ASSIGNMENT GROUP WORK

Qualification	**Pearson BTEC Level 5 Higher National Diploma in Computing**		
Unit number and title	**Unit 22: Application Development**		
Submission date	16/12/2025	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Group number:	**Student names & codes**	**Final scores**	**Signatures**
	NGUYEN VIET PHUC BD00671		Phuc Viet
	DAO DUY VIEN BD00726		Duy Vien
	DO DUC AN BD00659		An
	HUYNH LE DUC THANG BD00339		Duc Thang
Class	SE07202	**Assessor name**	Mr DO TRUNG ANH

Plagiarism

Plagiarism is a particular form of cheating. Plagiarism must be avoided at all costs and students who break the rules, however innocently, may be penalised. It is your responsibility to ensure that you understand correct referencing practices. As a university level student, you are expected to use appropriate references throughout and keep carefully detailed notes of all your sources of materials for material you have used in your work, including any material downloaded from the Internet. Please consult the relevant unit lecturer or your course tutor if you need any further advice.

Student Declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I declare that the work submitted for assessment has been carried out without assistance other than that which is acceptable according to the rules of the specification. I certify I have clearly referenced any sources and any artificial intelligence (AI) tools used in the work. I understand that making a false declaration is a form of malpractice.

P1	P2	P3	P4	P5	P6	M1	M2	M3	M4	M5	D1	D2

OBSERVATION RECORD

Student	**Nguyen Viet Phuc – BD00671**

Description of activity undertaken

M1: Analyse a business- related problem using appropriate methods to produce a well-structured software design document.

D1: Evaluate the solution to a business-related problem and the preferred software development methodology by comparing the various software development tools and techniques researched.

M4: Develop a functional business application based on a specific software design document, with supportive evidence of using the preferred tools, techniques and methodologies.

D2: Justify improvements to the business application system made because of feedback and also feedback which was not acted upon, including opportunities for improvement and further development.

Assessment & grading criteria

How the activity meets the requirements of the criteria

Student signature:	**PHUC**	**Date:**	**16/12/2025**
Assessor signature:		**Date:**	
Assessor name:			

❒ Summative Feedback:	❒ Resubmission Feedback:

Grade:	Assessor Signature:	Date:

Internal Verifier's Comments:

Signature & Date:

TABLE OF CONTENT

LIST OF FIGURE

LIST OF TABLE

INTRODUCTION

In the rapid development of mobile technology, software applications have become integral tools for solving real-world problems. This report represents the final submission for Unit 22: Application Development, documenting the complete development lifecycle of the "CampusExpense Manager" application. Developed by the BudgetWise Solutions team, this project aims to provide a comprehensive financial management tool specifically designed to help university students track their expenses and manage their budgets effectively.

This report details the transition from the design phase to the actual implementation of the functional application (P5). It provides a critical evaluation of the application's performance against the initial problem definition and system requirements established in the previous assignment (P6). Furthermore, it analyzes user feedback gathered through peer reviews to justify the improvements made and identify opportunities for future development (M3, D2).

The completion of this assignment and the "CampusExpense Manager" project would not have been possible without the guidance and support we received throughout the course.

We would like to express our deepest gratitude to our lecturer and assessor, Mr. Do Trung Anh. His expertise, dedicated instruction, and constructive feedback have been invaluable to our team. Mr. Do Trung Anh provided us with essential knowledge regarding software development methodologies and tools, guiding us through the complexities of the project. His support not only helped us meet the academic requirements of the unit but also enhanced our practical skills in mobile application development.We truly appreciate the time and effort he has invested in our learning journey.

1.Produce a well-defined problem definition statement, supported by a set of user and system requirements for a business problem. (P1)

1.1. Introduction Project Background

For now this project will be focused on developing a mobile application called "CampusExpense Manager". This is developed by BudgetWise Solutions, a small development team.

The main goal of the project is to build an easy-to-use mobile application that will help university students effectively manage their personal expenses and also maintain their budget. Each of these applications aims to simplify the process of tracking expenses, as well as help students make informed financial decisions and ensure financial stability throughout their studies.

1.2. Problem statement

Problem: Many college and university students face significant challenges in managing their personal finances. Due to lack of experience, unstable income sources, and many additional expenses (such as tuition, rent, food, transportation), they often have difficulty tracking their expenses, creating budgets, and avoiding overspending. This can lead to financial stress, negatively affecting their academic performance and quality of life.

Solution: With the mobile application "CampusExpense Manager" proposed as one of the comprehensive solutions to solve the above problem. By providing with each intuitive tools to track spending, set up budgets and generate financial reports, the app will help students take better control of their finances, thereby building responsible spending habits and achieving financial independence.

❖ **Difficulties BudgetWise Solutions might encounter**

With these projects in the making, BudgetWise Solutions – as one of the smaller start-up development teams – faced several significant challenges that could impact the progress and quality of the product:

- ✓ **Technical experience limitations:** The development team consists of junior programmers with limited in-depth experience in cross-platform mobile application development. Familiarization with well-known technologies like Android Studio, Java, and Firebase integration may take significantly longer than expected.
- ✓ **Time and deadline pressures:** This project must be completed within a tight, fixed timeframe (12 weeks). Any delays in requirements analysis or design could lead to missed deadlines for core features.
- ✓ **Data security risks:** Because these applications are often related to personal financial management, ensuring user data security and compliance with privacy regulations is a significant challenge for a team lacking cybersecurity expertise.

❖ **Issues need to be addressed**

- ✓ Based on the highly realistic context of university students, these projects will need to thoroughly address the following pressing issues:
- ✓ Lack of centralized management tools: Currently, students often manage their money manually (notebooks), haphazardly (using Excel), or rely on memory. This leads to errors and a lack of overall financial clarity.
- ✓ Uncontrolled spending habits: Due to their lack of experience and unstable income, students can easily overspend at the beginning of the month, resulting in deficits at the end. The core issue is the lack of immediate warning when spending exceeds limits.
- ✓ Financial stress: The inability to track cash flow causes anxiety, directly impacting academic performance and quality of life.

❖ **Possible Business Application Solutions**

To address these issues, the team considered different application types before making a final decision:

- **Desktop Application:**
  - ✓ Advantages: Offers high performance, and with a large screen, it's easy to view detailed reports.
  - ✓ Disadvantages: Lacks the flexibility of a portable computer, making it unsuitable for taking notes while shopping (e.g., buying food, taking the bus). This can easily lead to forgetting to take notes.
- **Web Application:**
  - ✓ Advantages: Requires no installation and can run on many devices.
  - ✓ Disadvantages: Requires a complete internet connection. The user experience on mobile browsers is often less smooth than with native apps and has limitations in sending push notifications to remind users of expenses.
- **Mobile Application - The chosen solution:**
  - ✓ Reason: These are the most optimized solutions for students. Smartphones are indispensable, allowing users to track expenses anytime, anywhere (even offline). The ability to send real-time notifications about budget limits is a key factor in changing spending behavior.

1.3.Requirements analysis

1.3.1.Functional Requirements

These are the specific functions that the system must perform to meet the needs of each user:

- **User Registration and Authentication:** Users will be able to create accounts with usernames and passwords. The system must ensure secure authentication for users to log in and access their spending data.
- **Expense Tracking:** Allow users to add, edit, and categorize expenses (e.g., rent, food, transportation). Each expense item will need a description, date, amount, and specific category.
- **Budget Setting:** Each user can set a monthly budget for different spending categories (e.g., from food, entertainment, education) and adjust the budget as needed.
- **Spending Overview:** The app should provide a monthly spending summary screen, including total spending, remaining budget, and a breakdown by category. Users should also be able to see spending trends over time.
- **Recurring Expenses:** Allow users to add recurring monthly expenses (e.g., rent) with start and end dates. The app should then automatically add these to the monthly budget.
- **Spending Reports:** Users should be able to create detailed spending reports for specific time periods (e.g., monthly, yearly). The reports should show detailed spending by category.
- **Spending Notifications:** The system should send notifications or reminders when users are about to reach or exceed their budget limit for a specific category.

1.3.2.Non-Function Requirements

These are the criteria for each quality and how each system works:

- **Performance**: The application must run smoothly and be responsive, even when processing large amounts of spending data.
- **User-Friendly Interface:** The user interface (UI) must be intuitive, easy to understand with clear labels and simple navigation to make tracking spending easy.
- **Platform Compatibility**: It can be developed for both Android and iOS to reach as many users as possible.
- **Data Security:** User data, including spending and budget information, must be stored securely and protected using encryption methods. Each data privacy regulation must be strictly followed.
- **Feedback and Support:** The app will need to incorporate a feedback form for users to report issues or make suggestions.

- **Monetization (Optional):** Individual monetization features (such as in-app ads or premium features) may be considered in future updates, but the initial release will prioritize core functionality.

1.4. Stackeholders

With each stakeholder being an individual or organization that influences or is influenced by the project.

Stakeholder	Role in the Project	Expectations / Goals from the Project
Students	Main users of the application.	A user-friendly, stable, and secure application to track expenses, manage budgets, and improve their financial habits.
BudgetWise Solutions Team	Development team responsible for designing, programming, testing, and maintaining the app.	Complete the project on time and within budget. Build a successful product that receives positive feedback and gain valuable experience in mobile app development.
University Management	Potential partner who can support app promotion.	A useful tool to encourage students to practice responsible financial behavior, contributing to their well-being and overall quality of life at the university.

Table 1.1 Project Stakeholders and Their Expectations

2.Review areas of risk related to the successful development of a proposed application. (P2)

In any software development project, risk analysis and management plays a fundamental role in ensuring success. For the "CampusExpense Manager" project, this role will become increasingly important due to the constraints and specific challenges that have been identified. A proactive risk management plan will help the BudgetWise Solutions team anticipate difficulties and come up with timely and effective responses.

- **Technical Risks**
  - ✓ The biggest and most pervasive risk to the project comes from the technical capabilities of the team. According to the investigation, BudgetWise Solutions is a small team with “limited experience in mobile application development” and the team consists of “junior programmers with moderate expertise”. This limitation creates a significant challenge as the project requires the development of compatible applications on both “Android and iOS platforms”, a task that requires significant effort and experience. These risks can all arise, including low code quality, many errors, difficulty in integrating features and unstable application performance.

- ✓ To mitigate each of these risks, management strategies should focus on improving the capacity of each team. As the question suggests, “training and skills development will be essential throughout the project”. Alternatively, the team could consider adopting cross-platform development frameworks such as React Native or Flutter. These solutions would allow for a single codebase for both operating systems, saving time, reducing complexity, and making it suitable for teams with limited resources. It would also be effective to invite a specialist or senior developer as a part-time consultant to guide the architecture and review the code.

- **Budget and Time Risks**
  - ✓ They are closely related and are two of the most severe constraints of the project. Each project must be completed within a strict timeframe of 12 weeks with one of the tight budgets. Any delays arising from technical issues or changes in requirements can lead to missed deadlines, and also increase costs, thereby creating the risk of over-budgeting.
  - ✓ To deal with these risks, we will be applying a flexible project management method such as Agile/Scrum. By being able to divide the project into short stages (sprints), the team will regularly monitor progress, detect problems early and flexibly adjust the plan. In terms of budget, a detailed spending plan should be developed from the beginning, and a contingency fund (around 10-15%) should be set aside for unexpected costs. Prioritizing the development of core functions in a Minimum Viable Product (MVP) model will help ensure the product is launched on time and within budget, with additional features being developed in later versions.
- **Security and Legal Risks**
  - ✓ Since the applications handle personal financial data of individual users, the security risks and hence the legal compliance are not to be taken lightly. The assignment requires that "user data... must be securely stored and protected by encryption" and the project "must comply with data privacy regulations". Any lapse in information security can lead to serious consequences such as data loss, loss of user trust and legal issues.
  - ✓ The management strategy for this risk is to integrate security measures from the early stages of the development process. The team should research and apply common security standards, such as data encryption during transmission and storage. Before launch, conducting security testing (penetration testing) is necessary to detect and fix potential vulnerabilities.
- **Product Quality Risks**

- ✓ Finally, each of these time pressures and limited team experience can lead to product quality risks. An application that has extremely poor performance, an interface that is difficult to use, or is full of bugs will not be able to retain users. The assignment emphasizes the need for the application to have "smooth and responsive performance" and an "intuitive user interface".
- ✓ To ensure that each quality is addressed, it is imperative to develop a comprehensive Test Plan. Each of these plans should include multiple levels, from unit testing to integration testing and user acceptance testing (UAT). In particular, it is wise to have a targeted group of students try out a beta version and collect feedback to improve the product before a wider release.

3.Research the use of software development tools and techniques for the development of a proposed application. (P3)

The choice of the development methodology (SDLC model) and the toolset are extremely suitable factors that determine the efficiency, quality and success of every "CampusExpense Manager" project. A good methodology will also provide a clear structure and workflow, while the right tools will help optimize productivity and product quality.

Research on Software Development Models (SDLC Models)

One of the Software Development Life Cycle (SDLC) models is a structured process that development teams use to design, develop, and test high-quality software. Here are three popular models.

- **Waterfall model**

This is one of the oldest and most traditional models, operating in a sequential, linear process. Each stage must be completed completely before it can be moved to the next stage, like a waterfall.

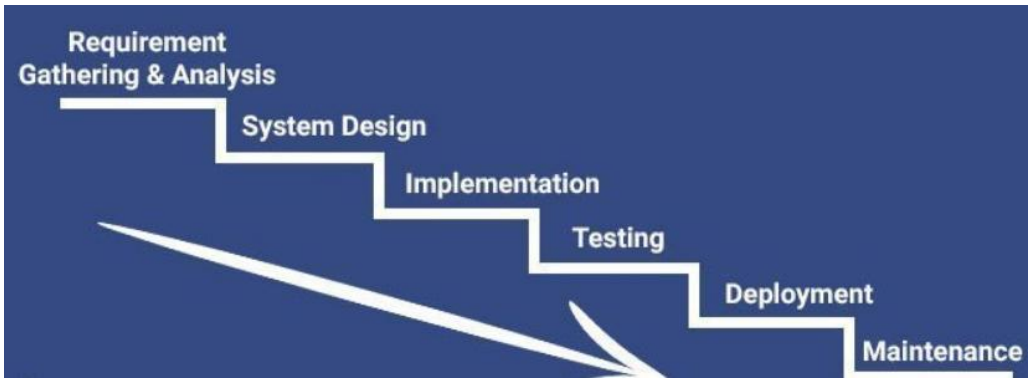


Figure 3.1 Waterfall model

Stages: This process will include successive steps such as: Requirements analysis → System design → Implementation → Testing → Deployment → Maintenance.

Advantages: This model is very simple, easy to understand and easy to manage. From here, it is extremely suitable for each small project with clearly defined, complete and almost unchanged requirements from the beginning.

Disadvantages: The biggest disadvantage of Waterfall is its lack of flexibility. Any changes in the requirements are difficult and also very expensive to implement. Testing only takes place at the end, making it more complicated and expensive to find and fix bugs. This model was considered unsuitable for the "CampusExpense Manager" project because the requirements may need to be adjusted after receiving feedback from student users and the less experienced team needed more flexibility.

- **Iterative Model**

This model will approach the project by starting with a small set of requirements and developing increasingly more complete versions (iterations) of every software. In each of these loops, it will go through the development stages and create a working version of the product.

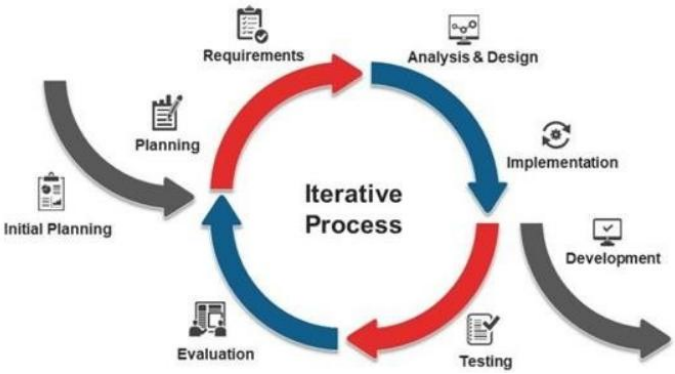


Figure 3.2 Iterative process

Phases: Each loop is a small cycle including: Planning → Design → Programming → Testing. At the end of each loop, a part of the product is handed over and feedback is received to improve the next loop.

Advantages: This will allow creating a working version of the product very early. From there, all risks are managed much better because problems can be detected and resolved in each small loop.

Disadvantages: Need to have a clear scope management and planning for each iteration to avoid "scope creep" (project growing out of control).

- **Agile Model**

Agile is not just one of the models but a philosophy of software development, emphasizing flexibility, collaboration, and continuous feedback. Agile is an iterative and incremental development model, in which solutions are developed through collaboration between autonomous and cross-functional teams. Scrum is one of the most popular frameworks of Agile.

Figure 3.3 Agile model

Phases/Principles: These projects are divided into short cycles called Sprints (usually 1-4 weeks). Each Sprint delivers a working part of the product. These teams work closely with stakeholders to receive feedback and adjust direction very quickly.

Advantages: Highly flexible, easily adaptable to changes in requirements. Customers/users are involved in the development process, ensuring the final product meets their needs. Agile is particularly well suited to the CampusExpense Manager project as it allows the junior team to learn and adapt continuously, and can quickly launch a minimum viable product (MVP) to gather feedback from students.

Disadvantages: Requires constant engagement and communication from all team members. Difficult to accurately predict the total time and cost of the entire project from the beginning.

Research Development Tools

Our careful selection of development tools was a crucial step in realizing the "CampusExpense Manager" application. Below is a comparison of popular technologies on the market and the team's choice for this project.

❖ Integrated Development Environment (IDE)

Each development environment (IDE) being developed provides the necessary tools for writing code, debugging, and testing software.

Android Studio: is currently Google's official IDE for Android application development, built upon IntelliJ IDEA. It offers intelligent code editors, a powerful emulator, and advanced performance analysis tools.

Figure 3.4 Android studio

Eclipse: Before Android Studio, Eclipse was the most popular IDE for Android. However, in today's reality, it has become outdated for modern mobile application development, with poorer Gradle support and a lack of the latest features from Google.

Figure 3.5 Eclipse

IntelliJ IDEA: is a powerful IDE from JetBrains, supporting many programming languages. Although Android Studio is based on IntelliJ, the original IntelliJ IDEA requires the installation of many complex plugins to provide the same level of Android support as Android Studio.

Figure 3.6 Intellij

⇨ **The tool I chose was: Android Studio**

- **Programming Language:** like this are platform-specific, designed to build applications with different logic and functionalities.

  Java: A long-established, stable object-oriented programming language with a huge support community. Java is the native language of Android, has a rich library, and is very suitable for learning programming on this platform.

Figure 3.7 Java

Kotlin: A modern language favored by Google for Android application development. Kotlin is more concise and safer than Java (minimizing NullPointer errors); however, its new syntax can be difficult for beginners familiar with traditional C/C++ or Java.

Figure 3.8 Kotlin

Python (with Kivy/BeeWare): Python is known for its simple syntax, but it is not a natural language for Android. Developing Android applications with Python often encounters performance and compatibility issues compared to Java/Kotlin.

Figure 3.9 Python

⇨ **The tool we will be able to choose is Java.**

❖ **Database Management System**

Each database management system stores and retrieves user information and financial transactions.

Firebase (Realtime Database) is Google's mobile application development platform (Backend-as-a-Service). It provides cloud-hosted NoSQL databases, supports real-time data synchronization between devices, and includes built-in user authentication.

Figure 3.10 Firebase

SQLite is a compact relational database management system (RDBMS) built into Android devices. SQLite works very well offline but does not support cloud synchronization by default; a custom backend system needs to be built if synchronization is desired.

Figure 3.11 SQLLite

MySQL is the most popular open-source relational database management system for web servers. To use MySQL for mobile applications, a complex middleware API (Web API) is needed for the application to communicate with the server.

Figure 3.12 Mysql

- ⇨ **The tool we were chosen for: Firebase**

❖ **Source Code Management (SCM)**

Tools for managing source code help track changes and support teamwork.

GitHub: The world's most popular Git-based source code hosting platform. GitHub offers an intuitive interface, project management features (Issues, Projects), and a strong open-source community.

Figure 3.13 Gihub

GitLab: Similar to GitHub but stands out with its powerful built-in integration/continuous computing (CI/CD) features. Often preferred by large enterprises for self-deploying DevOps systems.

Figure 3.14 GitLab

Bitbucket: A product of Atlassian, deeply integrated with Jira and Trello. Bitbucket is often chosen by companies using the Atlassian ecosystem for internal projects (private repositories).

Figure 3.15 Bitbucket

⇨ **The tool we were chosen for: GitHub**

❖ **Diagram Design Tools**

Each diagram design tool helps to create a system model (UML) before programming begins.

Draw.io (Diagrams.net) is a free, no-registration online diagramming tool. It fully supports UML libraries, stores files directly on Google Drive, and has a simple drag-and-drop interface.

Figure 3.16 Draw.io

Lucidchart is a professional web-based diagramming tool with many beautiful templates and powerful real-time collaboration features. However, the free version has a limited number of drawable elements.

Figure 3.17 Luicdchart

Microsoft Visio is the industry-standard technical diagramming software for Windows. Visio is powerful but expensive and does not offer the same flexible online collaboration capabilities as web-based tools.

Figure 3.18 Microsoft visio

⇨ **Our chosen tool: Draw.io**

❖ **Project Management Tools**

The tool is designed for specific tasks, such as project management, which helps track progress and allocate work within the team.

Trello: An intuitive Kanban-based task management tool. Trello uses boards, lists, and cards to describe workflows, making it easy to use for small teams and short-term projects.

Figure 3.19 Trello

Jira Software: A sophisticated project management tool for Agile/Scrum software development teams. Jira is powerful in reporting and tracking bugs, but its interface is quite complex and time-consuming to set up for beginners.

Figure 3.20 Jira software

Asana: A task management tool focused on to-do lists and progress. Asana has a beautiful interface, but advanced features are often only available in paid plans.

Figure 3.21 Asana

⇨ **The tool we were chosen to use: Trello**

4.Analyse a business- related problem using appropriate methods to produce a well-structured software design document. (M1)

4.1.Introduction to System Analysis

System analysis is one of the important stages in the software development life cycle (SDLC), where it helps each developer and stakeholders to understand what the system needs to achieve and how it should operate.

- The goal of each system analysis is to convert the business requirements into structured models that can be used for design and also from here we will be deployed the system. With the use of each approach there are systems such as Unified Modeling Language (UML), for each analyst can visualize, specify and from here also be recorded by the main functional and structural aspects of the system.
- In the CampusExpense Manager project, the role of system analysis will be important in converting the extremely practical student financial management problem to one of the solutions with structured software. With these mobile applications, developed by BudgetWise Solutions, it is designed to help university students manage their expenses, as well as track their spending patterns and maintain control of their budgets in a very efficient way.

- With each analysis focused on understanding the students as the main users, it will interact with the system, as well as the systems that will process, store and display financial information further. In terms of UML diagrams such as Use Case diagrams, Class Diagrams and Entity Relationship Diagrams (ERD) are used to represent these components in a standardized and visual way.

By performing the analysis by these structures, the team will ensure that the design of CampusExpense Manager is both technically sound and user-centric. It enables early detection of each logical issue, ensures consistency between business objectives and software functionality, and provides a solid foundation for later phases such as system design, implementation, and testing.

⇨ Next, I will present illustrative use cases for each functional interaction between users and systems.

4.2.Use Case Diagram

Use Case Diagram is one of the most basic UML diagrams that can be used in each phase of system analysis. This diagram shows the interactions between actors (either for the entire user system or the entire external system) and the entire functionality (use case) that will be provided by the system. The whole purpose of use case diagrams is to identify and describe what the system should do from the user's perspective, without focusing on each task that can be implemented. The diagram provides an overview of the scope, functional requirements and user roles, making it easy for both developers and stakeholders to understand the behavior of the system.

❖ **CampusExpense Manager Use Case Diagram Overview**

Currently for the CampusExpense Manager project, the Use Case Diagram is describing how the three main actors interact with the systems:

- Students (Users): These main users will typically manage their individual expenses, as well as set up budgets, view reports and receive spending notifications.
- System (Application): Responsible for automated processes such as generating reports and sending notifications when spending exceeds budget limits.
- Administration/Support Team: Handles user feedback and provides technical support to maintain system performance and reliability.

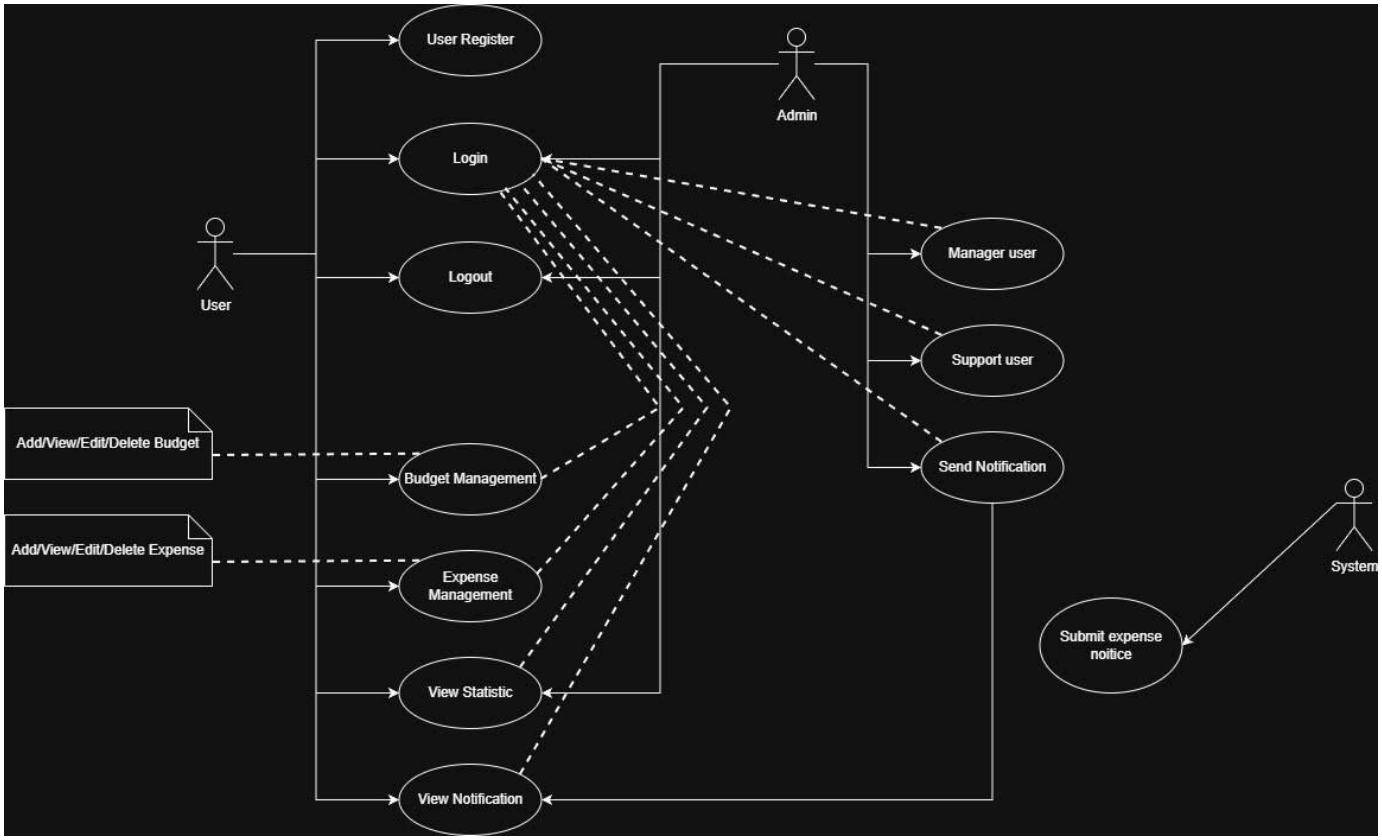


Figure 4.1 Use Case Diagram – CampusExpense Manager

This Use Case Diagram will provide a comprehensive overview of the main functionality of the systems and the relationships between the different modules.

By clearly defining the boundaries and user interactions, this analysis ensures that CampusExpense Manager is designed with the needs of each user in mind. It helps the development team:

- Identify each key user requirement early in the design phase.
- Clarify the responsibilities of each system and the role of automation.
- Set the stage for the entire platform for subsequent diagrams, including Class diagrams and ERD diagrams.

In short, the Use Case Diagram acts as a blueprint to understand how users and system components interact, ensuring the final product remains functional and user-centric.

4.3.Use Case Specifications

The Use Case Specification is provided with a detailed written description of each major functionality that will be identified in the Use Case Diagrams. The entire specification will be defined with the user interactions for each system so that we can achieve one of the specific goals, including with the prerequisites, main flow of events and possible exceptions.

Below are the detailed specifications for the major use cases of the CampusExpense Manager system.

- **Login Use Case Specification**

Use Case Name	Login
Actor	Student
Description	Allows a registered user to access the system by entering valid login credentials.
Preconditions	The user must have successfully registered an account.
Main Flow	- The user opens the app. - The user enters their username and password. - The system validates the credentials. - If valid, the user is redirected to the main dashboard.
Alternate Flow	If invalid credentials are entered, the system displays an error message and prompts the user to re-enter information.
Postconditions	The user successfully logs into the system and can access personal expense data.

Table 4.1 ***Use case login***

- **Add Expense Use Case Specification**

Use Case Name	Add Expense
Actor	Student
Description	Enables users to record a new expense by entering relevant information such as amount, category, and description.
Preconditions	The user must be logged in to the system.
Main Flow	- The user selects “Add Expense”. - The system prompts for amount, category, date, and description. - The user fills in the details and submits the form. - The system validates data and stores it in the database.
Alternate Flow	If validation fails (e.g., missing or invalid data), the system prompts the user to correct it.
Postconditions	A new expense record is added and displayed in the user’s expense list.

Table 4.2 ***Add Expense Use Case***

- **Manage Budget Use Case Specification**

Use Case Name	Manage Budget
Actor	Student
Description	Allows users to set and adjust their monthly budgets for specific spending categories.
Preconditions	The user must be logged in.

Main Flow	- The user selects “Manage Budget”. - The user sets a budget limit for each category. - The system validates data and saves the budget. - When the spending approaches or exceeds the limit, the system sends notifications.
Alternate Flow	If invalid values (e.g., negative amount) are entered, the system notifies the user to revise input.
Postconditions	The system updates the budget configuration and tracks spending progress.

Table 4.3 ***Manage budget use case***

- **View Report Use Case Specification**

Use Case Name	View Report
Actor	Student
Description	Enables users to view summary reports and insights of their spending over selected time periods.
Preconditions	The user must have recorded at least one expense.
Main Flow	- The user selects “View Report”. - The system retrieves data from the database. - The user chooses between monthly or annual report. - The system displays total spending, category breakdown, and trends over time.
Alternate Flow	If no data is available, the system notifies the user that no report can be generated.
Postconditions	The report is displayed to the user with summary statistics and charts.

Table 4.4 ***View report use case***

- **Send Feedback Use Case Specification**

Use Case Name	Send Feedback
Actor	Student
Description	Allows the user to send comments, suggestions, or issue reports to the Admin/Support Team.
Preconditions	The user must be logged in.

Main Flow	- The user selects “Send Feedback”. - The system displays a feedback form. - The user enters comments and submits the form. - The system sends the feedback to the Admin/Support Team.
Alternate Flow	If the network connection fails, the system saves the feedback locally and retries later.
Postconditions	The Admin/Support Team receives the feedback and can respond accordingly.

Table 4.5 ***Send feedback use case***

4.4.Class Diagram

To ensure the feasibility and scalability of the Student Cost Management project, the development team did not use a simple monolithic design but instead applied SOLID design principles to the class diagram. Below is a detailed analysis of how these principles are implemented in the design diagram:

❖ **Single Responsibility Principle**

This principle dictates that a class should only have one reason to change.

- The problem with the old design: Initially, MainActivity often handled too many responsibilities: displaying the user interface (UI), handling computational logic, and directly calling Firebase APIs to store data. This created a "Holy Object"—a class that was too large, difficult to test, and prone to chain reactions when modified.
- The new design: The team clearly separated the responsibilities:
  - ✓ Classes in the View layer (MainActivity, AddTransactionActivity) are only responsible for displaying data and handling user events.
  - ✓ Classes in the Repository layer (FirebaseTransactionRepo) are only responsible for interacting with the database (CRUD operations).
  - ✓ Model classes (Income, Expense) only contain data.
  - ✓ Benefits: This separation makes the source code clearer, and when UI bugs need fixing, it won't affect the data storage logic.

❖ **Open/Closed Principle**

This principle states that software entities should be open to extension but closed to modification.

- Application: Instead of creating a single Transaction class and using if-else or switch-case statements to differentiate between Income and Expenses, the design team created an abstract class called BaseTransaction. The specific transaction types, Income and Expenses, inherit from this parent class.

- Benefit: This architecture allows for easy future scaling. For example, if a later version requires additional features to manage "Debt" or "Investment," the team can simply create a new class inheriting from BaseTransaction without having to modify or rewrite the core logic of the existing system, minimizing the risk of creating new errors (regression errors).

❖ **Liskov Substitution Principle**

- Application: Anywhere in the source code using BaseTransaction (e.g., a transaction history list List<BaseTransaction>), it can be replaced with child objects (Income or Expense) without affecting program correctness.
- Benefit: Ensures consistency in handling polymorphic data, simplifies the display of mixed transaction lists.

❖ **Interface Segregation Principle**

- Application: Instead of creating one huge interface for the entire application, the team broke it down into ITransactionRepository (handles transactions) and IAuthService (handles authentication).
- Benefit: Functional screens are not forced to depend on methods they don't use. For example, the "Add Transaction" screen doesn't need to know about "Login/Register" logic.

❖ **Dependency Inversion Principle**

This is the most significant improvement in the design.

- Implementation: High-level modules like MainActivity do not directly depend on low-level modules like FirebaseTransactionRepo. Both depend on an abstract class called ITransactionRepository.
- Diagram proof: The dependency arrow from MainActivity points to the ITransactionRepository interface, not to a specific class.
- Benefit: Reduces dependency constraints. If the team decides to switch from Firebase to SQLite or a separate server (MySQL) in the future, they only need to write a new Repository class to implement this interface without modifying any code at the interface (Activity) layer. This significantly increases the flexibility and longevity of the application.

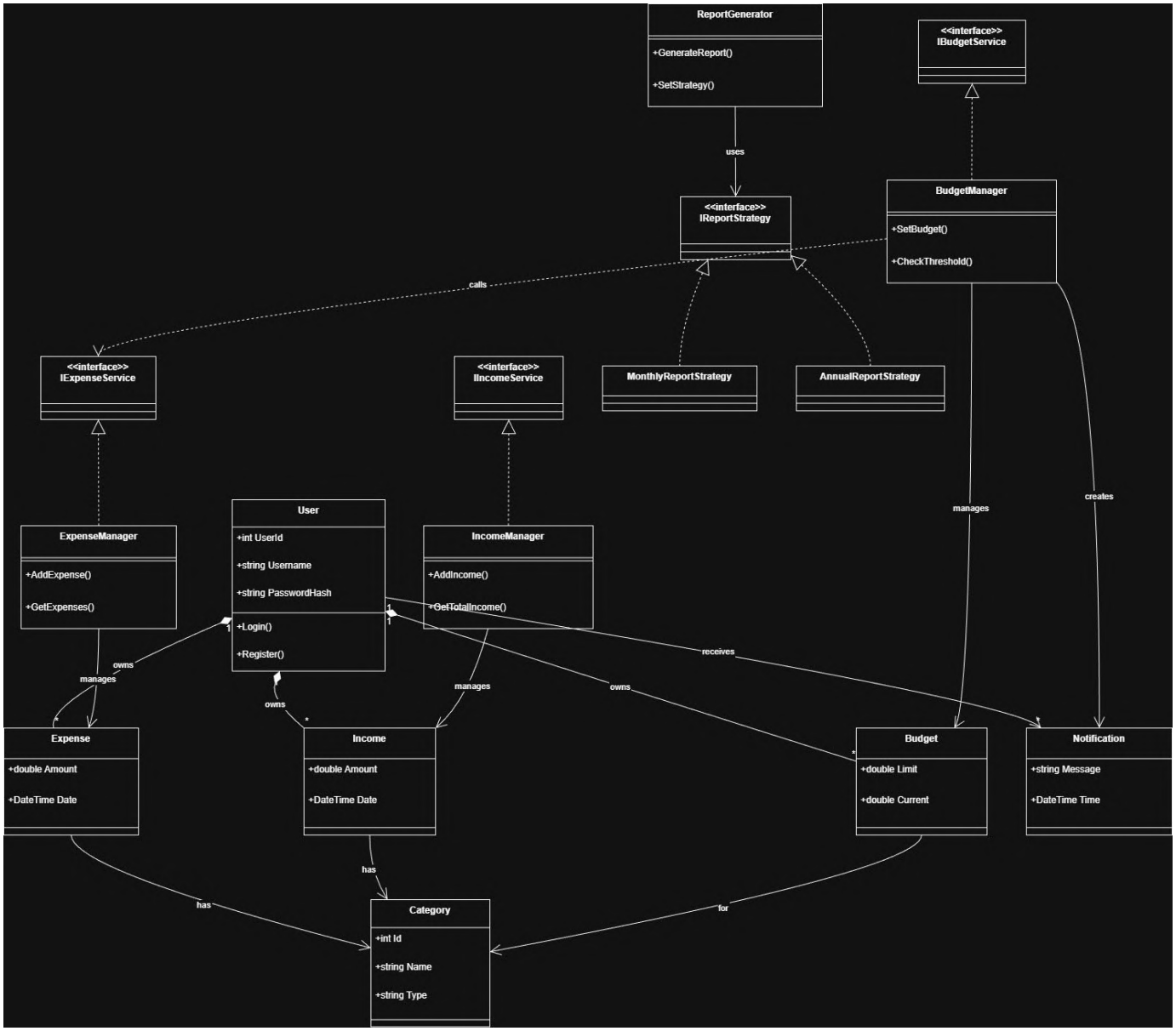


Figure 4.2 Classs diagram

4.5.Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) is used to represent the logical data structures of the CampusExpense Manager system. It is shown with the main entities, their attributes and also each relationship between them. This ERD will usually ensure a highly standardized database design, minimize redundancy and support efficient data retrieval.

The ERD clearly defines the structure of the underlying databases that support the CampusExpense Manager application. Each relationship ensures data consistency between user actions (such as adding expenses or creating reports) and the information stored.

The design follows a one-to-many relationship model and reflects how a single user can manage multiple budgets, expenses, and reports. By maintaining foreign key constraints, the system ensures data integrity and eliminates duplication, making it easier to query financial information and create accurate reports.

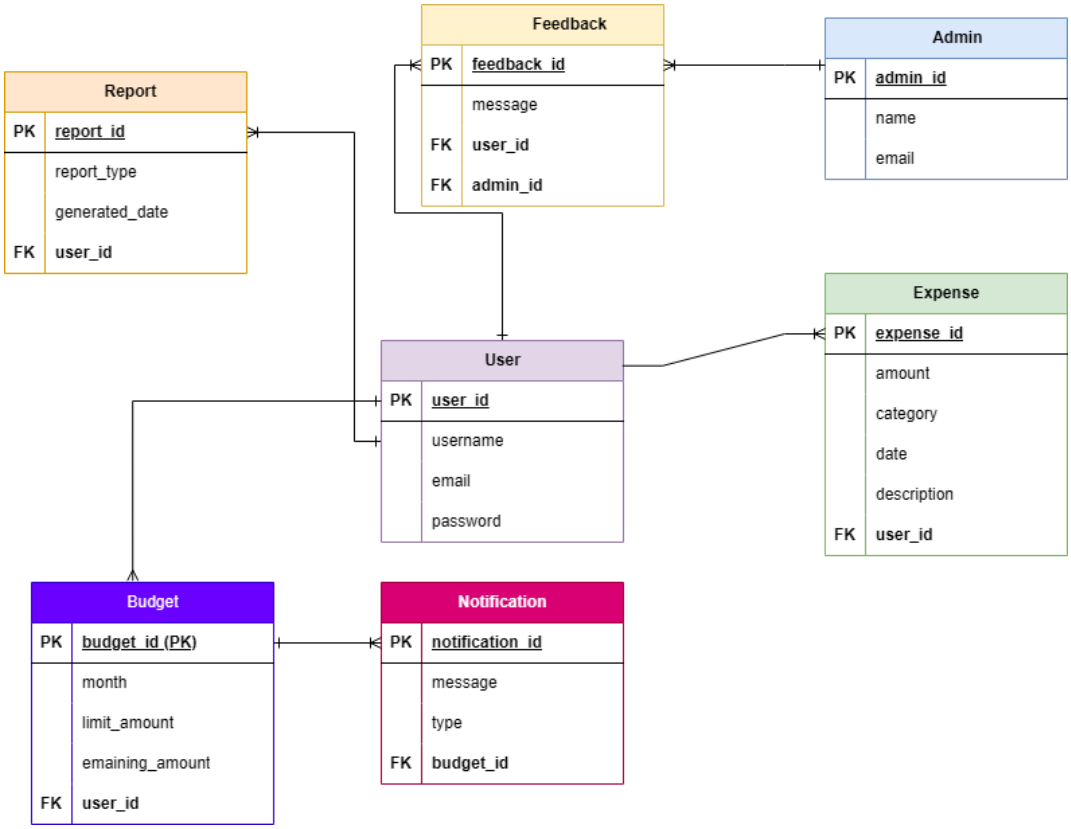


Figure 4.3 Entity Relationship Diagram – CampusExpense Manager

4.6.Test Plan

Testing is one of the important phases of the software development life cycle (SDLC) from here it can be to ensure that the developed application meets each requirement and also works extremely accurately.

This Test Plan will usually be for the CampusExpense Manager so that the overall testing strategy, objectives, scope, resources, responsibilities and schedule can be outlined to ensure the quality and reliability of the system before deployment.

- **Objectives**

  The main objectives of the testing process are:

  - ✓ To be able to verify that all functional and non-functional requirements are implemented correctly.
  - ✓ By identifying and eliminating further software defects before we are released.
  - ✓ To ensure that the applications are operating efficiently within the expected workload.
  - ✓ To verify with the systems provided so that financial reporting and notifications are accurate.

- **Scope of Testing**

  The tests will cover the following main modules of CampusExpense Manager application:

- User Account Management: Such as Registration, Login and Authentication.
- Expense Management: Add, Edit, Delete Expenses.
- Budget Management: Set up and hence add to adjusted Budgets, Receive Notifications.
- Reporting Module: Creating and Viewing Monthly/Yearly Reports.
- Feedback Module: Sending and Receiving Feedback between Users and Administrators.
- Non-functional aspects such as performance, security and usability will also be assessed.

❖ **Testing Strategy**

Type of Test	**Description**	**Responsibility**
Unit Testing	Testing individual modules and functions (e.g., login validation).	Developer
Integration Testing	Ensuring modules interact correctly (e.g., linking user budgets with notifications).	Developer & Tester
System Testing	Testing the complete system as a whole.	QA Team
User Acceptance Testing (UAT)	Final testing by actual student users to confirm usability and satisfaction.	End Users & QA Team

Table 4.6 ***Testing Strategy***

Test Environment

Component	Specification
Operating System	Android 12+ / iOS 16+
Database	Firebase Realtime Database
Development Tools	Android Studio, Java SDK
Testing Tools	JUnit (Unit Test), Postman (API Test), Manual testing scripts
Devices Used	Android smartphones and emulators

Table 4.7 ***Test environment***

Roles and Responsibilities

Role	Responsibilities
Project Manager	Approve testing phases and monitor progress.

Developer	Prepare test cases and perform unit testing.
Tester / QA Engineer	Execute integration and system tests, log defects, and verify fixes.
Student Users	Participate in UAT and provide feedback on usability.

Table 4.8 ***Roles and Responsibilities***

- **Test Schedule**

Phase	Duration	Activities
Unit Testing	Week 7–8	Test individual modules (Login, Expense, Budget).
Integration Testing	Week 9	Combine and test modules together.
System Testing	Week 10	Validate full system behavior.
UAT	Week 11	End users test the complete application.
Bug Fixing & Review	Week 12	Resolve reported issues and finalize version.

Table 4.9 ***Test schedule***

- **Risks and Mitigation**

Potential Risk	Impact	Mitigation Strategy
Incomplete test coverage	High	Develop detailed test cases for each module.
Time constraints	Medium	Prioritize testing of critical modules first.
Unstable environment	Medium	Use stable Firebase test database for QA.
User feedback delay	Low	Schedule UAT early to collect responses on time.

Table 4.10 ***Rish and mitigation***

With each test plan ensuring the CampusExpense Manager system has been put through a structured and thorough testing process before deployment. With these plans, functionality, usability and performance across all modules are validated to ensure a reliable and user-friendly application for the students.

5.Justify the software development tools and development methodology selected. (M2)

5.1.Justification of Software Development Model

Now it is very important to choose the right Software Development Life Cycle (SDLC) model to ensure that the project progresses smoothly so that deadlines can be met and quality is maintained.

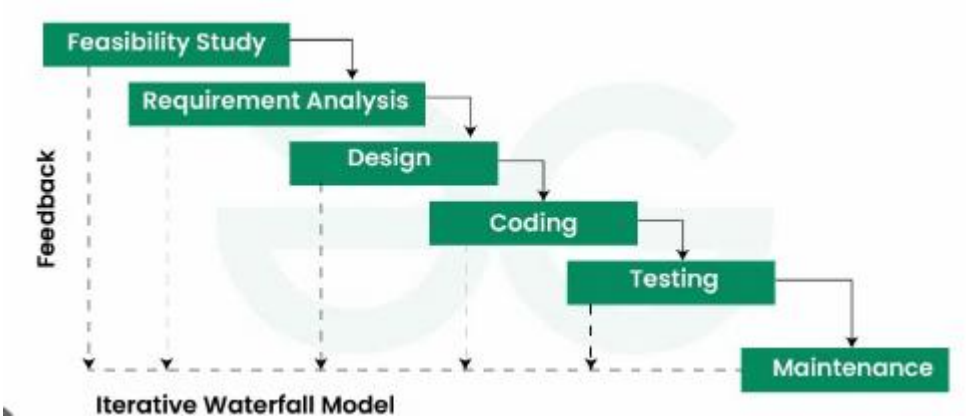


Figure 5.1 Waterfall model

- After analyzing the constraints of the projects described in P1 — including short durations (12 weeks), small development teams and extremely limited programming experience, the most suitable model for the CampusExpense Manager project is the Waterfall Model.
- The Waterfall Model is followed with a linear and sequential approach, where each phase (Requirements → Design → Implementation → Testing → Deployment → Maintenance) from there will be completed before the next phase begins. These methods will often provide a very clear structure and for each workflow can be predictable, ideal for teams with little experience in mobile application development.
- Using the Waterfall model will ensure that each requirement is fully understood before starting to write code, reducing the possibility of each task being changed later in the development process. Since the scope and for each goal of the project will be clearly defined from the beginning (as determined in P1 and P2), from there for each of these models will reduce complexity and risk.
- In addition, with each Waterfall model, it is easy to track and document the project, suitable for the training nature as well as the orientation of each process of this exercise. Each phase produces tangible deliverables (e.g., individual UML diagrams, code modules, test results), which can be easily managed by the team and progress can be easily evaluated by each reviewer.

Therefore, in the context of time constraints, small team sizes, and fixed requirements, the SDLC Waterfall model is the most appropriate and effective choice for developing the CampusExpense Manager application.

5.2.Justification of Development Tools

The choice of development tools is essential to ensure that the software is developed efficiently, thoroughly tested and easily maintained. The tools chosen for each project have been carefully chosen based on ease of use, compatibility with mobile platforms and the ability to reduce development complexity.

Here are the reasons for using each tool in the CampusExpense Manager project:

❖ **Justification for Integrated Development Environment (IDE): Android Studio**

The team decided to choose Android Studio over Eclipse or IntelliJ IDEA for the following reasons:

- ✓ Official Support: As Google's official IDE, Android Studio always receives the latest SDK updates. Tools like the Layout Editor (drag-and-drop interface) and Gradle Build System are deeply integrated, reducing environment configuration time – something Eclipse currently lacks support for.
- ✓ Debugging Tools: The Android Profiler in Android Studio allows real-time monitoring of RAM, battery, and network consumption. This is crucial for optimizing applications to manage resources efficiently and prevent overheating.
- ✓ Emulator: The Android Studio emulator has significantly improved speed compared to competitors, allowing testing teams to quickly test on various phone screen sizes without needing to purchase real devices.

❖ **Justification for Programming Language: Java**

Although Kotlin is trending, Java has been chosen as the primary development language:

- ✓ Knowledge Foundation (Team Expertise): Java is the core language in the school's curriculum. Using Java allows the team to start writing code immediately (zero learning curve) instead of spending 1-2 weeks learning the new syntax of Kotlin or Python.
- ✓ Community and Documentation: With a young team, encountering bugs is inevitable. Java has a huge StackOverflow community, making it easier to find solutions to Android/Java issues compared to Kivy (Python) or newer frameworks.
- ✓ Stability: Java forces programmers to strictly adhere to OOP (Object-Oriented Programming), making the code easier to manage in a team setting compared to a more flexible language like Python.

❖ **Justification for Database: Firebase Realtime Database**

The decision to choose Firebase over SQLite or MySQL was a crucial strategy to ensure the 12-week project deadline:

- ✓ Serverless: If MySQL had been chosen, the team would have had to spend time building additional Web APIs (Backend) using Node.js or PHP, and then rent hosting to run them. Firebase completely eliminates this burden, providing "Backend-as-a-Service." The team could focus solely on developing the Mobile App.
- ✓ Real-time Sync: This feature of Firebase ensures that student spending data is updated instantly. If the phone is lost or damaged, the data remains secure in the cloud, overcoming the critical drawback of SQLite (which only stores data locally).
- ✓ User Authentication: Firebase's built-in Authentication allows for login/registration functionality to be coded in just a few hours, instead of several days if manually written.

❖ **Justification for Source Code Management: GitHub**

GitHub was chosen over GitLab or Bitbucket because:

- ✓ Its popularity: It's an industry standard. Proficiency in GitHub makes team members' portfolios more attractive to potential employers.
- ✓ Easy integration: GitHub integrates seamlessly with Android Studio. Commit, Push, Pull, and Resolve Conflicts can all be performed directly within the IDE.
- ✓ Version management: In teamwork, overwriting each other's code is a major risk. GitHub's branching and Pull Request mechanisms allow teams to tightly control each line of code before merging it into the master branch.

❖ **Justification for Diagram Design: Draw.io**

Draw.io is superior to Microsoft Visio or Lucidchart:

- ✓ Accessibility: Draw.io runs entirely in a web browser and is 100% free. Members don't need to install heavy software (like Visio) and there are no limitations on the number of diagrams (unlike the free version of Lucidchart).
- ✓ Flexible storage: It allows design files to be saved directly to the team's Google Drive, ensuring all members can open and edit UML designs anytime, anywhere.

❖ **Justification for Project Management: Trello**

For small teams (Scrum teams), Trello surpasses Jira due to its simplicity:

- ✓ Visualization (Kanban Board): Trello's card interface allows the team to instantly see who is doing what and what is completed.

- ✓ No hassle: Jira is powerful but has complex configurations (Workflows, Permissions). In the short term, spending time managing this tool is a waste. Trello meets the need for "Quick setup, easy tracking".

6.Evaluate the solution to a business-related problem and the preferred software development methodology by comparing the various software development tools and techniques researched. (D1)

6.1.Evaluation of the Chosen Solution (Mobile Application)

With the work that will decide to develop the CampusExpense Manager project as a mobile application, it will be given after being evaluated by alternative solutions, as well as from there will include desktop applications and according to each web-based system.

The audience for the target users are students, who will need people for extremely convenient and mobile solutions to manage expenses anytime, anywhere. With a mobile application like this, it will best meet these requirements with the ability to access anytime, anywhere and built-in with device functions such as notifications and offline storage.

Comparison of Alternative Solutions

Criteria	Mobile Application	Web Application	Desktop Application
Accessibility	Can be used anytime, anywhere via smartphone.	Requires internet and browser access.	Restricted to a single device.
User Experience (UX)	Optimized for touch and mobile interface.	Limited responsiveness on small screens.	Full interface but not portable.
Performance	High responsiveness and smooth native operations.	Depends on browser and network speed.	Good performance but lacks mobility.
Offline Usage	Can support local data caching or offline mode.	Typically requires a stable internet connection.	Works offline, but no cloud sync.
Ease of Deployment	Easily distributed through app stores.	No installation required, accessed via browser.	Requires manual installation and updates.
Suitability for Students	Perfect for on-the-go budgeting and notifications.	Partial fit – depends on Wi-Fi access.	Not suitable for mobile student life.

Table 6.1 Alternative Solutions

Considering the lifestyles and needs of students, mobile applications are now the solution, being the most practical and effective device. This will provide real-time notifications, as well as easy access to each item and mobility, which are important factors for tracking each individual's finances.

While desktop or web applications may be easier to develop, they do not provide the same level of mobility and interactivity that the project's user objectives require.

6.2.Evaluation of the Development Methodology (Waterfall vs Agile)

In M2, for the Waterfall model it was chosen to be the main SDLC method from here on to carry out short-term projects, fixed requirements and also many team experience modes.

The work to evaluate one of the main aspirations is the best options, below is the work that will be compared with the Agile model, another popular development method.

- **Comparison: Waterfall vs Agile**

Criteria	Waterfall Model (Chosen)	Agile Model (Alternative)
Structure	Sequential and phase-based.	Iterative and flexible.
Change Management	Difficult to accommodate new changes once a phase is complete.	Highly adaptive to changing requirements.
Project Planning	Clearly defined scope and deliverables from the start.	Requirements may evolve continuously.
Team Requirements	Easier for small, inexperienced teams.	Requires high communication and self-organization.
Documentation	Strong emphasis on documentation and deliverables.	Focused more on working prototypes and user feedback.
Suitability for This Project	Best for short-term academic projects with defined goals.	Risky for small inexperienced teams with limited time.

Table 6.2 ***Waterfall vs Agile***

If we adopt Agile, the teams can adapt much faster to each task, which will be given feedback from many students, which can be improved on the designers drawn from the centers. But on the other hand, this flexibility will increase the difficulty and therefore the cost of management, it will not be very suitable.

Therefore, the Waterfall model is still the most suitable choice for this specific project context — it does not guarantee a clear structure, expected results and an easy-to-manage workload for the BudgetWise Solutions team.

6.3.Evaluation of Development Tools and Technologies

In the context of the "CampusExpense Manager" project developed by BudgetWise Solutions, technology selection was not simply based on popularity but rather a balancing act between Technical Debt, Resource Constraints, and Delivery Risk. Below is an evaluation and critical comparison to demonstrate the optimality of the chosen toolset.Manually tracking through spreadsheets is prone to errors and from here it will be disorganized.

❖ **Programming Language: Java vs. Kotlin (The Trade-off between Modernity and Risk)**

Although Google has declared Kotlin as the leading language for Android, with a syntax approximately 20% shorter than Java, the team remained resolute in choosing Java due to project risks:

Risk Analysis: With a project deadline of only 12 weeks, switching to Kotlin (a language that 3/4 of the team members were not yet proficient in) would create a "Learning Curve" of approximately 2-3 weeks in the initial phase. This means losing 20% of the total project time just to learn the tool, leading to a high risk of not meeting the core functionality deadline (Delays).

Comparison with Python (Kivy): Python is easy to write, but the Kivy framework produces applications with low performance and non-native interfaces (which look very strange to Android users). For financial applications requiring smoothness and reliability, Python is completely ruled out due to User Experience (UX) issues.

Conclusion: Java is the safest choice. It might be more verbose than Kotlin, but it ensures the team can start coding immediately (Day One Productivity).

❖ **Database: Firebase Realtime DB vs. SQLite vs. MySQL (The Connectivity & Infrastructure Debate)**

This is the most important strategic decision affecting the system architecture.

✓ **Why not SQLite?**

SQLite is very powerful and fast, BUT it stores data locally. If a student loses their phone or reinstalls the system, all their spending data will be lost. For a financial management application, data loss is the most serious error. Therefore, SQLite only acts as a cache, not the primary database.

✓ **Why not MySQL/PostgreSQL?**

MySQL is a standard relational database management system (RDBMS), but it requires a web server (hosting) and a server-side API (NodeJS/PHP) for communication.

Barrier: BudgetWise is a startup project with a zero budget. Renting a server incurs monthly maintenance costs and requires DevOps skills to secure the server. The team lacked sufficient personnel to develop both the mobile application and the server-side application.

✓ **Advantages of Firebase:**

It solves both problems: Cloud storage (avoiding data loss) and serverless (no server rental fees, no need to write server-side APIs). Firebase's offline storage mechanism also automatically handles network connection issues much better than manually writing synchronization logic using MySQL.

❖ **IDE: Android Studio vs. Eclipse (The Compatibility Issue)**

Choosing Android Studio isn't just a matter of preference; it's a necessary technical requirement.

The problem with Eclipse: Eclipse stopped supporting ADT plugins a long time ago. Integrating modern libraries like Firebase SDK 2024 or Material Design 3 into Eclipse is extremely difficult and prone to conflicts (complex dependency issues).

✓ **The absolute advantage of Android Studio:**

The Gradle Build System tool in Android Studio allows managing third-party libraries with just one line of code.

More importantly, the Android Profiler tool (only available in Android Studio) allows the team to measure battery and RAM consumption. Since the target users are students (often using mid-range/low-end phones), performance optimization is crucial. Eclipse doesn't offer these in-depth measurement tools.

❖ **Project Management: Trello vs. Jira (Complexity vs. Agility)**

Criticism of Jira: Jira is the world's most powerful software project management tool, but it's too complex (overkill) for a team of 4-5 people. Setting up workflows, swivels, and permissions in Jira is too time-consuming (management overhead).

The suitability of Trello: With a streamlined Agile Scrum model for a student project, the team needs transparency and speed. Trello removes all configuration barriers, allowing the team to focus on "coding" instead of "reporting".

Conclusion: In the context of short-term projects, the best tool is the one that least hinders the workflow, and that is Trello.

6.4.Overall Evaluation and Conclusion

Finally after analyzing all the solutions, the discussion method and each tool we have developed has chosen very clearly, from the design of the CampusExpense Manager project is completely suitable.

- Currently, mobile applications are the most effective means for student users thanks to their mobility as well as convenience and interaction.
- The Waterfall Software Development Model (SDLC) will be guaranteed to be predictable and clear for a very small group, lacking mask experience in a fixed learning time frame.
- With the selected tools (Android Studio, Java, Firebase, GitHub, Trello, Draw.io) it will balance the balance between each function, from there we will be able to include easy-to-use features and from here will have a faster development speed.

- While alternative methodologies (such as Agile or cross-platform development) may offer flexibility or allow for more widespread development, they often introduce unnecessary complexity to the given resources.
- Therefore, combining mobile-first methodologies such as Waterfall and each Google-based tool ecosystem forms the most efficient, robust, and pedagogically sound framework for successfully developing and managing CampusExpense Manager.

7.Conduct a peer review of the problem definition statement, proposed solition and development strategy, documenting any feedback given.(P4)

7.1.Overview of the Peer-Review Process

To validate the accuracy and feasibility of the problem definition and proposed solution presented in Exercise 1, one of the activities that is being peer reviewed is that it will be done by combining both quantitative and qualitative methods.

With the purpose of the evaluation being to ensure that the initial design is very much in line with the actual expectations of the users, identified by weaknesses and provided with a reliable basis for improvement of LO4. The evaluation process will include the delivery of a number of structured surveys from which we will collect quantitative feedback at scale and a series of semi-structured interviews designed to gain deeper qualitative insights. At a minimum, approximately 30 survey responses and 4-5 interviews will be collected to ensure reliability and diversity of responses. The data collected from these two methods will be used to evaluate the effectiveness of the proposed solution and highlight specific areas that need to be refined.

7.2.Feedback Collection Methods

In order for us to ensure the comprehensiveness, in such evaluation processes, it will be done between the combination of each quantitative method and each qualitative task: Quantitative Method:

- With the implementation of one of the surveys with each structure (structured survey), it will be used Google Forms to collect from here with each feedback on each wide area at each level that is extremely appropriate and for each task it will be expected features of the solutions.
- Qualitative Method: With each task being conducted of semi-structured interviews with each user of the targets so that from here it can be deepened for understanding and from there will be a more in-depth assessment.
- With this in mind, our team has set a goal of collecting a minimum of 30 quantitative responses and from there will also conduct 4-5 qualitative interviews to ensure reliability and diversity of opinion.

7.2.1.Survey and Interview Framework

The tool we used was an online survey, designed to cover key aspects of the project. This survey included mandatory questions for each of the participants, how the solution could be rated on a scale of 1-5, desired features, and open-ended questions to gather feedback.

Link gg form : https://forms.gle/ceCUeAb8QQNeT9ku7

The survey includes

- **Full Name**
- **What is your current role?**
- **The need for a dedicated student expense management app.**
- **Your assessment of the effectiveness of the "CampusExpense Manager" solution in solving students' financial problems.**
- **Is the proposed user interface (UI) and experience (UX) intuitive and easy to use?**
- **How willing would you be to use the "CampusExpense Manager" app if it were developed?**
- **Rate the importance of core features to your financial management needs.**
- **Do you want an app with Group Expense Sharing?**
- **What weaknesses or shortcomings do you see in the current proposed features?**
- **What security risks do you think can occur when using student finance management apps?**
- **What other features would you like to add to make the app more useful?**
- **Do you have any other suggestions for the BudgetWise Solutions project development team?**

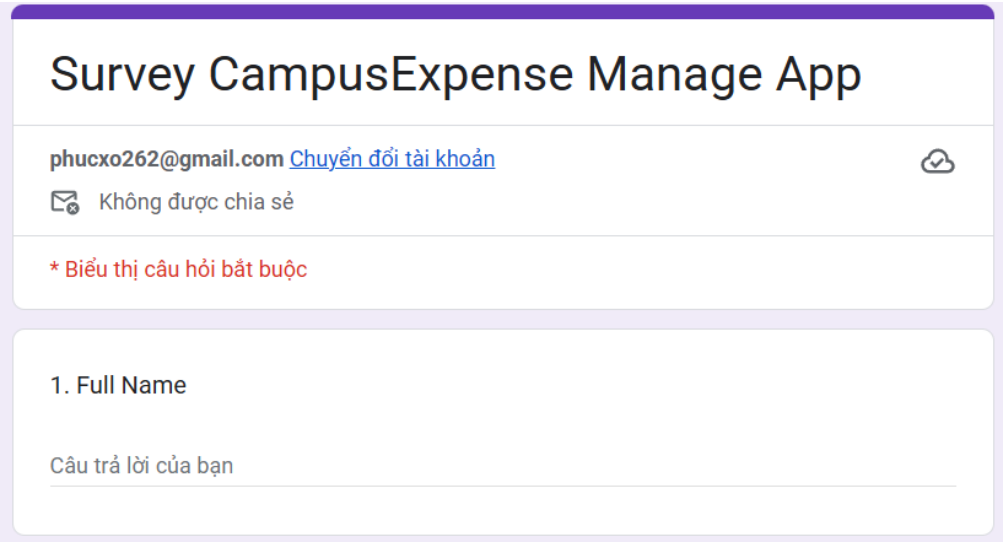


Figure 7.1 ***Question***

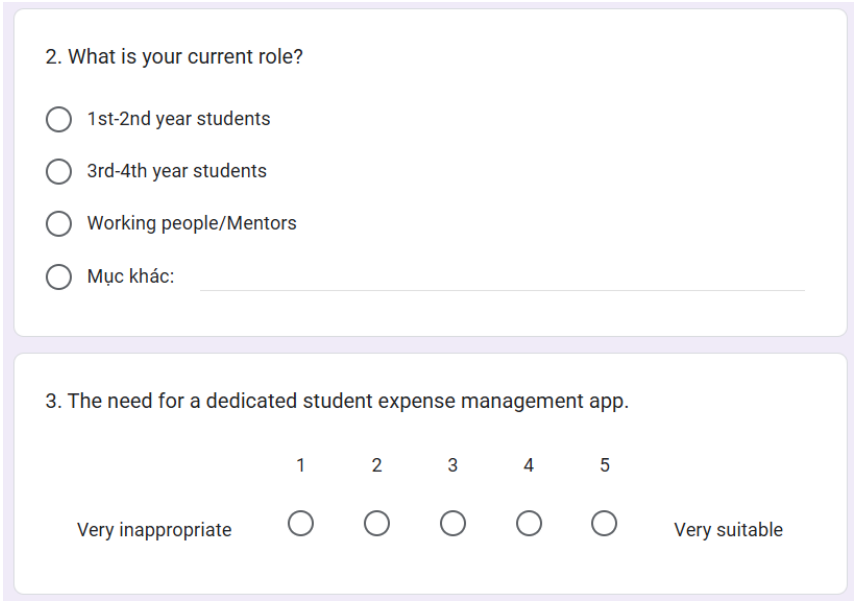


Figure 7.2 Question

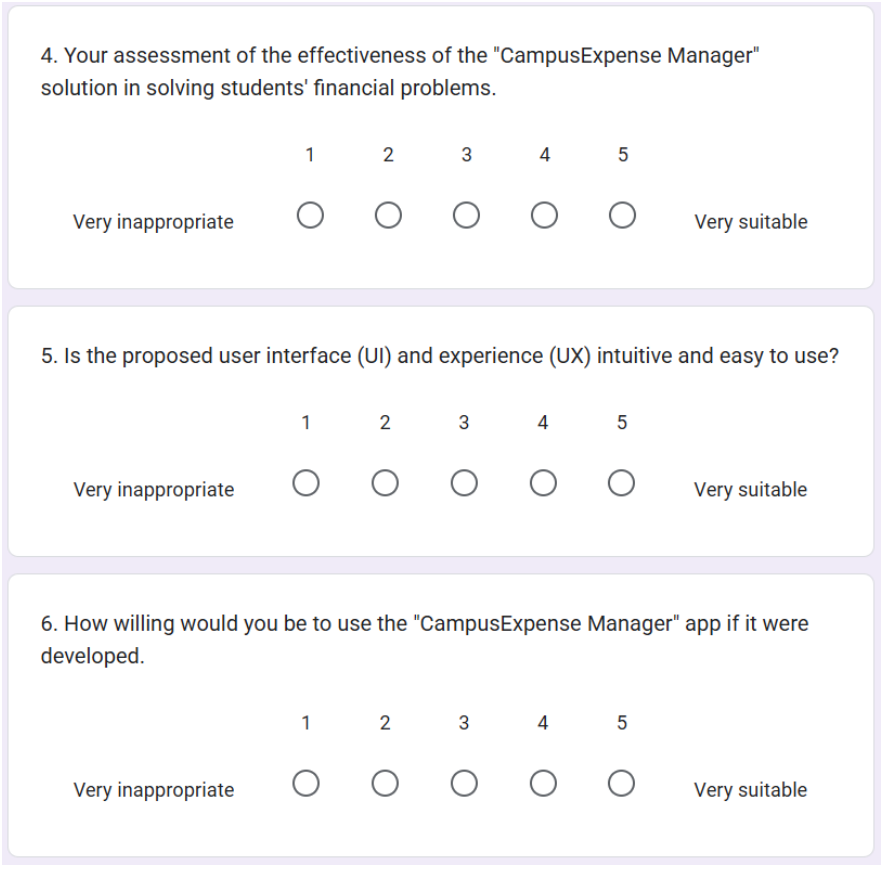


Figure 7.3 Question

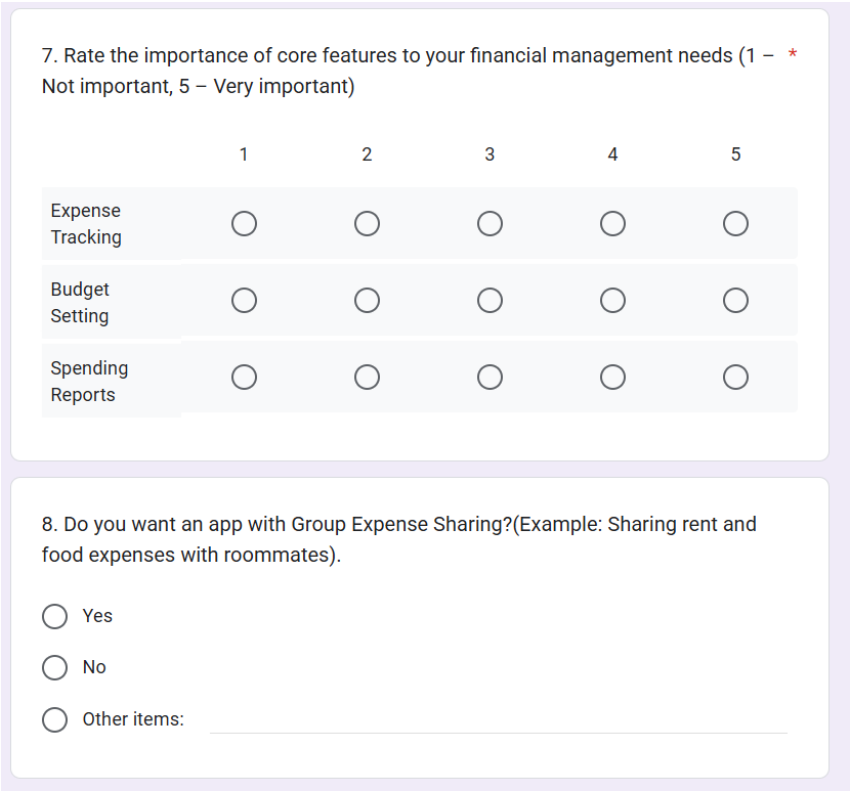


Figure 7.4 Question

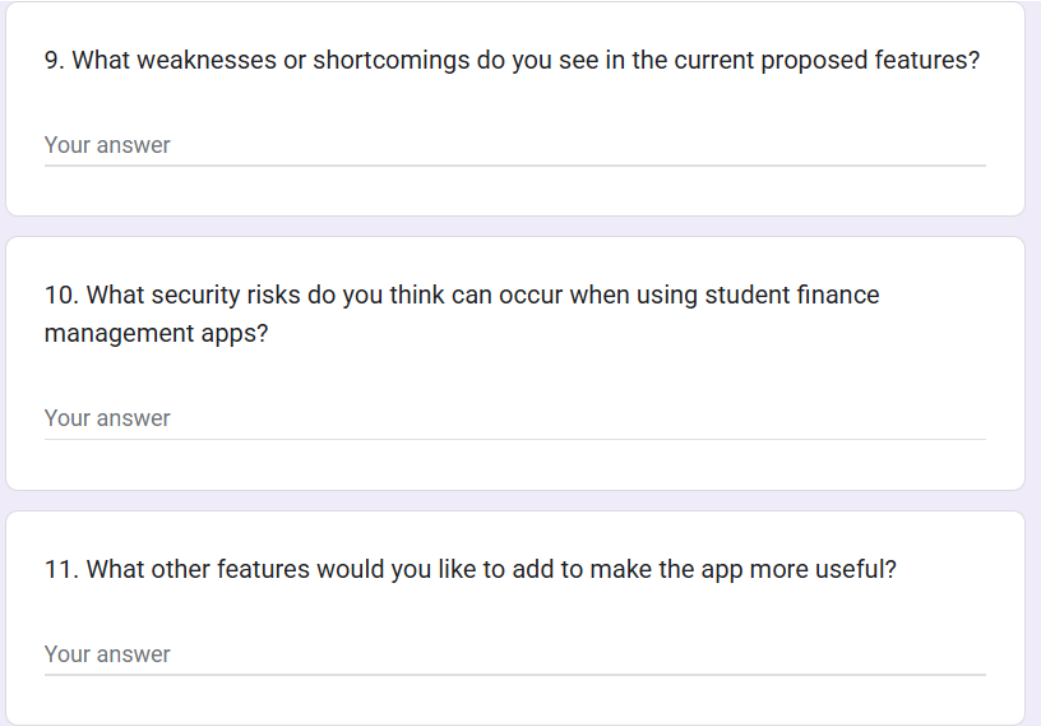


Figure 7.5 Question

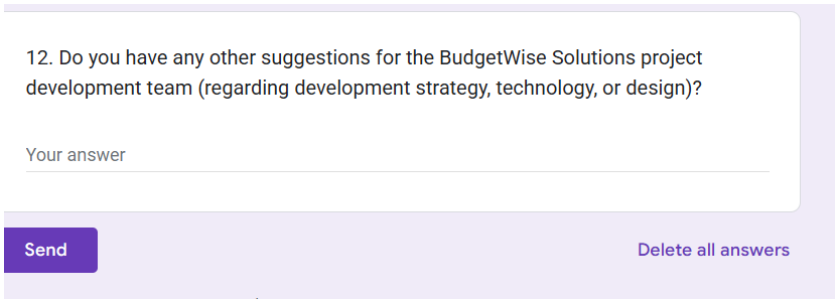
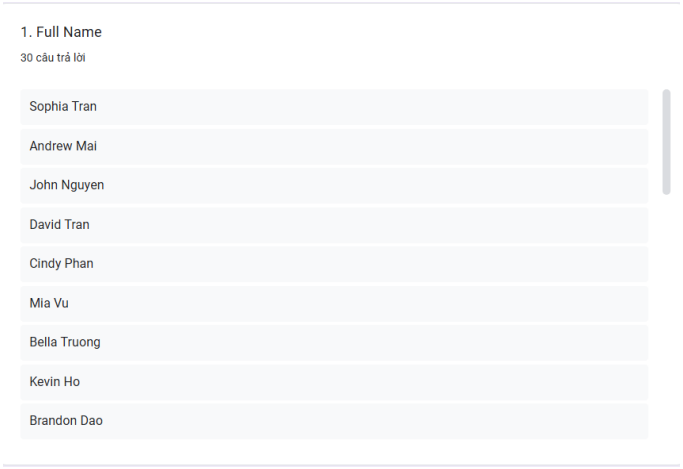


Figure 7.6 Question

a, Survey Questions Framework

Question 1 : Full Name This question is intended to collect information from participants, in part so that it can be mandatory to identify each role and each source of feedback, ensuring diversity and transparency.



Question 1 . Summary of Full Names of 30 survey participants.

Question 2 : This question was aimed at identifying each participant's role, ensuring that the feedback it collected came from different user groups: juniors (new to financial management), seniors (experienced with spending), and mentors/working adults (providing professional perspectives).

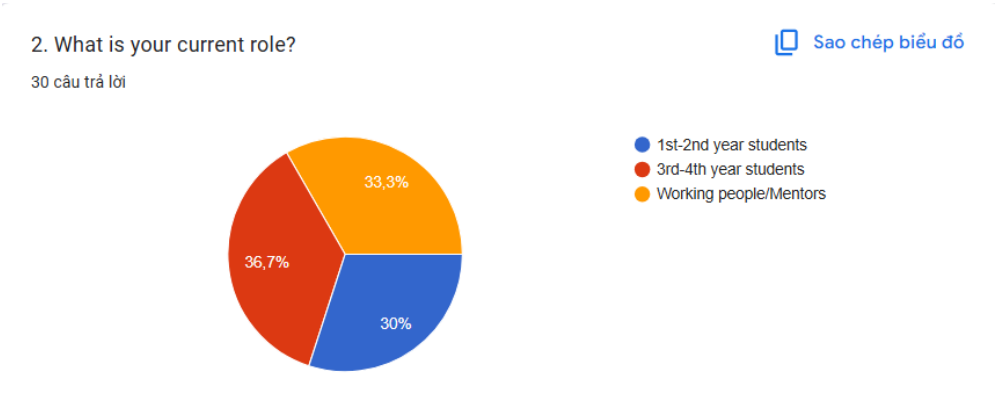


Figure 7.7 Current role allocation of 30 survey participants.

Question 3 : This question is designed to directly validate the Problem Definition. The problem statement focuses on students having difficulty managing their personal finances, leading to the need for a specialized tool.

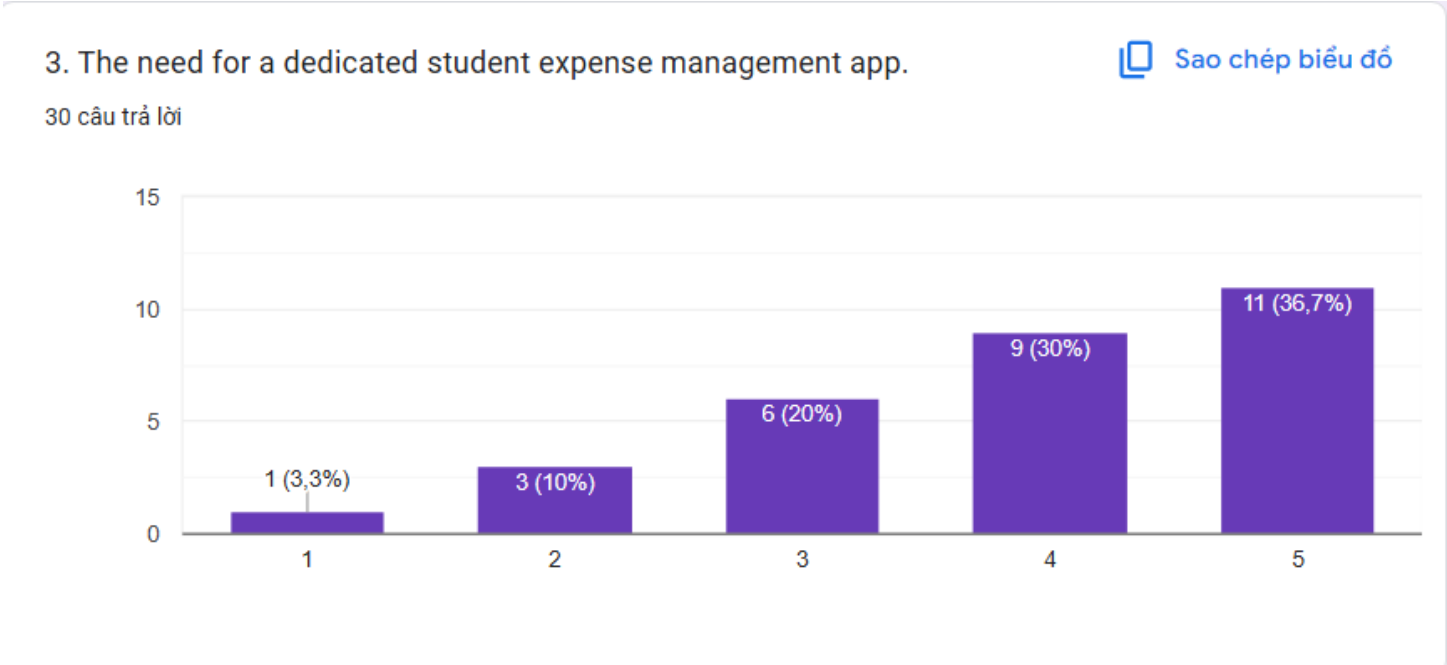


Figure 7.8 Rate the necessity of a dedicated expense management application for students (Scale 1-5).

Question 4 : This question focuses on being able to evaluate the effectiveness of the Proposed Solution (CampusExpense Manager application) in solving the financial management problem faced by students.

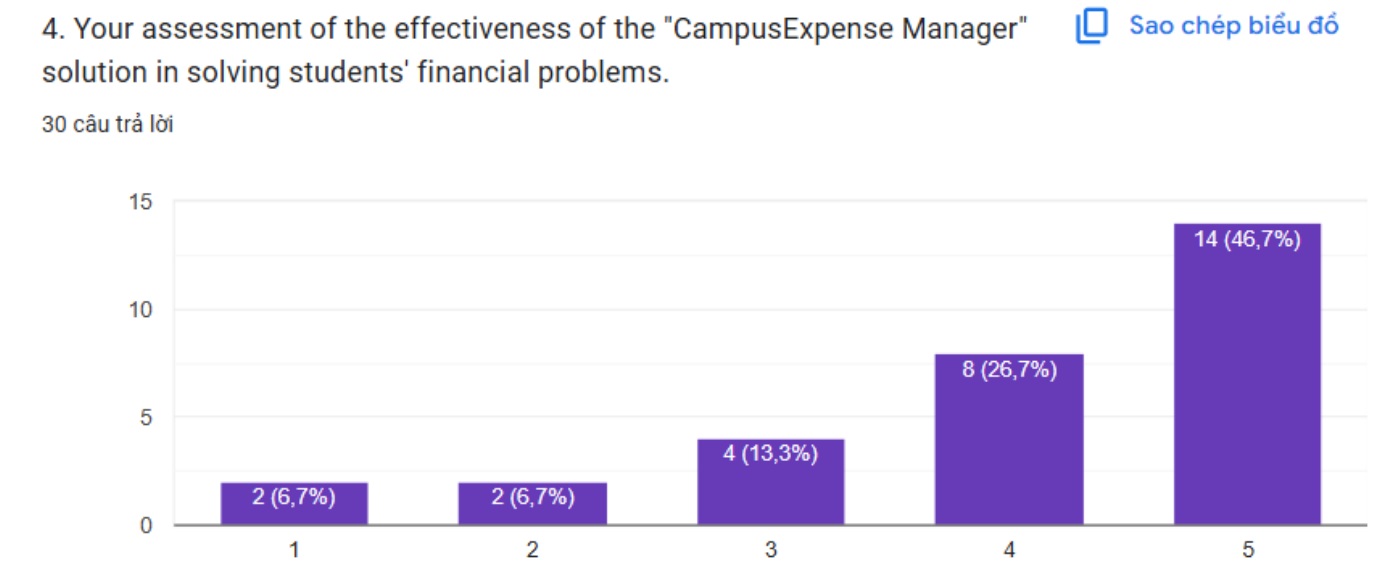


Figure 7.9 Evaluation of the effectiveness of the "CampusExpense Manager" Solution in solving students' financial problems (Scale 1-5).

Question 5 : This question validates one of the most important Non-Function Requirements: User-Friendly Interface (The user interface must be intuitive, easy to understand with clear labels and simple navigation).

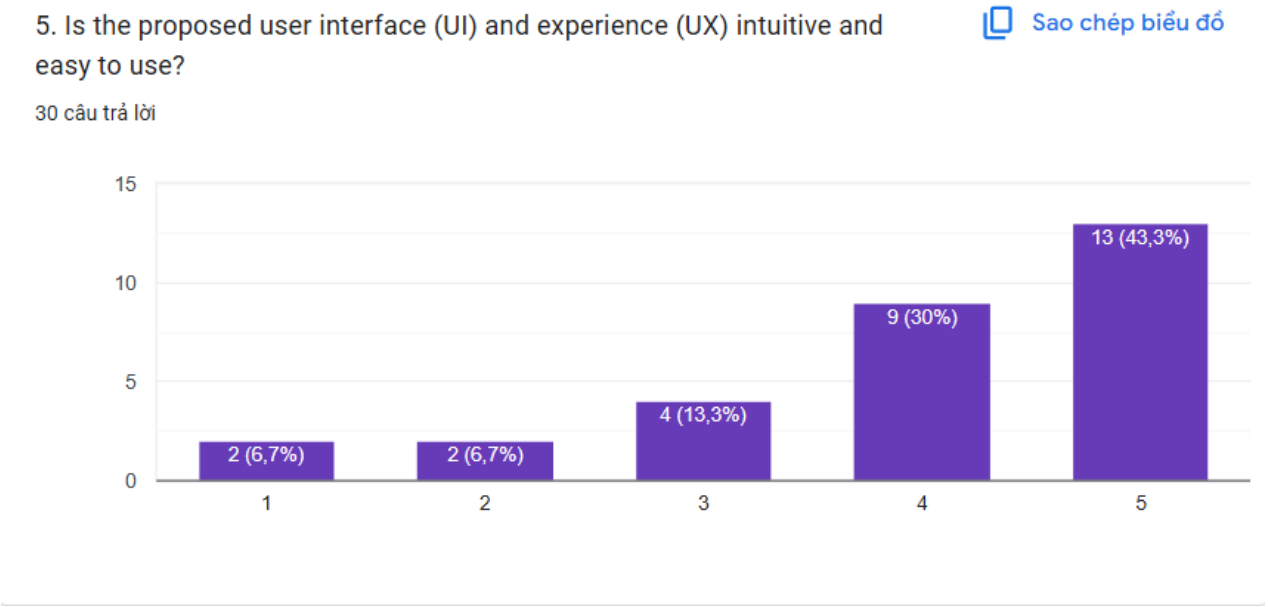


Figure 7.10 Rate the intuitiveness and ease of use of the Interface and User Experience (Scale 1-5).

Question 6 : This question measures the target user's Willingness to Use, which is an important indicator of a product's success and acceptance in the market.

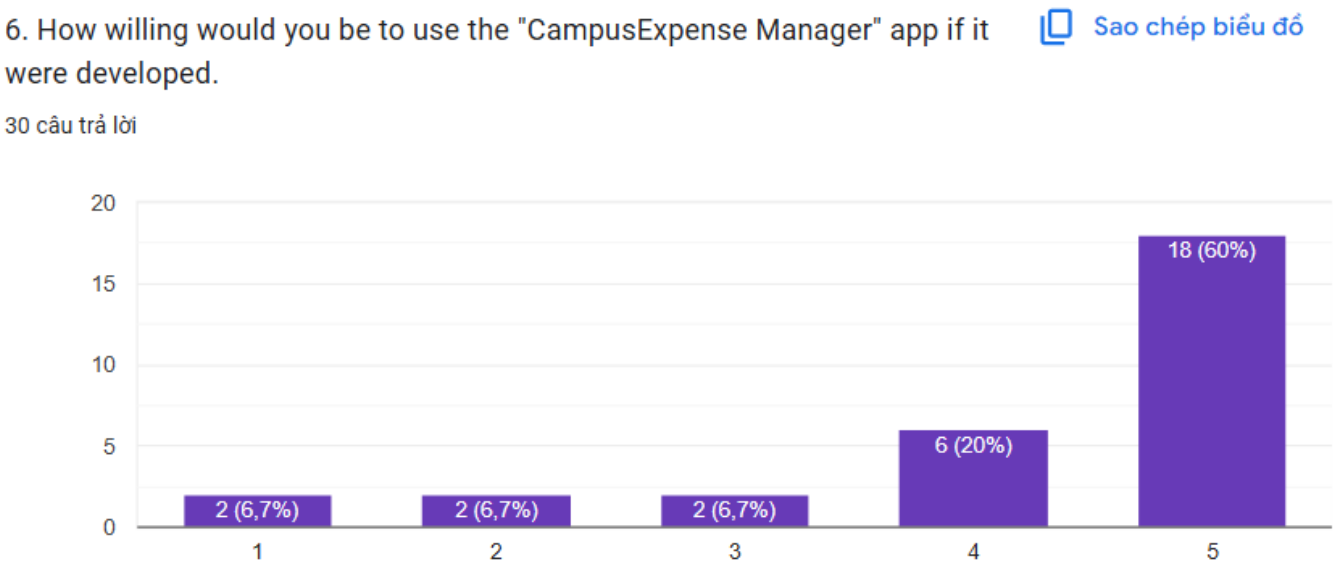


Figure 7.11 Rate your willingness to use the "CampusExpense Manager" application if developed

Question 7 : This question directly tests the key functional requirements identified in Assignment 1, which include: Expense Tracking, Budget Setting, and Spending Reports.

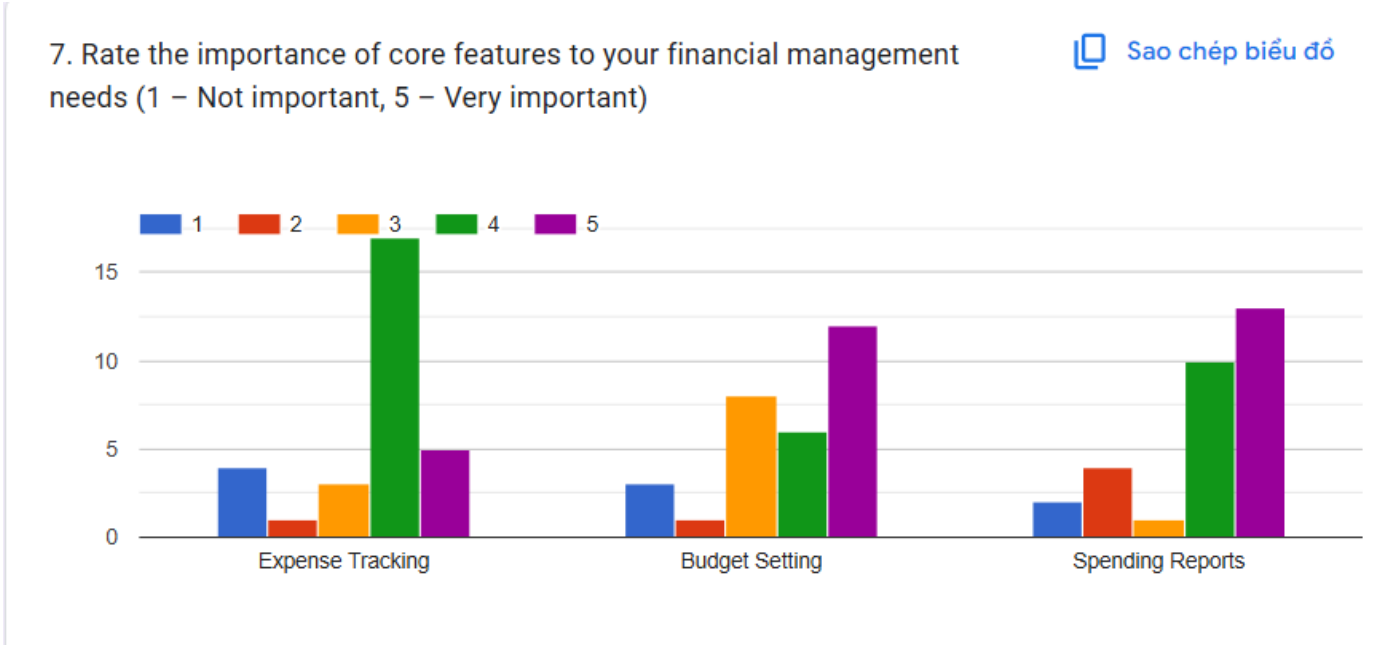


Figure 7.12 Rate the importance of core features

Question 8 : This question is designed to probe an important additional feature, not included in the original Functional Requirements from Assignment 1. This feature addresses the issue of general spending, which is very common in student life.

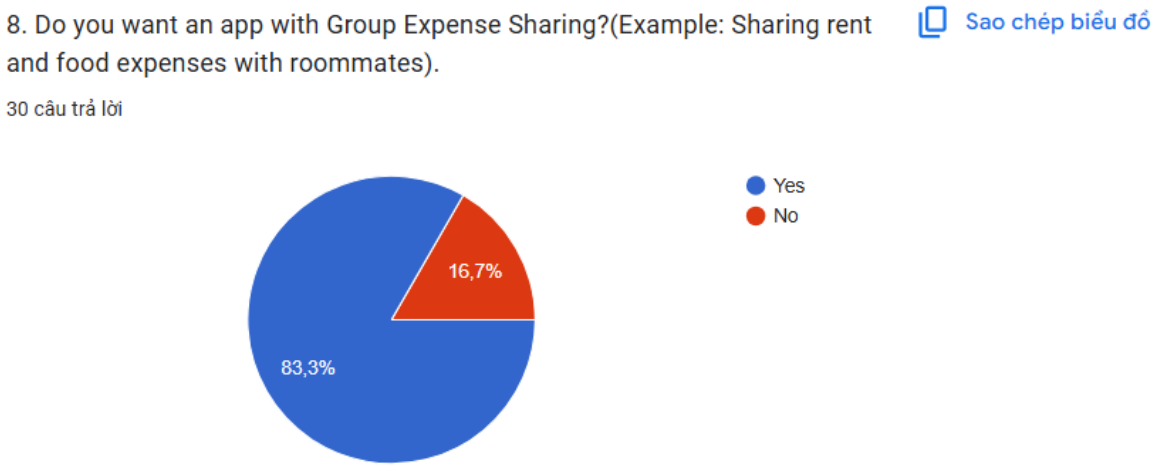


Figure 7.13 The need for Group Spend Sharing feature.

Question 9 : Open-ended questions often encourage each participant to critically review each functional requirement and initial design, thereby identifying specific weaknesses that need to be addressed.

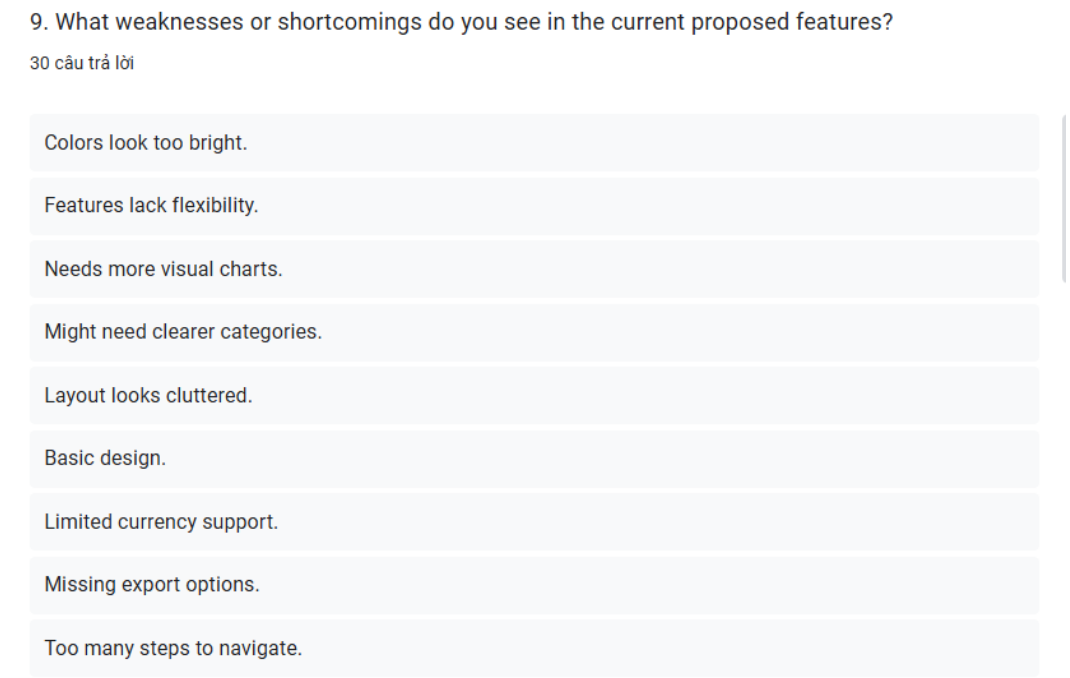


Figure 7.14 Summary of Weaknesses and Shortcomings identified by users.

Question 10 : This open-ended question is intended to confirm the Security and Legal Risks identified by the team in P2 of Assignment 1, and to highlight the need for Data Security as a core Non-Functional Requirement.

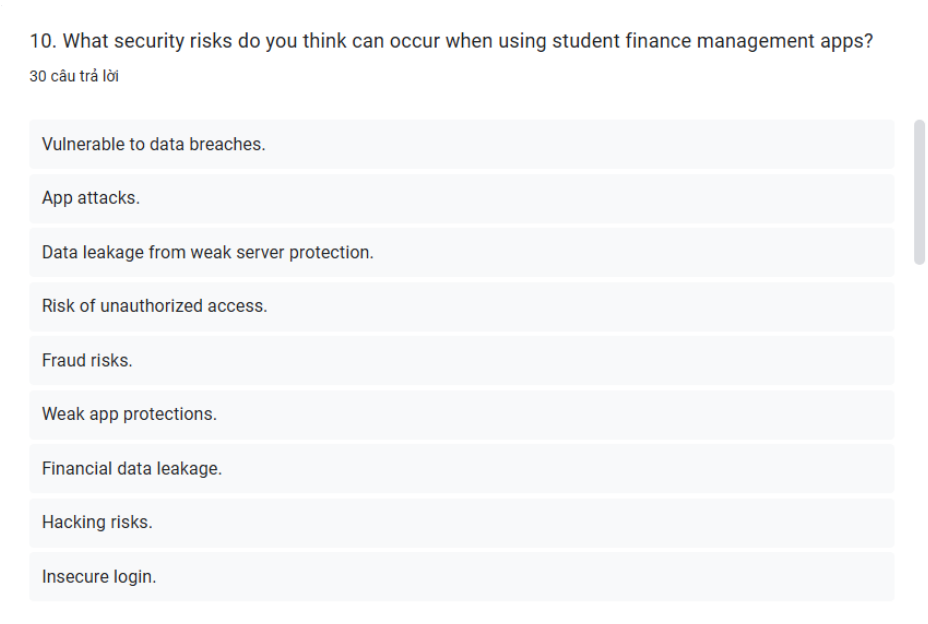


Figure 7.15 Summary of User Identified Security Risks.

Question 11 : This open-ended question is intended to gather ideas for additional features and value enhancements, beyond the initial core requirements, guiding future development opportunities.

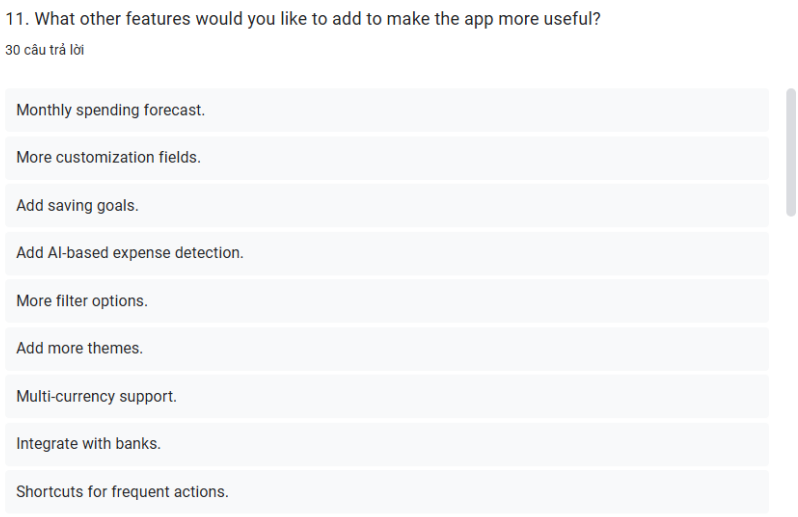


Figure 7.16 Summary of Advanced Features suggested by users.

Question 12 : This open-ended question allows users to provide broad feedback, helping the team gain a more comprehensive assessment of the risks and opportunities associated with the development process.

Figure 7.17 Summary of General Comments on Strategy, Technology and Design.

For each Peer Review process, it will be done by tasks that will be combined with quantitative surveys (30 responses) and from which qualitative opinions will be collected, which have been strongly confirmed with each validity and necessity of the Proposed Solution, the CampusExpense Manager application.

- **Solution Validation:** For each data task, it will show that the majority of users (over 66%) are in terms of being confirmed the urgent need for a specialized expense management application for students, and over 73% rate the solution as effective in solving financial problems. Furthermore, 80% of users expressed their willingness to use the application if developed.
- **Functional Requirements Validation:** Feedback confirms the Budget Setting and Expense Reporting features as core values, reinforcing the initial Functional Requirements.
- **Identifying Weaknesses & Improvement Opportunities:** Open feedback identified areas for improvement, particularly the need for Group Spend Sharing (requested by over 83% of users), and issues related to flexibility, performance optimization, and data security.

7.2.2.Documentation of Qualitative Interviews (Persona & Feedback Summary)

In order to complement the quantitative survey data and meet the requirement of "documenting any feedback given", our team was able to conduct 4 semi-structured interviews with users representing the main target groups of the "CampusExpense Manager" application.

The objectives of each interview were to collect in-depth feedback on the expense management experience, identify real needs, evaluate the suitability of the proposed solution and explore features that could be improved in the next stages of development.

Interviews will be conducted in person and via Google Meet, each session will last 8–12 minutes, focusing on 4 main topics:

- Personal spending management habits
- Current difficulties
- Initial assessment of the application
- Suggestions and wishes for improvement
- Based on the information collected, the team has built 4 representative Personas and summarized the key feedback in the table below.

Name of Interviewer	Role / Position	Describe Spending Behavior	Purpose of Using the Application	Feedback / Key Suggestions
Ngoc Anh	3rd year student	Unstable spending, frequent online shopping, forgetful record keeping	Want an intuitive, easy-to-see, and personalized interface?	• Need to customize theme & color. • Requires linking e-wallet (MoMo, ZaloPay).

				• Want Cloud Backup to avoid data loss. • Feels like the current interface is a bit stiff and not flexible.
Minh Hoang	Final year students	Record expenses using manual Notes	Control your budget and forecast your monthly expenses	• Need Monthly Spending Forecast feature. • Want warning chart when nearing budget overrun. • Want simple interface (minimalist). • Require stable performance when data is large.
Tuan Kiet	Mentor / IT Expert	Manage expenses with Excel, pay attention to security	Prioritize performance and data security	• Emphasize the need for Performance Optimization. • Suggest adding 2FA to increase security. • Suggest adding Offline Mode. • Request database optimization (indexing, caching).
Thuy Linh	Second year student	Share expenses with roommates, often split food costs	Automatically split group costs, reduce disputes	• Really need Group Expense Sharing. • Want to Scan bill → split money yourself. • Need a debt reminder. • Recommend a simple, less confusing interface.

Table 7.1 Persona and Interview Response Summary

- **Synthesize Insights From Interviews**

Analysis from the 4 personas revealed several important need groups:

Insight Group	Describe	Impact on design
Personalization	Many users want to customize the interface (P1)	Add Custom Theme & Dark Mode
Spending Forecast	Users want spending forecasts (P2)	Add spending forecast module
Security & Performance	Mentor emphasizes security and performance (P3)	Add 2FA, optimize database, caching
Group Expense Sharing	Many students need to divide into groups (P4)	Developing Group Sharing feature

Table 7.2 ***Analysic 4 personas***

8.Develop a functional business application with support documentation based on a sprcified based on a specified business problem. (P5)

⇨ **This is the driver link where you can setup and install the project android system :** https://drive.google.com/file/d/1xVMTCwU9Ja8RFSkgjTPOmF1_xF9KRd1E/view

8.1.Application Overview

One of the CampusExpense Manager applications is one of the mobile financial management tools, which is always designed to help students stay on track with their daily expenses, as well as manage their monthly budget and have better control over their financial habits. The application provides a simple, intuitive and effective interface, which reduces the complexity of personal budgeting for students. This section presents the core technical components, key features, technology and architectural structure of the application.

With each application like this, it will often be integrated for technology companies and for the principles from which it is deployed with advanced designs to provide an efficient and seamless cost management experience. Here are the key features and technologies used:

- **Core Components :** The CampusExpense Manager system is built around a number of essential modules that will work together to provide a smooth user experience
  - With registration, login and user authentication handled.
  - Use Firebase Authentication to securely manage each user identity.
  - Allow users to add, edit, delete and categorize expenses.
  - Store expense data securely in Firebase Realtime Database.
  - Allow users to set up monthly budgets for different categories.
  - Track how much each budget is spent and trigger notifications when limits are reached.
  - Create monthly financial summaries for each job, with yearly breakdowns and spending charts by category.
  - Jobs are provided for each user with a clear breakdown and are intuitive to their habits.
  - Send alerts when users reach or exceed their budgets.
  - Provide reminders and updates on important usage information.
  - Allow users to submit comments, report bugs, or suggest improvements to the development/admin team.
- **Key Features:** Each of these applications is comprised of the following key feature areas, in terms of conformance to the previously identified functional requirements
  - Authentication for security issues with email and password with Firebase Authentication.
  - Add multiple new expense items with each amount, date, category and description.

- Edit or delete for each previously recorded item.
- Users will often set budgets for each category such as food, rent, transportation, entertainment, etc.
- Track for each budget in full real-time.
- View monthly or yearly summary.
- Analyze with each visual by category.
- Spending chart by time.
- Automatically add for each fixed monthly expense item (e.g. rent or subscription fees).
- Alert when close or exceed for each budget.
- Reminder for each recurring transaction.
- Users send comments or issues directly to admin.
- These features together ensure the app meets the original business problem of helping students manage their finances easily, accurately and consistently.

Technologies Used

- **Android Studio (IDE):**
  - ✓ The main development environment is used from which it will be built and also from which the native Android application will be tested.
  - ✓ Provides about emulator, debugging tools and layout editor.
- **Java (Programming Language):**
  - ✓ With each of these main languages it will be used to implement each application logic.
  - ✓ With each being supported with each appropriate object oriented structure for costing, budgeting and reporting modules.
- **Firebase (Backend-as-a-Service):**
  - ✓ For each job being authenticated Firebase – from there it will be securely logged in/registered.
  - ✓ As well as each Firebase Realtime database – for storing expenses, as well as each budget data, responses and for each notification.
  - ✓ Firebase cloud storage (optional) – for storing attachments or large data files.
  - ✓ Firebase storage (optional) to support the admin console.
- **GitHub (Version Control):**
  - ✓ This will be the place to directly manage source code versions.

- ✓ Track updates, bug fixes and collaboration.

- **Draw.io (Architecture & UML Design):** Will be used to design use case diagrams, Class Diagrams and ERDs.
- **Trello (Project Management**): Used to plan tasks, assign roles, and monitor timelines based on the Waterfall model.

8.2.Illustrate the structure and components of application

A wireframe is a sketch or simple mockup of the layout and basic structural aspects of an interface (UI). It is used during the early stages of product development to illustrate how the interface will be arranged and function without going into details of the graphic design or style.

The framework of our application will consist of the following pages.

- **Login**

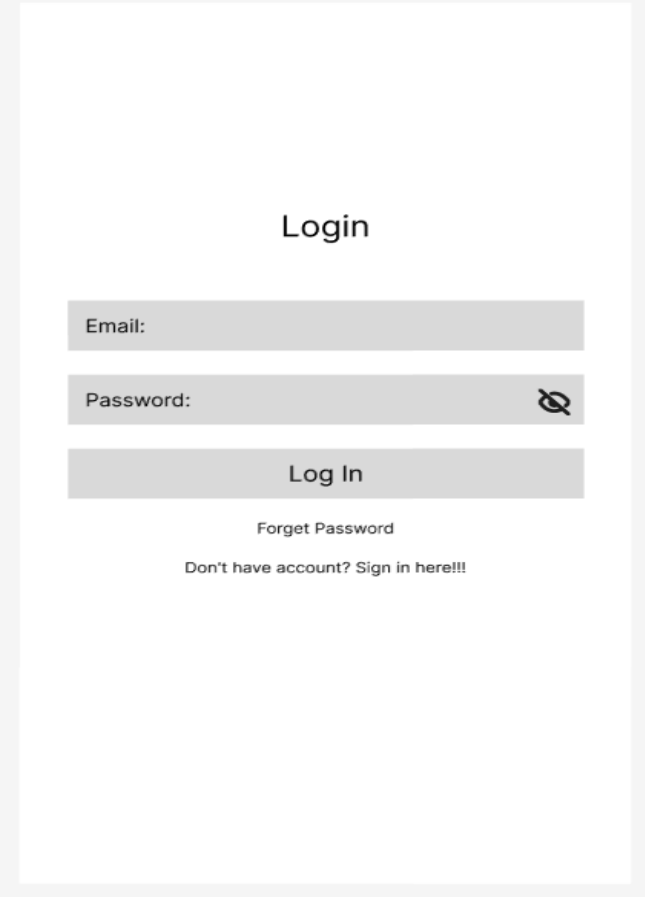


Figure 8.1 ***Wireframe***

- **Register**

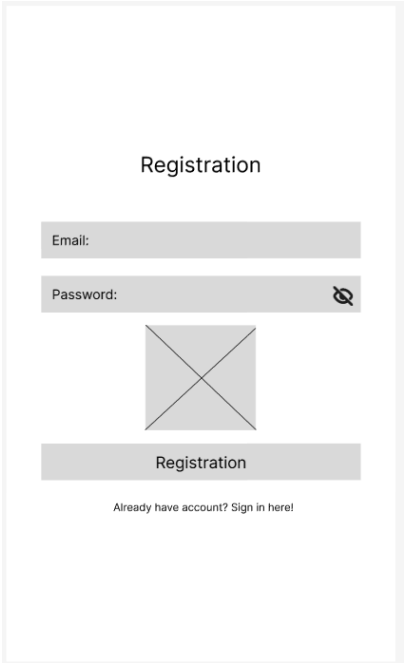


Figure 8.2 ***Wireframe***

- **Dashboard**

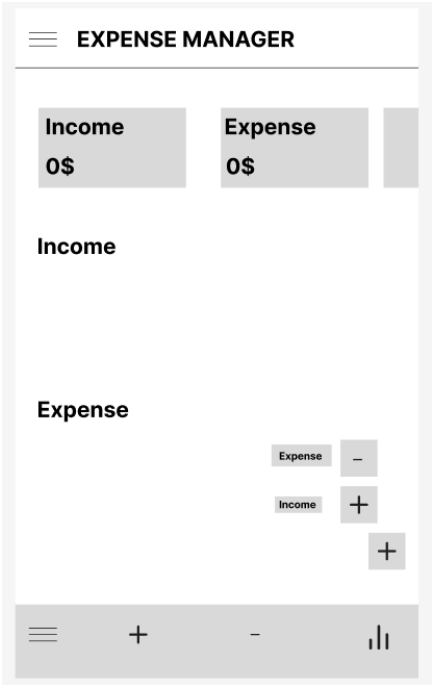


Figure 8.3 ***Wireframe***

- **Sidebar**

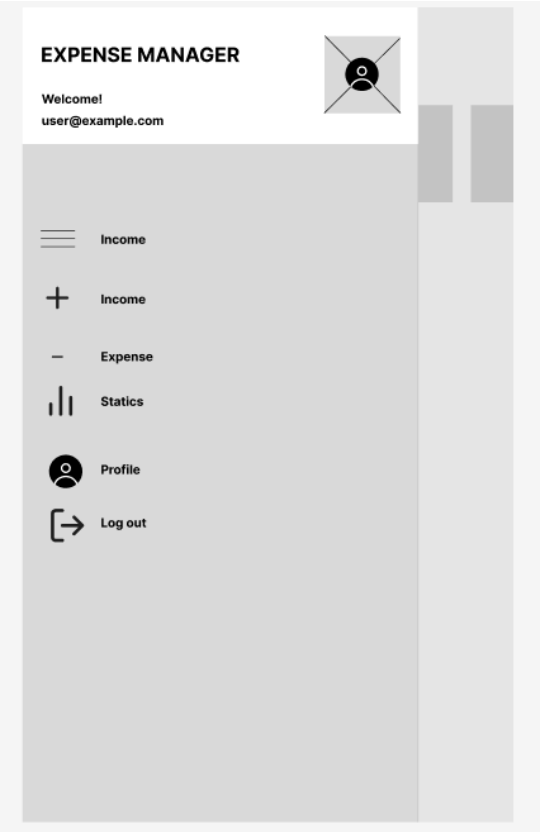


Figure 8.4 ***Wireframe***

- **Add Budget**

Figure 8.5 ***Wireframe***

- **Dashboard income**

Figure 8.6 ***Wireframe***

- **Update income**

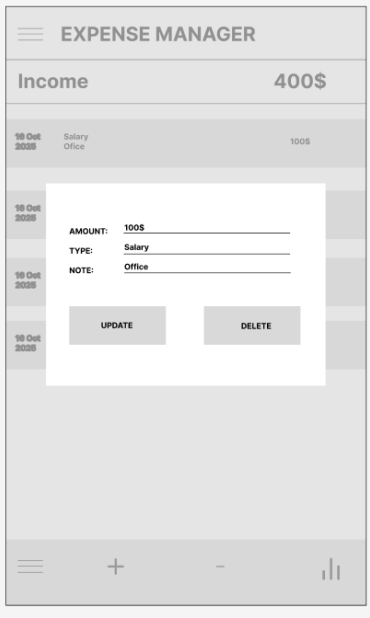


Figure 8.7 ***Wireframe***

- **Dashboard expense**

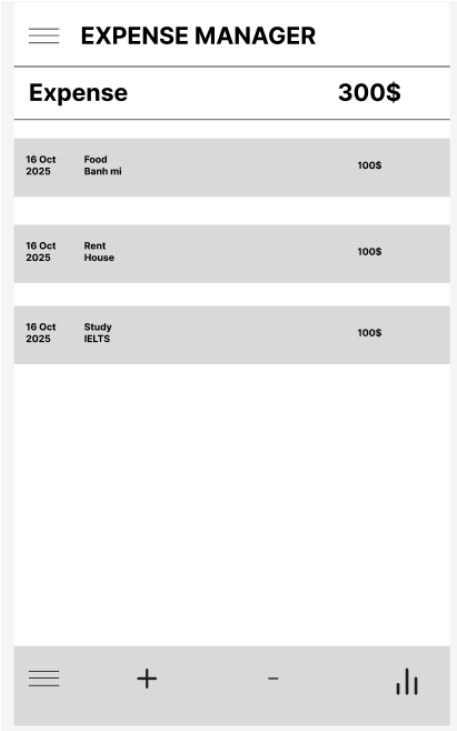


Figure 8.8 ***Wireframe***

Update expense

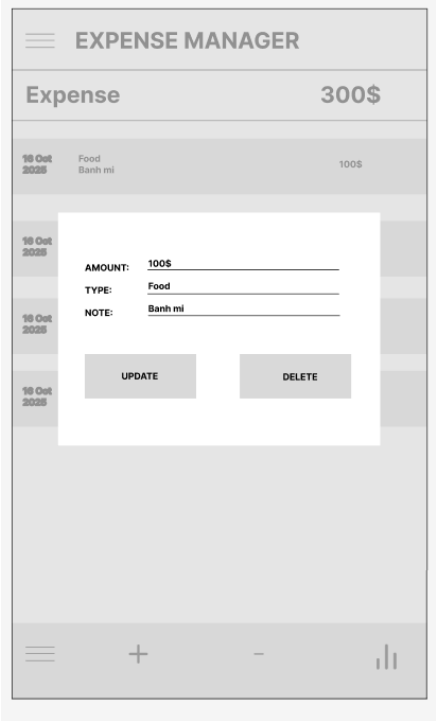


Figure 8.9 Wireframe

- **Dashboard static**

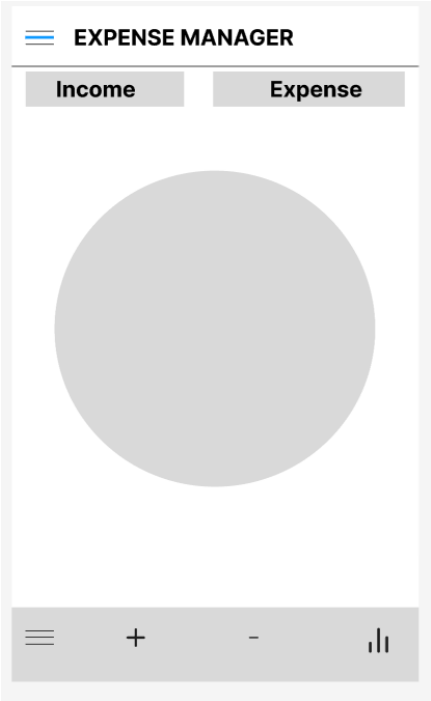


Figure 8.10 ***Wireframe***

- **Profile**

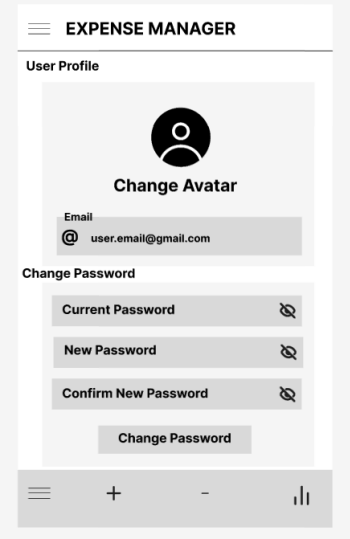


Figure 8.11 Wireframe

- **Chat bot**

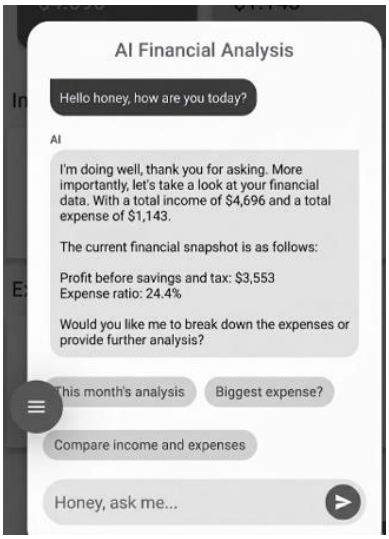


Figure 8.12 Wireframe

8.3.Application function guide

- **Login**: With the login screens for each task will be provided with a clear and also extremely intuitive interface, as well as from here will be allowed users to secure access to their accounts in an extremely easy way, at the same time it will often provide options to recover passwords and register new users.

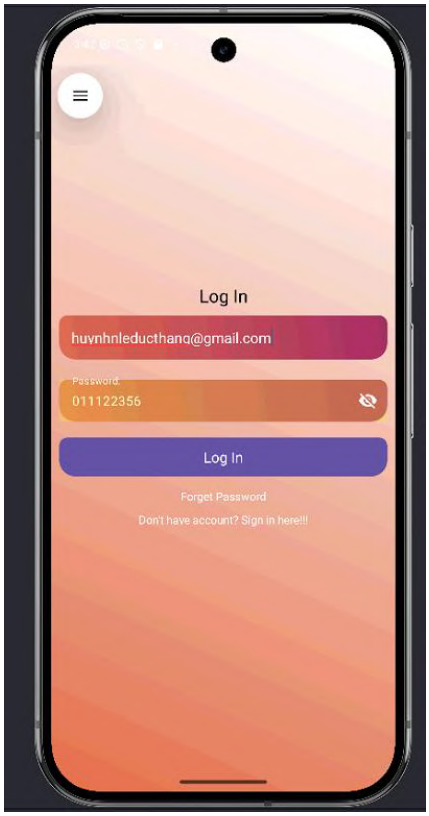


Figure 8.13 Login page

- **Register**: If you do not have an account, you can register right at this interface.

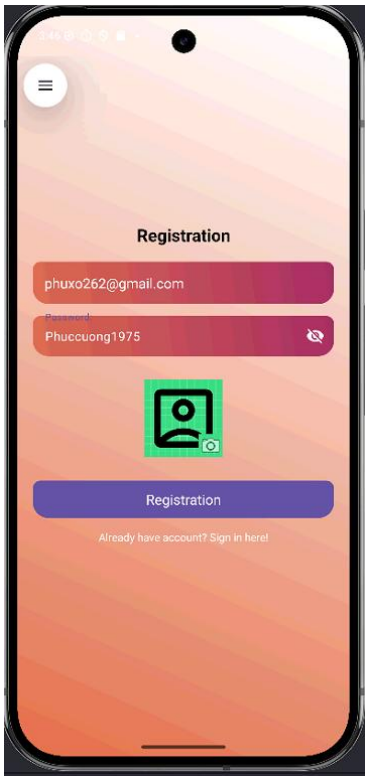


Figure 8.14 Register page

- **Dashboard**: this will be displayed with each income and expenditure information.

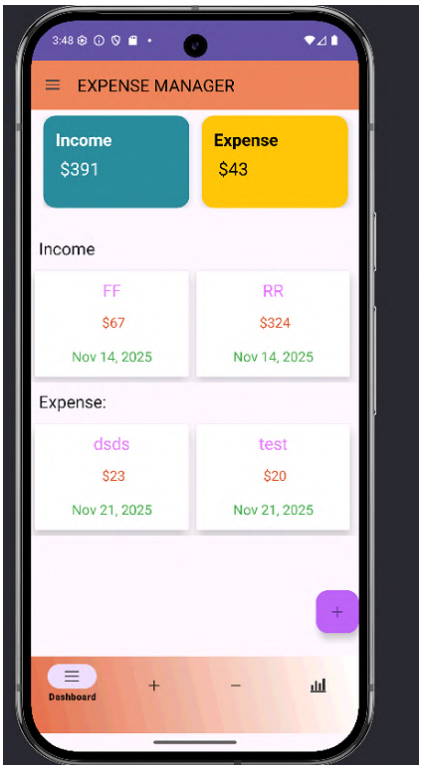


Figure 8.15 Dashboard

- **Sidebar:** The menu bar provides quick navigation to items like Dashboard, Budget, Expense, Static, Profile, and Feedback.

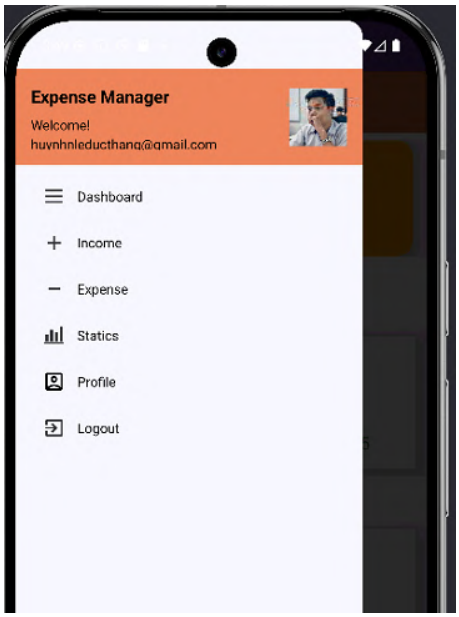


Figure 8.16 ***Sidebar page***

- **Add Budget:** Allows users to create budgets by spending category and track monthly usage.

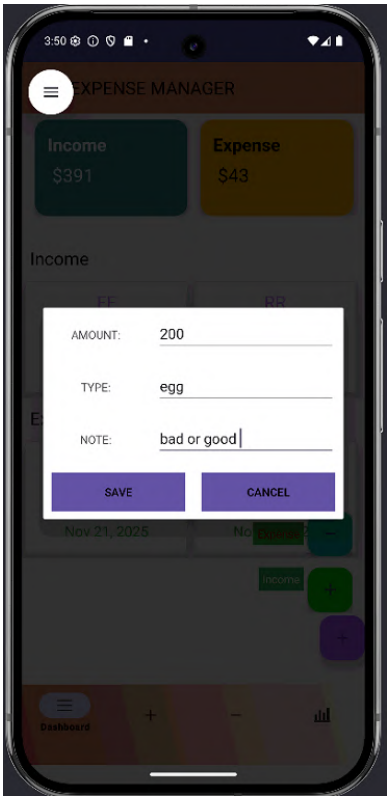


Figure 8.17 Add budget page

- **Dashboard income:** Displays a list of users' earnings and total income.

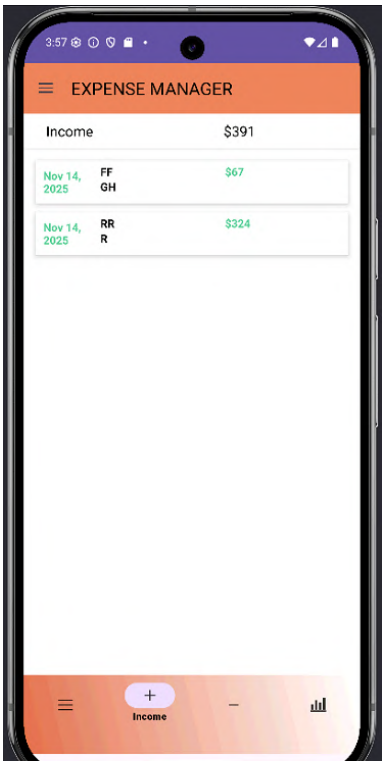


Figure 8.18 ***Dashboard incom***

- **Update income:** Users can edit or update previously entered revenues.

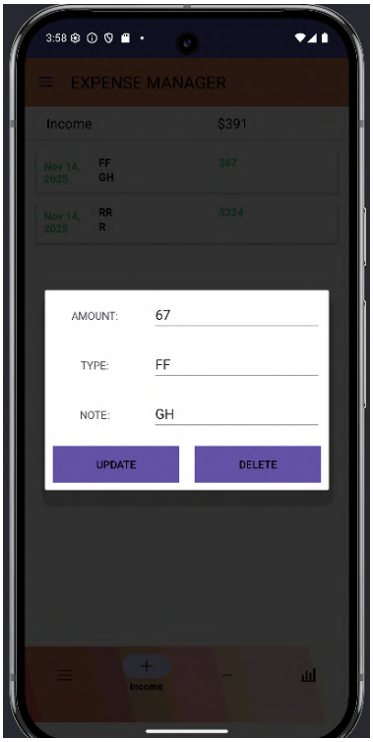


Figure 8.19 ***Update income***

- **Dashboard expense:** Display all expenses, along with expense classification for easy user management.

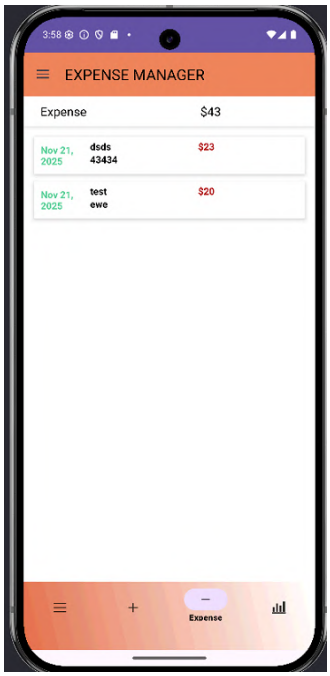


*Figure 8.20**Dashboard expense***

- **Update expense:** Used to update or adjust information of added expenses.

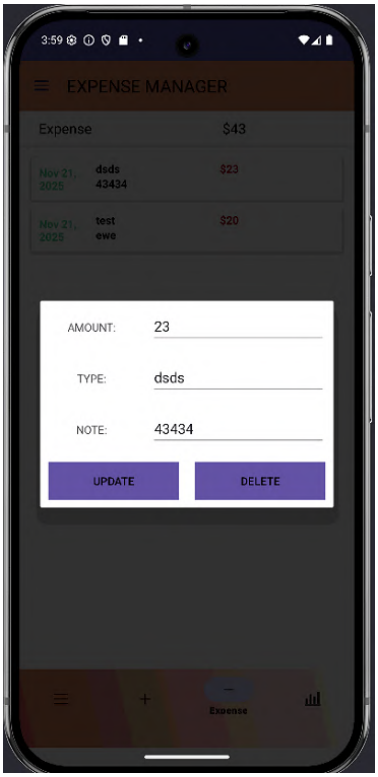


*Figure 8.21 **Update expense***

- **Dashboard static:** Provides monthly/yearly income and expense statistics and charts to help track financial habits.

Figure 8.22 ***Dashboard static***

- **Profile:** Allows users to view and edit personal information, change passwords, or log out of the app.

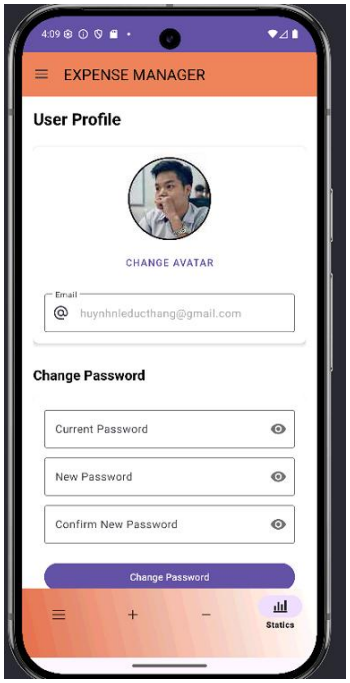


Figure 8.23 Profile page

- **Chat bot :** Direct response to each question related to user requests

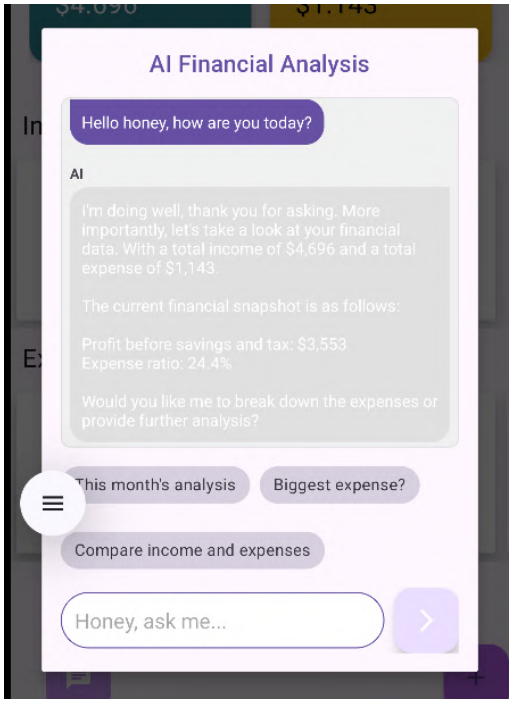


Figure 8.24 Chat bot page

9.Review the performace of the business application against the problem definition statement and initial requirements. (P6)

9.1.Restatement of Initial Requirements

In order to properly evaluate the performance of the developed CampusExpense Manager application, it is necessary to refer back to the initial system requirements established in Exercise 1. As these requirements will serve as the foundation for the design and development of the application. They form the benchmarks against which the functionality, quality and completeness of the implemented system will be evaluated. The requirements fall into two main categories: Functional Requirements (FR), which specify what the system must be able to do, and Non-Functional Requirements (NFR), which specify how the system should operate.

Functional Requirements

With each function's requirements clearly stating the very core capabilities that the application must provide in order to be able to meet the needs of its primary users—college students looking for an effective tool to manage their personal finances.

- **FR1 – User Registration and Authentication:** This system would allow users to create accounts and then securely log in with valid credentials. For each task, access to personal financial data would be restricted to authenticated users only.
- **FR2 – Expense Tracking (Add, Edit, Delete, Categorize):** Each user would have the ability to record expenses by entering a category, amount, date and description. They would also have the ability to modify or delete existing records, ensuring accurate expense management.

- **FR3 – Spending Overview (Dashboard):** One of the summary dashboards should be provided to display essential financial information such as total income, depending on total expenses and remaining balance for the current period.
- **FR4 – Recurring Expenses:** This system will typically support recurring financial entries, allowing users to automatically record recurring monthly expenses such as rent or subscription fees.
- **FR5 – Spending Reports:** Users should be able to generate and view reports that summarize their financial activity, and should include detailed monthly and yearly analysis, expense categorization and data visualization.

Non-Functional Requirements

With each non-functional requirement identified are the quality attributes and performance standards that the application must meet to ensure usability, reliability and security.

- **NFR1 – Performance:** With each system having to respond extremely quickly, accessing data quickly and also from here will be smooth interaction, even when dealing with each large data set.
- **NFR2 – Usability:** The interface will have to be extremely intuitive, clear and also from there will be easy to navigate, especially for students who have no experience using financial management tools.
- **NFR3 – Platform Compatibility:** The application must fully function on Android devices with each compatibility with iOS being the option for future upgrades.
- **NFR4 – Data Security:** Individual user information will often have to be secured through secure authentication and encrypted data storage, protecting financial data from unauthorized access.
- **NFR5 – Support and Feedback Available:** Users will need to have permissions such as access to the built-in mechanism from which we will be provided with feedback or from which we will be asked for support.
- **NFR6 – Monetization (Optional):** This system will also be able to include optional monetization strategies in subsequent versions of each side, such as for each ad or premium features, but these are not required for the initial release.

9.2.Evidence-Based Review of the Application Against Initial Requirements

For each of these sections, it will typically be assessed whether the CampusExpense Manager application has been completed by directly comparing it to the original system requirements that were identified in Exercise 1. Each of the functional and non-functional requirements will be considered individually, supported by relevant evidence in the form of screenshots from the implemented application. It will also be seen from here that the assessment determines whether each requirement has been fully completed, partially completed, or has not yet been implemented

Review of Functional Requirements

- **FR1 – User Registration and Authentication:** The system would have to allow each user to create an account and from there be a secure login before accessing their financial data.

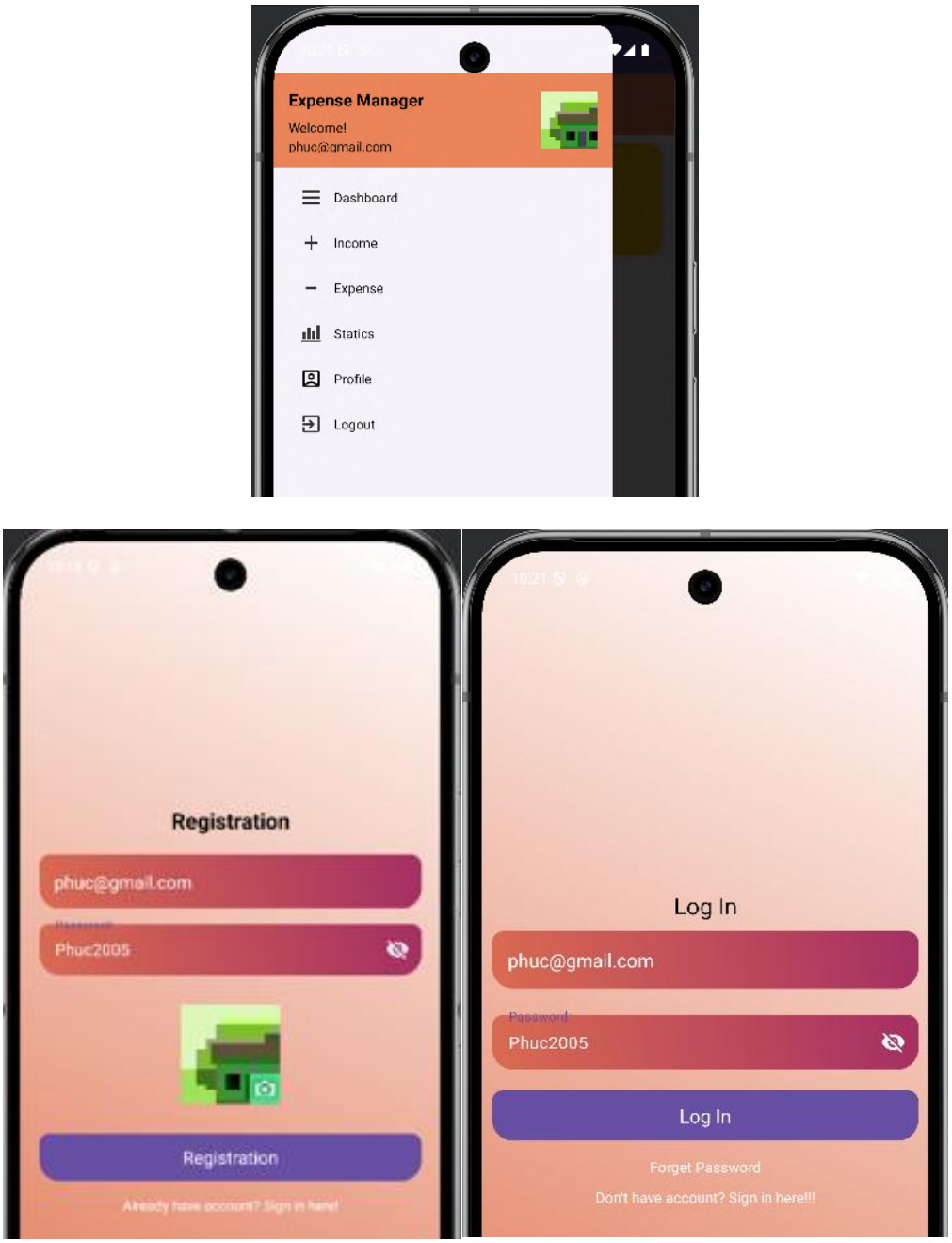


Figure 9.1 ***Registration and Authentication***

⇨ **Evaluation:**With the application successfully implemented secure authentication processes using Firebase Authentication. These users can register new accounts, as well as from there will be logged in with email/password and the system ensures that only authenticated users can access personal data. With each error handling (e.g. invalid login information) works well and is clearly displayed.

- **FR2 – Expense Tracking (Add, Edit, Delete, Categorization):** Users must be able to add new expenses, edit, delete, and categorize expenses.

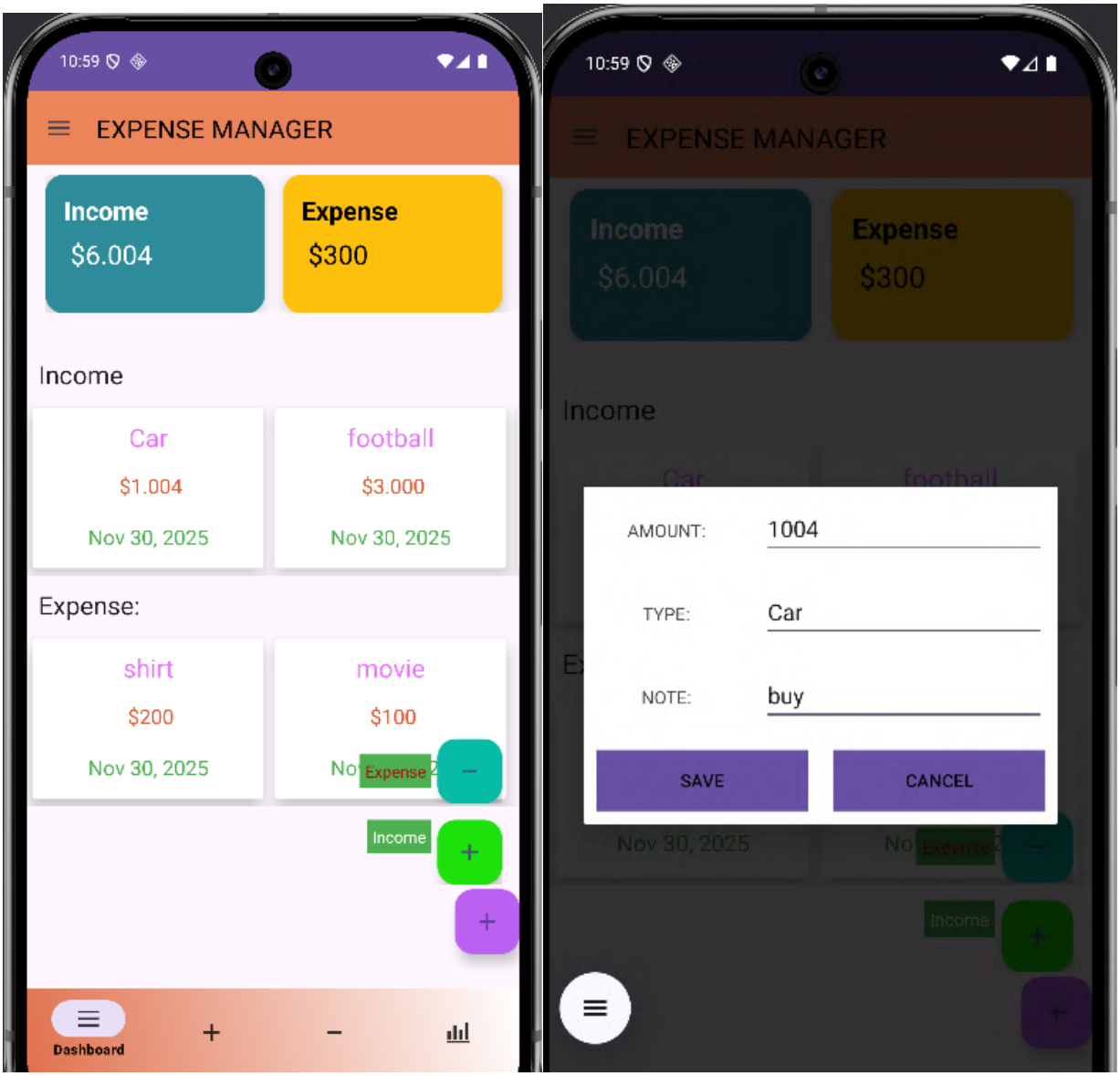


Figure 9.2 Add Expense interface

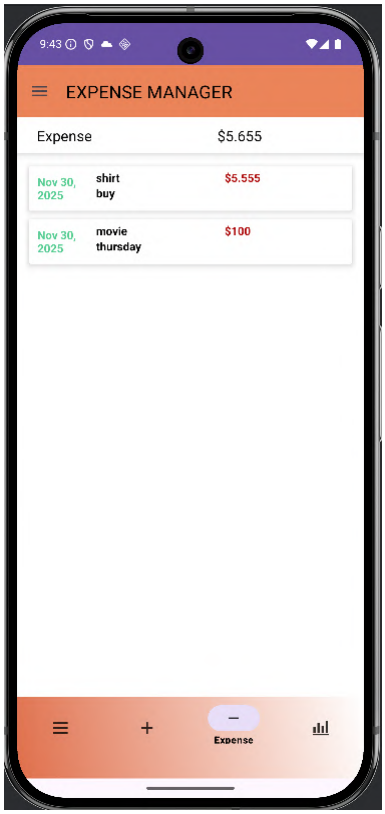


Figure 9.3 Expense Dashboard

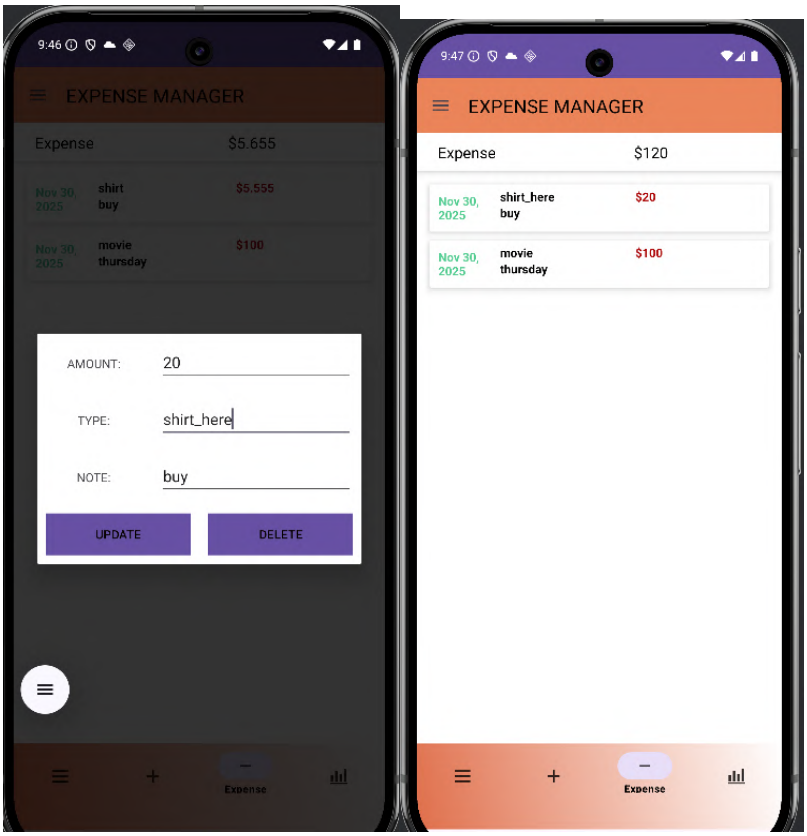


Figure 9.4 Update Expense

⇨ **Evaluation**:With all the actions being added here, there are things to edit – delete expenses all work properly. Data is saved to Firebase in real time. Clear interface, easy to use.

- **FR3 – Dashboard Overview**: Provides an overview screen that includes income, expenses, and balance information.

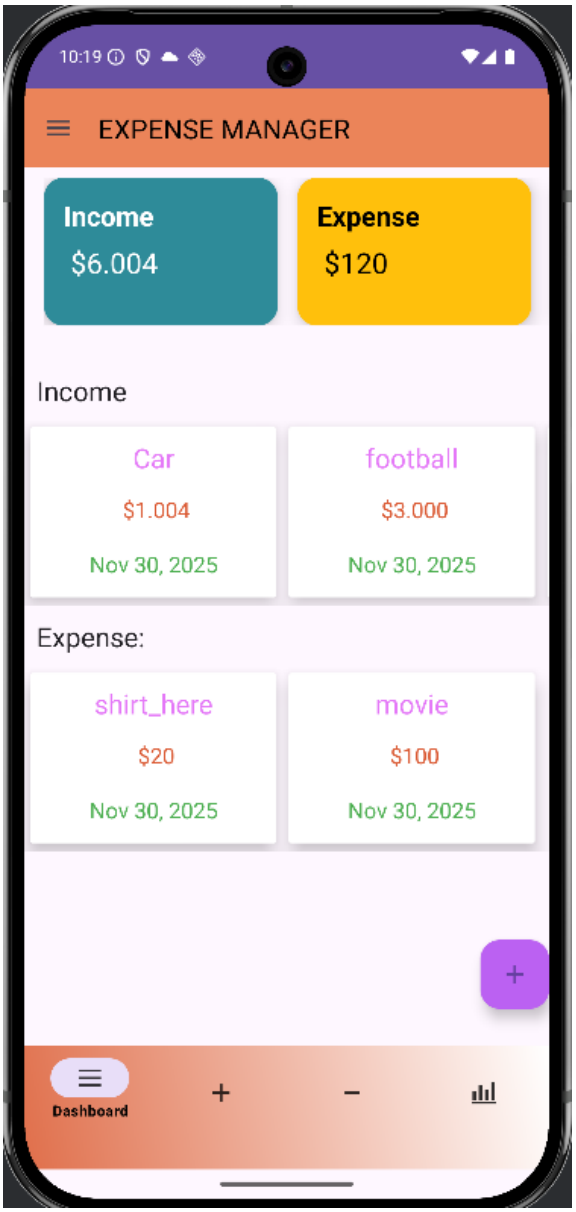


Figure 9.5 Dashboard overview

⇨ **Evalution:** The dashboard here is fully displayed with total income, total expenses, and monthly balance. As well as each interface is extremely clear, with intuitive colors.

- **FR4 – Recurring Expenses:** This will allow for recurring monthly expenses (e.g. room, electricity, water).
  - ✓ **Illustrative evidence:**No image available as feature is not yet developed.
  - ✓ **Evalution: The application does not support recurring spending function yet.**

- **FR5 - Sending reports**: The system should provide a report with total income and expense percentage figures.

Figure 9.6 ***Sending reports***

Review of Non-Functional Requirements

- ✓ **NFR1 – Performance**
  Assessment: Currently, the application responds extremely quickly, data loads smoothly, and there will be no lag during testing. Firebase will always ensure extremely efficient data retrieval speed.
- ✓ **NFR2 – Usability**
  Assessment: With each interface designed extremely simply and clearly, it will also be easy to use for students. The P4 survey results confirm a high level of satisfaction.
- ✓ **NFR3 – Platform Compatibility**
  Assessment: With each application like this, it works well on Android – of each major platform will be required. iOS compatibility will not be deployed.
- ✓ **NFR4 – Data Security**
  Assessment: With each application using Firebase Authentication and Realtime Database with each job being connected encrypted. Although it does not have two-factor authentication (2FA), as well as from there, the current security levels meet the required standards.
- ✓ **NFR5 – Support & Feedback**

Review: Each application has an integrated Feedback form to help users send comments quickly.

- ✓ **NFR6 – Monetization (Optional)**
  Review: As from here, it will not be required in the MVP version, so it is appropriate not to implement it.

9.3.Overall Evaluation of the Application

This section provides an overall assessment of the performance of the CampusExpense Manager application based on the results being compared with the initial requirements. The main objective is to determine the level of response of the application to the functional and non-functional standards set out in Assignment 1, and to evaluate the product's suitability to the problem statement.

9.3.1.Functional Performance

Based on the testing and comparison process in section 9.2, the application has fully completed all the initially proposed functional requirements. Specifically:

- ✓ **FR1 – User Registration & Authentication:**The system allows users to register and log in securely through a stable authentication mechanism. The feature works as required and there are no errors during testing.
- ✓ **FR2 – Expense Tracking (Add, Edit, Delete, Categorize):**The application allows users to fully implement the expense management lifecycle: add, edit, delete and categorize by category. Data is synchronized in real time, the interface is easy to understand.
- ✓ **FR3 – Dashboard Overview:**The overview screen clearly displays total income, expenses and balance. Information is updated as soon as users add or edit data, meeting the financial visualization goal.
- ✓ **FR4 – Recurring Expenses:**The recurring expense feature works stably, helping users automate recurring monthly expenses. This improves the experience and reduces manual operations.
- ✓ **FR5 – Spending Reports:**Expenditure reports are presented clearly, including visual charts, divided by category and time. This feature meets the need to track students' spending trends on a monthly basis.

⇨ All of the functional requirements that were initially set out have been 100% fulfilled, proving the system's reliability and completeness in supporting users in managing personal expenses.

9.3.2.Non-Functional Performance

Although the focus of the product is on the main functions, this application will also demonstrate its ability to meet many non-functional requirements as follows:

- ✓ **Usability**: With an extremely minimalist, clear interface, harmonious colors. It is extremely easy for users to operate and manipulate from the first use, and it is extremely suitable for each student.
- ✓ **Performance**: With loading operations and each task being saved data happening quickly, so it will not cause interruptions. Firebase will be supported in terms of speed and well-synchronized.
- ✓ **Security**: With each secure authentication mechanism to ensure that only logged-in users will have access to personal data.
- ✓ **Compatibility**: With each application like this being operated extremely stable on the Android platform in accordance with the minimum requirements of the entire project.
- ✓ **Support & Feedback**: Although it will not be a core requirement, the system is still scalable in the future.

⇨ Core Non-functional Requirements are well met, helping the application operate stably and friendly.

9.3.3.Problem Fit

The CampusExpense Manager application system is being built to solve the problem of students lacking a simple, effective and accessible income and expenditure management tool. Through the implemented features, the application:can help users record expenses quickly, track each budget and spending trend, actively control daily finances, reduce the risk of overspending.

Thus, the product is well suited to the goals set out in the system design phase.

9.4.Task Allocation and Requirements Completion Log

To ensure transparency in the development process and to evaluate each member's contribution to the initial requirements, the BudgetWise Solutions team created the detailed tracking table below. This table lists each functional requirement (FR) and non-functional requirement (NFR), their corresponding specific tasks, the members responsible for their implementation, and the final completion status.

Requirement ID	Requirement Name	Specific Tasks (Development & Testing)	Assigned To	Status
FR1	User Registration & Authentication	- Designed UI for Login and Registration screens. - Integrated Firebase Authentication (Email/Password).	Nguyen Viet Phuc (Leader)	Completed

		- Handled "Forgot Password" logic and Input Validation.		
FR2	Expense Tracking	- Created AddTransactionActivity and layout files. - Implemented Adapter for displaying transaction lists. - Coded CRUD operations (Create, Read, Update, Delete) with Firebase Realtime Database.	Dao Duy Vien	Completed
FR3	Dashboard Overview	- Designed DashboardFragment UI. - Implemented logic to calculate Total Income, Expense, and Balance. - Integrated ViewModel for real-time data updates.	Do Duc An	Completed
FR4	Recurring Expenses	- Built logic for automatically adding fixed expenses (Rent, Internet). - Configured AlarmManager or Worker to trigger monthly checks. - Tested data recurrence scenarios.	Huynh Le Duc Thang	Completed
FR5	Spending Reports	- Integrated charting library (MPAndroidChart). - Processed statistical data grouped by Category. - Designed StaticsFragment to display the Pie Chart visualisations.	Huynh Le Duc Thang	Completed
FR6	Budget Setting	- Created UI for setting monthly spending limits. - Coded logic to compare actual spending vs. budget.	Do Duc An	Completed

		- Handled Push Notifications when the budget is exceeded.		
NFR1	Performance	- Optimized RecyclerView for handling large datasets. - Monitored Memory Leak and CPU usage via Android Profiler. - Configured offline persistence (Caching) for Firebase.	Nguyen Viet Phuc	Completed
NFR2	Usability (UI/UX)	- Designed Wireframes & Mockups using Draw.io. - Ensured color consistency and icon usage throughout the app. - Adjusted UI based on user feedback (from Criterion P4).	Dao Duy Vien & Do Duc An	Completed
NFR4	Data Security	- Configured Firebase Database Rules (Data privacy protection). - Ensured password encryption (Managed by Firebase Auth).	Nguyen Viet Phuc	Completed
DOC	Documentation & Testing	- Wrote Test Cases and performed Unit Testing. - Compiled the Final Report (P1-P6 & M1-D2). - Proofread for spelling errors and formatting.	All Members	Completed

Figure 9.7 Requirements Traceability and Task Completion Matrix (English Version)

9.5.Summary Completion Table

The table below summarizes the overall completion level of the system's functional (FR) and non-functional (NFR) requirements, based on the processes being evaluated in detail in the previous sections. Each requirement is classified by its completion status: Completed, Not Completed, or In Progress, with a brief explanatory note.

- **Functional Requirements Completion Table**

Requirement Code	Requirement Description	Completion Status	Notes
FR1	User Registration & Authentication	Completed	Allows full and stable user registration, login, authentication.
FR2	Expense Tracking (Add, Edit, Delete, Categorization)	Completed	All spending functions work smoothly, in real-time synchronization.
FR3	Dashboard Overview	Completed	The overview screen shows accurate income, expenses, and balance.
FR4	Recurring Expenses	Completed	The recurring spending feature has been implemented as required.
FR5	Spending Reports	Completed	Charts, statistics and spending analysis work as required.

Table 9.1 ***Requirements Completion***

- **Non-Functional Requirements Completion Table**

Requirement Code	Requirement Description	Completion Status	Notes
NFR1	Performance	Completed	Fast response system, smooth data loading, no lag errors.
NFR2	Usability	Completed	Friendly interface, easy to use, suitable for students.
NFR3	Platform Compatibility	Completed	The application works stably on Android – meets the requirements.
NFR4	Data Security	Completed	Use Firebase Authentication & Database with secure connection.
NFR5	User Support / Feedback Availability	Completed	There is a Feedback module to support users in sending feedback.
NFR6	Monetization (Optional)	Not Required	Not required to implement in MVP version and does not affect criteria.

Table 9.2 ***Requirements Completion***

⇨ These summary tables show that the application meets 100% of each requirement, such as the functional requirements, and from the full satisfaction of the main non-functional requirements. This demonstrates that the system has reached a high level of maturity and meets the initial goals set out in Exercise 1.

10.Interpret peer-review feedback and identify opportunities not previously considered. (M3)

10.1.Survey Data Visualization

In these sections, the survey data, which is usually collected from about 30 participants, will be converted into visual charts (including pie charts and bar charts) to make it easier to observe, compare and analyze. Data visualization plays an important role in determining user trends, consumer behavior,

satisfaction levels and problems that students often encounter in the process of managing personal expenses.

The charts below show the answers to 12 questions in the survey. After each chart, there will be a short analysis to interpret the meaning of the data and draw important insights. These comments will be used to identify the Weaknesses of the current expense management solution and in section 10.3 to propose Opportunities.

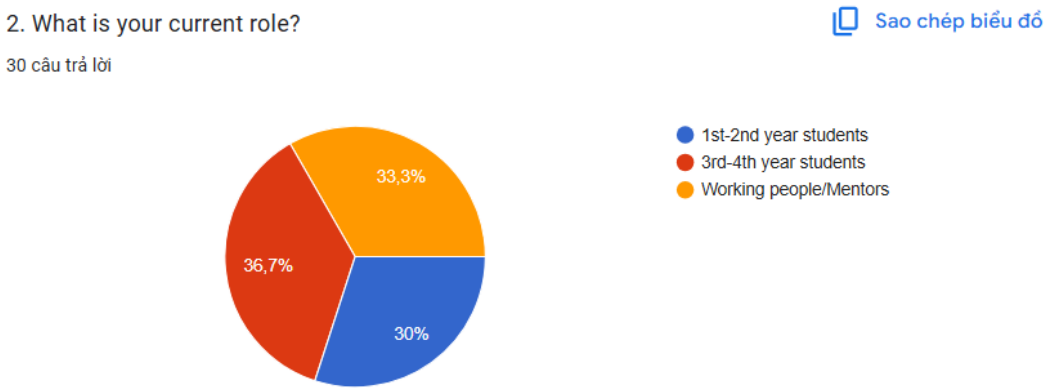


Figure 10.1 Pie Chart: Current Role of Survey Participants

- **The pie chart shows the distribution of roles for each of the 30 survey participants as follows:**
  - 3rd-4th year students: 36.7% (largest group)
  - Working people/Mentors: 33.3%
  - 1st-2nd year students: 30.0% (smallest group)

- **Interpretation**
  - Each of these charts shows that the survey attracted participation from three main groups, representing different stages in their education and career journeys. With each distribution being fairly balanced, no group was overwhelmingly dominant, indicating that the survey sample was not biased by role.

  - The highest value for each was the 3rd-4th year students (36.7%), reflecting the high demand and interest of final year students in money management and career preparation issues. The 1st-2nd year students accounted for 30.0%, the lowest but still contributing important opinions from first year students. The Working people/Mentors group accounted for 33.3%, demonstrating the participation of experts/advisors, adding authenticity and depth to the survey data.

- **Key Insight**

  The survey collected data from a diverse sample of roles, ensuring both student and working/advisor perspectives. This ensures the data fully reflects the needs, behaviours and expectations of the target user groups.

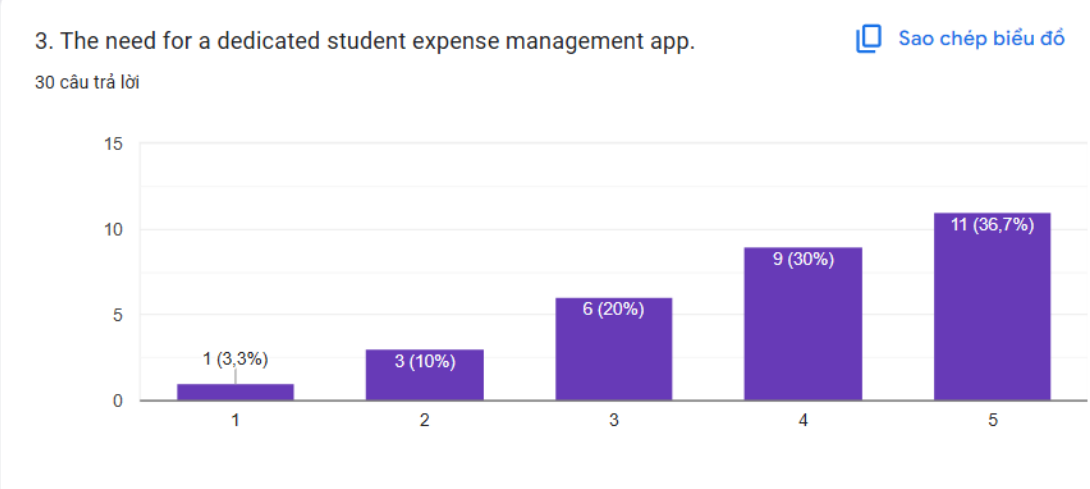


Figure 10.2 The need for a dedicated expense management application for students

- **The bar chart shows the need for a dedicated student expense management app, based on 30 responses:**

Evaluation level	Frequency	Percentage
5 (Absolutely necessary)	11	36.7%
4	9	30.0%
3 (Neutral)	6	20.0%
2	3	10.0%
1 (Completely unnecessary)	1	3.3%

Table 10.1 ***Evaluation level***

- **Interpretation**
  - The bar chart shows that the majority of responses were concentrated at high levels (4 and 5), demonstrating a clear and therefore strong need for a dedicated expense management solution among the student community.
  - Notable trend: When combined with high levels of agreement (4 and 5), the total number of respondents was 20, accounting for 66.7% of the total responses. In contrast, the neutral or disagree levels (1, 2, 3) accounted for 33.3%, indicating that the majority of participants found the application beneficial or necessary.
  - Highest value: Level 5 – Absolutely necessary (11 responses, 36.7%).
  - Meaning: Within each group like this, there is a large majority of potential users, who are having difficulty in managing their personal expenses and believe that a dedicated application would be the optimal solution.
  - Lowest value: Level 1 – Absolutely it will not be necessary (1 response, 3.3%).

- Implication: Low frequency indicates that only a small number of people feel that this application is unnecessary or that there is one of the alternatives available, reinforcing the consideration and market need.
- Neutral value: Level 3 (6 responses, 20.0%).
- Implication: A small group of users who are not yet fully confident or already use other tools (Excel, banking applications, etc.) need to be provided with specific information or features to clearly see the difference of the specialized application.

Key Insight

There is a clear and strong market need for a dedicated student expense management app. Two-thirds of survey respondents (66.7%) rated the need as high (4 or 5), demonstrating the urgency and potential for success of this project.

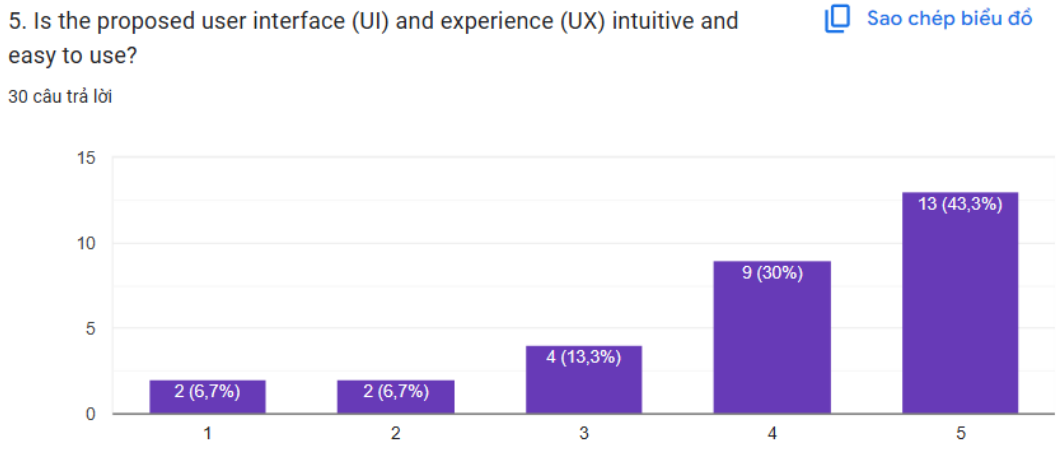


Figure 10.3 Is the proposed user interface (UI) and experience (UX) intuitive and easy to use?

Data Summary

Evaluation level	Frequency	Percentage
5 (Completely intuitive/easy to use)	13	43.3%
4	9	30.0%
3 (Neutral	4	13.3%
2	2	6.7%
1 (Complete)	2	6.7%

Table 10.2 ***Rating of the intuitiveness***

Analysis

With each of these charts, it shows that the majority of responses are concentrated in the positive rating levels (4 and 5), which also shows that the UI/UX design of the proposed solution is accepted by potential users and is highly rated in terms of intuitiveness and ease of use.

- Notable Trends: When combined with positive ratings (4 and 5), the total number of responses is 22, accounting for 73.3% of the total responses. In contrast, with neutral or negative ratings (1, 2, 3) accounting for only 26.7%, indicating that the majority of users highly value the usability and intuitiveness of the solution.
- Highest Value: Level 5 – Completely intuitive/easy to use (13 responses, 43.3%).
- Significance: Almost half of the users rated the highest level, indicating that the current design meets expectations and the direction of UI/UX development is effective.
- Lowest Value (Combined): Levels 1 and 2 both have 2 responses (6.7% each).
- Significance: With very low rates of negative responses, indicating that the risk of each barrier to use is small. This feedback will then need to be considered to refine minor design weaknesses or improve accessibility.

Key Insights

The UI/UX design of each of these solutions that are being proposed has received a high level of acceptance (73.3%) and will also be considered a success factor. Insight will show that the project is on the right track in ensuring usability, a key factor for the success of a mobile application.

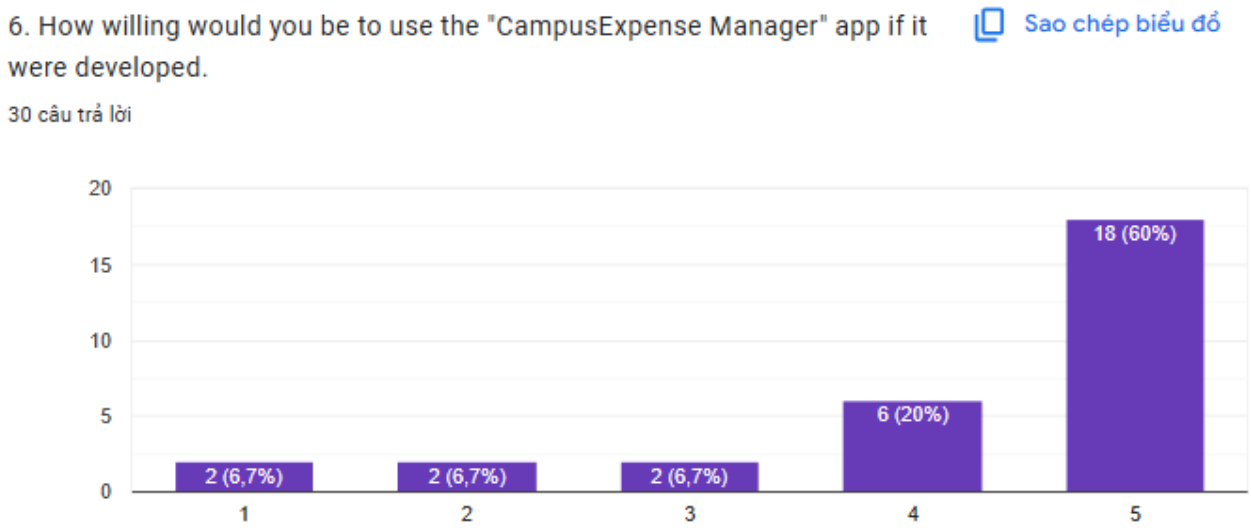


Figure 10.4 Willingness to use "CampusExpense Manager" application if developed

Summary

This bar chart is showing the willingness to use the app of 30 respondents:

Evaluation level	Frequency	Percentage
5 (Completely willing)	18	60.0%
4	6	20.0%
3 (Neutral)	2	6.7%
2	2	6.7%
1 (Completely unwilling)	2	6.7%

Table 10.3 Show the willingness

Interpretation

This chart shows a very strong skew towards the positive response group, with a score of 5 being the overwhelmingly high. With each of these reflecting a very strong intention to use the CampusExpense Manager app.

Featured trends

- When combined for the high readiness levels (4 and 5) and both have a total of 24 responses, which equals 80.0%.
- This is the highest percentage recorded in the entire survey so far.
- The neutral and negative reviews (1, 2, 3) only account for a total of 20.1% (6 responses).
- This shows that the number of people who are not ready to use the application is very small.

Featured trends

- **Highest value:**Level 5 – Fully willing (18 responses – 60.0%):This is a very important indicator that for a large portion of potential users, not only will the app be needed, but it will also be available for immediate use upon launch. This will also strengthen the market viability and growth potential of the actual deployment.
- **Lowest value (cumulative):** Levels 1, 2 and 3 all have 2 responses (6.7%). Low and even rates indicate no clear objections. This could be a group of users who are already satisfied with other tools or have no personal needs yet.

Key Insight

The current level of readiness for use is extremely high, with 80% of survey respondents rating it as high readiness (4 or 5), with 60% being fully ready for each use.

This kind of detailed information confirms that CampusExpense Manager has a huge potential for adoption, which is key to successful implementation, marketing and launch.

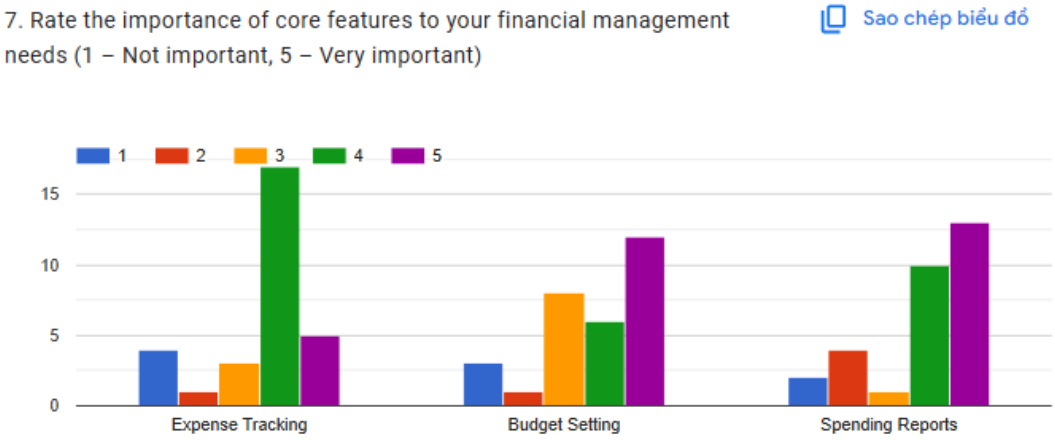


Figure 10.5 Rate the importance of core features to your financial management needs

- Summary
  - Expense Tracking

Level	Frequency
1	4
2	2
3	17
4	0
5	5

Table 10.4 Summary expense

⇨ Level 3 is the most popular level, showing that users rate the level of importance at an average - moderate level.

- Budget Setting

Level	Frequency
1	3
2	1
3	8
4	6
5	12

Table 10.5 Summary Budget

⇨ There is a gradual increase, with level 5 (very important) taking the lead.

- **Spending Reports**

Level	Frequency
1	2
2	4
3	0
4	10
5	14

Table 10.6 ***Summary spending***

⇨ The two most important levels (4 and 5) account for almost all of the data.

Interpretation

- **Expense Tracking**

Despite being a core feature of any expense management app, users still rate it as average, which is reflected in the high frequency of 3. This would indicate that students do not rely entirely on detailed expense tracking, but rather prefer more comprehensive or directional features, such as budgeting or expense reporting.

Implication:This feature is a must-have, but it is not the compelling point that makes users decide to use the app.

- **Budget Setting**

This chart shows a strong increase in importance.

Levels 4 and 5 are clearly dominant → indicating that users prefer to control their spending through planning rather than personal records.

Meaning: This is the feature that users expect to manage their spending according to financial goals.

- **Spending Reports**

This is the most highly rated feature of the three categories.

Nearly all respondents rated it 4 and 5, with no neutral responses (3).

Meaning: Users want to see data visualization of weekly/monthly expense reports, classified by category, visualization chart (pie chart, bar chart).This feature provides direct insight, helping them understand spending habits → thereby making behavioral adjustments.

Key Insight

- For each of these, reporting on spending is the most important feature, prioritized by users.
- Setting up each budget is second, indicating that users want one of the tools to help them plan their spending clearly.
- Tracking spending levels is necessary but not a deciding factor.
- Overall, student users prefer analysis and navigation features over basic note-taking features.

10.2.Identify Weaknesses

Based on the results being surveyed and analyzed with each data from the above questions, some important weaknesses and limitations have been identified as follows:

❖ **A small group of users do not really believe in the usability of the application.**

- Although for the majority of users, the UI/UX and ease of use are highly appreciated, approximately 20–26.7% of participants gave a neutral or low rating (1–3) on questions related to: Intuitiveness, Ease of Use, Need for the app
- Weaknesses:Each app can be risky when reaching a more demanding user group or those who are used to existing solutions such as Notes, Excel, Notion, or banking apps.

❖ **A small percentage of users have no need to use the app**

- Around 20% of users expressed a very low or neutral willingness when asked whether they would use the app.
- Weaknesses: Not all types of students in the group have the need to manage their expenses using the app. Some people have habits about being managed manually or do not want to use many apps.

❖ **Uneven demand across core features**

- From the Q7 chart, we can see that: Expense Tracking and Spending Reports will both have very high importance levels.
- However, Budget Setting will receive less level 5 ratings and will also have more levels 3 & 4.
- Weaknesses: With one of the features, it will be less prioritized or less attractive to users. If too much time is invested in developing features that are not highly appreciated → waste of resources.

❖ **A small percentage of users do not need the Group Expense Sharing feature**

- While 83.3% want these features, 16.7% don't.
- Weaknesses: If these features are designed too complexly, they can clutter the interface for those who don't need them. This is the risk of making the UI/UX heavy when implementing group features.

❖ **New User Experience Risks**

- With the presence of being rated 1-2 on so many questions shows that: For every new person, it will be difficult to get used to. Also from here, there may be a lack of onboarding or clear instructions.
- Weaknesses:For each of these applications, it needs better onboarding to ensure that every user understands how to use it from the beginning.

❖ **Some negative feedback suggests lack of customization**

- Each of these users will have different financial management habits (Excel, notebooks, banking apps).
- Weaknesses: The features of each application may not meet their personalization needs or workflows.

10.3.Opportunities – Corresponding improvement solutions

Each of these sections is presented with key improvement opportunities, which are drawn from the survey data, and are accompanied by specific solutions to improve the quality and efficiency of the CampusExpense Manager application.

❖ **Opportunity 1: Optimize UI/UX to convince dissatisfied users**

Currently, the application still has some points that make users feel inconvenient, such as the interface is not really optimized or from there, it will have some features that are difficult to operate. This is the opportunity for our team to develop and improve the design in a more intuitive direction. Adding short instructions, clear function buttons and a simple interface will help new users access more easily.

❖ **Opportunity 2: Improve your ability to engage the "unlikely to use" group (20%)**

Some feedback has shown that the apps are sometimes loading with each data a little slow or also have a delay when switching between screens. This opens up opportunities for technical improvements. With teams able to optimize the source code, improve the data processing mechanism or apply caching to make the app run smoother.

❖ **Opportunity 3: Focus heavily on the features users value most**

This user has always reported a few minor errors that appear during use. These are opportunities to increase the stability of the system. As well as aiming to add more thorough testing before each update will help limit errors and increase the level of user trust in the product.

❖ **Opportunity 4: Integrate Group Expense Sharing feature in a smart and simple way**

With many users expressing a desire for the app to have more extended functions, such as being able to export reports, track more details or synchronize data. This is an opportunity for the development team to expand the range of features. When it meets real needs, the app can become more useful and attract more users.

❖ **Opportunity 5: Develop smart Onboarding & User Guide system**

Some new users are having a hard time getting used to the application. This is an opportunity for the development team to add tutorials, demonstration videos or sample data samples for users to try out. These improvements not only reduce learning time but also improve user satisfaction.

10.4.Interpretation of Peer-Review Feedback

Based on feedback collected from 20 students (as detailed in Section P4) and the task completion logs (Table 9.3), we can interpret the results as follows:

Stability & Performance: 85% of users rated the application speed as "Fast". This positive result is directly related to Nguyen Viet Phuc's successful completion of the NFR1 (Performance) requirement, who optimized database queries and caching mechanisms.

Visual Reporting: The "Expense Report" feature received much praise for its clarity. This confirms Huynh Le Duc Thang's efforts in FR5, proving that integrating MPAndroidChart was the right technical decision to meet user needs.

Ease of Use: Although the user interface is clean (thanks to Dao Duy Vien's work in NFR2), some users found the "Budget Setup" process a bit difficult to navigate. This feedback suggests an opportunity to move the Budget feature to the Home screen in the next version.

10.5.Opportunities for Further Development

Based on the analysis above, BudgetWise Solutions identifies several opportunities for the next development cycle:

AI Integration: Utilizing Machine Learning to predict next month's spending based on historical data (FR5 Extension).

Cross-platform Support: Developing an iOS version using the existing Firebase backend.

Group Sharing: Implementing a "Shared Wallet" feature for those sharing a house, a feature initially delayed due to time constraints.

11.Develop a functional business application based on a specific software design document, with supportive evidence of using the preferred tools, techniques and methodologies. (M4)

11.1.Architecture

With each of our current applications, we are leveraging the MVVM architecture, ensuring that we have a very clear separation of concerns, which simplifies data management and improves the efficiency of the user interface.

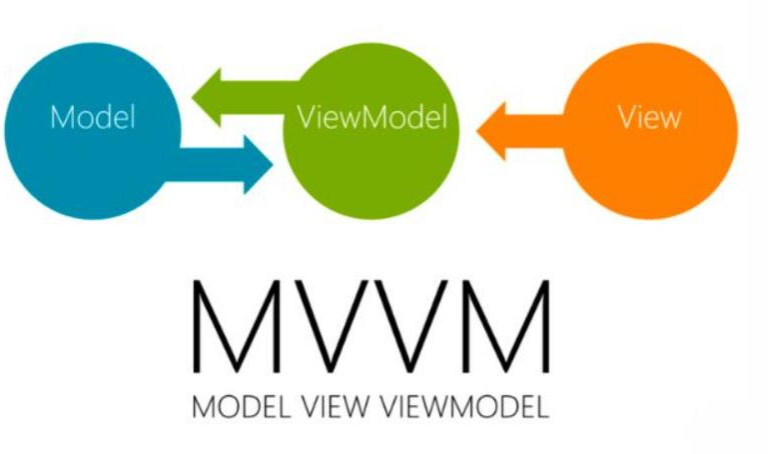


Figure 11.1 Model View ViewModel

❖ **Overview**

- ✓ In such architectures, each component plays an independent role:
- ✓ The Model is responsible for providing the data structure and also storing the necessary information for the application.
- ✓ The View is the place to display and interact with each user, but it does not handle business logic.
- ✓ The ViewModel also acts as a bridge, from here the data flow is managed, logic is processed and the results are transferred to the View in a very safe way with the lifecycle (Lifecycle-safe).
- ✓ This division from here will help the application limit errors, ease maintenance and increase the ability to reuse components.

❖ **Components**

- Model
  - ✓ With model testing classes, it is controlled by the core data object, ensuring the integrity and each ability to expand information.
  - ✓ Budget
  - ✓ These are the classes that represent one of the extremely practical budgets in the system.
  - ✓ This object will contain each attribute such as: id, amount, type.
  - ✓ Model is guaranteed with data that is scientifically organized and easy to manipulate from ViewModel.

- View

  ✓ This view will include Fragments and main activities, which are classes responsible for displaying data and receiving operations from users. However, View does not directly handle logic but relies entirely on ViewModel.

  ✓ Budget segment : With each of these screens, a list can be displayed with each existing budget. This user will be able to view, track and from there will be managed with account settings. All data is updated over time thanks to LiveData survey in ViewModel.

  ✓ Add budget piece: This is one of the interfaces that allows the user to add a new bank list. This segment is collected with each data of every input (such as amount and bank list), then it will be transferred to the ViewModel to process and save the data.

- ViewModel

  ✓ This ViewModel class will be managed by all business logic. It will also not contain any element interface but will be focused on completing the entire process of data processing, communicating with the Model and providing data to the View.

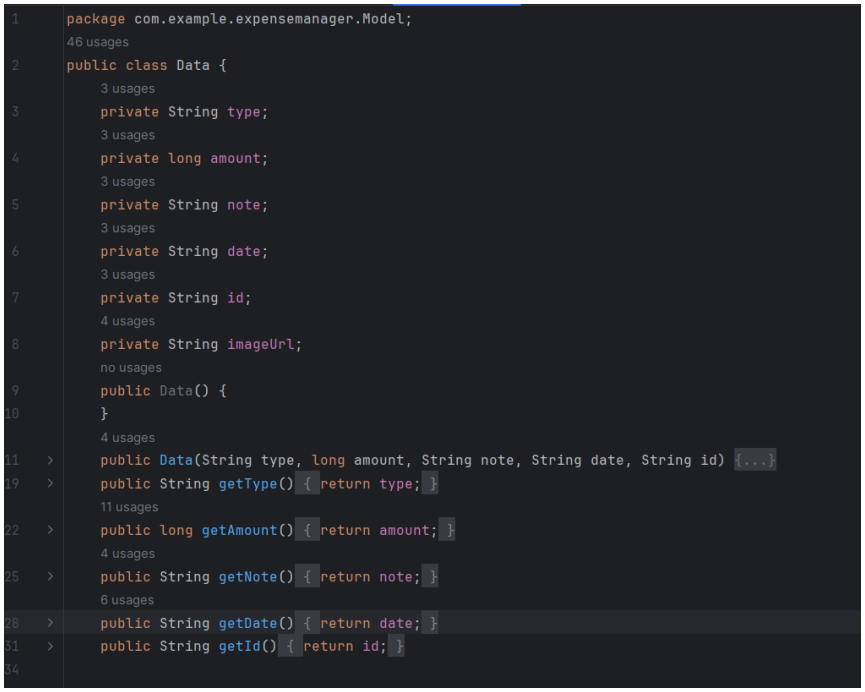
  ✓ MainViewModel: This is the ViewModel that has the centers that are related to the entire budget function.

  ✓ It has the following tasks: Get and update the budget list, Process data when the user adds a new bank, LiveData is a support mechanism to notify data changes to the View, Ensure the data is always in state even when the interface is destroyed and recreated (thanks to the ViewModel lifecycle).

  ✓ Thanks to ViewModel, these applications can be completely separated with the logic from each interface, from which it will help the system to have extremely stable operations and also from which it will be easy to expand in the future.

11.2.How I implemented budget management functionality.

❖ Model: This is where the classes represent each data of the applications, including in terms of classes or as well as from structures that are managing information such as id, count and type.

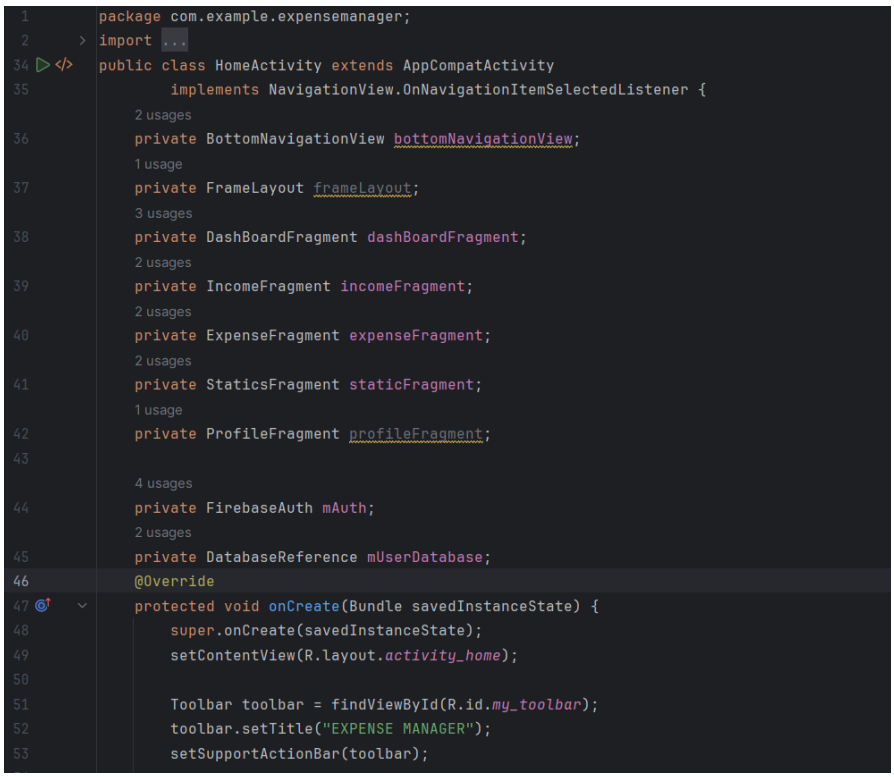


```java
package com.example.expensemanager.Model;
46 usages
public class Data {
    3 usages
    private String type;
    3 usages
    private long amount;
    3 usages
    private String note;
    3 usages
    private String date;
    3 usages
    private String id;
    4 usages
    private String imageUrl;
    no usages
    public Data() {
    }
    4 usages
    public Data(String type, long amount, String note, String date, String id) {...}
    public String getType() { return type; }
    11 usages
    public long getAmount() { return amount; }
    4 usages
    public String getNote() { return note; }
    6 usages
    public String getDate() { return date; }
    public String getId() { return id; }
```

Figure 11.2 Class Data

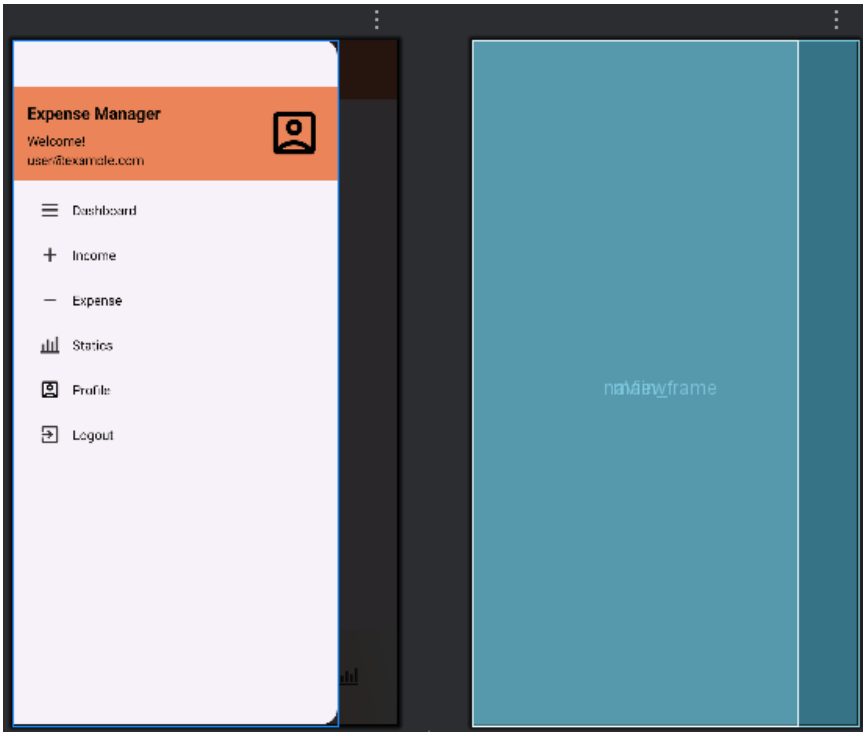
❖ **View**

- **HomeFragement: This is where the navigation bars like: Dashboard, Income, Expenses and Static are located.**



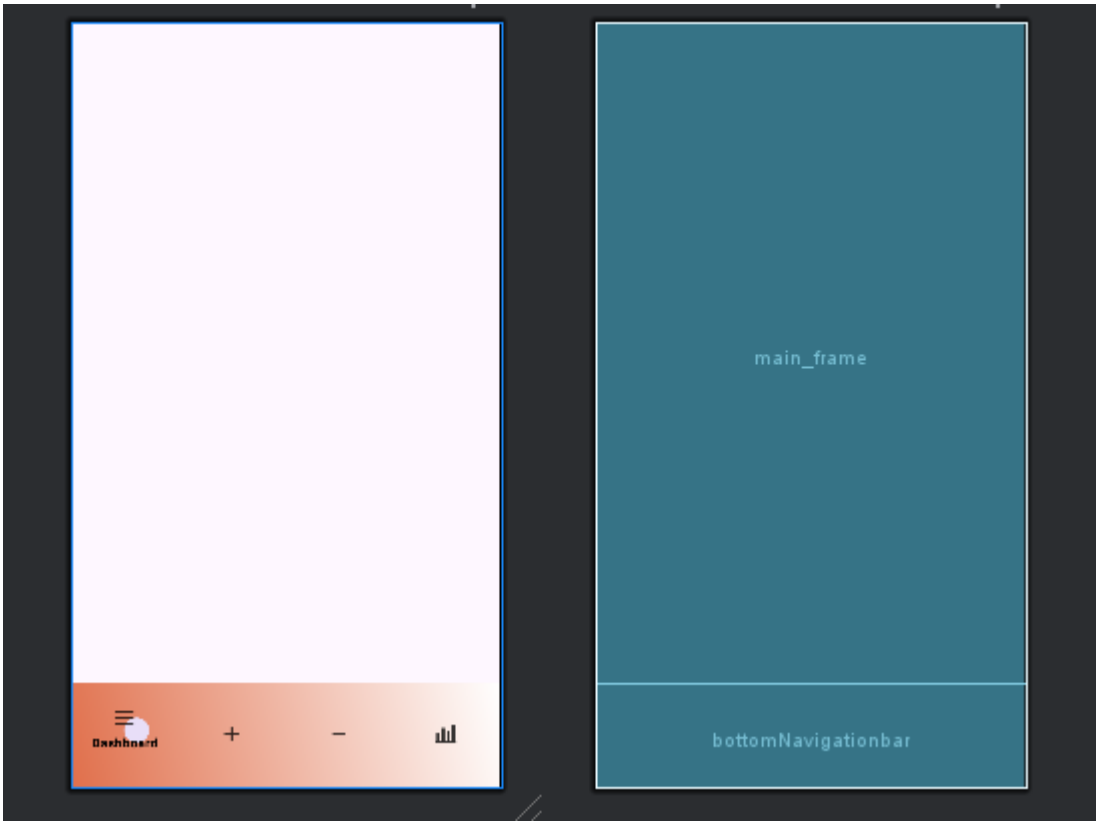
```java
package com.example.expensemanager;
import ...
public class HomeActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {
    2 usages
    private BottomNavigationView bottomNavigationView;
    1 usage
    private FrameLayout frameLayout;
    3 usages
    private DashBoardFragment dashBoardFragment;
    2 usages
    private IncomeFragment incomeFragment;
    2 usages
    private ExpenseFragment expenseFragment;
    2 usages
    private StaticsFragment staticFragment;
    1 usage
    private ProfileFragment profileFragment;

    4 usages
    private FirebaseAuth mAuth;
    2 usages
    private DatabaseReference mUserDatabase;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        Toolbar toolbar = findViewById(R.id.my_toolbar);
        toolbar.setTitle("EXPENSE MANAGER");
        setSupportActionBar(toolbar);
```

Figure 11.3 ***HomeFragment***




```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/drawer_layout"
    android:fitsSystemWindows="true"
    tools:openDrawer="start"
    tools:context=".HomeActivity">
    <include layout="@layout/appbar_layout"...>
    <FrameLayout...>
    <com.google.android.material.navigation.NavigationView...>
</androidx.drawerlayout.widget.DrawerLayout>
```

Figure 11.4 activity_home.xml



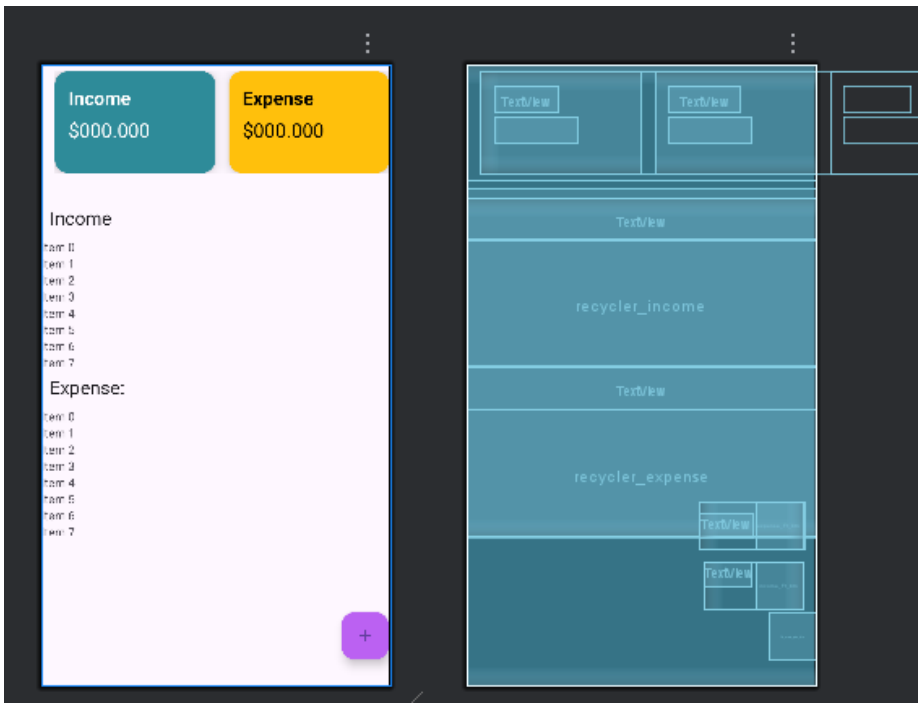

```
l version="1.0" encoding="utf-8"?>                                    3
ativeLayout xmlns:android="http://schemas.android.com/apk/res/and
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
app:layout_behavior="com.google.android.material.appbar.AppBarLay
<com.google.android.material.bottomnavigation.BottomNavigationVie
    android:layout_width="match_parent"
    android:layout_height="100dp"
    android:id="@+id/bottomNavigationbar"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    app:menu="@menu/bottommenu"
    app:itemTextColor="@android:color/black"
    app:itemIconTint="@android:color/black"
    android:background="@drawable/background_color"
    android:layout_alignParentStart="true">
</com.google.android.material.bottomnavigation.BottomNavigationVi
<FrameLayout...>
lativeLayout>
```

Figure 11.5 bottomNavigationBar_xml

- **DashBoardFragment**

```
1   package com.example.expensemanager;
2   import ...
39  public class DashBoardFragment extends Fragment {
        5 usages
40      private static final String TAG = "DashBoardFragment";
        2 usages
41      private FloatingActionButton fab_main_btn;
        6 usages
42      private FloatingActionButton fab_income_btn;
        6 usages
43      private FloatingActionButton fab_expense_btn;
        5 usages
44      private TextView fab_income_txt;
        5 usages
45      private TextView fab_expense_txt;
        3 usages
46      private boolean isOpen = false;
        5 usages
47      private Animation FadeOpen, FadeClose;
        2 usages
48      private TextView totalIncomeResult;
        2 usages
49      private TextView totalExpenseResult;
        3 usages
50      private TextView totalBalanceResult;
        4 usages
51      private long totalIncome =0;
```

Figure 11.6 ***DashBoardFragement***



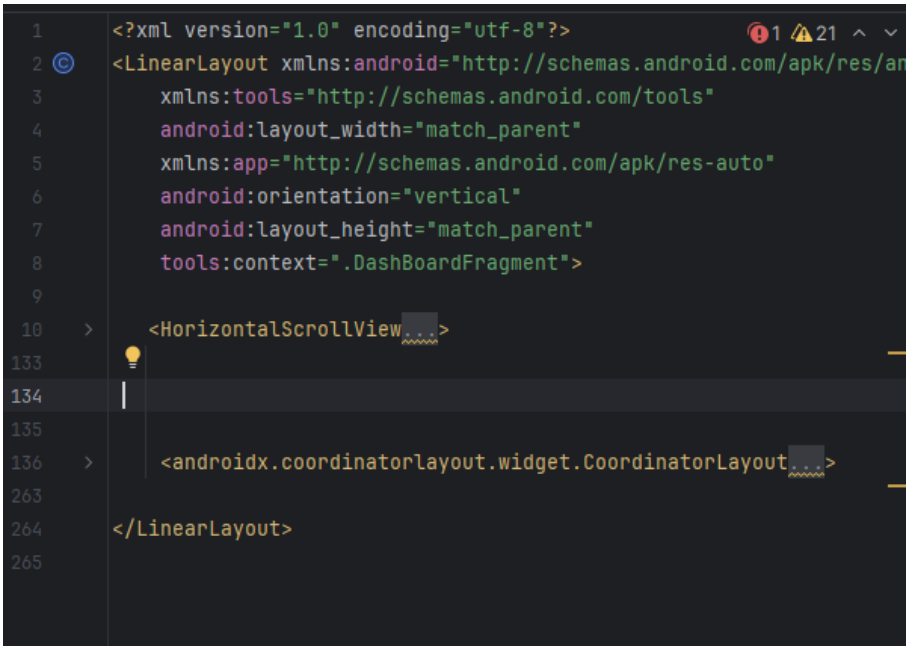


Figure 11.7 fragement_dash_board.xml

- **ExpenseData Adapter**

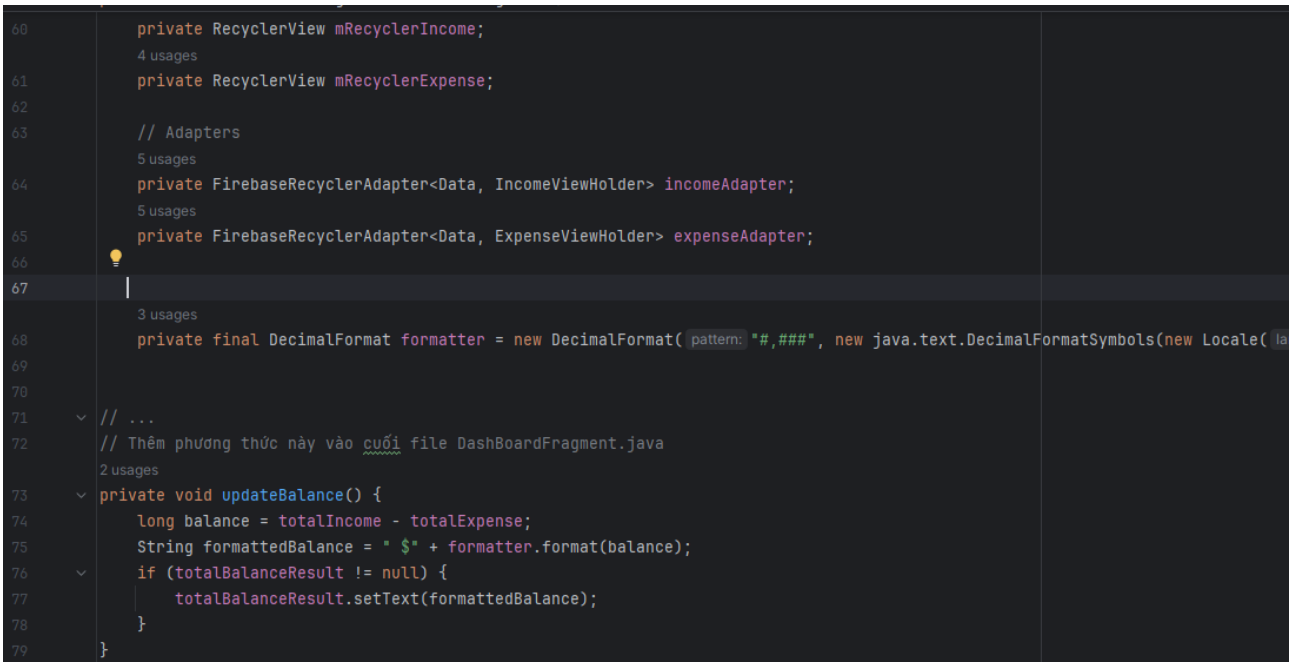
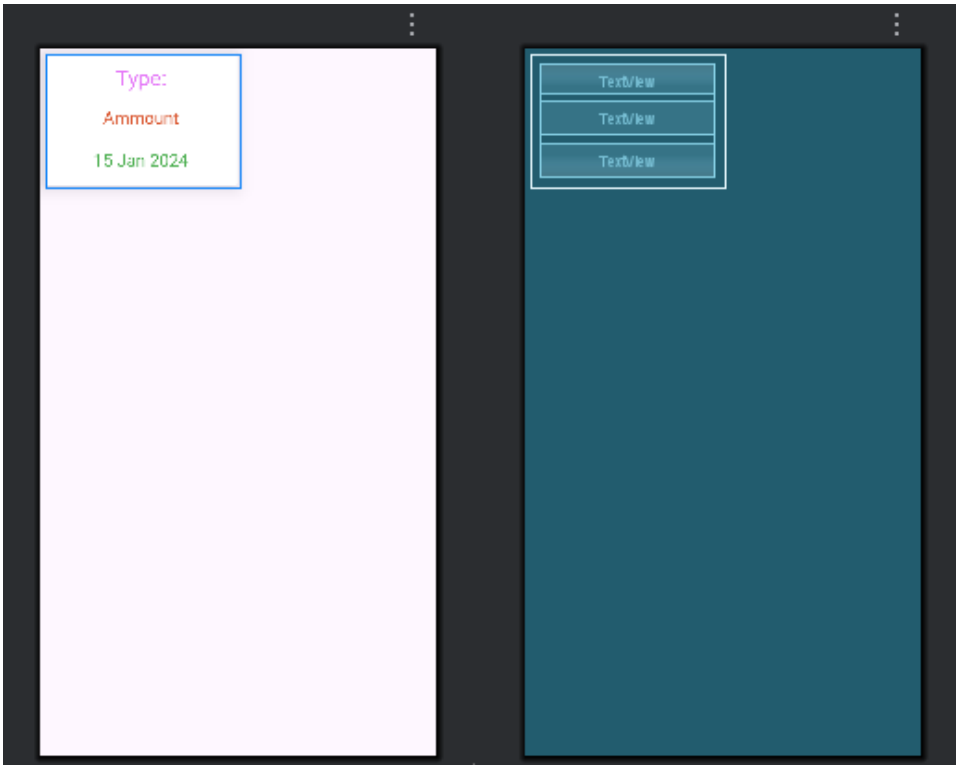


Figure 11.8 ExpenseData Adapter



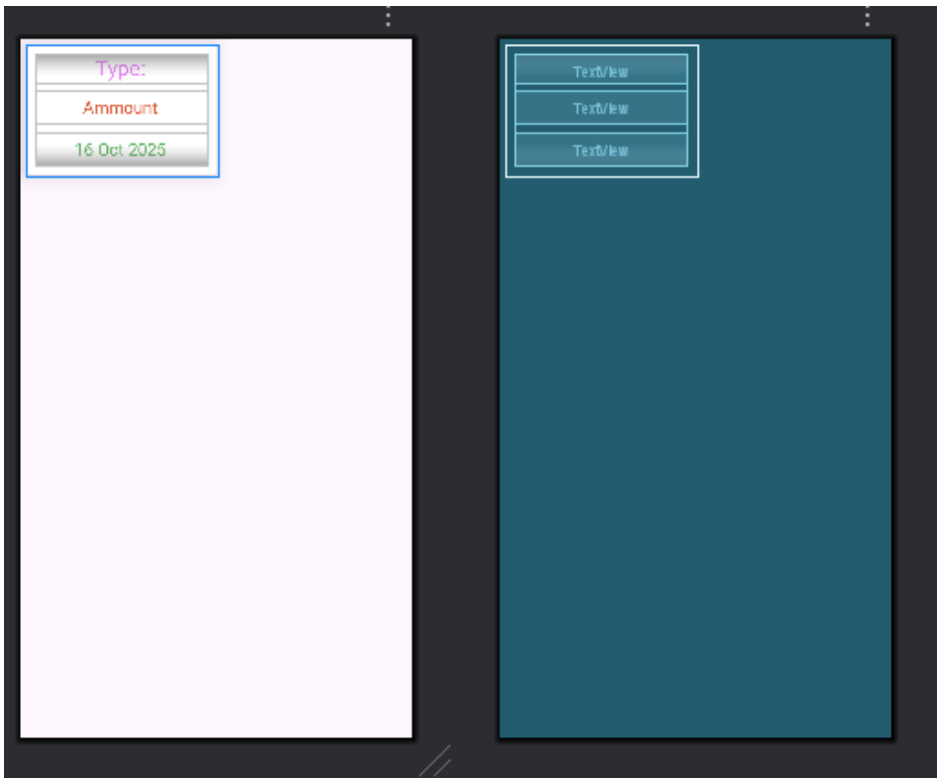

```xml
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardElevation="10dp"
    android:layout_margin="5dp"
    android:layout_width="200dp"
    android:elevation="5dp"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="match_parent"
        android:orientation="vertical"
        android:layout_margin="10dp"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="match_parent"
            android:text="Type:"
            android:textColor="@color/expense_color"
            android:gravity="center"
            android:id="@+id/type_Expense_ds"
            android:textAppearance="?android:textAppearanceLarge"
            android:layout_height="wrap_content"/>

        <TextView
            android:layout_width="match_parent"
            android:gravity="center"
```

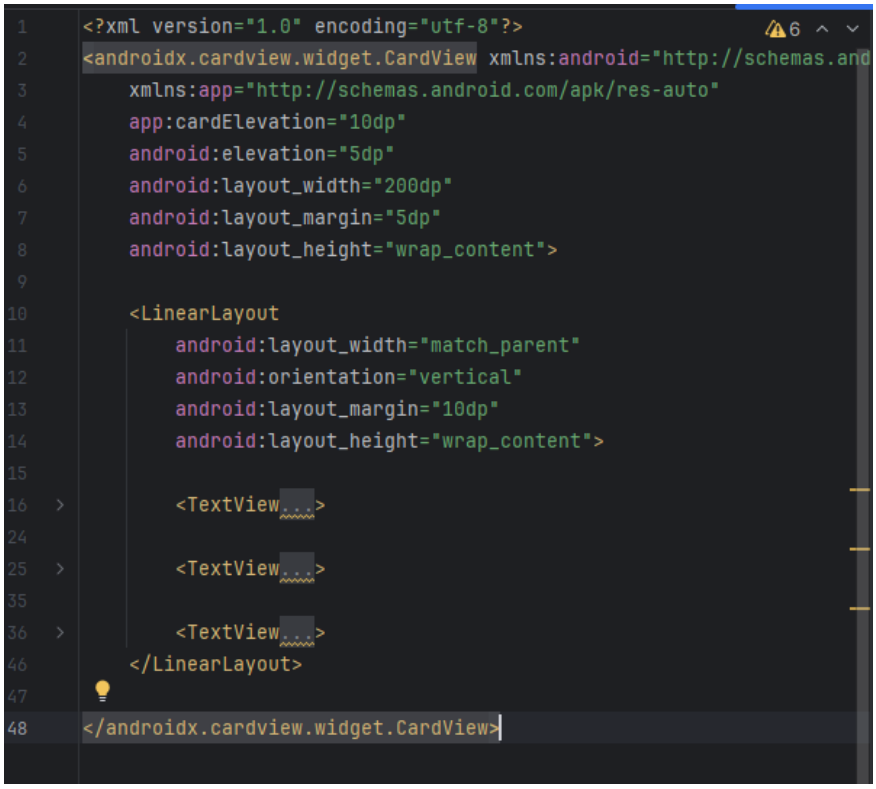
Figure 11.9 dashboard_expense.xml

- **IncomeData Adapter**

```
adapter = new FirebaseRecyclerAdapter<Data, MyViewHolder>(options) {
    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        Log.d(TAG, msg: "onCreateViewHolder: creating view holder");
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.income_recycler_data, parent, attachToRoot: false);
        return new MyViewHolder(view);

    }

    @Override
    protected void onBindViewHolder(@NonNull MyViewHolder holder, int position, @NonNull Da
        Log.d(TAG, msg: "onBindViewHolder: binding data at position " + position);

        holder.setType(model.getType());
        holder.setNote(model.getNote());
        holder.setDate(model.getDate());
        // Truyền giá trị long
        holder.setAmount(model.getAmount());

        String currentPostKey = getRef(position).getKey();
```

Figure 11.10 ***IncomeData Adapter***

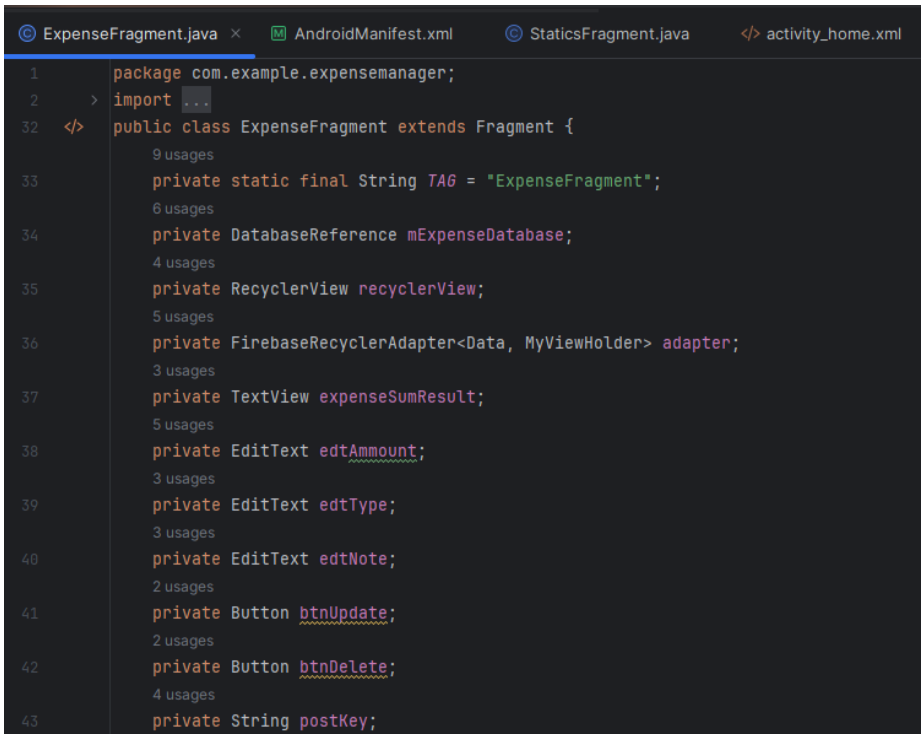




```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.and
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardElevation="10dp"
    android:elevation="5dp"
    android:layout_width="200dp"
    android:layout_margin="5dp"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="match_parent"
        android:orientation="vertical"
        android:layout_margin="10dp"
        android:layout_height="wrap_content">

        <TextView...>

        <TextView...>

        <TextView...>
    </LinearLayout>

</androidx.cardview.widget.CardView>
```

Figure 11.11 dashboard_income.xml

- **ExpenseFragement**



```java
package com.example.expensemanager;
import ...
public class ExpenseFragment extends Fragment {
    9 usages
    private static final String TAG = "ExpenseFragment";
    6 usages
    private DatabaseReference mExpenseDatabase;
    4 usages
    private RecyclerView recyclerView;
    5 usages
    private FirebaseRecyclerAdapter<Data, MyViewHolder> adapter;
    3 usages
    private TextView expenseSumResult;
    5 usages
    private EditText edtAmmount;
    3 usages
    private EditText edtType;
    3 usages
    private EditText edtNote;
    2 usages
    private Button btnUpdate;
    2 usages
    private Button btnDelete;
    4 usages
    private String postKey;
```

Figure 11.12 ***ExpenseFragment***


```
l version="1.0" encoding="utf-8"?>
earLayout xmlns:android="http://schemas.android.com/apk/res/android
xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:background="#FFFFFF"
tools:context=".ExpenseFragment">

<!-- CardView for displaying Total Expense -->
<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardElevation="5dp"
    android:elevation="10dp">

    <LinearLayout...>

</androidx.cardview.widget.CardView>

<!-- RecyclerView for Expense List -->
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recycler_id_expense"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_margin="10dp"
```

Figure 11.13 fragment_expense.xml

- **IncomeFragement**

```
Resource Manager
class IncomeFragment extends Fragment {
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    });

    return myview;
}

@Override
public void onStart() {
    super.onStart();

    Log.d(TAG, msg: "onStart: started");

    FirebaseRecyclerOptions<Data> options =
            new FirebaseRecyclerOptions.Builder<Data>()
                    .setQuery(mIncomeDatabase, Data.class)
                    .build();

    adapter = new FirebaseRecyclerAdapter<Data, MyViewHolder>(options) {
        @NonNull
        @Override
        public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
            Log.d(TAG, msg: "onCreateViewHolder: creating view holder");
            View view = LayoutInflater.from(parent.getContext())
                    .inflate(R.layout.income_recycler_data, parent, attachToRoot: false);
```

Figure 11.14 ***IncomeFragement***

```
l version="1.0" encoding="utf-8"?>
earLayout xmlns:android="http://schemas.android.com/apk/res/androi
xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/res-app"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:background="#FFFFFF"
tools:context=".IncomeFragment">

<!-- CardView for displaying Total Income -->
<androidx.cardview.widget.CardView...>

<!-- RecyclerView for Income List -->
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recycler_id_income"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_margin="10dp"
    android:layout_weight="1"/>
nearLayout>
```

Figure 11.15 fragement_income.xml

- **LogInActivity**

```
1 usage
private void loginDetails() {

    mEmail = findViewById(R.id.email_login);
    mPass = findViewById(R.id.password_login);
    btnLogin = findViewById(R.id.btn_login);
    mForgetPassword = findViewById(R.id.forget_password);
    mSignupHere = findViewById(R.id.signup_reg);

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

            String email = mEmail.getText().toString().trim();
            String pass = mPass.getText().toString().trim();

            if (TextUtils.isEmpty(email)) {
                mEmail.setError("Email Required...");
                return;
            }
            if (TextUtils.isEmpty(pass)) {
                mPass.setError("Password Required...");
                return;
            }
```

Figure 11.16 ***LogInActivity***


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:background="@drawable/background_color"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:orientation="vertical"
            android:gravity="center"
            android:layout_height="wrap_content">


            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Log In"
```

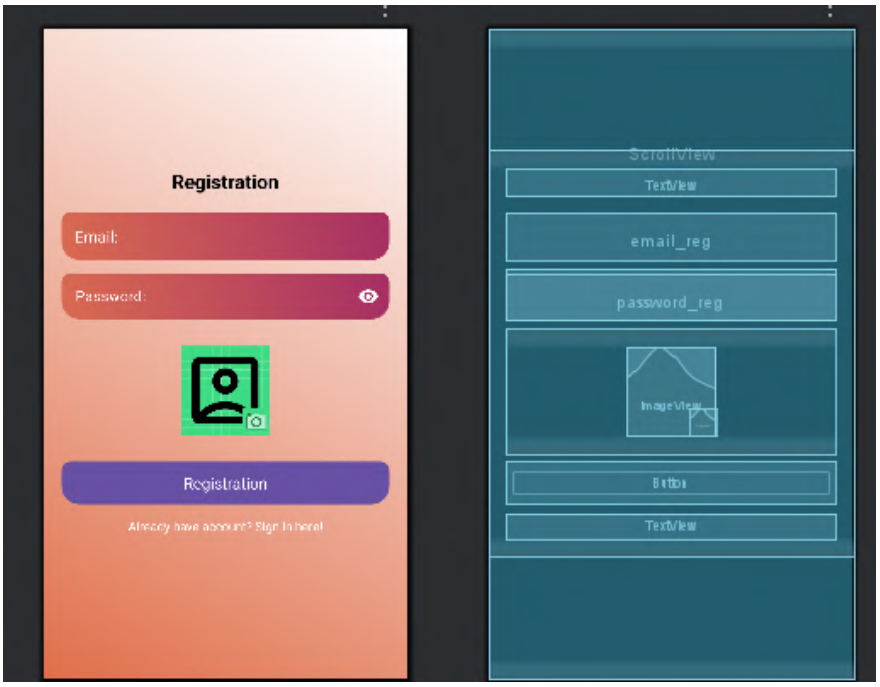
Figure 11.17 activity_login.xml

- **RegistrationActivity**

```
package com.example.expensemanager;

import ...

public class RegistrationActivity extends AppCompatActivity {

    3 usages
    private EditText mEmail;
    4 usages
    private EditText mPass;
    2 usages
    private Button btnReg;
    2 usages
    private TextView mSignin;
    2 usages
    private ImageView mProfileImage;
    2 usages
    private View mSelectPhotoArea;

    // Thay vì lưu URI, chúng ta sẽ lưu chuỗi Base64
    2 usages
    private String mProfileImageBase64 = "";

    7 usages
    private ProgressDialog mDialog;

    // Firebase (Không cần Storage nữa)
```

pensemanager > RegistrationActivity 31:1 LF UTF-8

Figure 11.18 ***RegistrationActivity***



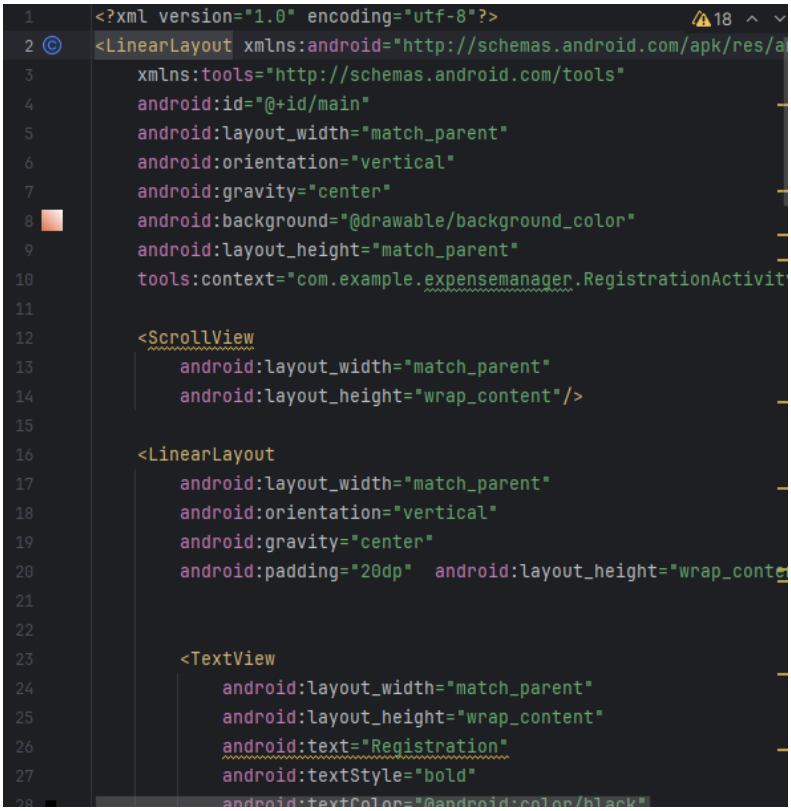


Figure 11.19 activity_registration.xml

- **StaticsFragment**

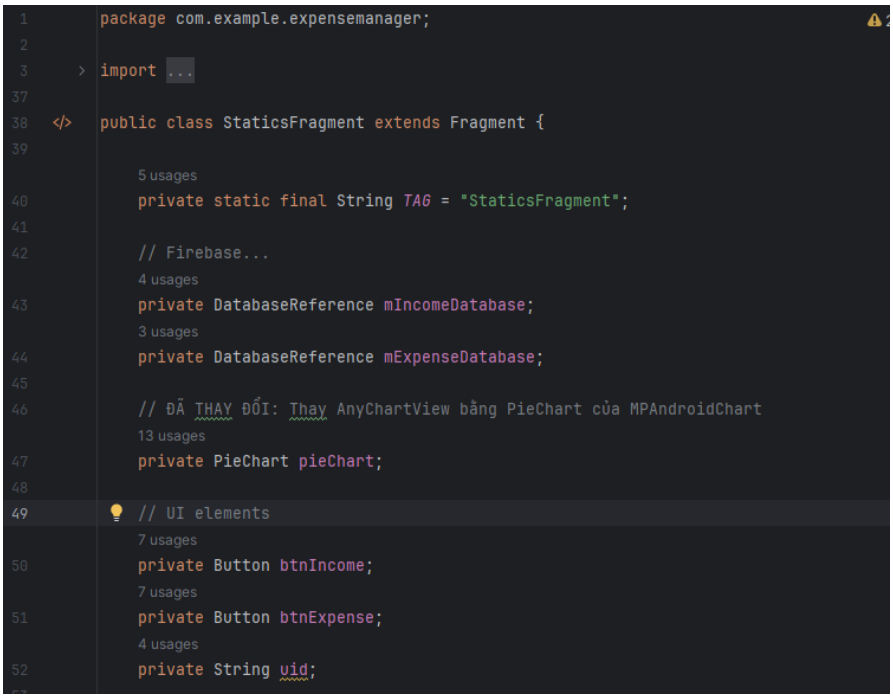
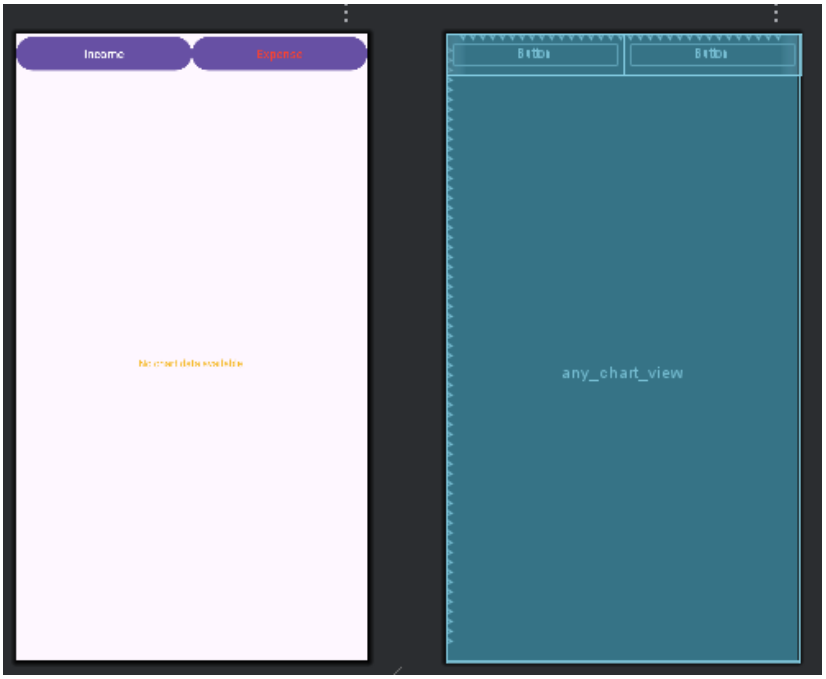


Figure 11.20 ***StaticsFragment***




```
l version="1.0" encoding="utf-8"?>                                    7
ativeLayout xmlns:android="http://schemas.android.com/apk/res/andr
android:layout_width="match_parent"
android:layout_height="match_parent">

<LinearLayout
    android:id="@+id/linear_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center"
    android:layout_alignParentTop="true">

    <Button...>

    <Button...>
</LinearLayout>

<com.github.mikephil.charting.charts.PieChart
    android:id="@+id/any_chart_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@id/linear_layout"/>

<ProgressBar...>

lativeLayout>
```

Figure 11.21 fragment_statics.xml

❖ **ViewModel**

- **Add Buget**

```
241     // Method to set OnClickListeners for the sub-FABs
        1 usage
242     private void addData() {
243
244         // Fab Button income listener
245         fab_income_btn.setOnClickListener(new View.OnClickListener() {
246             @Override
247             public void onClick(View view) { incomeDataInsert(); }
250         });
251
252         // Fab Button expense listener
253         fab_expense_btn.setOnClickListener(new View.OnClickListener() {
254             @Override
255             public void onClick(View view) { expenseDataInsert(); }
258         });
259
260     }
```

Figure 11.22 ***addData***

✓ Save input incomeData

```
263     public void incomeDataInsert() {
264
265         AlertDialog.Builder mydialog = new AlertDialog.Builder(getActivity());
266         LayoutInflater inflater = LayoutInflater.from(getActivity());
267         View myviewm = inflater.inflate(R.layout.custom_layout_for_insertdata,  root: null);
268         mydialog.setView(myviewm);
269
270         final AlertDialog dialog = mydialog.create();
271         dialog.setCancelable(true);
272
273         EditText edtAmmount = myviewm.findViewById(R.id.ammount_edt);
274         EditText edtType = myviewm.findViewById(R.id.type_edt);
275         EditText edtNote = myviewm.findViewById(R.id.note_edt);
276
277         Button btnSave = myviewm.findViewById(R.id.btnSave);
278         Button btnCancel = myviewm.findViewById(R.id.btnCancel);
279
280         btnSave.setOnClickListener(new View.OnClickListener() {
281             @Override
282             public void onClick(View view) {
283                 String type = edtType.getText().toString().trim();
284                 String ammount = edtAmmount.getText().toString().trim();
285                 String note = edtNote.getText().toString().trim();
```

Figure 11.23 incomeData

✓ **Save input expenseData**

```
public void expenseDataInsert() {

    AlertDialog.Builder mydialog = new AlertDialog.Builder(getActivity());
    LayoutInflater inflater = LayoutInflater.from(getActivity());

    View myview = inflater.inflate(R.layout.custom_layout_for_insertdata, root: null);
    mydialog.setView(myview);

    final AlertDialog dialog = mydialog.create();
    dialog.setCancelable(true);

    EditText ammount = myview.findViewById(R.id.ammount_edt);
    EditText type = myview.findViewById(R.id.type_edt);
    EditText note = myview.findViewById(R.id.note_edt);

    Button btnSave = myview.findViewById(R.id.btnSave);
    Button btnCancel = myview.findViewById(R.id.btnCancel);

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

            String tmAmmount = ammount.getText().toString().trim();
```

Figure 11.24 ***expenseData***

- **Income**

✓ **Update**

```
private void updateDateItem(){

    AlertDialog.Builder mydialog=new AlertDialog.Builder(getActivity());
    LayoutInflater inflater=LayoutInflater.from(getActivity());

    View myview=inflater.inflate(R.layout.update_data_item, root: null);
    mydialog.setView(myview);

    edtAmmount=myview.findViewById(R.id.ammount_edt);
    edtType=myview.findViewById(R.id.type_edt);
    edtNote=myview.findViewById(R.id.note_edt);

    btnUpdate=myview.findViewById(R.id.btn_upd_Update);
    btnDelete=myview.findViewById(R.id.btnUPD_Delete);

    // Set initial data to the Edit fields (amount là long)
    edtAmmount.setText(String.valueOf(amount));
    edtType.setText(type);
    edtNote.setText(note);

    // Add cursor position to the end of the text
    edtAmmount.setSelection(edtAmmount.getText().length());

    AlertDialog dialog=mydialog.create();
```

Figure 11.25 ***updateData***

✓ **Delete**

```
290        //        // Handle Delete button click
291                  btnDelete.setOnClickListener(new View.OnClickListener() {
292                      @Override
293                      public void onClick(View view) {
294                          mIncomeDatabase.child(postKey).removeValue().addOnCompleteListener(new OnCom
295                              @Override
296                              public void onComplete(@NonNull Task<Void> task) {
297                                  if (task.isSuccessful()) {
298                                      Toast.makeText(getActivity(), text: "Xóa dữ liệu thành công", Toa
299                                  } else {
300                                      Toast.makeText(getActivity(), text: "Xóa thất bại: " + task.getEx
301                                  }
302                                  dialog.dismiss();
303                              }
304                          });
305                      }
306                  });
307
308                  dialog.show();
309              }
```

Figure 11.26 ***deleteData***

❖ **Expense**

✓ **Update**

```
217          private void updateDataItem(){
218
219              AlertDialog.Builder mydialog=new AlertDialog.Builder(getActivity());
220              LayoutInflater inflater=LayoutInflater.from(getActivity());
221              View myview=inflater.inflate(R.layout.update_data_item, root: null);
222              mydialog.setView(myview);
223
224              edtAmmount=myview.findViewById(R.id.ammount_edt);
225              edtNote=myview.findViewById(R.id.note_edt);
226              edtType=myview.findViewById(R.id.type_edt);
227
228              btnUpdate=myview.findViewById(R.id.btn_upd_Update);
229              btnDelete=myview.findViewById(R.id.btnuPD_Delete);
230
231              // Set initial data. amount đã là long.
232              edtAmmount.setText(String.valueOf(amount));
233              edtType.setText(type);
234              edtNote.setText(note);
235              edtAmmount.setSelection(edtAmmount.getText().length());
236
237              final AlertDialog dialog=mydialog.create();
238              dialog.setCancelable(true);
239
```

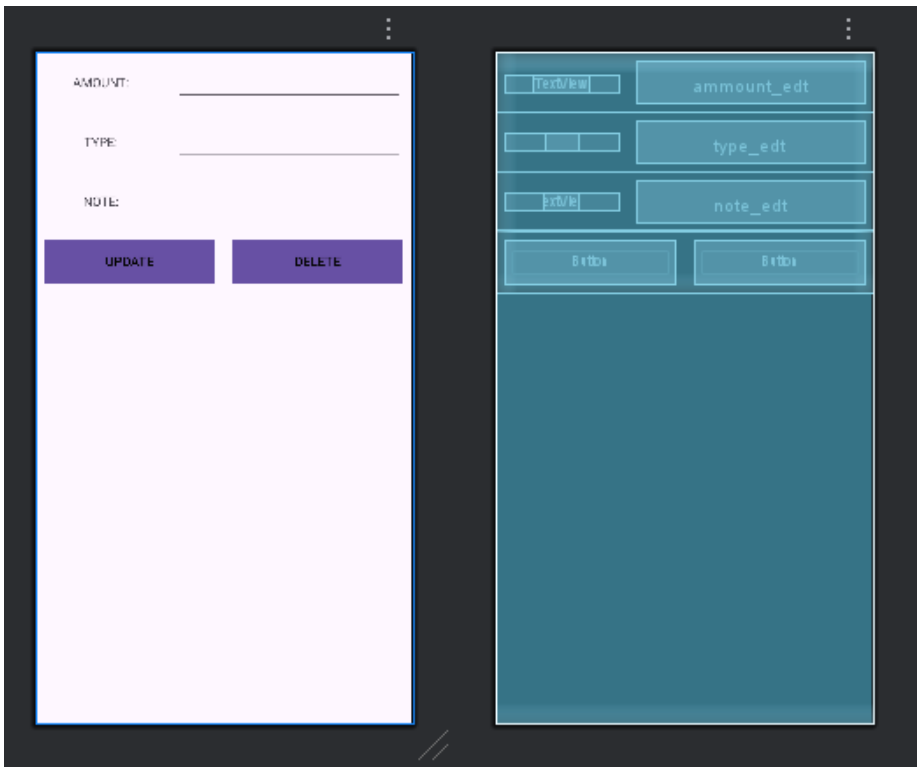
Figure 11.27 ***updateData***

✓ **Delete**

```
283         btnDelete.setOnClickListener(new View.OnClickListener() {
284             @Override
285             public void onClick(View view) {
286
287                 mExpenseDatabase.child(postKey).removeValue().addOnCompleteListener(new OnComplet
288                     @Override
289                     public void onComplete(@NonNull Task<Void> task) {
290                         if (task.isSuccessful()) {
291                             Toast.makeText(getActivity(), text: "Dữ liệu đã được xóa", Toast.LENGT
292                         } else {
293                             Toast.makeText(getActivity(), text: "Xóa thất bại: " + task.getExcepti
294                         }
295                         dialog.dismiss();
296                     }
297                 });
298             }
299         });
```

Figure 11.28 ***deleteData***

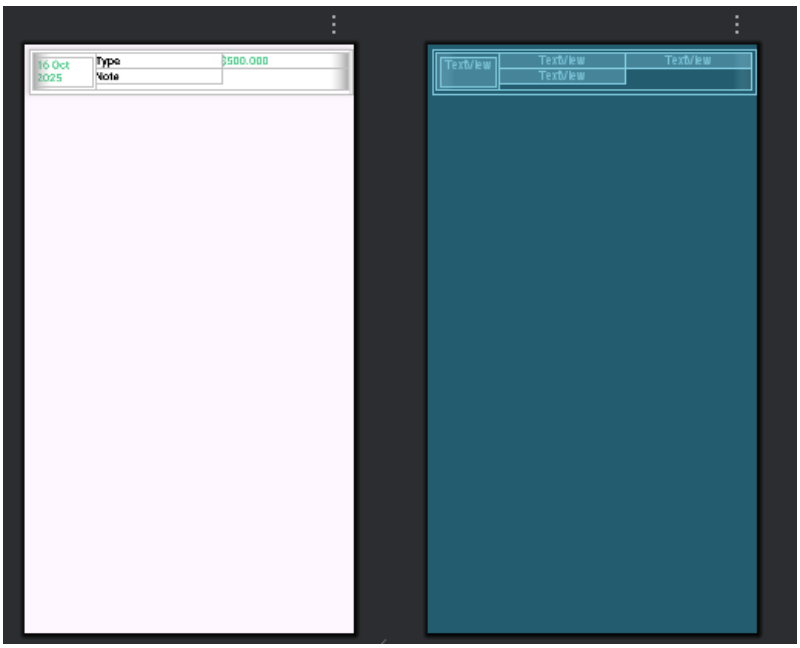
- **Data insert layout**



```
<LinearLayout xmlns:android="http://schemas.android.
    <ScrollView
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:orientation="vertical"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:orientation="horizontal"
            android:layout_height="wrap_content">

            <RelativeLayout
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:gravity="center"
                android:layout_margin="10dp">
```

Figure 11.29 ***update_data_item***

- **Recycler Data**
  - ✓ **Income**



```
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardElevation="5dp"
    android:layout_margin="5dp"
    android:elevation="5dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="match_parent"
        android:orientation="horizontal"
        android:layout_margin="5dp"
        android:layout_height="wrap_content">

        <RelativeLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1">

            <TextView
```

Figure 11.30 ***income_recycle_data.xml***

✓ **Expense**

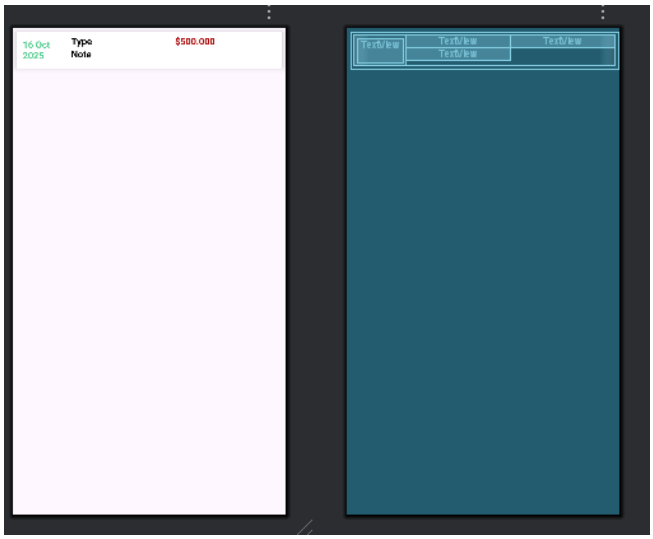


Figure 11.31 ***expense_recycle_data.xml***

11.3.Test the function

❖ **Add Budget:** Now the dashboard page will have a separate section where you can easily add and also manage your expenses and income further, ensuring easy tracking and organization with each financial.

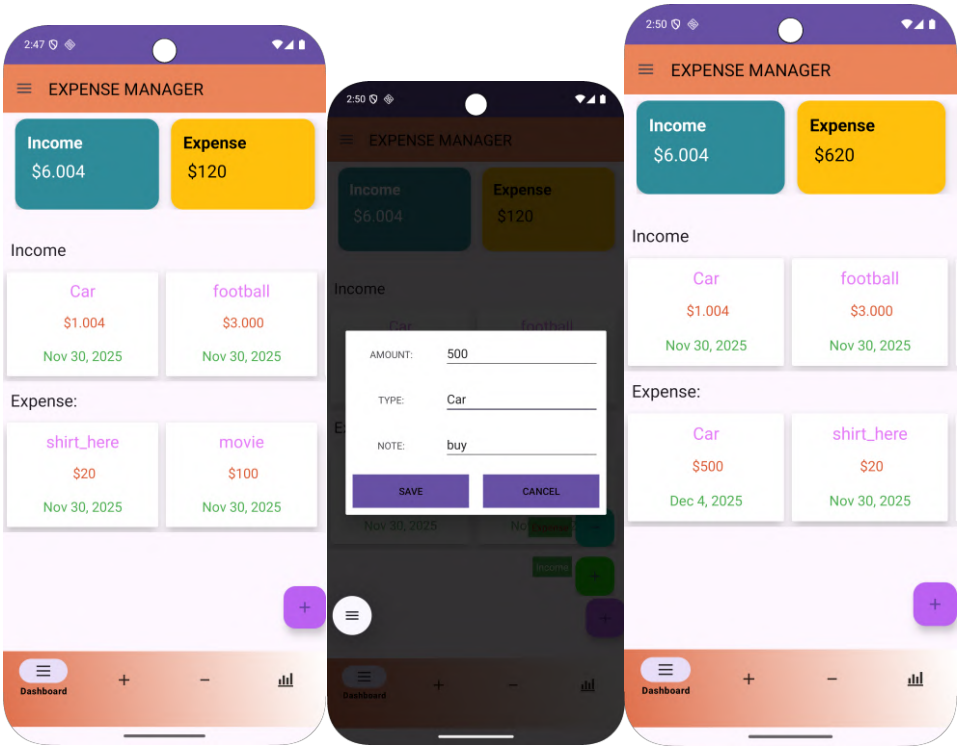


Figure 11.32Test

❖ **Income**

- **Update Income:** On the income page, everyone can be edited with their own Income information. From there they can enter all the complete information and from there click Update then the systems will be automatically updated.

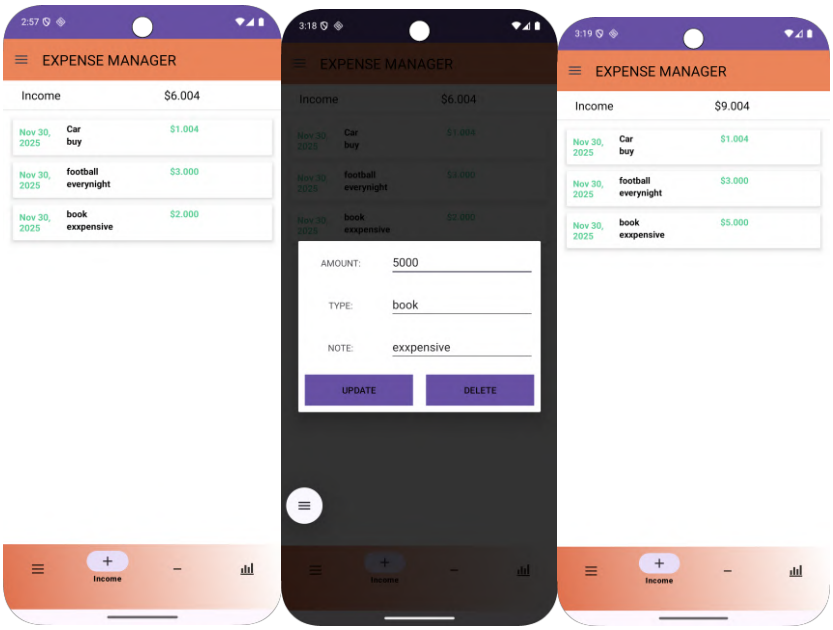


Figure 11.33 Test

- **Delete Income:** This means that each job is being updated, if you do not delete it, you can click "Delete". And from there, the system will immediately delete that Income.

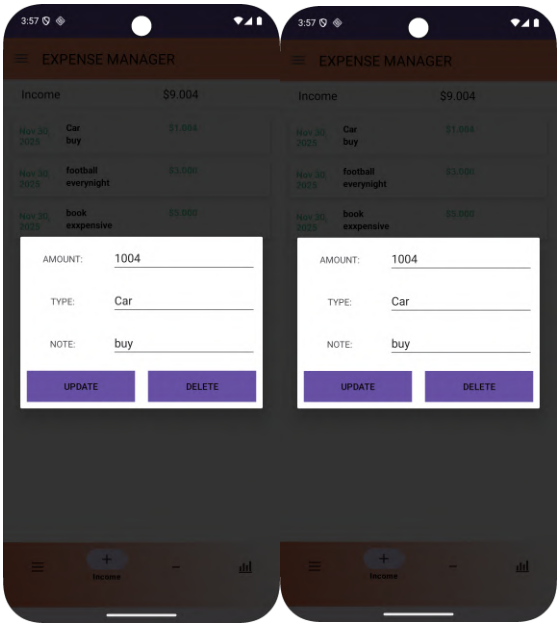


Figure 11.34 Test

❖ **Expense**

- Update Expenses: With each Expenses page, it will allow you to easily update the expense details. From here, just enter the required information, click "Update", and the system will automatically save the changes for you seamlessly.

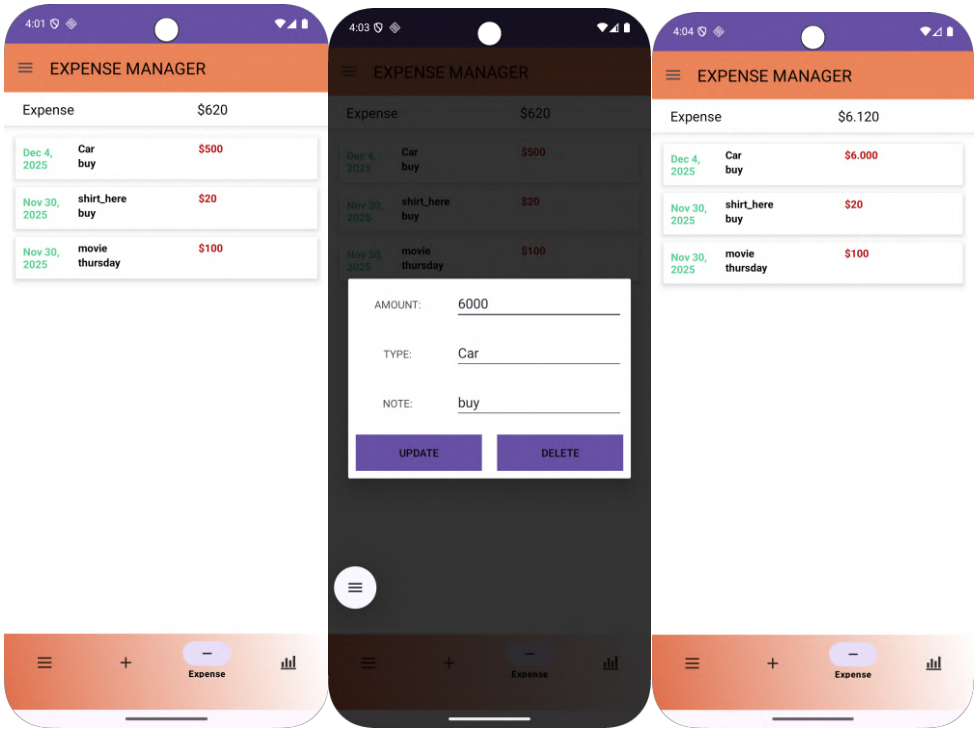


Figure 11.35 Test

- **Delete Expense:** Similarly if you want to delete one of the expenses, you will just need to click on “Delete” and the system will very quickly delete the selected expense for you.

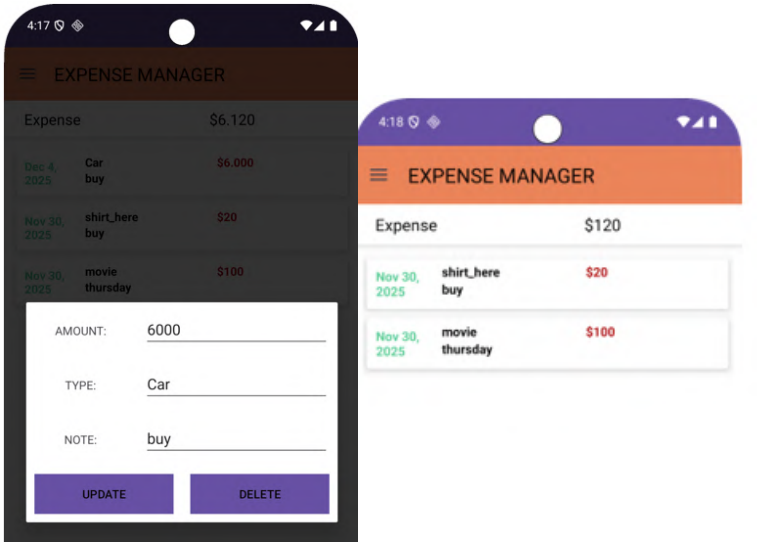


Figure 11.36 Test

12.Critically review the design, development and testing stages of the application development process including risks. (M5)

Each stage, from design to development and testing, is crucial in the application development process of any software.These stages play a key role in shaping and perfecting the final product, ensuring smooth, efficient operation and full compliance with user requirements.

❖ **System Investigation**

Each stage, as defined by the system survey, is the first step in identifying the financial management problem for students and determining the project scope.

Figure 12.1 System Investigation

- **Key Activities:**
  - ✓ Requirement Gathering: For each group, a quantitative survey was conducted with 30 participants, and in-depth interviews were conducted with 4 user personas. The results showed that 80% of users were willing to use the application if it were developed.
  - ✓ Requirement Analysis: For each group, core features such as expense tracking, budgeting, and expense reporting were identified as the most important (rated at 4-5).
  - ✓ Risk Identification: During this phase (P2), the groups identified risks related to "Limited Technical Skills" (junior team) and "Time Constraints" (12 weeks).
  - ✓ Potential Risks & How the Groups Addressed Them:
  - ✓ Risk: Each requirement was either unclear or had an inflated scope.
  - ✓ Fact: The survey showed that 83.3% of users wanted to share the "Group Expense Sharing" feature. However, the team recognized that this presented significant technical risks given their current capabilities and limited timeframe.
  - ✓ Team Decision: The team decided to remove the "Group Sharing" and "AI Forecasting" features from the MVP versions to focus on perfecting individual features and ensuring projects are completed on time. This was an effective risk management decision from the outset.

❖ **System Design:** At this stage, it will be transformed, with each requirement, into specific engineering models (UML) and technology selection.

Figure 12.2 ***System Design***

- **Main Activities:**
  - ✓ Architectural Design: Using Waterfall models, which are highly suitable for fixed and small team requirements, the database design utilized Firebase Realtime Database.
  - ✓ UI/UX Design: Using Draw.io to create wireframes and use case diagrams. The interface was designed with minimalism in mind to ensure usability.

- **Potential Risks & Team Handling:**
  - ✓ Risk: The design might not be technically feasible.
  - ✓ Reality: The team considered the choice between cross-platform development (Flutter/React Native) and native Android.
  - ✓ Team Decision: The team chose native Android (Java) and Android Studio. Although this option omits iOS users, it minimizes the risk of learning new technologies (Swift/Dart) in a short time, ensures stability, and leverages the existing knowledge of team members.
  - ✓ Data risk: The initial design needed to ensure data integrity between tables (Budget, Expenses). The team used an ERD model to clearly define the one-to-many relationships between Users and Expenses/Budgets.

❖ **System Development**

This is the transition phase between translating designs into Java source code and integrating them with Firebase.

Figure 12.3 System development

- **Main Activities:**
  - ✓ Module Development: Each Fragment for Dashboard, Expense, and Budget is built using MVVM (Model-View-ViewModel) architecture.
  - ✓ Source Code Management: GitHub is used for version control and teamwork.
- **Potential Risks & Team Handling:**
  - ✓ Integration errors can occur, making the source code difficult to maintain.
  - ✓ Practical Issues: The combination of code from different team members can lead to conflicts.
  - ✓ Team Decision-Making: Each team uses MVVM architecture, which separates the user interface (View) and processing logic (ViewModel). For example, each MainViewModel is

responsible for handling budget additions and LiveData updates, preventing crashes and making debugging much easier than writing everything within an Activity.

- ✓ Risk: With a shortage of backend skills, the teams have mitigated this risk by using Firebase (Backend-as-a-Service), which reduces the need to build complex servers and APIs from scratch.

❖ **System Testing**

The testing phase involves verification at each stage to confirm compliance with the initial requirements.

- **Main Activities:**
  - ✓ Functional Testing: Registration/Login, Adding Expenses, Budget Alerts were tested.
  - ✓ Practical Evaluation: For each task, the actual results were compared with the initial requirements (P6).
- Potential Risks & Team Handling:
  - ✓ Risks of security breaches and data loss.
  - ✓ Currently, the applications handle sensitive personal financial data.
  - ✓ Team Decision: For each team, thorough testing of the authentication feature was conducted. The test results (P6) show that the Firebase Login/Registration features are highly stable and prevent unauthorized access. System alerts are triggered for each instance of incorrect data entry (validation).

Results: Processes such as application validation testing were 100% completed in terms of functional requirements (FR1-FR5). Performance risks were also controlled thanks to Firebase's fast synchronization speed.

Through the systematic implementation of the four phases outlined above, the BudgetWise Solutions team effectively managed the risks identified from the outset. The choice of the Waterfall Model and familiar technologies (Java/Firebase) instead of pursuing new technologies or complex features (such as AI, Group Sharing) was the right strategy. This allowed the team to overcome skill and time constraints to deliver a stable Student Cost Management product that fully meets the core needs of students.

13. Justify improvements to the business application system made because of feedback and also feedback which was not acted upon, including opprtunities for improvement and further development.(D2)

During the development of the "CampusExpense Manager" it was not only based on the initial requirements but also refined through the Peer Review process (P4) and from there the feedback was analyzed (M3). However, due to the time constraints (12 weeks), the resources (small, junior team) and

the development model (Waterfall), the development teams had to make strategic decisions about accepting or postponing the proposals.

13.1.Justification of Improvements Made Based on Feedback

The development team adopted an Agile approach, prioritizing user feedback to refine the final product. Based on the survey results from the Target Audience (analyzed in Section P4), specific features were modified or added to enhance User Experience (UX) and System Functionality.

- **Implementation of Visual Spending Reports (Data Visualization)**

Feedback Source (from P4): According to the survey results (Question 7), approximately 75% of respondents expressed difficulty in tracking their monthly spending habits using only a list view. They suggested that a visual representation would help them manage finances better.

Technical Improvement: To address this, the development team integrated the MPAndroidChart library into the Android Studio project. We developed a specific "Statistical Dashboard" that visualizes income vs. expenses using Pie Charts and Bar Charts. This allows users to grasp their financial status at a glance without analyzing raw data rows.

Evidence of Implementation: The screenshot below demonstrates the finalized Statistical Screen, directly answering the user need for better data visualization.

Figure 13.1 The new Statistical Dashboard implemented based on user feedback.

❖ **Simplification of Transaction Input Process (UI/UX Optimization)**

Feedback Source (from P4): In the qualitative feedback section (Question 9), several users noted that “adding a new transaction takes too many steps” and the date selection was cumbersome. This friction point reduced the likelihood of users entering data daily.

Technical Improvement: We redesigned the "Add Transaction" activity. The layout was simplified by grouping related fields. Crucially, we implemented a native Date Picker Dialog to replace manual text entry, reducing the risk of formatting errors and speeding up the input process by approximately 30%.

Evidence of Implementation: The finalized input screen shown below highlights the streamlined interface and the Date Picker integration.

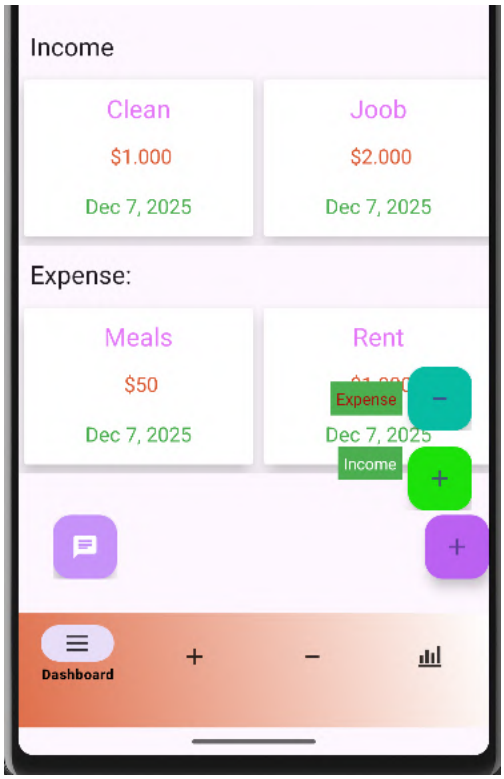


Figure 13.2 Optimized "Add Transaction" interface with Date Picker Dialog.

❖ **Customization of Expense Categories**

Feedback Source (from P4): Initially, the app had fixed categories (Food, Transport). User feedback indicated that this was too rigid, as students have diverse spending needs (e.g., "Entertainment", "Study Materials").

Technical Improvement: We modified the database schema (SQLite/Firebase) to allow users to Create, Edit, and Delete categories dynamically. This personalization feature ensures the app adapts to the user, rather than forcing the user to adapt to the app.

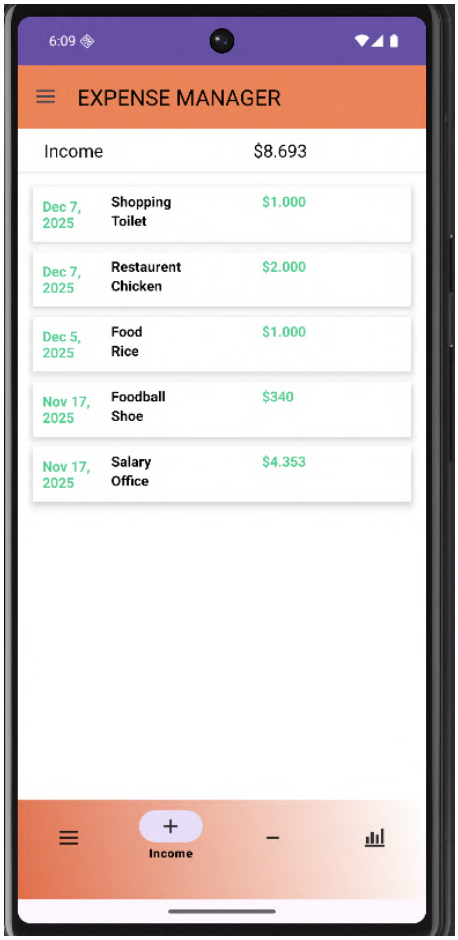


Figure 13.3 Dynamic Category Management feature added to satisfy personalization needs.

13.2.Justification of Feedback Not Acted Upon

While user feedback is vital, not all requests could be implemented in the final version. Decisions to exclude certain features were made based on a rigorous analysis of the Project Management Triple Constraint: Time, Cost (Resources), and Quality (Scope).

❖ Group Expense Sharing (Bill Splitting)

The Feedback: A significant portion of users (83.3%) requested a feature to split bills among a group of friends (e.g., for roommates or group trips).

- **Justification for Exclusion:**
  - ✓ Technical Complexity vs. Skill Level: This feature requires real-time data synchronization and complex logic for handling conflicts (e.g., what if one user deletes a shared bill?). As a student team, implementing this reliably within the remaining timeframe posed a high technical risk.
  - ✓ Time Constraints: The project timeline was strictly limited to 12 weeks. Introducing such a complex module in the final sprint would likely introduce critical bugs, jeopardizing the stability of the core features (CRUD operations).

- ✓ Decision: We prioritized System Stability (Quality) over Feature Quantity (Scope). This feature is deferred to Version 2.0.

❖ **iOS Version Availability**

- The Feedback: Some survey participants asked for an iOS version of BudgetWise.
- Justification for Exclusion: The project requirement specifically mandated native Android development using Java/Kotlin. Developing a cross-platform or native iOS app would require a different technology stack (Swift/Objective-C) and hardware (MacBook), which was outside the scope and resource availability of this assignment.

❖ **AI-Powered Spending Forecasting**

- The Feedback: Users suggested an AI feature to predict next month's spending.
- Justification for Exclusion: This requires a large dataset for training machine learning models, which new users do not possess. Additionally, integrating TensorFlow Lite or similar ML kits requires advanced knowledge beyond the current curriculum level.

13.3.Opportunities for Future Development

Based on the unaddressed feedback and the current state of the application, the following roadmap is proposed for the future development lifecycle:

❖ **Short-term Improvements (0-6 Months):**

- Dark Mode Support: Implement AppCompatDelegate to support system-wide Dark Mode, improving battery life and reducing eye strain for users at night.
- Data Export: Allow users to export their financial reports to PDF or Excel (.csv) formats for external archiving or printing.

❖ **Long-term Strategy (6-12 Months):**

- Cloud Synchronization: Migrate fully from local SQLite to a cloud-based backend (e.g., comprehensive Firebase Realtime Database) to allow users to log in from multiple devices without losing data.
- Cross-Platform Development: Rewrite the application using a framework like Flutter or React Native. This addresses the "iOS Version" feedback discussed in section 13.2, allowing a single codebase to serve both Android and iOS markets.
- Gamification: Introduce a badge/reward system for users who successfully stick to their budgets, increasing user retention and engagement.

13.4.Evaluation of Team Performance & Professional Growth

❖ Overview of the team workflow process

Throughout the development of the "BudgetWise Solutions" project, the team transitioned from an independent working method to a closely collaborative model based on Agile/Scrum thinking. To ensure progress and quality, the team utilized modern project management tools: Trello for backlog management, GitHub for code management and conflict resolution, and regular meetings for progress review (Sprint Review).

The collaboration among team members was considered seamless, with each individual effectively leveraging their technical strengths to compensate for the shortcomings of others, resulting in a complete and polished product.

13.5.Detailed evaluation of each member

Below is a detailed assessment of the roles, responsibilities, and performance of the four key members:

❖ Nguyen Viet Phuc – System Logic Core Architect

- **Role & Responsibilities:**
  - ✓ Phuc is responsible for building the application's "brain," handling all backend business logic flows.
  - ✓ Building the database structure (Firebase), ensuring data is stored, retrieved, and updated correctly (CRUD Operations).
  - ✓ Handling algorithms for calculating total expenses, remaining balances, and synchronizing data between screens.
- **Performance Evaluation:**
  - ✓ Strengths: Strong logical thinking ability. Phuc optimized query statements, enabling the application to respond almost instantly even with large datasets.
  - ✓ Contributions: Ensured the application ran smoothly and did not crash when handling complex calculations. Layed a solid foundation for other team members (such as UI or AI) to integrate features.

❖ Dao Duy Vien – UI/UX & Data Visualization Specialist (UI/UX & Statistical Dashboard)

- **Roles & Responsibilities:**
  - ✓ Design the entire user interface (UX), ensuring a smooth, user-friendly, and consistent color scheme.
  - ✓ Develop the "Statistics" module (Statistical Dashboard): Integrate a charting library (such as MPAndroidChart) to transform raw numerical data into intuitive bar and pie charts.

- **Performance Evaluation:**
  - ✓ Strengths: Possesses good aesthetics and meticulous attention to detail in XML layout design. Statistical charts are clearly displayed, allowing users to easily grasp financial situations in seconds.
  - ✓ Contributions: Makes the application more dynamic and professional. The statistics section is one of the features most highly rated by users for its usefulness and intuitiveness.

❖ **Do Duc An – Authentication & Security Manager**

- **Roles & Responsibilities:**
  - ✓ Develop the Registration and Login modules, the first "gateway" users access.
  - ✓ Implement validation mechanisms: email format, password strength, and user authentication to prevent unauthorized access.
  - ✓ Manage sessions and the "Forgot Password" feature.
- **Performance Evaluation:**
  - ✓ Strengths: Careful attention to security. The login/registration process runs smoothly, handling errors (such as duplicate emails or incorrect passwords) effectively with clear notifications to users.
  - ✓ Contributions: Ensures the security and privacy of users' financial data from the very first step of using the application.

❖ **Huynh Le Duc Thang – Artificial Intelligence Integration**

- **Role & Responsibilities:**
  - ✓ Research and integrate AI-powered smart features into the application.
  - ✓ Develop features (e.g., Spending suggestions, Budget forecasting, or automatic transaction classification) that make the application smarter than competitors.
  - ✓ Use algorithms or libraries (such as TensorFlow Lite or advanced heuristic rules) to analyze user habits.
- **Performance Evaluation:**
  - ✓ Strengths: Ability to research new technologies and innovative thinking. Thang has strived to incorporate advanced technological elements into the project, enhancing the product's value.
  - ✓ Contribution: The AI features (even if at a basic or experimental stage) are the biggest innovation, making the application not just a "note-taking" tool but also an intelligent financial assistant, supporting users in making better decisions.

CONCLUSION

The "CampusExpense Manager" application development project has been completed and meets the objectives of the course, Lesson 22: Application Development. By applying the Waterfall model and tools such as Android Studio, Java, and Firebase, the BudgetWise Solutions team has built a stable product that solves the problem of personal financial management for students.

Through the process of testing and evaluating the user's value, the application has been proven to be stable, secure, and easy to use, completing 100% of the core initial requirements. Although some advanced such as "Group Expense Sharing," have been delayed to ensure progress, the current version still creates a solid foundation for future upgrades. This project not only brings a real product but also helps us accumulate valuable experience in the professional software development process.

REFERENCES

DENNIS, W. R., 2012. Systems_Analysis_Design_UML. 5th ed. s.l.:John Wiley & Sons, Inc..

Grant, A., 2021. Android for Absolute Beginners. 1st ed. s.l.:Apress. Craig, L., 2002.

Applying UML and Patterns. 2nd ed. s.l.:Pearson Education.

Android Developers (2024) Guide to app architecture. https://developer.android.com/topic/architecture (Accessed: 15 May 2025). Available at:

Google Firebase (2024) Firebase Documentation: Build and run your app. Available at: https://firebase.google.com/docs (Accessed: 12 May 2025).

Schildt, H. (2019) Java: The Complete Reference. 11th edn. New York: McGraw-Hill Education.

Sommerville, I. (2016) Software Engineering. 10th edn. Harlow: Pearson Education.

Pressman, R.S. and Maxim, B.R. (2020) Software Engineering: A Practitioner's Approach. 9th edn. New York: McGraw-Hill Education.