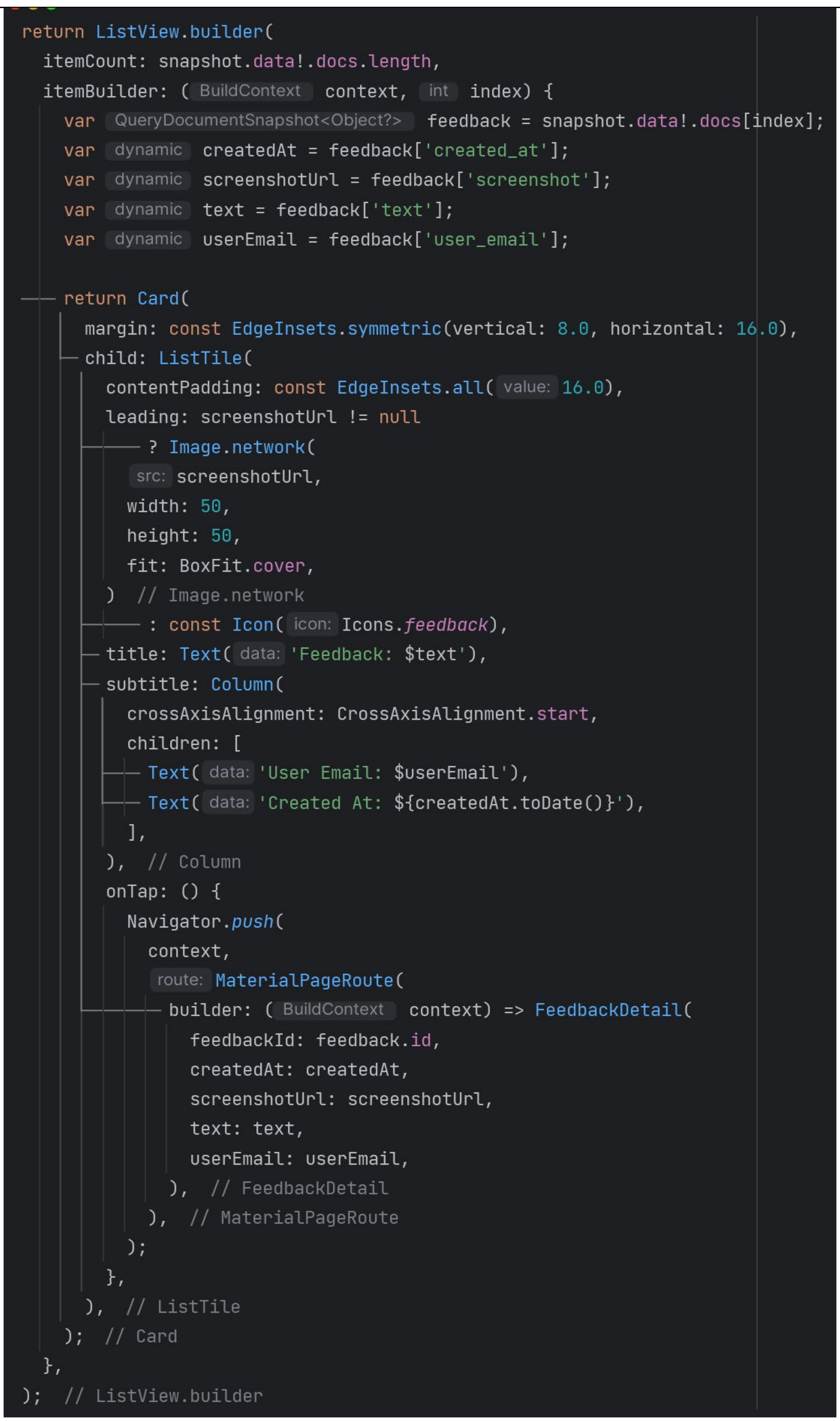
85-107



```
Future<void> _createManager(String email, String password) async {
  try {
    UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );

    await _firestore.collection( collectionPath: 'users').doc(userCredential.user!.uid).set( data: {
      'email': email,
      'role': 'manager',
      'username': '',
      'phoneNo': '',
      'bio': '',
      'organization': '',
    });

    ScaffoldMessenger.of(context)
        .showSnackBar( snackBar: const SnackBar(content: Text( data: 'Manager added successfully')));
  } catch (e) {
    ScaffoldMessenger.of(context)
        .showSnackBar( snackBar: SnackBar(content: Text( data: 'Failed to add manager: $e')));
  }
}
```

The UserListScreen and ManagerListScreen classes enable an admin to manage users and managers, respectively, there are some sample interfaces shows in Figure 5.3.29 and Figure 5.3.2=30. Both screens display lists of users or managers from Firestore and provide options to update, delete, or add new accounts. The **_updateUser** and **_updateManager** functions allow the admin to modify user details such as email, username, phone number, and bio. In the case of managers, password and organization can also be updated. These changes are made in both Firestore and Firebase Authentication. The **_deleteUser** and **_deleteManager** functions delete a user or manager's account and their related data from Firestore, and optionally their files from Firebase Storage. The account is also removed from Firebase Authentication. The **_createUser** and **_createManager** functions add new user or manager accounts to Firebase Authentication and Firestore. Both classes offer a simple interface with a 'ListView' displaying users or managers, along with options to edit, delete, or add accounts via dialogs.

View_Feedback.dart

Lines	Code
25-71	

```
return ListView.builder(
  itemCount: snapshot.data!.docs.length,
  itemBuilder: ( BuildContext  context,  int  index) {
    var  QueryDocumentSnapshot<Object?>  feedback = snapshot.data!.docs[index];
    var  dynamic  createdAt = feedback['created_at'];
    var  dynamic  screenshotUrl = feedback['screenshot'];
    var  dynamic  text = feedback['text'];
    var  dynamic  userEmail = feedback['user_email'];

    return Card(
      margin: const EdgeInsets.symmetric(vertical: 8.0, horizontal: 16.0),
      child: ListTile(
        contentPadding: const EdgeInsets.all( value: 16.0),
        leading: screenshotUrl != null
            ? Image.network(
          src: screenshotUrl,
          width: 50,
          height: 50,
          fit: BoxFit.cover,
        )  // Image.network
            : const Icon( icon: Icons.feedback),
        title: Text( data: 'Feedback: $text'),
        subtitle: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text( data: 'User Email: $userEmail'),
            Text( data: 'Created At: ${createdAt.toDate()}'),
          ],
        ),  // Column
        onTap: () {
          Navigator.push(
            context,
            route: MaterialPageRoute(
              builder: ( BuildContext  context) => FeedbackDetail(
                feedbackId: feedback.id,
                createdAt: createdAt,
                screenshotUrl: screenshotUrl,
                text: text,
                userEmail: userEmail,
              ),  // FeedbackDetail
            ),  // MaterialPageRoute
          );
        },
      ),  // ListTile
    );  // Card
  },
);  // ListView.builder
```

The FeedbackList class displays a list of feedback entries fetched from Firestore in real-time, a sample interface shows in Figure 5.3.31. Each feedback includes details such as text, user email, creation date, and an optional screenshot. When a feedback entry is tapped, the user is navigated to a detailed view for more information. In this detailed view, the feedback ID, user email, feedback text, and creation date are displayed, along with a clickable screenshot if available. The screenshot can be viewed in full screen using the FullScreenImage class, which employs the PhotoView widget to allow zooming and panning. This setup provides a streamlined way to review and interact with feedback, including visual elements, enhancing feedback review for administrators.

5.3.4 User Pages UI

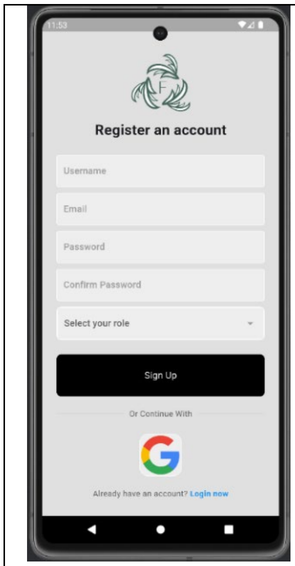


Figure 5.3.1 Register page

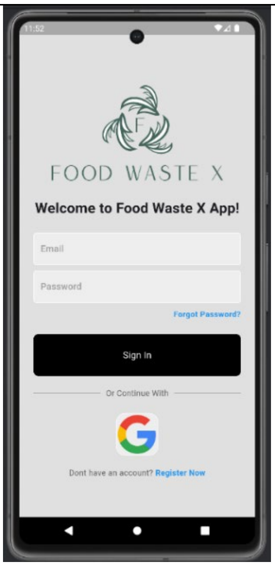


Figure 5.3.2 Login Page

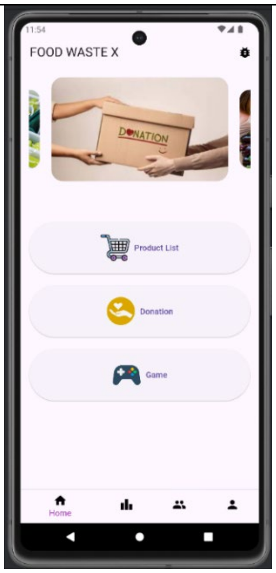


Figure 5.3.3 Home Page

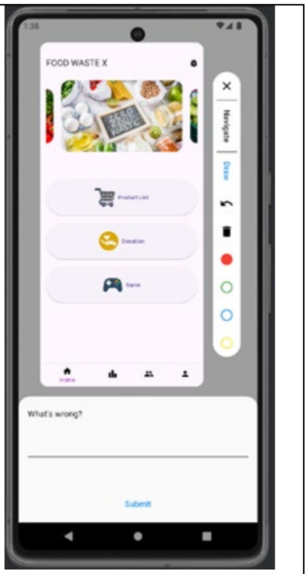


Figure 5.3.4 Feedback

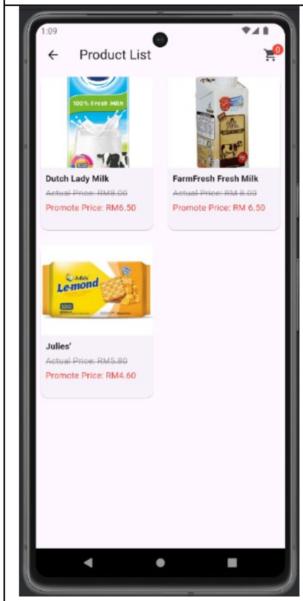


Figure 5.3.5 Product List Page

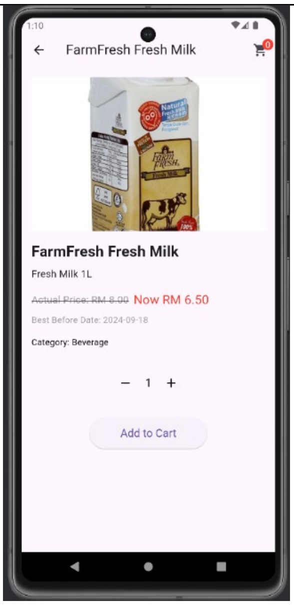


Figure 5.3.6 Product Details Page

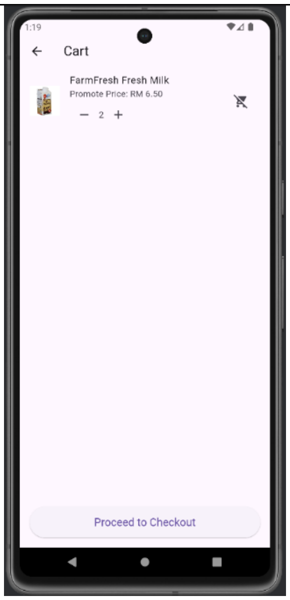


Figure 5.3.7 Cart Page

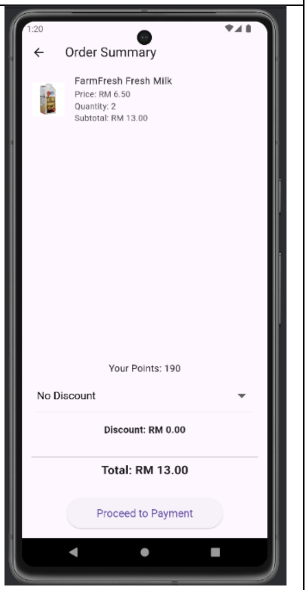
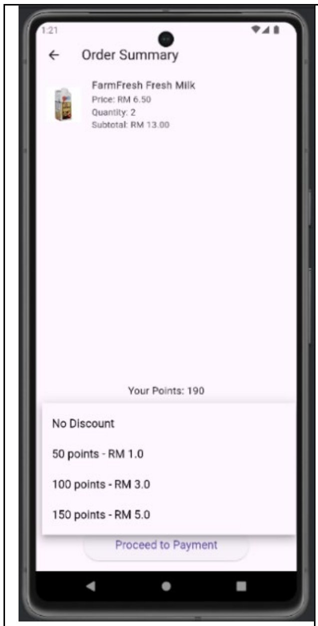


Figure 5.3.8 Order Summary Page



Order Summary

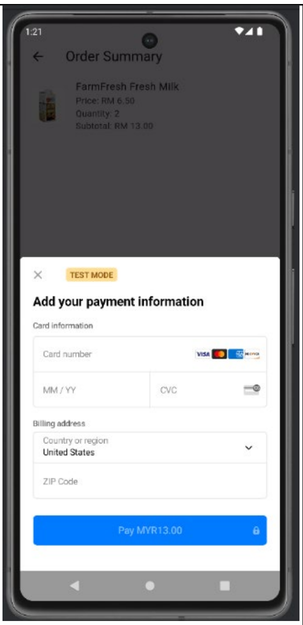
FarmFresh Fresh Milk
Price: RM 6.50
Quantity: 2
Subtotal: RM 13.00

Your Points: 190

No Discount
50 points - RM 1.0
100 points - RM 3.0
150 points - RM 5.0

Proceed to Payment

Figure 5.3.9 Discount Option



Order Summary

FarmFresh Fresh Milk
Price: RM 6.50
Quantity: 2
Subtotal: RM 13.00

TEST MODE

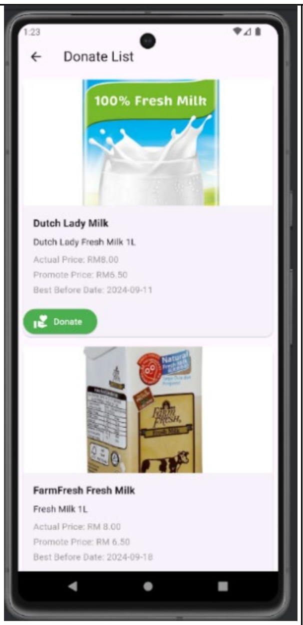
Add your payment information

Card information
Card number
MM / YY
CVC

Billing address
Country or region
United States
ZIP Code

Pay MYR13.00

Figure 5.3.10 Cart Payment



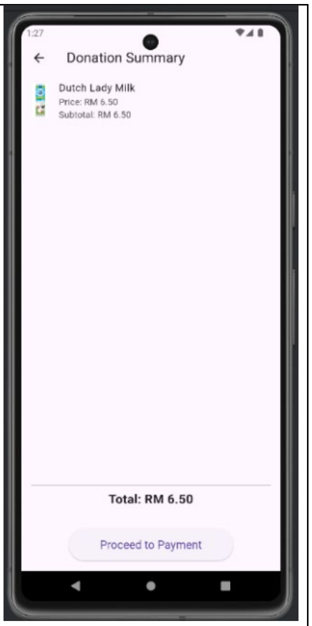
Donate List

Dutch Lady Milk
Dutch Lady Fresh Milk 1L
Actual Price: RM8.00
Promote Price: RM6.50
Best Before Date: 2024-09-11

Donate

FarmFresh Fresh Milk
Fresh Milk 1L
Actual Price: RM 8.00
Promote Price: RM 6.50
Best Before Date: 2024-09-18

Figure 5.3.11 Donation Page



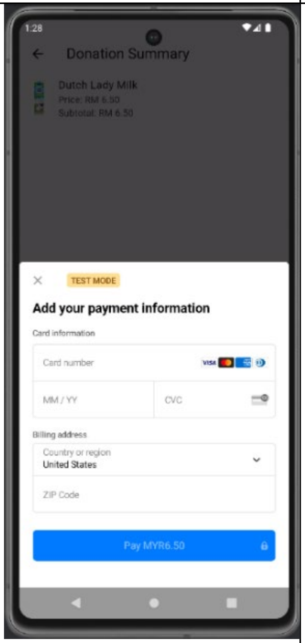
Donation Summary

Dutch Lady Milk
Price: RM 6.50
Subtotal: RM 6.50

Total: RM 6.50

Proceed to Payment

Figure 5.3.12 Donation Summary Page



Donation Summary

Dutch Lady Milk
Price: RM 6.50
Subtotal: RM 6.50

TEST MODE

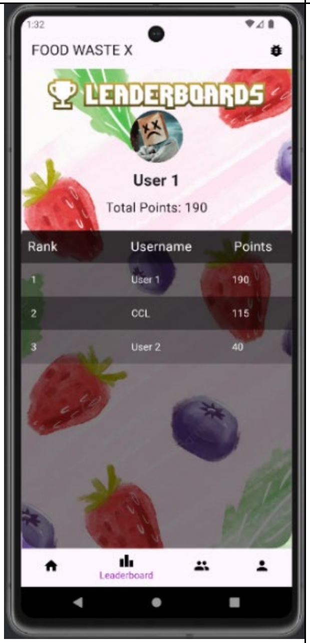
Add your payment information

Card information
Card number
MM / YY
CVC

Billing address
Country or region
United States
ZIP Code

Pay MYR6.50

Figure 5.3.13 Donation Payment

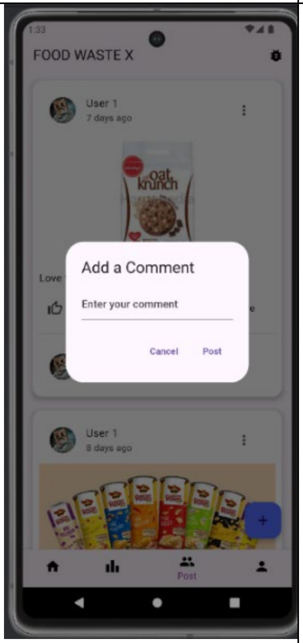


FOOD WASTE X

User 1
Total Points: 190

Rank	Username	Points
1	User 1	190
2	CCL	115
3	User 2	40

Figure 5.3.14 Leaderboard Page



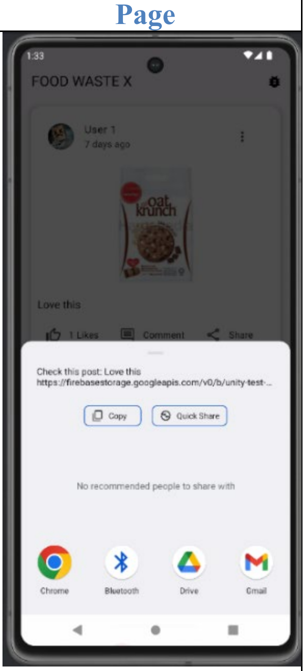
FOOD WASTE X

User 1
7 days ago

Add a Comment
Enter your comment
Cancel Post

User 1
8 days ago

Figure 5.3.15 Comment



FOOD WASTE X

User 1
7 days ago

Love this

1 Likes Comment Share

Check this post: Love this
https://firebasestorage.googleapis.com/v0/b/unity-test-...

Copy Quick Share

No recommended people to share with

Figure 5.3.16 Share Post

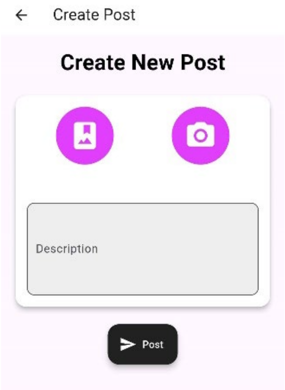
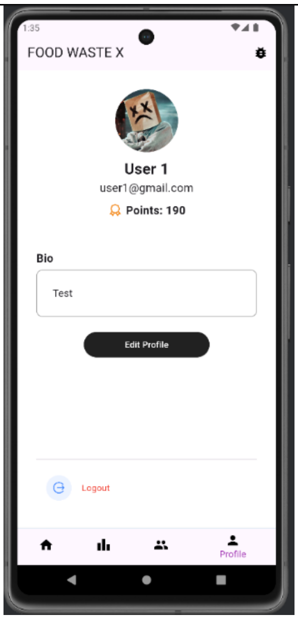


Figure 5.3.17 Create Post

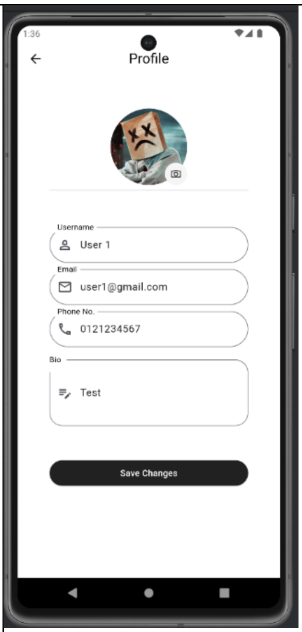


User 1
user1@gmail.com
Points: 190

Bio

Test

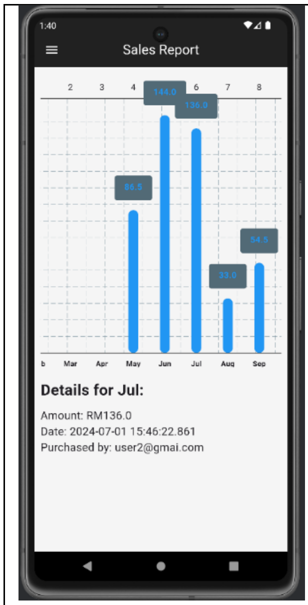
Figure 5.3.18 Profile Page



Username: User 1
Email: user1@gmail.com
Phone No.: 0121234567
Bio: Test

Figure 5.3.19 Edit Profile Page

5.3.5 Manager Pages UI



Details for Jul:

Amount: RM136.0
Date: 2024-07-01 15:46:22.861
Purchased by: user2@gmai.com

Figure 5.3.20 Sales Report Page

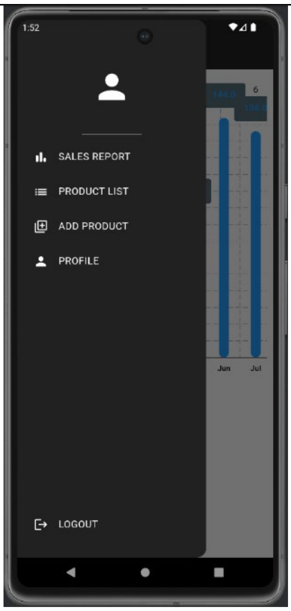
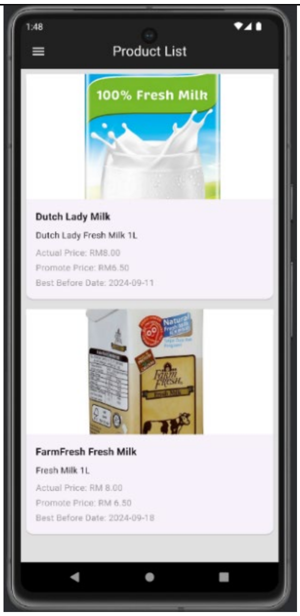


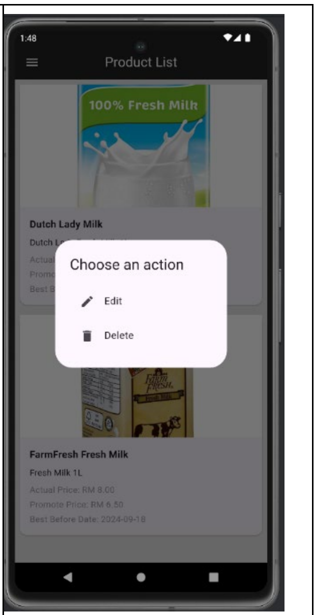
Figure 5.3.21 Menu



Dutch Lady Milk
Dutch Lady Fresh Milk 1L
Actual Price: RM8.00
Promote Price: RM6.50
Best Before Date: 2024-09-11

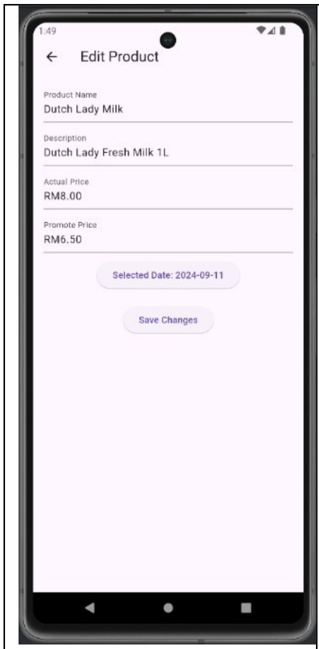
FarmFresh Fresh Milk
Fresh Milk 1L
Actual Price: RM 8.00
Promote Price: RM 6.50
Best Before Date: 2024-09-18

Figure 5.3.22 Manager Product List Page



Choose an action
Edit
Delete

Figure 5.3.23 Popup Selection



Product Name
Dutch Lady Milk

Description
Dutch Lady Fresh Milk 1L

Actual Price
RM8.00

Promote Price
RM6.50

Figure 5.3.24 Edit Product Page

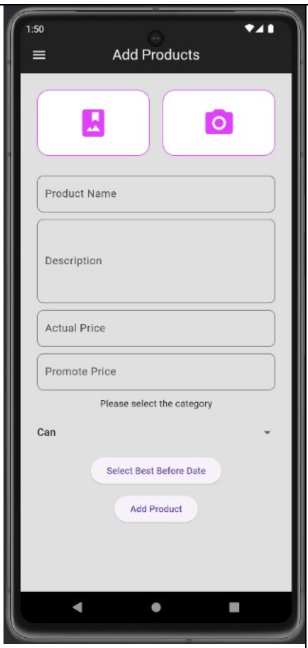
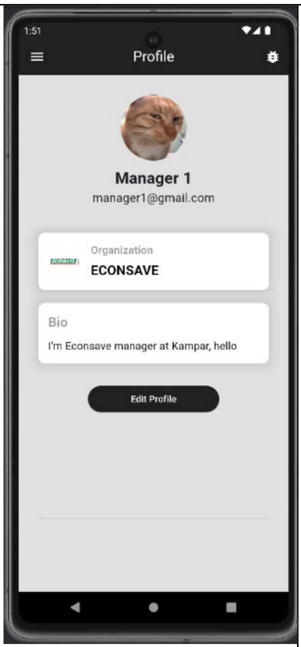


Figure 5.3.25 Add Product Page

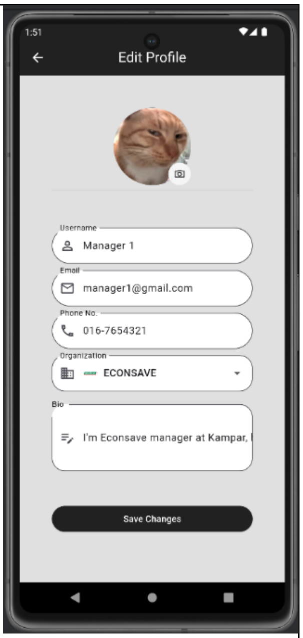


Manager 1
manager1@gmail.com

Organization
ECONSAVE

Bio
I'm Econsave manager at Kampar, hello

Figure 5.3.26 Manager Profile Page



Username
Manager 1

Email
manager1@gmail.com

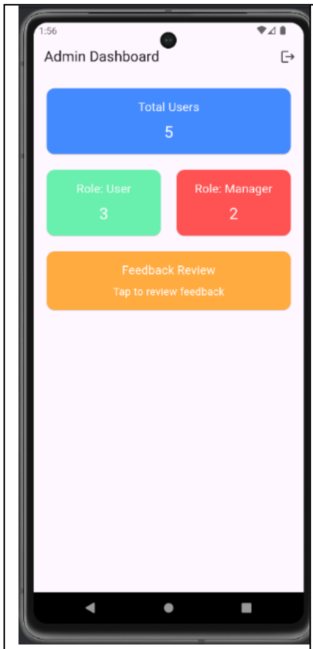
Phone No.
016-7654321

Organization
ECONSAVE

Bio
I'm Econsave manager at Kampar, I

Figure 5.3.27 Edit Manager Profile Page

5.3.6 Admin Pages UI



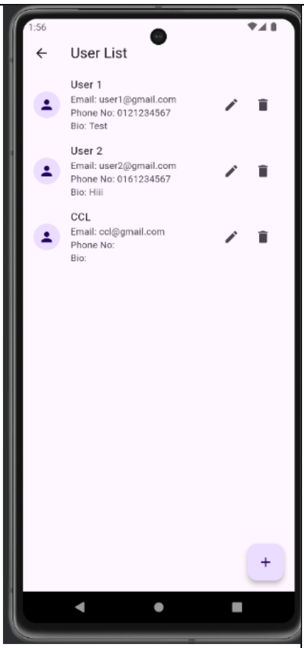
Total Users
5

Role: User
3

Role: Manager
2

Feedback Review
Tap to review feedback

Figure 5.3.28 Admin Dashboard

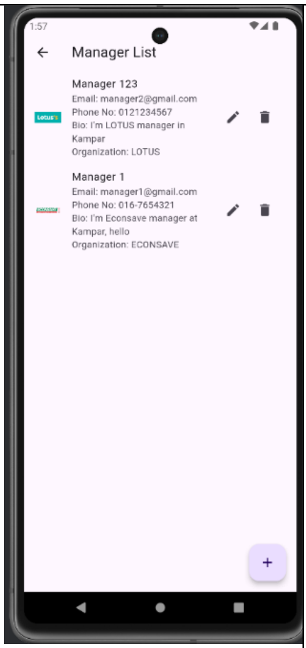


User 1
Email: user1@gmail.com
Phone No: 0121234567
Bio: Test

User 2
Email: user2@gmail.com
Phone No: 0161234567
Bio: Hiii

CCL
Email: ccl@gmail.com
Phone No:
Bio:

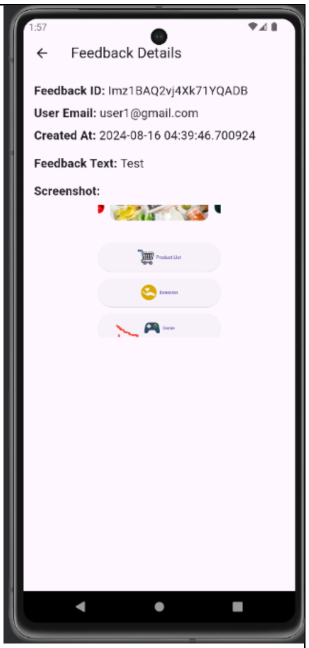
Figure 5.3.29 User List



Manager 123
Email: manager2@gmail.com
Phone No: 0121234567
Bio: I'm LOTUS manager in Kampar
Organization: LOTUS

Manager 1
Email: manager1@gmail.com
Phone No: 016-7654321
Bio: I'm Econsave manager at Kampar, hello
Organization: ECONSAVE

Figure 5.3.30 Manager List



Feedback ID: lmz1BAQ2vj4Xk71YQADB

User Email: user1@gmail.com

Created At: 2024-08-16 04:39:46.700924

Feedback Text: Test

Screenshot:

Figure 5.3.31 Feedback Details

5.4 Implementation Issues and Challenges

In this project, I encountered several issues and challenges during the development process. First and foremost, learning new development tools, such as Unity and Flutter, proved to be quite demanding. Using Unity to design the game elements, including the background, game mechanics, and scripting to handle the interactions between game objects, was time-consuming. Understanding Unity's components and how to properly implement them required extensive research and experimentation, especially as I was unfamiliar with the platform at the beginning. Additionally, designing a UI for the game was particularly challenging. Creating a visually appealing and user-friendly UI is crucial for enhancing user experience, especially since the app relies heavily on keeping users engaged over long periods of gameplay. Crafting a seamless and intuitive UI took considerable time and effort.

One of the significant challenges I faced was integrating the Unity game into the Flutter app. While Flutter is widely known for its cross-platform capabilities, integrating Unity-based game components into the Flutter framework proved to be a complex and ultimately unsuccessful task. The main issue stemmed from compatibility challenges between Unity's real-time rendering engine and Flutter's UI framework, which are built on entirely different architectures.

During the integration process, I encountered numerous errors and roadblocks that could not be resolved. These included technical issues related to rendering, communication between the two platforms, and difficulties in embedding Unity's game engine within a Flutter environment. Despite attempting several potential solutions, such as using third-party packages and bridging techniques, the integration could not be completed successfully. The tools and resources available for integrating Unity with Flutter are still limited, which made it particularly difficult to find a stable solution. As a result, I was unable to proceed with incorporating the game into the Flutter app.

In conclusion, the major challenge of the project was the failure to integrate the Unity game into the Flutter app. Despite investing considerable time in troubleshooting and testing various methods, I was unable to resolve the technical barriers between the two platforms.

CHAPTER 6 System Evaluation and Discussion

Testing will be conducted from two key perspectives: system testing and usability testing, to ensure both the functional integrity and user-friendliness of the application. System testing involves isolating individual modules or components to verify their compliance with predefined standards, identifying potential flaws or coding issues early in the development process. This method ensures that each part of the application performs as intended on a technical level. In contrast, usability testing focuses on gathering feedback from actual users to evaluate the application's user interface and overall experience, therefore invited 5 users to test the application. Through interactive sessions or questionnaires, usability testing examines factors such as user satisfaction, ease of navigation, and clarity of instructions. By integrating these two approaches, developers gain a comprehensive understanding of both the application's technical stability and user-cantered design, enabling them to identify and resolve issues more effectively before the application is released.

6.1 System Testing

6.1.1 Registration

Table 4 Registration Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Fill in all the field	Username: Tester Email: test@gmail.com Password: 123456 Confirm Password: 123456 Role: User	Register successful and navigate to user home page	Successfully registered and go to home page	Pass
2.	Only fill in the email and password	Username: Email:	Register successful and	Successfully registered and	Pass

		test1@gmail.com Password: 123456 Confirm Password: 123456 Role:	navigate to user home page	go to user home page	
3.	Without fill in email	Username: Tester 2 Email: Password: 123456 Confirm Password: 123456 Role: User	Cannot register and will show error message	Popup message" Email and Password cannot be empty"	Pass
4.	Without fill in password	Username: Tester 3 Email: tester3@gmail.com Password: Confirm Password: Role: User	Cannot register and will show error message	Popup message" Email and Password cannot be empty"	Pass
5.	Password and Confirm Password not same	Username: Tester 4 Email: tester4@gmail.com	Cannot register and show error message	Popup message" Password do not match"	Pass

		Password: 123456 Confirm Password: 121234 Role: User			
6.	Wrong email format	Username: Tester 5 Email: tester5@gmail Password: 123456 Confirm Password: 123456 Role: User	Cannot register and show error message	Popup message "The email address is badly formatted"	Pass
7.	Select role "Manager"	Username: Manager 1 Email: manager1@gmail Password: 123456 Confirm Password: 123456 Role: Manager	Register successful and navigate to sales report page	Successfully register and go to sales report page	Pass

6.1.2 Login

Table 5 User Login Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Fill with valid user's account email and password	Email: user1@gmail.com Password: 123456	Login Successful and navigate to user home page	Login successful and navigate to user home page	Pass
2.	Fill with invalid user's account password	Email: user1@gmail.com Password: 678910	Cannot login and show error message	Popup message "invalid credential"	Pass
3.	Fill with invalid user's account email	Email: user12@gmail.com Password: 123456	Cannot login and show error message	Popup message "invalid credential"	Pass
4.	Empty email and password	Email: Password:	Cannot login and show error message	Popup message "Email and Password be empty"	Pass
5.	Google account login by click the Google button		Login successful and navigate to user home page	Login successful and navigate to user home page	Pass
6.	Fill in manager's account email and password	Email: manager1@gmail.com Password: 123456	Login successful and navigate to sales report page	Login successful and go to sales report page	Pass

6.1.3 Add to Cart and Check Out

Table 6 Cart and Check Out Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Add product to cart	1 product in cart with price RM 6.50	Calculate total amount equal to RM 6.50 in check out	Pay RM 6.50 successful	Pass

2.	Add 3 same products to cart	3 products in cart each RM 6.50	Calculate total amount equal to RM 19.50	Pay RM 19.50 successful	Pass
3.	Add 3 different products to cart	1st product with RM 6.50, 2nd product with RM 4.60, and 3rd product with RM 8.00	Calculate total amount equal to RM 19.10	Pay RM 19.10 successful	Pass
4.	Add 3 same products to cart	3 products in cart each RM 6.50 and apply voucher RM 1.00	Total amount equal to RM 19.50 and pay RM 18.50 successful	Pay RM 18.50 successful	Pass
5.	Add 3 same products to cart	3 products in cart with RM 6.50 and apply voucher RM 3.00	Total amount equal to RM 19.50 and pay RM 16.50 successful	Pay RM 16.50 successful	Pass

6.1.4 Card Payment

Table 7 Card Payment Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Insert valid card information	Card number: 4242 4242 4242 4242 Expired date: 10/27 Cvv: 123 Country: Malaysia	Payment Successful	Paid Successful	Pass
2.	Insert invalid card number	Card number: 1111 4242 4242 4242 Expired date: 10/27	Show error message “Your card number is invalid”	Show message “Your card number is invalid”	Pass

		Cvv: 123 Country: Malaysia			
3.	Insert invalid Expired Date	Card number: 1111 4242 4242 4242 Expired date: 10/20 Cvv: 123 Country: Malaysia	Show error message “Your card’s expiration year is invalid”	Show error message “Your card’s expiration year is invalid”	Pass
4.	Insert invalid country	Card number: 4242 4242 4242 4242 Expired date: 10/27 Cvv: 123 Country: United Kingdom	Unable to click pay button	Unable to click pay button	Pass
5.	Empty card information	Card number: Expired date: Cvv: Country: Malaysia	Unable to click pay button	Unable to click pay button	Pass

6.1.5 Create Post

Table 8 Create Post Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Create New Post with image and description	Image: image 1 Description: Good	Successfully post	Successfully post	Pass

2.	Create New Post without image and description	Image: Description:	Cannot post and show error message	Cannot post and show error message in description box	Pass
3.	Create New Post without image	Image: Description: Good	Cannot post and show error message	Cannot post and show error message "Please select an image"	Pass
4.	Create New Post without description	Image: image 2 Description:	Cannot post and show error message	Cannot post and show error message in description box	Pass

6.1.6 Post

Table 9 Post Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Like, comment, and share post	Like:1 Comment: Nice Share	Successful like the post, post comment, and share the post	Successful like the post, post comment, and share the post	Pass
2.	Edit the post	Image: image 5 Description: Very Good	Successful edit the post image and description	Successful change the post image and description	Pass
3.	Edit other user's posts		No option for edit other user's posts	No option for edit other user's posts	Pass
4.	Delete the post		Successful delete post	Successful remove the post	Pass
5.	Delete other user's posts		No option for delete other user's posts	No option for delete other user's posts	Pass

6.	Edit comment	Comment: So good	Successful edit comment	Successful change the comment	Pass
7.	Edit other user's comment	Comment: So bad	No option for edit other user's comment	No option for edit other user's comment	Pass
8.	Delete comment		Successful delete comment	Successful remove comment from the post	Pass
9.	Delete other user's comment		No option for delete other user's comment	No option for delete other user's comment	Pass

6.1.7 Edit User Profile

Table 10 Edit User Profile Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Edit all the information with valid data	Profile Picture: picture 123 Username: User 123 Phone No.: 0191234567 Bio: Hello, I'm Alex	Successful edit the username, phone number, and bio	Changed the username, phone number and bio	Pass
2.	Edit phone number with invalid format	Profile Picture: picture 123 Username: User 123 Phone No.: 0191234567123 Bio: Hello, I'm Alex	Unsuccessful edit phone number	Cannot change the phone number	Pass

3.	Remove the username phone number, and bio	Profile Picture: picture 123 Username: Phone No.: Bio:	Successful edit the username, phone number, and bio	Changed the username, phone number and bio	Pass

6.1.8 Add Product

Table 11 Add Product Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Insert all the valid data	Image: image 1 Product Name: Dutch Lady Milk Description: Dutch Lady fresh milk (1L) Actual Price: RM 8.00 Promote Price: RM 6.50 Date: 2024-09-19	Successful add a product	Successful add a product	Pass
2.	Without insert product name	Image: image 1 Product Name: Description: Dutch Lady fresh milk (1L) Actual Price: RM 8.00 Promote Price: RM 6.50 Date: 2024-09-19	Show error message	Show error message “Please fill in all fields before adding a product”	Pass

3.	Without insert product description	Image: image 1 Product Name: Dutch Lady Milk Description: Actual Price: RM 8.00 Promote Price: RM 6.50 Date: 2024-09-19	Successful add a product	Successful add a product	Pass
4.	Without insert actual price	Image: image 1 Product Name: Dutch Lady Milk Description: Dutch Lady fresh milk (1L) Actual Price: Promote Price: RM 6.50 Date: 2024-09-19	Show error message	Show error message "Please fill in all fields before adding a product"	Pass
5.	Without select a date	Image: image 1 Product Name: Dutch Lady Milk Description: Dutch Lady fresh milk (1L) Actual Price: RM 8.00 Promote Price: RM 6.50 Date:	Show error message	Show error message "Please fill in all fields before adding a product"	Pass

6.1.9 Edit Product

Table 12 Edit Product Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Edit all the product details	Product Name: Dutch Lady Chocolate Milk Description: Dutch Lady chocolate milk (1L) Actual Price: RM 8.50 Promote Price: RM 7.50 Date: 2024-09-20	Successful update the product details	Successful update the product details	Pass
2.	Edit the product name to empty	Product Name: Description: Dutch Lady fresh milk (1L) Actual Price: RM 8.00 Promote Price: RM 6.50 Date: 2024-09-19	Show error message	Show error message "Please fill in all require fields"	Pass
3.	Edit the product description to empty	Product Name: Dutch Lady Milk Description: Actual Price: RM 8.00 Promote Price: RM 6.50 Date: 2024-09-19	Successful update the product details	Successful update the product details	Pass

4.	Edit the actual price to empty	Product Name: Dutch Lady Milk Description: Dutch Lady fresh milk (1L) Actual Price: Promote Price: RM 6.50 Date: 2024-09-19	Show error message	Show error message "Please fill in all require fields"	Pass
5.	Edit the promote price to empty	Product Name: Dutch Lady Milk Description: Dutch Lady fresh milk (1L) Actual Price: RM 8.00 Promote Price: Date: 2024-09-19	Show error message	Show error message "Please fill in all require fields"	Pass

6.1.10 Edit Manager Profile

Table 13 Edit Manager Profile Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Edit all the manager profile details	Profile Picture: picture 111 Username: User 123 Phone No.: 0121234567 Organization: ECONSAVE Bio: Hello, I'm John	Successful update profile details	Update all the profile details success	Pass

2.	Edit the username to empty	Profile Picture: picture 111 Username: Phone No.: 0121234567 Organization: ECONSAVE Bio: Hello, I’m John	Successful update profile details	Update all the profile details success	Pass
3.	Edit the phone number with invalid format	Profile Picture: picture 111 Username: User 123 Phone No.: 012123456711 Organization: ECONSAVE Bio: Hello, I’m John	Show error message	Show error message “Please enter a valid phone number format”	Pass
4.	Edit the bio to empty	Profile Picture: picture 111 Username: User 123 Phone No.: 0121234567 Organization: ECONSAVE Bio:	Successful update profile details	Update all the profile details success	

6.1.11 Admin Dashboard

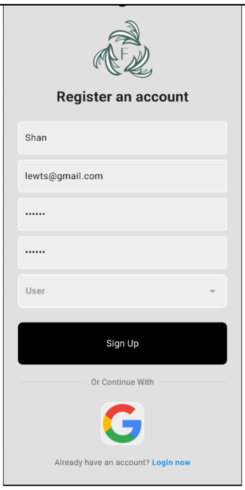
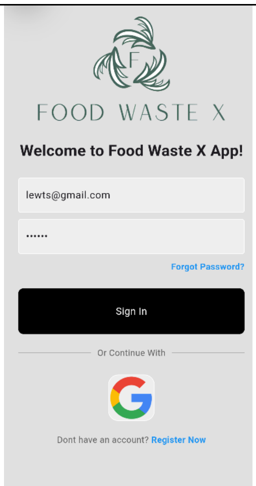
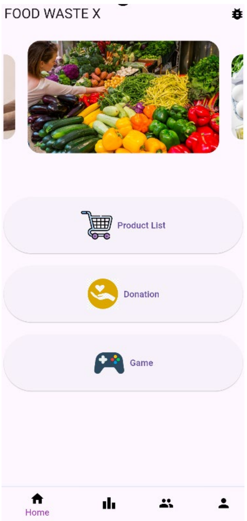
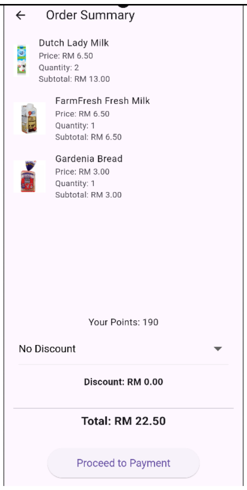
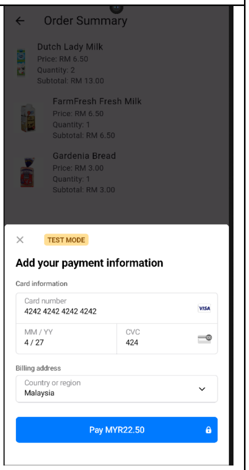
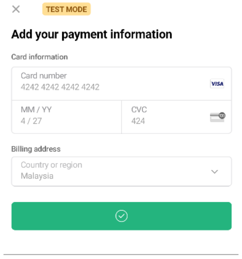
Table 14 Admin Dashboard Testing

No.	Test Case Description	Test Data	Expected Outcome	Actual Outcome	Pass/Fail
1.	Edit all user details	Email: user1@gmail.com Username: User 1 Phone No.: 0121234567 Bio: Hi, I'm Steve	Successful update user details	Update user details success	Pass
2.	Delete the user details		Delete all the details of the user	Delete all the details of the user	Pass
3.	Edit all manager details	Email: manager1@gmail.com Username: Manager 1 Phone No.: 0161234567 Bio: Hi, I'm Olivia	Successful update manager details	Update manager details success	Pass
4.	Delete the manager details		Delete all the details of the manager	Delete all the details of the manager	Pass

6.2 User Testing

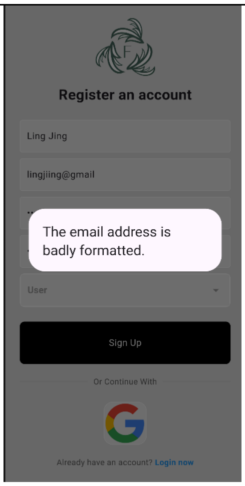
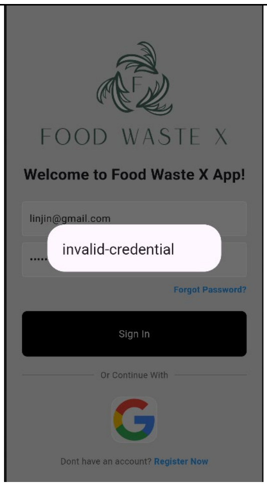
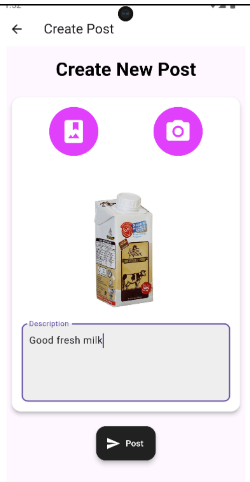
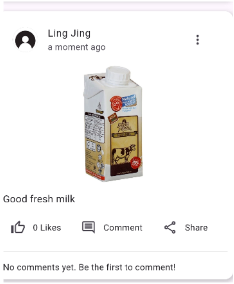
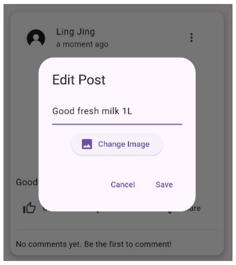
6.2.1 User 1: Lew Tian Shan

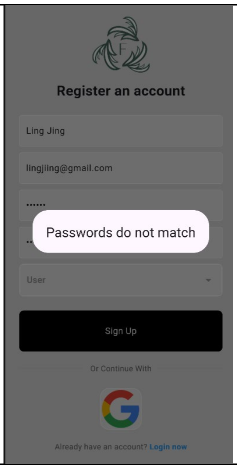
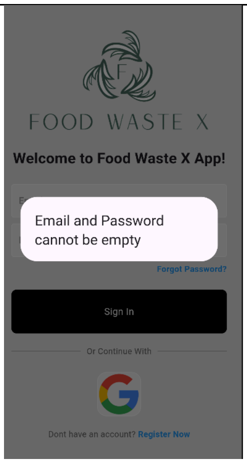
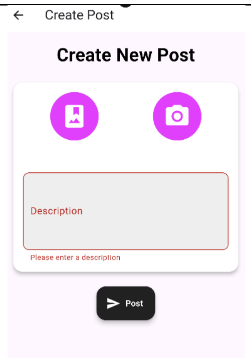
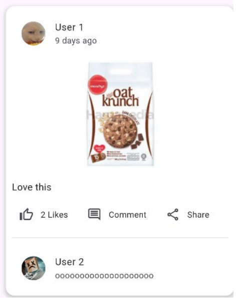
Table 15 User 1 Testing

Test Case	Register	Login	Add To Cart	Payment
Description	Fill all the fields with valid data	Fill in valid account credential	Add a few products	Fill in valid card information
Expected Outcome	Successful register an account	Successful login	Add a few products and show calculated total amount	Successful pay the total amount
Actual Outcome	Successful register an account	Successful login and	Shown calculated total amount	Successful paid the total amount
Result	 Register an account Shan lewts@gmail.com ······ ······ User Sign Up Or Continue With Already have an account? Login now	 FOOD WASTE X Welcome to Food Waste X App! lewts@gmail.com ······ Forgot Password? Sign In Or Continue With Dont have an account? Register Now  FOOD WASTE X Product List Donation Game Home	 Order Summary Dutch Lady Milk Price: RM 6.50 Quantity: 2 Subtotal: RM 13.00 FarmFresh Fresh Milk Price: RM 6.50 Quantity: 1 Subtotal: RM 6.50 Gardenia Bread Price: RM 3.00 Quantity: 1 Subtotal: RM 3.00 Your Points: 190 No Discount Discount: RM 0.00 Total: RM 22.50 Proceed to Payment	 Order Summary TEST MODE Add your payment information Card information Card number 4242 4242 4242 4242 MM / YY 4 / 27 CVC 424 Billing address Country or region Malaysia Pay MYR22.50  TEST MODE Add your payment information Card information Card number 4242 4242 4242 4242 MM / YY 4 / 27 CVC 424 Billing address Country or region Malaysia
Pass/Fail	Pass	Pass	Pass	Pass

6.2.2 User 2: Ling Jing

Table 16 User 2 Testing

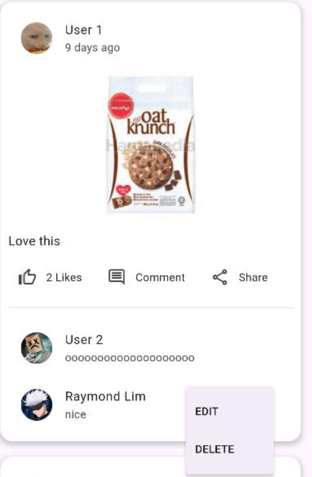
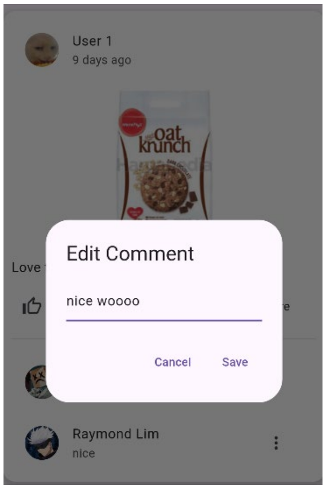
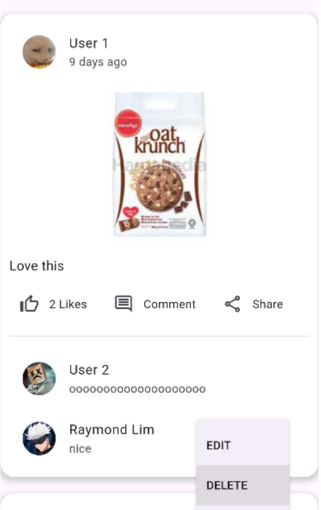
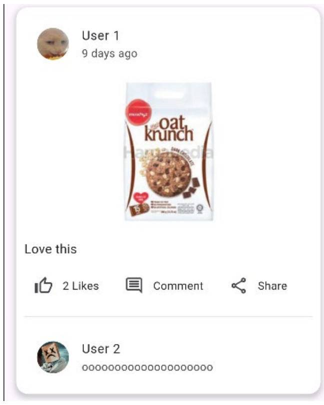
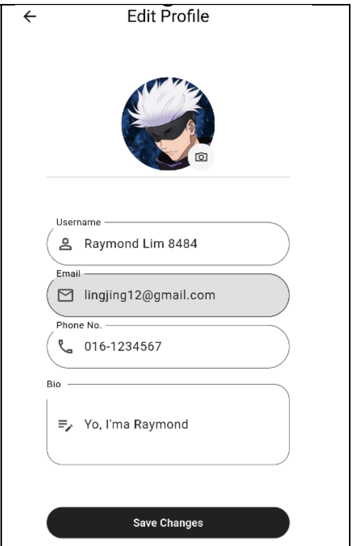
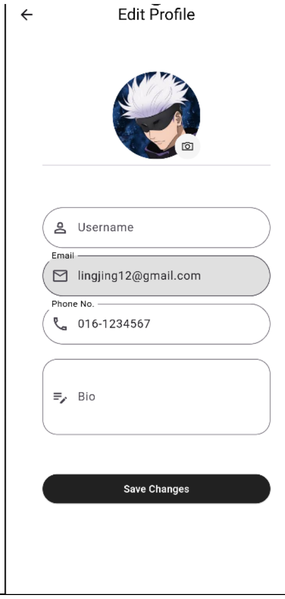
Test Case	Register	Login	Create Post	Edit Post
Description	Fill in with invalid email format and password that does not match with confirm password	Fill in valid account credential and empty input	Select an image and write description for post. Also try not fill any fields.	Edit post description, also try to edit other user's post.
Expected Outcome	Unable to register an account and show error message	Login not success and show error message	For fill in all fields, successful post. For not fill in any fields, unable to post and show error message.	Update post success for own post Unable to edit other user's post
Actual Outcome	Popup error message "The email is badly formatted" and "passwords do not match"	Popup error message "invalid credential" and "Email and Password cannot be empty"	Successful post when fill in all the fields Unable post when not fill in any fields	Update post success for own post No option to edit other user's post
Result	 Register an account Ling Jing lingjiing@gmail The email address is badly formatted. User Sign Up Or Continue With Already have an account? Login now	 FOOD WASTE X Welcome to Food Waste X App! linjin@gmail.com invalid-credential Forgot Password? Sign In Or Continue With Dont have an account? Register Now	 Create Post Create New Post Description Good fresh milk Post	 Ling Jing a moment ago Good fresh milk 0 Likes Comment Share No comments yet. Be the first to comment!  Edit Post Good fresh milk 1L Change Image Cancel Save No comments yet. Be the first to comment!

	 Register an account Ling Jing lingjiing@gmail.com Passwords do not match User Sign Up Or Continue With Already have an account? Login now	 FOOD WASTE X Welcome to Food Waste X App! Email and Password cannot be empty Forgot Password? Sign In Or Continue With Dont have an account? Register Now	 Create New Post Please enter a description 	 User 1 9 days ago Love this User 2 oooooooooooooooooooo
Pass/Fail	Pass	Pass	Pass	Pass

6.2.4 User 4: Chong ZhenWei

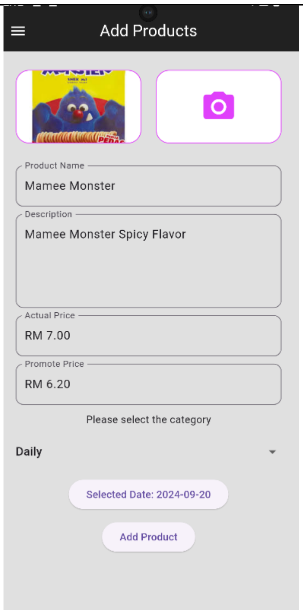
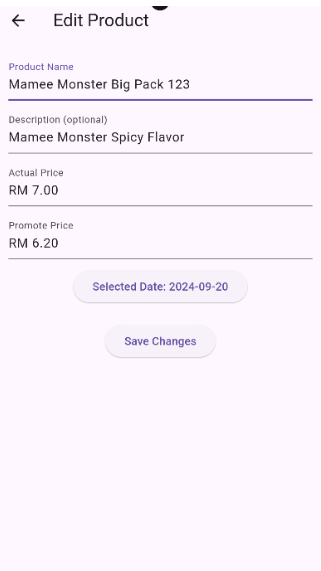
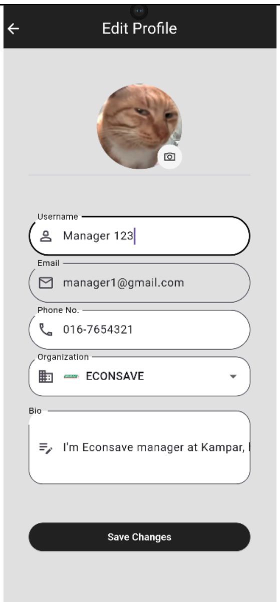
Table 17 User 3 Testing

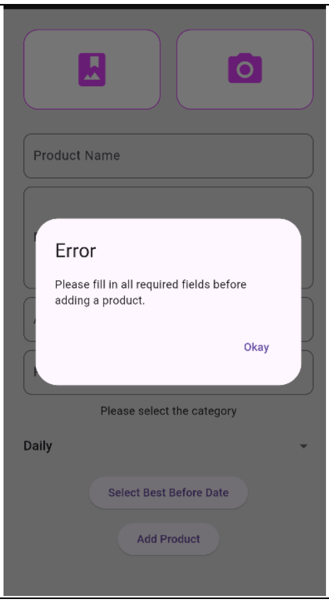
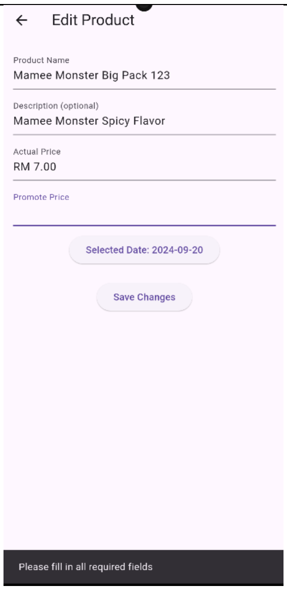
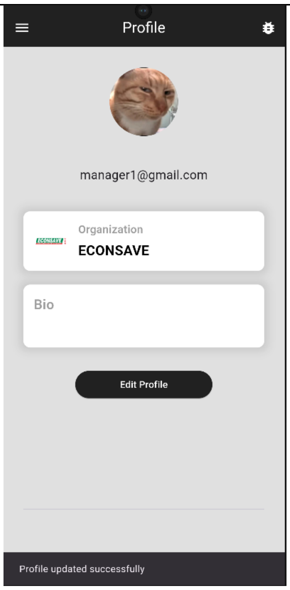
Test Case	Edit Comment	Delete Comment	Edit User Profile
Description	Edit own comment and try edit other user's comment	Delete own comment and try delete other user's comment	Edit all user profile details and try change username and bio to empty
Expected Outcome	Successful update comment Unable edit other user 's comment	Successful delete comment Unable delete other user's comment	Successful update the user profile details
Actual Outcome	Update comment success No option for edit other user's comment	Delete comment success No option for delete other user's comment	Successful update the user profile details

Result	 User 1 9 days ago Love this 2 Likes Comment Share User 2 oooooooooooooooooooo Raymond Lim nice EDIT DELETE  User 1 9 days ago Edit Comment nice woooo Cancel Save Raymond Lim nice	 User 1 9 days ago Love this 2 Likes Comment Share User 2 oooooooooooooooooooo Raymond Lim nice EDIT DELETE  User 1 9 days ago Love this 2 Likes Comment Share User 2 oooooooooooooooooooo	 Edit Profile Username: Raymond Lim 8484 Email: lingjing12@gmail.com Phone No.: 016-1234567 Bio: Yo, I'ma Raymond Save Changes  Edit Profile Username Email: lingjing12@gmail.com Phone No.: 016-1234567 Bio Save Changes
Pass/Fail	Pass	Pass	Pass

6.2.3 User 3: Raymond Lim Weng Kuin

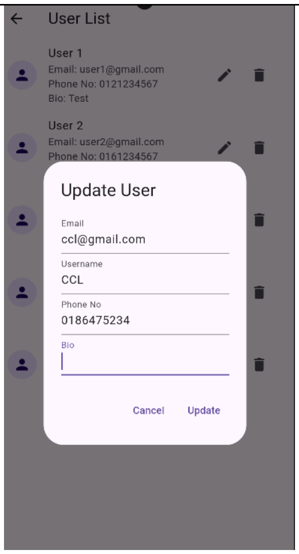
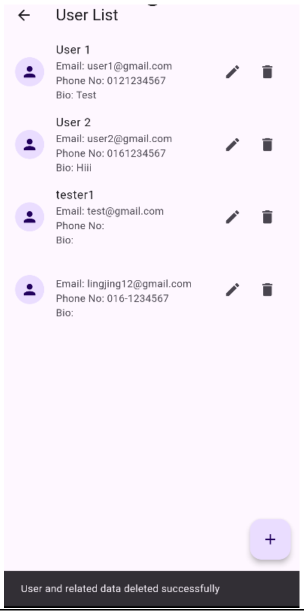
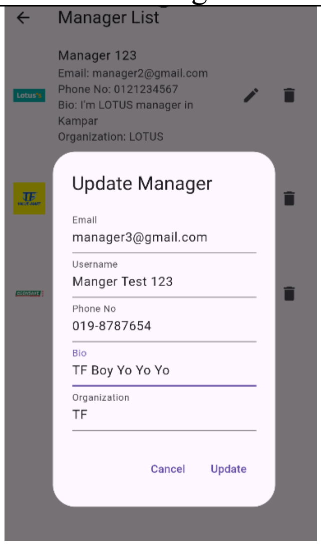
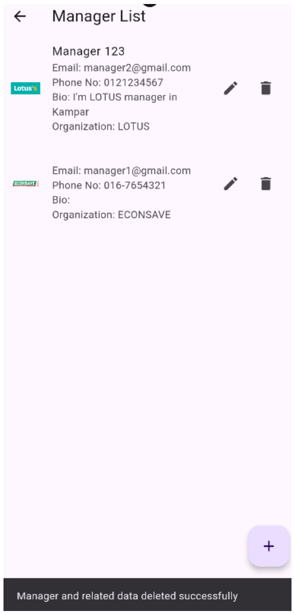
Table 18 User 4 Testing

Test Case	Add Product	Edit Product	Edit Manager Profile
Description	Fill in all fields and try not fill in any fields	Change all the product details and try not fill the promote price	Edit all manager profile details and try change username and bio to empty
Expected Outcome	Successful add a product Unable to add product and show error message	Successful update the product details Unable to save the changes and show error message	Successful update the manager profile details
Actual Outcome	Successful add a product when fill in all fields Unable to add product and show error message "Please fill in all required fields before adding a product"	Successful update the product details when fill in all the fields Unable to save the changes and show error message "Please fill in all required fields"	Successful update the manager profile details
Result	 Add Products Product Name: Mamee Monster Description: Mamee Monster Spicy Flavor Actual Price: RM 7.00 Promote Price: RM 6.20 Please select the category Daily Selected Date: 2024-09-20 Add Product	 Edit Product Product Name: Mamee Monster Big Pack 123 Description (optional): Mamee Monster Spicy Flavor Actual Price: RM 7.00 Promote Price: RM 6.20 Selected Date: 2024-09-20 Save Changes	 Edit Profile Username: Manager 123 Email: manager1@gmail.com Phone No.: 016-7654321 Organization: ECONSAVE Bio: I'm Econsave manager at Kampar, Save Changes

	 Product Name Error Please fill in all required fields before adding a product. Okay Please select the category Daily Select Best Before Date Add Product	 Edit Product Product Name Mamee Monster Big Pack 123 Description (optional) Mamee Monster Spicy Flavor Actual Price RM 7.00 Promote Price Selected Date: 2024-09-20 Save Changes Please fill in all required fields	 Profile manager1@gmail.com Organization ECONSAVE Bio Edit Profile Profile updated successfully
Pass/Fail	Pass	Pass	Pass

6.2.5 User 5: Yong Wei Lok

Table 19 User 5 Testing

Test Case	Admin Dashboard – Edit and Delete User Details	Admin Dashboard – Edit and Delete Manager Details
Description	Edit and delete user details	
Expected Outcome	Successful update user details Successful delete user details	Successful update manager details Successful delete manager details
Actual Outcome	Successful update user details Successful delete user details	Successful update manager details Successful delete manager details
Result	 User List User 1 Email: user1@gmail.com Phone No: 0121234567 Bio: Test User 2 Email: user2@gmail.com Phone No: 0161234567 Update User Email: ccl@gmail.com Username: CCL Phone No: 0186475234 Bio:  User List User 1 Email: user1@gmail.com Phone No: 0121234567 Bio: Test User 2 Email: user2@gmail.com Phone No: 0161234567 Bio: Hiii tester1 Email: test@gmail.com Phone No: Bio: Email: lingjing12@gmail.com Phone No: 016-1234567 Bio: User and related data deleted successfully	 Manager List Manager 123 Email: manager2@gmail.com Phone No: 0121234567 Bio: I'm LOTUS manager in Kampar Organization: LOTUS Update Manager Email: manager3@gmail.com Username: Manger Test 123 Phone No: 019-8787654 Bio: TF Boy Yo Yo Yo Organization: TF  Manager List Manager 123 Email: manager2@gmail.com Phone No: 0121234567 Bio: I'm LOTUS manager in Kampar Organization: LOTUS Email: manager1@gmail.com Phone No: 016-7654321 Bio: Organization: ECONSAVE Manager and related data deleted successfully
Pass/Fail	Pass	Pass

6.3 Objectives Evaluation

The "Food Waste Mobile Application with Gamification" effectively meets its outlined objectives, demonstrating a comprehensive approach to addressing food waste through a gamified mobile solution. The first objective, which focuses on studying existing food waste gamified applications, is achieved through an extensive literature review. The document analyses a range of applications, such as the Too Good To Go app and similar gamification solutions, identifying both strengths and limitations in current industry practices. This thorough evaluation provides a solid foundation for the subsequent development of a unique application.

The second objective, which aims to propose gamification elements and data management solutions for addressing food waste, is well-executed. The application integrates game mechanics like points, leaderboards, and challenges, which are designed to motivate users to engage in sustainable food practices. Furthermore, the use of Firebase for data management ensures efficient handling of user and supermarket data, supporting scalability and functionality across various markets.

In line with the third objective, the project successfully develops a mobile application with engaging gamification features. These include leaderboards, badges, and competitions, which enhance user motivation. The design caters to both end users and supermarkets, addressing the problem from both supply and demand perspectives. Additionally, the application's potential for multi-market expansion is highlighted, increasing its overall impact.

Finally, the fourth objective, which involves evaluating the application through usability testing, is also met. User feedback demonstrates that the application is easy to use and enjoyable, effectively encouraging sustainable behavior. The results show that the gamified elements, combined with the user-friendly interface, enhance engagement and raise awareness about food waste.

6.4 Concluding Remark

The "Food Waste Mobile Application with Gamification" provides a well-rounded solution to the growing problem of food waste, particularly within supermarkets. By successfully integrating gamification elements such as points, challenges, and leaderboards, the application encourages users to adopt more sustainable food consumption habits. Its design effectively targets both consumers and suppliers, addressing the issue from both ends of the supply chain. The application's scalability and multi-market potential, combined with its user-friendly interface and positive usability feedback, demonstrate its effectiveness in fostering long-term behavioral change. Overall, the project achieves its objectives and contributes meaningfully to efforts in reducing food waste and promoting environmental sustainability.

CHAPTER 7 Conclusion and Recommendation

7.1 Conclusion

The development of the Food Waste Mobile Application with Gamification demonstrates a promising solution to the pressing issue of food waste in supermarkets. By integrating gamification techniques such as points, challenges, and rewards, the app effectively transforms routine activities into engaging experiences that encourage users to adopt more sustainable food consumption practices. This approach not only motivates users to reduce food waste but also fosters a sense of community through features like leaderboards and social interaction, promoting collective action toward environmental sustainability.

The application stands out by addressing both the supply and demand aspects of food waste, something that existing solutions often overlook. On the supply side, supermarket managers can monitor and manage food products nearing expiration, offering incentives to reduce waste. On the demand side, consumers are encouraged to purchase near-expiry products at discounted rates, helping to prevent unnecessary disposal while also benefiting economically. This dual approach enhances the application's impact by creating a bridge between retailers and consumers in the fight against food waste.

Usability testing revealed that the application is user-friendly, easy to navigate, and enjoyable, successfully engaging users in sustainable behaviour without feeling burdensome. By leveraging gamification, the app taps into intrinsic motivations such as competition, achievement, and reward, which drives long-term user engagement. The results of the tests also indicated increased awareness among users about the environmental and social consequences of food waste, highlighting the app’s effectiveness as both an educational tool and a behavioural change agent.

Overall, the Food Waste Mobile Application with Gamification marks a significant contribution to the field of food waste management. It not only provides a practical tool for supermarkets and consumers but also promotes a cultural shift toward more sustainable consumption patterns. By harnessing the power of technology and gamification, this project brings forward a scalable and impactful solution that aligns with global sustainability goals, particularly the aim to halve food waste by 2030.

7.2 Recommendation

While the Food Waste Mobile Application with Gamification has shown significant potential to reduce food waste through user engagement, there are several areas for future enhancement that could further increase its impact, functionality, and scalability.

1. Expansion of User Base and Markets: The application could be expanded to target other sectors of the food supply chain, such as restaurants, food manufacturers, and wholesalers. By broadening its scope, the app can increase its user base and further reduce food waste across different industries and regions.

2. Integration of Advanced Analytics: Incorporating data analytics and machine learning can provide more personalized user experiences. The app could track individual behaviours and suggest actions like purchasing near-expiry items, reducing food waste more efficiently. Predictive analytics could also help supermarkets better manage stock, reducing overall waste.

3. Competition with AI: A new feature could involve users competing against AI in the gamified tasks. For example, the AI could simulate different food waste management scenarios or challenge users to beat its scores in finding near-expiry products. This would add an extra layer of competition and engagement, encouraging users to improve their food waste management skills and strategies.

4. Stage Progression with Target Points: Introducing a structured level system would require users to clear each stage before advancing to the next. To proceed, users must achieve a minimum target score, such as 30 points or higher. This challenge-based progression would motivate users to perform better, adding an element of strategy and goal setting, while also ensuring continuous engagement throughout the app.

6. Partnerships with Food Banks and Charities: Expanding the app’s social impact through collaborations with food banks and charities could create a direct link for users to donate near-expiry products. Streamlining the donation process would enhance the app’s community and sustainability focus.

REFERENCES

[1] "FOOD WASTE INDEX REPORT 2021," Mar. 2021.

[2] FAO, IFAD, UNICEF, WFP, and and WHO, "The State of Food Security and Nutrition in the World 2023," *The State of Food Security and Nutrition in the World 2023*, Jul. 2023.

[3] J. Buzby, "Food Waste and its Links to Greenhouse Gases and Climate Change," *U.S. DEPARTMENT OF AGRICULTURE*, 24-Jan-2022. [Online]. Available: https://www.usda.gov/media/blog/2022/01/24/food-waste-and-its-links-greenhouse-gases-and-climate-change#:~:text=Food%20loss%20and%20waste%20also,even%20more%20potent%20greenhouse%20gas. [Accessed: 27-Nov-2023].

[4] "Goal 12 RESPONSIBLE CONSUMPTION AND PRODUCTION," *United Nations Development Programme*. [Online]. Available: https://www.undp.org/sustainable-development-goals/responsible-consumption-and-production. [Accessed: 03-Dec-2023].

[5] G. Farr-Wharton, J. H. J. Choi, and M. Foth, "Food talks back: Exploring the role of mobile applications in reducing domestic food wastage," in *Proceedings of the 26th Australian Computer-Human Interaction Conference, OzCHI 2014*, 2014, pp. 352–361.

[6] E. O. Adewuyi and K. Adefemi, "Behavior Change Communication Using Social Media: A Review."

[7] R. S. Alsawaier, "The effect of gamification on motivation and engagement," *International Journal of Information and Learning Technology*, vol. 35, no. 1. Emerald Group Publishing Ltd., pp. 56–79, 2018.

[8] "FOOD WASTE INDEX REPORT 2021," 2021.

[9] J. Hong, A. Jaegler, and O. Gergaud, "Mobile applications to reduce food waste in supply chains: a systematic literature review," *British Food Journal*, vol. 126, no. 2, pp. 509–530, Jan. 2024.

[10] N. M. Tuah, S. Khadijah, A. Ghani, S. Darham, and S. Sura, "A Food Waste Mobile Gamified Application Design Model using UX Agile Approach in Malaysia," *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 13, no. 5, 2022.

[11] T. H. Nguyen, "Employing gamification to support sustainable food consumption Analysis and Redesign of the Too Good To Go mobile app," 2020.

[12] "The Octalysis Framework for Gamification & Behavioral Design." [Online]. Available: https://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/. [Accessed: 27-Jan-2024].

[13] K. Venessa and H. W. Aripradono, "Usage of Gamification and Mobile Application to Reduce Food Loss and Waste: A Case Study of Indonesia," *Journal of Information Systems and Informatics*, vol. 5, no. 1, pp. 102–122, Feb. 2023.

[14] N. P. Cechetti, E. A. Bellei, D. Biduski, J. P. M. Rodriguez, M. K. Roman, and A. C. B. De Marchi, “Developing and implementing a gamification method to improve user engagement: A case study with an m-Health application for hypertension monitoring,” *Telematics and Informatics*, vol. 41, pp. 126–138, Aug. 2019.

[15] E. Närvänen, N. Mesiranta, M. Mattila, and A. Heikkinen, “Introduction: A framework for managing food waste,” *Food Waste Management: Solving the Wicked Problem*. Springer International Publishing, pp. 1–24, 03-Sep-2019.

[16] “Meet Android Studio | Android Developers.” [Online]. Available: https://developer.android.com/studio/intro. [Accessed: 25-Jan-2024].

[17] “Flutter - Build apps for any screen.” [Online]. Available: https://flutter.dev/. [Accessed: 25-Jan-2024].

[18] “Dart overview | Dart.” [Online]. Available: https://dart.dev/overview. [Accessed: 25-Jan-2024].

[19] “Unity - Introduction.” [Online]. Available: https://www.tutorialspoint.com/unity/unity_introduction.htm. [Accessed: 05-Feb-2024].

[20] “What is the Visual Studio IDE? | Microsoft Learn.” [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022. [Accessed: 10-Apr-2024].

FINAL YEAR PROJECT WEEKLY REPORT

(Project 2)

Trimester, Year: Y3S3	**Study week no.: 2&3**
Student Name & ID: Chai Cun Lin 20ACB03887	
Supervisor: Ms Tseu Kwan Lee	
Project Title: Food Waste Mobile Application With Gamification	

1. WORK DONE

- In Week 2, I discuss with supervisor for the to do list arrangement for following weeks.
- In Week 3, redesigned the login and register page, also added the forgot password function.

2. WORK TO BE DONE

- Dashboard for admin and manager
- Redesign social post page, profile page, product and donation page.

3. PROBLEMS ENCOUNTERED

No

4. SELF EVALUATION OF THE PROGRESS

Need to learn how to create chart in Flutter for the dashboard.

Supervisor's signature

Student's signature

Trimester, Year: Y3S3	**Study week no.: 5&6**
Student Name & ID: Chai Cun Lin 20ACB03887	
Supervisor: Ms Tseu Kwan Lee	
Project Title: Food Waste Mobile Application With Gamification	

1. WORK DONE

- In Week 5, discuss how the design of admin page UI
- In Week 6, done redesigned the product list page, become grid view.

2. WORK TO BE DONE

- Add product detail’s view, add to cart function, create the admin home page and manager pages.
- Redesign social post page and donation page.

3. PROBLEMS ENCOUNTERED

No.

4. SELF EVALUATION OF THE PROGRESS

Don’t know what should show in the admin page.

Supervisor’s signature

Student’s signature

Trimester, Year: Y3S3	**Study week no.: 7&9**
Student Name & ID: Chai Cun Lin 20ACB03887	
Supervisor: Ms Tseu Kwan Lee	
Project Title: Food Waste Mobile Application With Gamification	

1. WORK DONE

- In Week 7, done design the UI for profile and added report error function.
- In Week 9, added edit profile function.

2. WORK TO BE DONE

- Add product's details and add to cart function
- Redesign social post page and donation page.
- Payment function for items in cart.
- Change the manager home page to the sales report page.

3. PROBLEMS ENCOUNTERED

No.

4. SELF EVALUATION OF THE PROGRESS

Need to speed up development.

Supervisor's signature

Student's signature

Trimester, Year: Y3S3	**Study week no.: 10&11**
Student Name & ID: Chai Cun Lin 20ACB03887	
Supervisor: Ms Tseu Kwan Lee	
Project Title: Food Waste Mobile Application With Gamification	

1. WORK DONE

- Completed the product details page and the add to cart function.
- Added make payment function in the cart page.
- Done the sales report page and set as manager home page

2. WORK TO BE DONE

- Discount using points that user has.
- Redesign social post page

3. PROBLEMS ENCOUNTERED

No

4. SELF EVALUATION OF THE PROGRESS

Require testing the application.

Supervisor's signature

Student's signature

Trimester, Year: Y3S3	**Study week no.: 12**
Student Name & ID: Chai Cun Lin 20ACB03887	
Supervisor: Ms Tseu Kwan Lee	
Project Title: Food Waste Mobile Application With Gamification	

1. WORK DONE

- Complete the discount function and redesigned the social post page.

2. WORK TO BE DONE

- Test the application and fix the bugs

3. PROBLEMS ENCOUNTERED

No

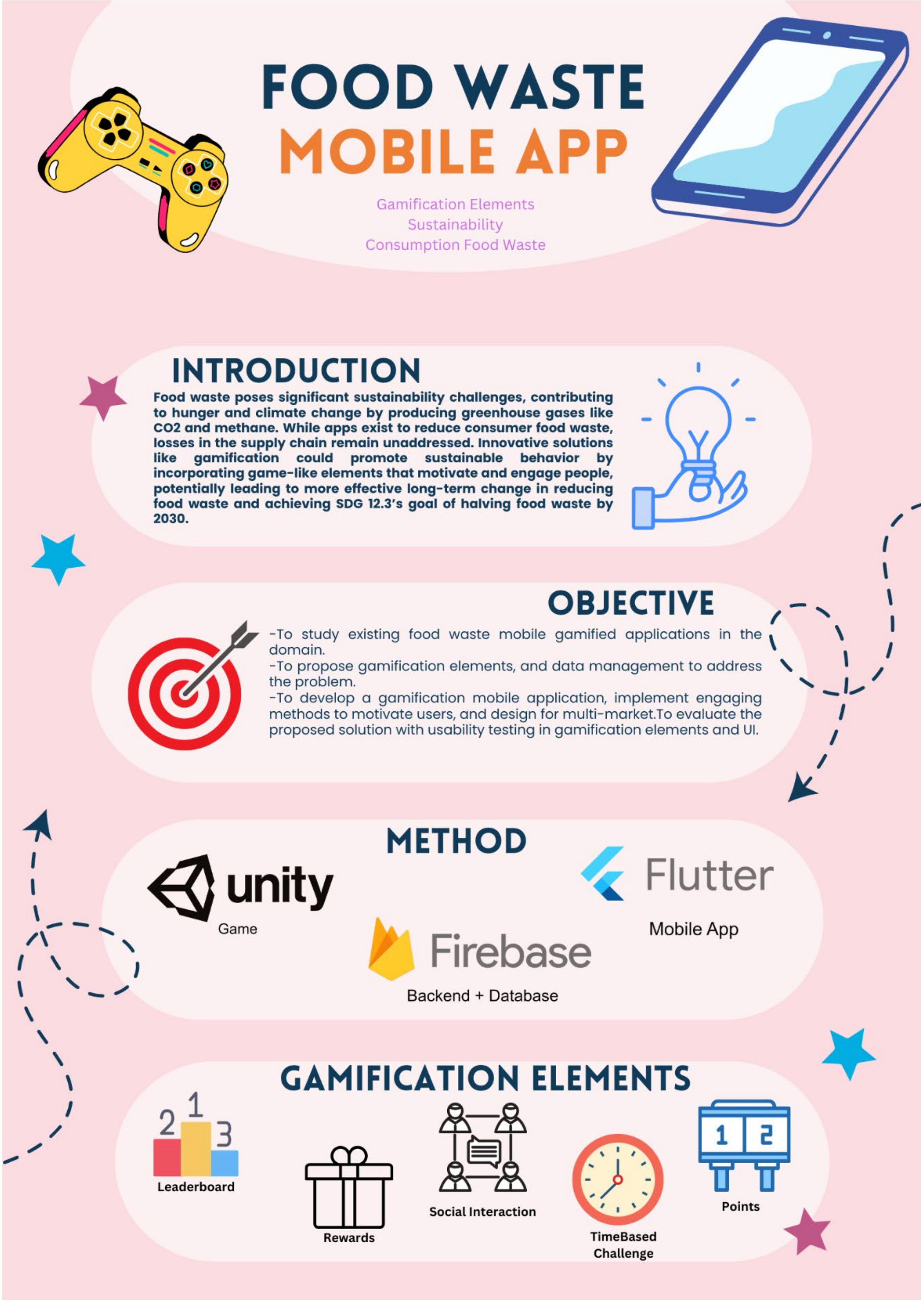
4. SELF EVALUATION OF THE PROGRESS

No enough testing the application

Supervisor’s signature

Student’s signature

POSTER



FOOD WASTE MOBILE APP

Gamification Elements
Sustainability
Consumption Food Waste

INTRODUCTION

Food waste poses significant sustainability challenges, contributing to hunger and climate change by producing greenhouse gases like CO2 and methane. While apps exist to reduce consumer food waste, losses in the supply chain remain unaddressed. Innovative solutions like gamification could promote sustainable behavior by incorporating game-like elements that motivate and engage people, potentially leading to more effective long-term change in reducing food waste and achieving SDG 12.3's goal of halving food waste by 2030.

OBJECTIVE

-To study existing food waste mobile gamified applications in the domain.
-To propose gamification elements, and data management to address the problem.
-To develop a gamification mobile application, implement engaging methods to motivate users, and design for multi-market.To evaluate the proposed solution with usability testing in gamification elements and UI.

METHOD

Game

Backend + Database

Mobile App

GAMIFICATION ELEMENTS

Leaderboard

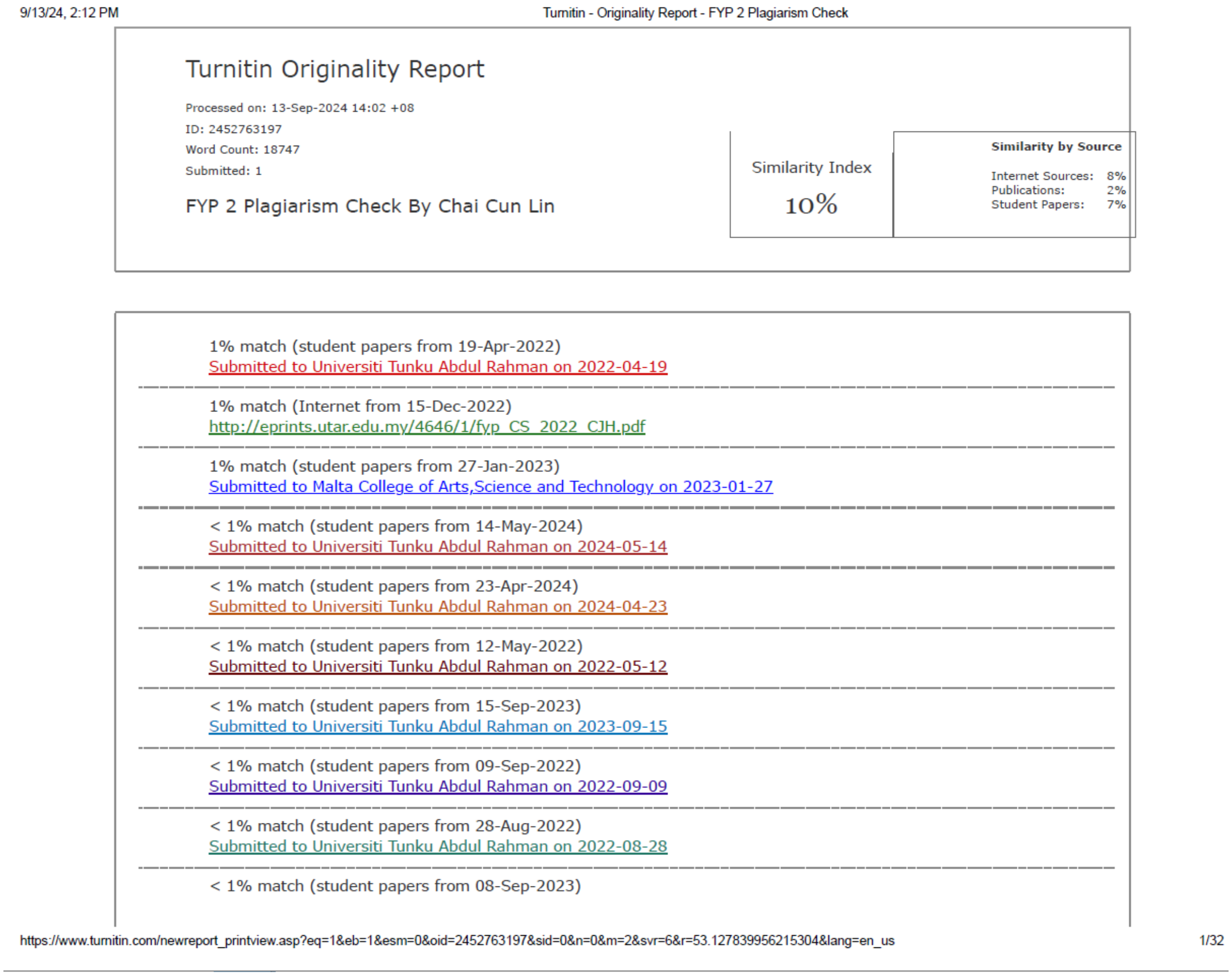
Rewards

Social Interaction

TimeBased Challenge

Points

PLAGIARISM CHECK RESULT



9/13/24, 2:12 PM Turnitin - Originality Report - FYP 2 Plagiarism Check

Turnitin Originality Report

Processed on: 13-Sep-2024 14:02 +08
ID: 2452763197
Word Count: 18747
Submitted: 1

FYP 2 Plagiarism Check By Chai Cun Lin

Similarity Index
10%

Similarity by Source

Internet Sources:	8%
Publications:	2%
Student Papers:	7%

1% match (student papers from 19-Apr-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-04-19

1% match (Internet from 15-Dec-2022)
http://eprints.utar.edu.my/4646/1/fyp_CS_2022_CJH.pdf

1% match (student papers from 27-Jan-2023)
Submitted to Malta College of Arts,Science and Technology on 2023-01-27

< 1% match (student papers from 14-May-2024)
Submitted to Universiti Tunku Abdul Rahman on 2024-05-14

< 1% match (student papers from 23-Apr-2024)
Submitted to Universiti Tunku Abdul Rahman on 2024-04-23

< 1% match (student papers from 12-May-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-05-12

< 1% match (student papers from 15-Sep-2023)
Submitted to Universiti Tunku Abdul Rahman on 2023-09-15

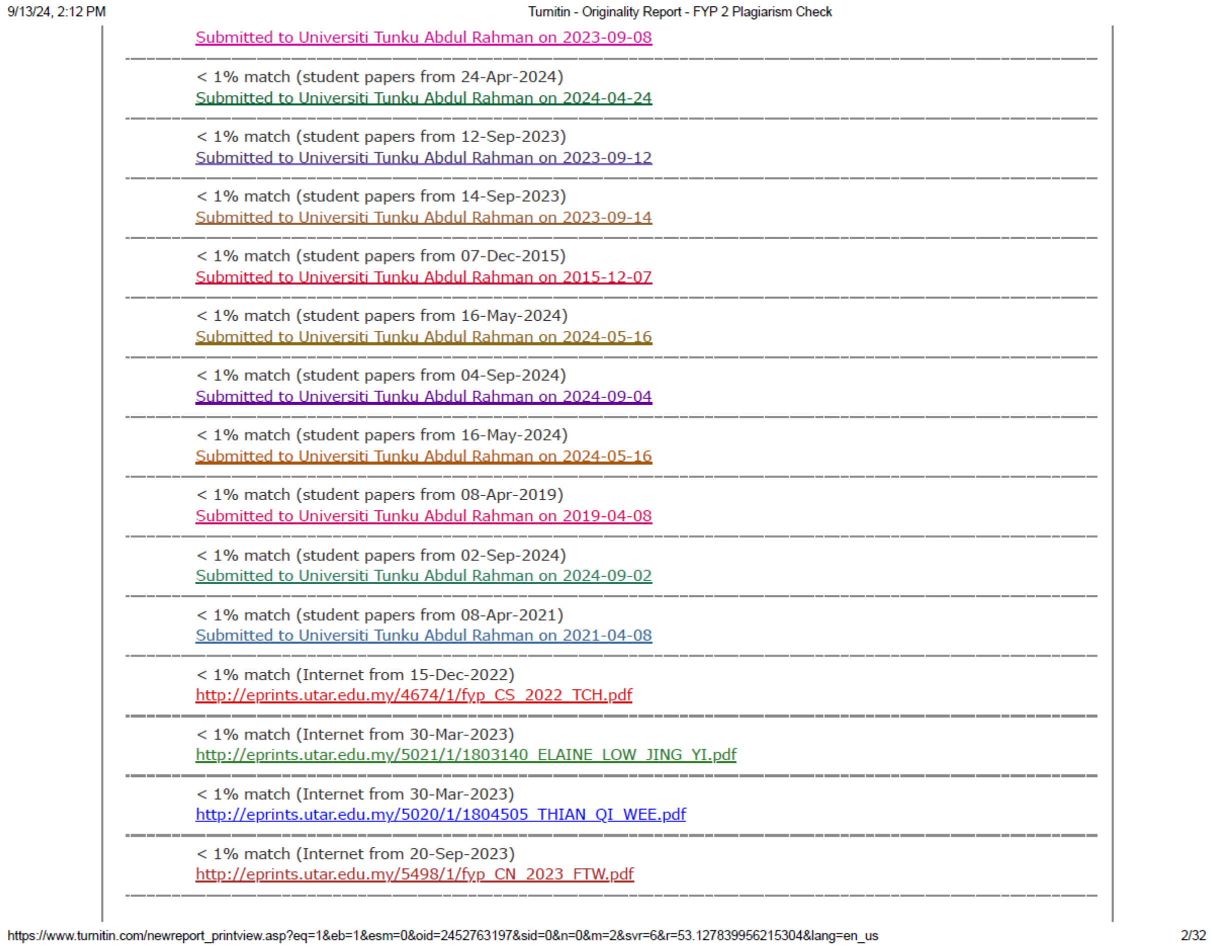
< 1% match (student papers from 09-Sep-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-09-09

< 1% match (student papers from 28-Aug-2022)
Submitted to Universiti Tunku Abdul Rahman on 2022-08-28

< 1% match (student papers from 08-Sep-2023)

https://www.turnitin.com/newreport_printview.asp?eq=1&eb=1&esm=0&oid=2452763197&sid=0&n=0&m=2&svr=6&r=53.127839956215304&lang=en_us 1/32

PLAGIARISM CHECK RESULT



9/13/24, 2:12 PM Turnitin - Originality Report - FYP 2 Plagiarism Check

Submitted to Universiti Tunku Abdul Rahman on 2023-09-08

< 1% match (student papers from 24-Apr-2024)
Submitted to Universiti Tunku Abdul Rahman on 2024-04-24

< 1% match (student papers from 12-Sep-2023)
Submitted to Universiti Tunku Abdul Rahman on 2023-09-12

< 1% match (student papers from 14-Sep-2023)
Submitted to Universiti Tunku Abdul Rahman on 2023-09-14

< 1% match (student papers from 07-Dec-2015)
Submitted to Universiti Tunku Abdul Rahman on 2015-12-07

< 1% match (student papers from 16-May-2024)
Submitted to Universiti Tunku Abdul Rahman on 2024-05-16

< 1% match (student papers from 04-Sep-2024)
Submitted to Universiti Tunku Abdul Rahman on 2024-09-04

< 1% match (student papers from 16-May-2024)
Submitted to Universiti Tunku Abdul Rahman on 2024-05-16

< 1% match (student papers from 08-Apr-2019)
Submitted to Universiti Tunku Abdul Rahman on 2019-04-08

< 1% match (student papers from 02-Sep-2024)
Submitted to Universiti Tunku Abdul Rahman on 2024-09-02

< 1% match (student papers from 08-Apr-2021)
Submitted to Universiti Tunku Abdul Rahman on 2021-04-08

< 1% match (Internet from 15-Dec-2022)
http://eprints.utar.edu.my/4674/1/fyp_CS_2022_TCH.pdf

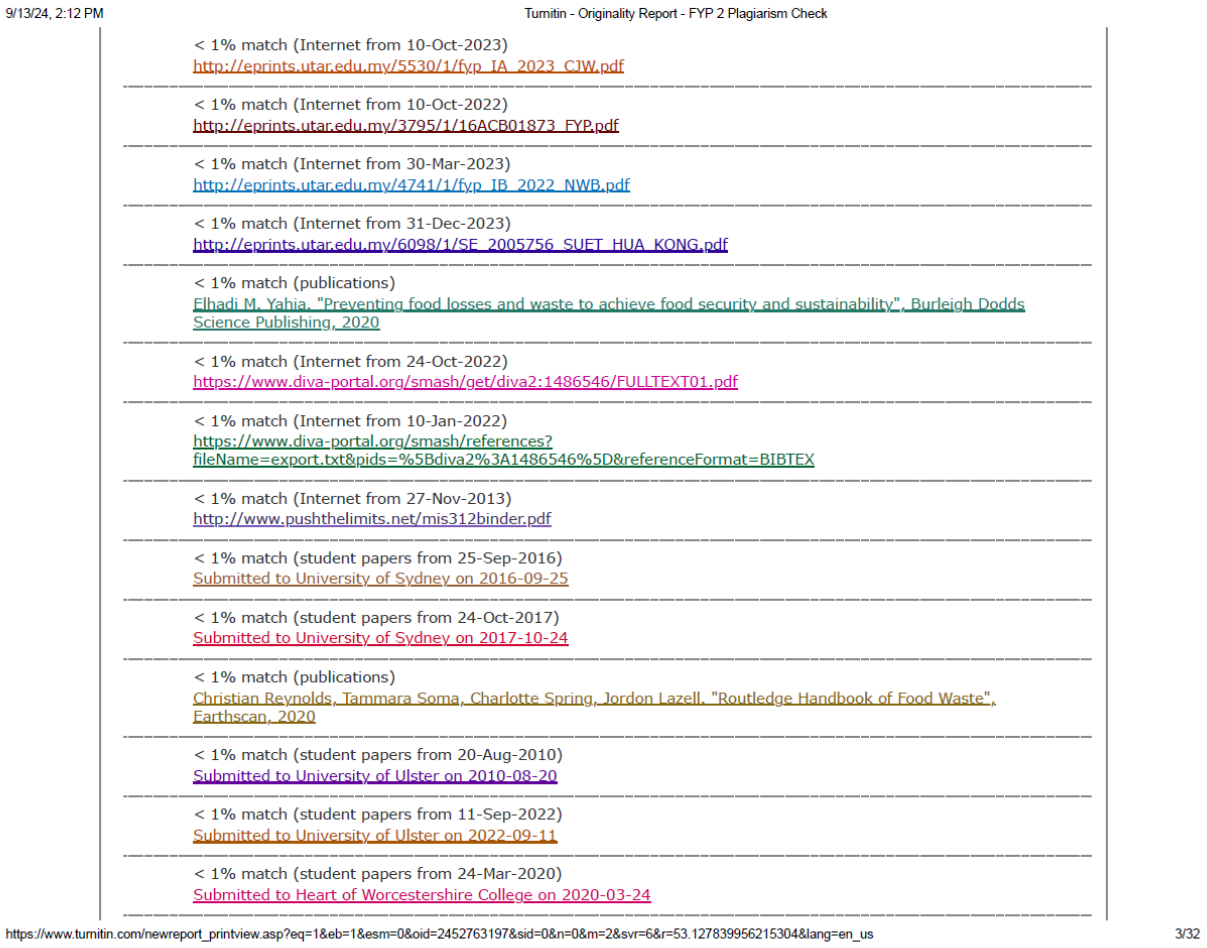
< 1% match (Internet from 30-Mar-2023)
http://eprints.utar.edu.my/5021/1/1803140_ELAINE_LOW_JING_YI.pdf

< 1% match (Internet from 30-Mar-2023)
http://eprints.utar.edu.my/5020/1/1804505_THIAN_QI_WEE.pdf

< 1% match (Internet from 20-Sep-2023)
http://eprints.utar.edu.my/5498/1/fyp_CN_2023_FTW.pdf

https://www.turnitin.com/newreport_printview.asp?eq=1&eb=1&esm=0&oid=2452763197&sid=0&n=0&m=2&svr=6&r=53.127839956215304&lang=en_us 2/32

PLAGIARISM CHECK RESULT



9/13/24, 2:12 PM Turnitin - Originality Report - FYP 2 Plagiarism Check

< 1% match (Internet from 10-Oct-2023)
http://eprints.utar.edu.my/5530/1/fyp_IA_2023_CJW.pdf

< 1% match (Internet from 10-Oct-2022)
http://eprints.utar.edu.my/3795/1/16ACB01873_FYP.pdf

< 1% match (Internet from 30-Mar-2023)
http://eprints.utar.edu.my/4741/1/fyp_IB_2022_NWB.pdf

< 1% match (Internet from 31-Dec-2023)
http://eprints.utar.edu.my/6098/1/SE_2005756_SUET_HUA_KONG.pdf

< 1% match (publications)
Elhadi M. Yahia. "Preventing food losses and waste to achieve food security and sustainability", Burleigh Dodds Science Publishing, 2020

< 1% match (Internet from 24-Oct-2022)
https://www.diva-portal.org/smash/get/diva2:1486546/FULLTEXT01.pdf

< 1% match (Internet from 10-Jan-2022)
https://www.diva-portal.org/smash/references?fileName=export.txt&pids=%5Bdiva2%3A1486546%5D&referenceFormat=BIBTEX

< 1% match (Internet from 27-Nov-2013)
http://www.pushthelimits.net/mis312binder.pdf

< 1% match (student papers from 25-Sep-2016)
Submitted to University of Sydney on 2016-09-25

< 1% match (student papers from 24-Oct-2017)
Submitted to University of Sydney on 2017-10-24

< 1% match (publications)
Christian Reynolds, Tammara Soma, Charlotte Spring, Jordon Lazell. "Routledge Handbook of Food Waste", Earthscan, 2020

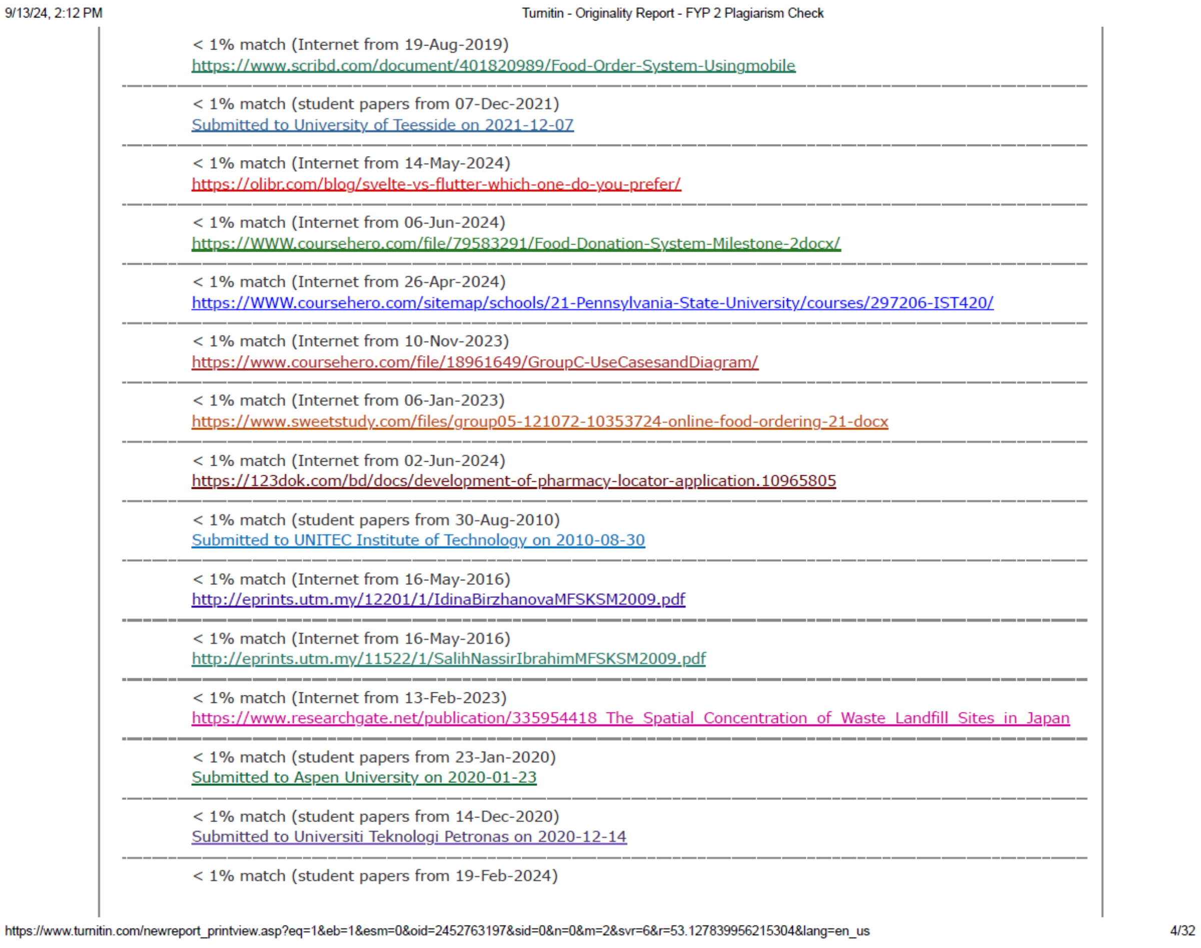
< 1% match (student papers from 20-Aug-2010)
Submitted to University of Ulster on 2010-08-20

< 1% match (student papers from 11-Sep-2022)
Submitted to University of Ulster on 2022-09-11

< 1% match (student papers from 24-Mar-2020)
Submitted to Heart of Worcestershire College on 2020-03-24

https://www.turnitin.com/newreport_printview.asp?eq=1&eb=1&esm=0&oid=2452763197&sid=0&n=0&m=2&svr=6&r=53.127839956215304&lang=en_us 3/32

PLAGIARISM CHECK RESULT



9/13/24, 2:12 PM Turnitin - Originality Report - FYP 2 Plagiarism Check

< 1% match (Internet from 19-Aug-2019)
https://www.scribd.com/document/401820989/Food-Order-System-Usingmobile

< 1% match (student papers from 07-Dec-2021)
Submitted to University of Teesside on 2021-12-07

< 1% match (Internet from 14-May-2024)
https://olibr.com/blog/svelte-vs-flutter-which-one-do-you-prefer/

< 1% match (Internet from 06-Jun-2024)
https://WWW.coursehero.com/file/79583291/Food-Donation-System-Milestone-2docx/

< 1% match (Internet from 26-Apr-2024)
https://WWW.coursehero.com/sitemap/schools/21-Pennsylvania-State-University/courses/297206-IST420/

< 1% match (Internet from 10-Nov-2023)
https://www.coursehero.com/file/18961649/GroupC-UseCasesandDiagram/

< 1% match (Internet from 06-Jan-2023)
https://www.sweetstudy.com/files/group05-121072-10353724-online-food-ordering-21-docx

< 1% match (Internet from 02-Jun-2024)
https://123dok.com/bd/docs/development-of-pharmacy-locator-application.10965805

< 1% match (student papers from 30-Aug-2010)
Submitted to UNITEC Institute of Technology on 2010-08-30

< 1% match (Internet from 16-May-2016)
http://eprints.utm.my/12201/1/IdinaBirzhanovaMFSKSM2009.pdf

< 1% match (Internet from 16-May-2016)
http://eprints.utm.my/11522/1/SalihNassirIbrahimMFSKSM2009.pdf

< 1% match (Internet from 13-Feb-2023)
https://www.researchgate.net/publication/335954418_The_Spatial_Concentration_of_Waste_Landfill_Sites_in_Japan

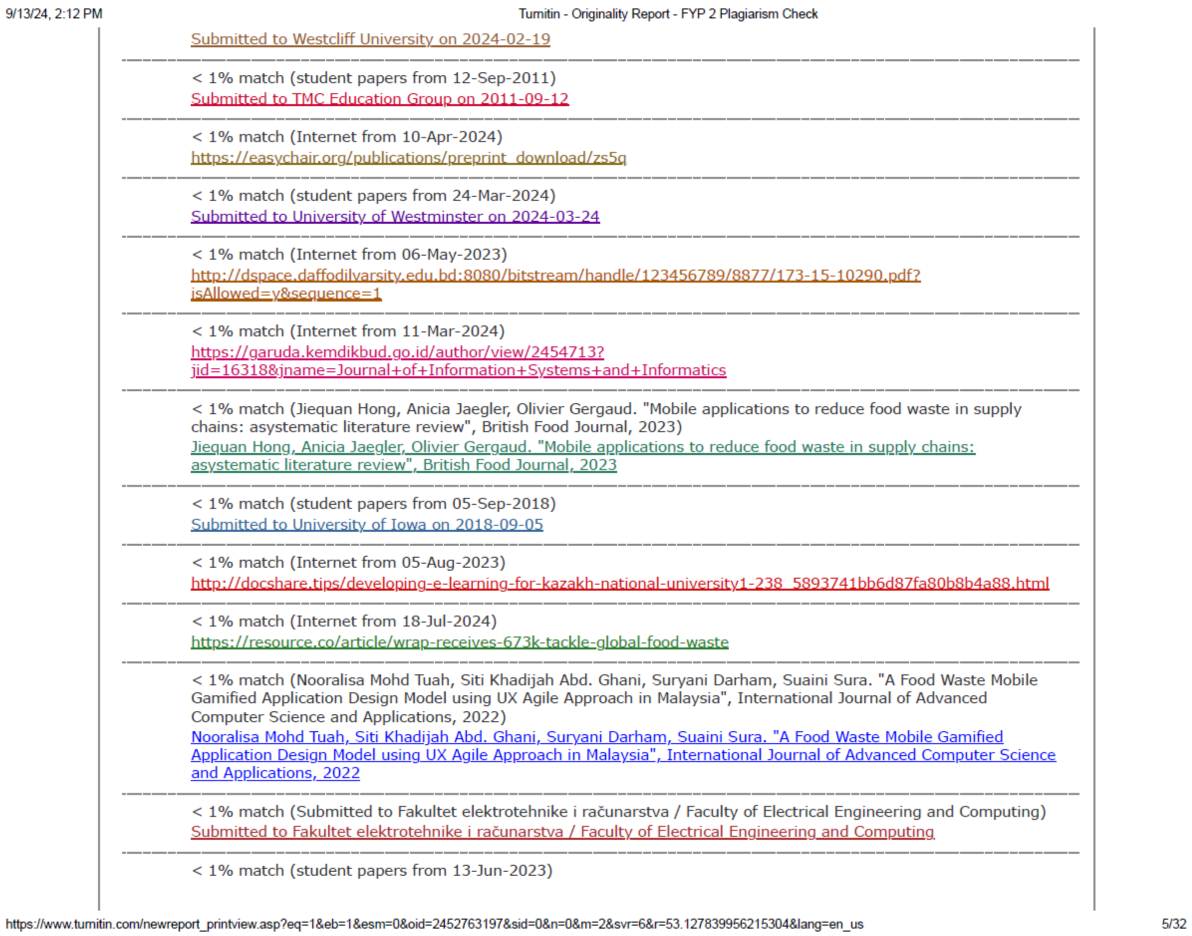
< 1% match (student papers from 23-Jan-2020)
Submitted to Aspen University on 2020-01-23

< 1% match (student papers from 14-Dec-2020)
Submitted to Universiti Teknologi Petronas on 2020-12-14

< 1% match (student papers from 19-Feb-2024)

https://www.turnitin.com/newreport_printview.asp?eq=1&eb=1&esm=0&oid=2452763197&sid=0&n=0&m=2&svr=6&r=53.127839956215304&lang=en_us 4/32

PLAGIARISM CHECK RESULT



9/13/24, 2:12 PM Turnitin - Originality Report - FYP 2 Plagiarism Check

Submitted to Westcliff University on 2024-02-19

< 1% match (student papers from 12-Sep-2011)
Submitted to TMC Education Group on 2011-09-12

< 1% match (Internet from 10-Apr-2024)
https://easychair.org/publications/preprint_download/zs5q

< 1% match (student papers from 24-Mar-2024)
Submitted to University of Westminster on 2024-03-24

< 1% match (Internet from 06-May-2023)
http://dspace.daffodilvarsity.edu.bd:8080/bitstream/handle/123456789/8877/173-15-10290.pdf?isAllowed=y&sequence=1

< 1% match (Internet from 11-Mar-2024)
https://garuda.kemdikbud.go.id/author/view/2454713?jid=16318&jname=Journal+of+Information+Systems+and+Informatics

< 1% match (Jiequan Hong, Anicia Jaegler, Olivier Gergaud. "Mobile applications to reduce food waste in supply chains: asystematic literature review", British Food Journal, 2023)
Jiequan Hong, Anicia Jaegler, Olivier Gergaud. "Mobile applications to reduce food waste in supply chains: asystematic literature review", British Food Journal, 2023

< 1% match (student papers from 05-Sep-2018)
Submitted to University of Iowa on 2018-09-05

< 1% match (Internet from 05-Aug-2023)
http://docshare.tips/developing-e-learning-for-kazakh-national-university1-238_5893741bb6d87fa80b8b4a88.html

< 1% match (Internet from 18-Jul-2024)
https://resource.co/article/wrap-receives-673k-tackle-global-food-waste

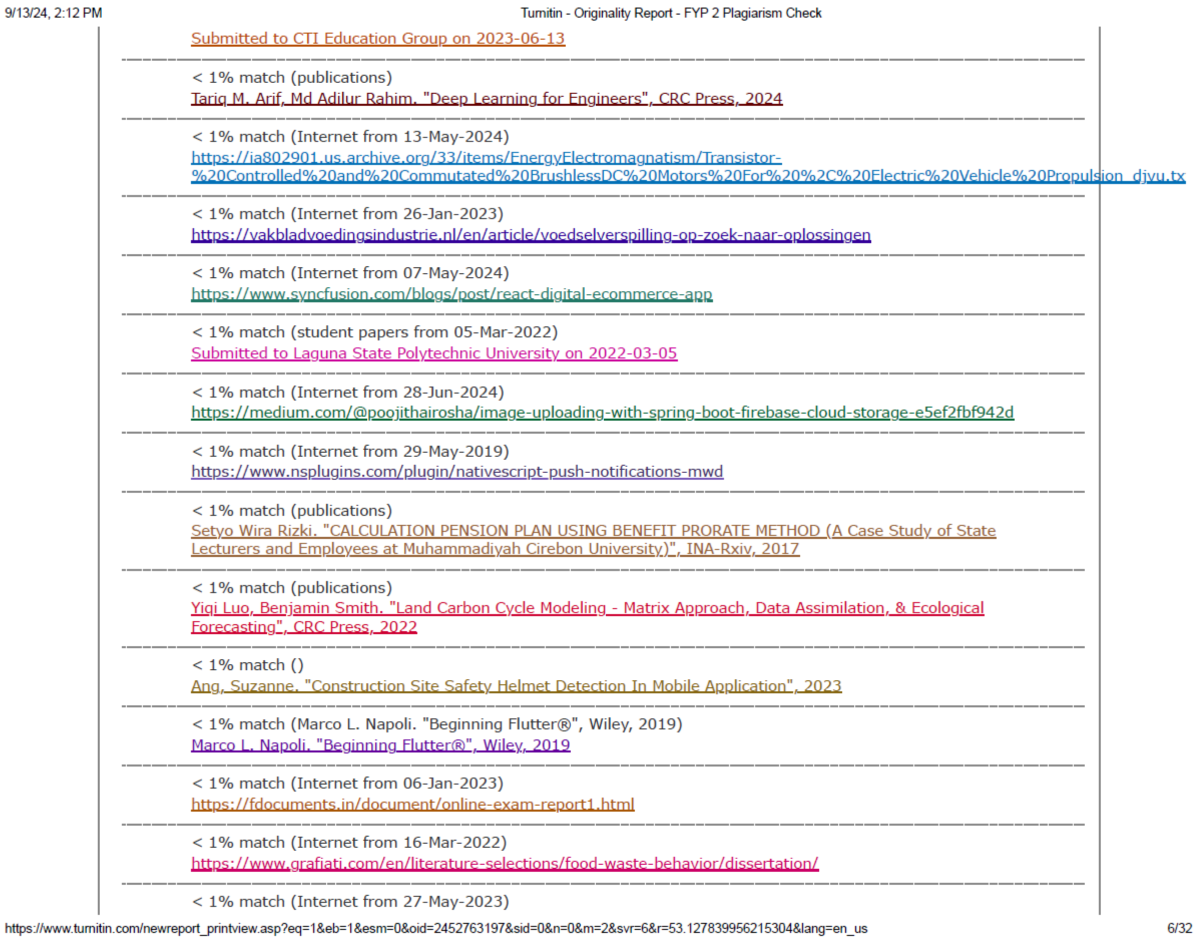
< 1% match (Nooralisa Mohd Tuah, Siti Khadijah Abd. Ghani, Suryani Darham, Suaini Sura. "A Food Waste Mobile Gamified Application Design Model using UX Agile Approach in Malaysia", International Journal of Advanced Computer Science and Applications, 2022)
Nooralisa Mohd Tuah, Siti Khadijah Abd. Ghani, Suryani Darham, Suaini Sura. "A Food Waste Mobile Gamified Application Design Model using UX Agile Approach in Malaysia", International Journal of Advanced Computer Science and Applications, 2022

< 1% match (Submitted to Fakultet elektrotehnike i računarstva / Faculty of Electrical Engineering and Computing)
Submitted to Fakultet elektrotehnike i računarstva / Faculty of Electrical Engineering and Computing

< 1% match (student papers from 13-Jun-2023)

https://www.turnitin.com/newreport_printview.asp?eq=1&eb=1&esm=0&oid=2452763197&sid=0&n=0&m=2&svr=6&r=53.127839956215304&lang=en_us 5/32

PLAGIARISM CHECK RESULT



9/13/24, 2:12 PM Turnitin - Originality Report - FYP 2 Plagiarism Check

Submitted to CTI Education Group on 2023-06-13

< 1% match (publications)
Tariq M. Arif, Md Adilur Rahim. "Deep Learning for Engineers", CRC Press, 2024

< 1% match (Internet from 13-May-2024)
https://ia802901.us.archive.org/33/items/EnergyElectromagnatism/Transistor-%20Controlled%20and%20Commutated%20BrushlessDC%20Motors%20For%20%2C%20Electric%20Vehicle%20Propulsion_djvu.tx

< 1% match (Internet from 26-Jan-2023)
https://vakbladvoedingsindustrie.nl/en/article/voedselverspilling-op-zoek-naar-oplossingen

< 1% match (Internet from 07-May-2024)
https://www.syncfusion.com/blogs/post/react-digital-ecommerce-app

< 1% match (student papers from 05-Mar-2022)
Submitted to Laguna State Polytechnic University on 2022-03-05

< 1% match (Internet from 28-Jun-2024)
https://medium.com/@poojithairosha/image-uploading-with-spring-boot-firebase-cloud-storage-e5ef2fbf942d

< 1% match (Internet from 29-May-2019)
https://www.nsplugins.com/plugin/nativescript-push-notifications-mwd

< 1% match (publications)
Setyo Wira Rizki. "CALCULATION PENSION PLAN USING BENEFIT PRORATE METHOD (A Case Study of State Lecturers and Employees at Muhammadiyah Cirebon University)", INA-Rxiv, 2017

< 1% match (publications)
Yiqi Luo, Benjamin Smith. "Land Carbon Cycle Modeling - Matrix Approach, Data Assimilation, & Ecological Forecasting", CRC Press, 2022

< 1% match ()
Ang, Suzanne. "Construction Site Safety Helmet Detection In Mobile Application", 2023

< 1% match (Marco L. Napoli. "Beginning Flutter®", Wiley, 2019)
Marco L. Napoli. "Beginning Flutter®", Wiley, 2019

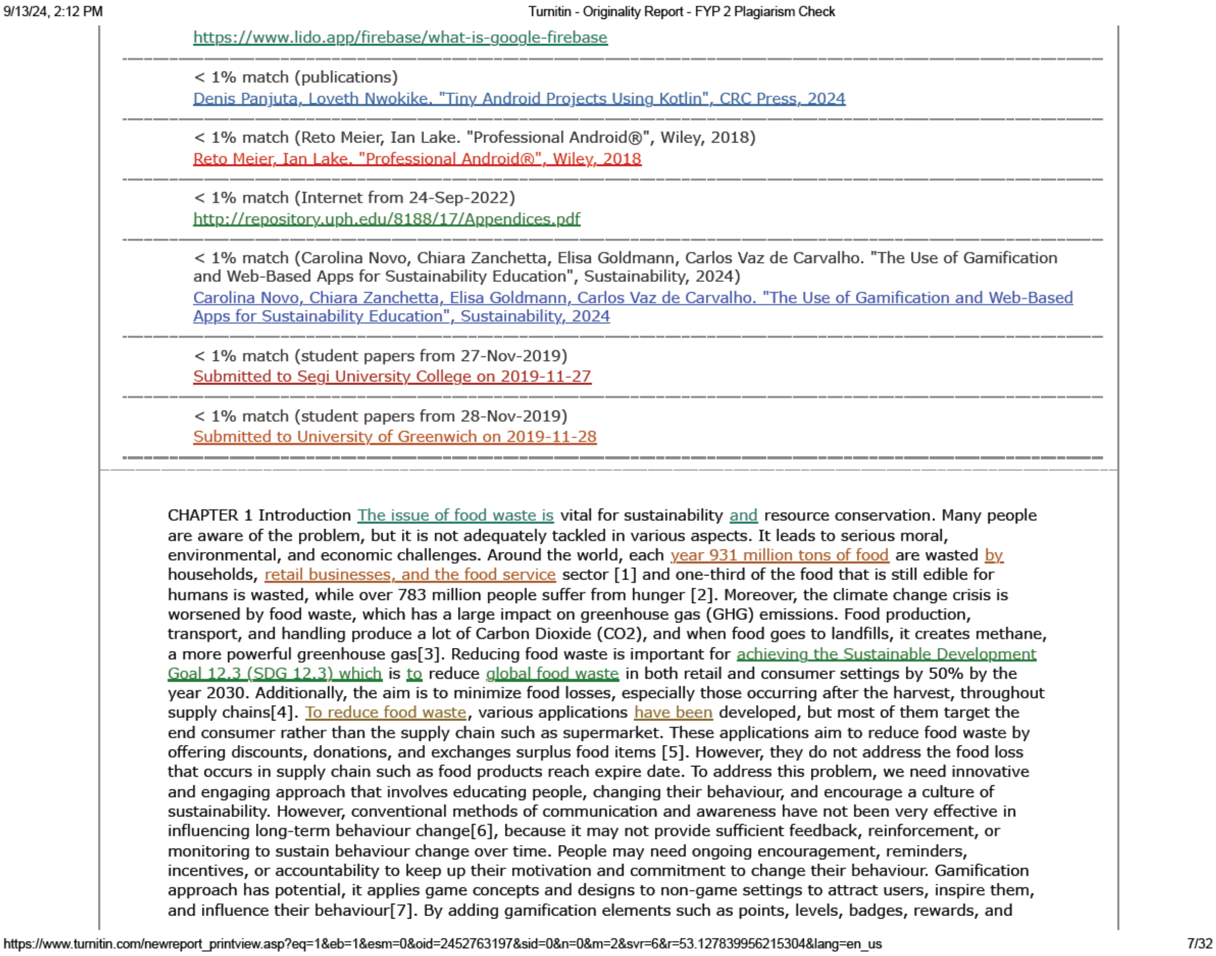
< 1% match (Internet from 06-Jan-2023)
https://fdocuments.in/document/online-exam-report1.html

< 1% match (Internet from 16-Mar-2022)
https://www.grafiati.com/en/literature-selections/food-waste-behavior/dissertation/

< 1% match (Internet from 27-May-2023)

https://www.turnitin.com/newreport_printview.asp?eq=1&eb=1&esm=0&oid=2452763197&sid=0&n=0&m=2&svr=6&r=53.127839956215304&lang=en_us 6/32

PLAGIARISM CHECK RESULT



9/13/24, 2:12 PM Turnitin - Originality Report - FYP 2 Plagiarism Check

https://www.lido.app/firebase/what-is-google-firebase

< 1% match (publications)
Denis Panjuta, Loveth Nwokike. "Tiny Android Projects Using Kotlin", CRC Press, 2024

< 1% match (Reto Meier, Ian Lake. "Professional Android®", Wiley, 2018)
Reto Meier, Ian Lake. "Professional Android®", Wiley, 2018

< 1% match (Internet from 24-Sep-2022)
http://repository.uph.edu/8188/17/Appendices.pdf

< 1% match (Carolina Novo, Chiara Zanchetta, Elisa Goldmann, Carlos Vaz de Carvalho. "The Use of Gamification and Web-Based Apps for Sustainability Education", Sustainability, 2024)
Carolina Novo, Chiara Zanchetta, Elisa Goldmann, Carlos Vaz de Carvalho. "The Use of Gamification and Web-Based Apps for Sustainability Education", Sustainability, 2024

< 1% match (student papers from 27-Nov-2019)
Submitted to Segi University College on 2019-11-27

< 1% match (student papers from 28-Nov-2019)
Submitted to University of Greenwich on 2019-11-28

CHAPTER 1 Introduction The issue of food waste is vital for sustainability and resource conservation. Many people are aware of the problem, but it is not adequately tackled in various aspects. It leads to serious moral, environmental, and economic challenges. Around the world, each year 931 million tons of food are wasted by households, retail businesses, and the food service sector [1] and one-third of the food that is still edible for humans is wasted, while over 783 million people suffer from hunger [2]. Moreover, the climate change crisis is worsened by food waste, which has a large impact on greenhouse gas (GHG) emissions. Food production, transport, and handling produce a lot of Carbon Dioxide (CO2), and when food goes to landfills, it creates methane, a more powerful greenhouse gas[3]. Reducing food waste is important for achieving the Sustainable Development Goal 12.3 (SDG 12.3) which is to reduce global food waste in both retail and consumer settings by 50% by the year 2030. Additionally, the aim is to minimize food losses, especially those occurring after the harvest, throughout supply chains[4]. To reduce food waste, various applications have been developed, but most of them target the end consumer rather than the supply chain such as supermarket. These applications aim to reduce food waste by offering discounts, donations, and exchanges surplus food items [5]. However, they do not address the food loss that occurs in supply chain such as food products reach expire date. To address this problem, we need innovative and engaging approach that involves educating people, changing their behaviour, and encourage a culture of sustainability. However, conventional methods of communication and awareness have not been very effective in influencing long-term behaviour change[6], because it may not provide sufficient feedback, reinforcement, or monitoring to sustain behaviour change over time. People may need ongoing encouragement, reminders, incentives, or accountability to keep up their motivation and commitment to change their behaviour. Gamification approach has potential, it applies game concepts and designs to non-game settings to attract users, inspire them, and influence their behaviour[7]. By adding gamification elements such as points, levels, badges, rewards, and

https://www.turnitin.com/newreport_printview.asp?eq=1&eb=1&esm=0&oid=2452763197&sid=0&n=0&m=2&svr=6&r=53.127839956215304&lang=en_us 7/32

PLAGIARISM CHECK RESULT

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

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Chai Cun Lin
ID Number(s)	20ACB03887
Programme / Course	Computer Science
Title of Final Year Project	Food Waste Mobile Application With Gamification

Similarity	**Supervisor's Comments** **(Compulsory if parameters of originality exceed the limits approved by UTAR)**
Overall similarity index: 10 **%** **Similarity by source** Internet Sources: 8 % Publications: 2 % Student Papers: 7 %	
Number of individual sources listed of more than 3% similarity: 0	

Parameters of originality required, and limits approved by UTAR are as Follows:

(i) Overall similarity index is 20% and below, and

(ii) Matching of individual sources listed must be less than 3% each, and

(iii) Matching texts in continuous block must not exceed 8 words

Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

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

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

Signature of Supervisor

Name: Tseu Kwan Lee

Date: 13/09/2024

Signature of Co-Supervisor

Name:

Date:

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB03887
Student Name	Chai Cun Lin
Supervisor Name	Tseu Kwan Lee

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

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

____

(Signature of Student)
Date: 13/09/2024

FYP 2 CHECKLIST