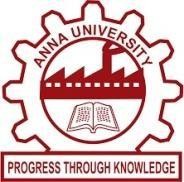
**PERSONAL EXPENDITURE TRACKER**

#### A DESIGN PROJECT REPORT

***submitted by***

# MUBINA BEGUM M

# OVIYA S

# RANJANI S

***in partial fulfillment for the award of the degree***

***of***

# BACHELOR OF ENGINEERING

***in***

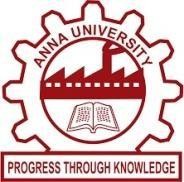
# COMPUTER SCIENCE AND ENGINEERING

**K RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

**(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)**

**Samayapuram – 621 112**

#### JUNE 2025

**PERSONAL EXPENDITURE TRACKER**

#### A DESIGN PROJECT REPORT

***submitted by***

# MUBINA BEGUM M (811722104094)

# OVIYA S (811722104106)

# RANJANI S (811722104121)

***in partial fulfillment for the award of the degree of***

# BACHELOR OF ENGINEERING

***in***

# COMPUTER SCIENCE AND ENGINEERING

**K RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

**(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)**

**Samayapuram – 621 112**

#### JUNE 2025

**K RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

## (Autonomous)

**SAMAYAPURAM – 621 112**

# BONAFIDE CERTIFICATE

Certified that this project report titled **“PERSONAL EXPENDITURE TRACKER”** is Bonafide work of **MUBINA BEGUM M (811722104094), OVIYA S (811722104106), RANJANI S (811722104121)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

Dr. A. Delphin Carolina Rani, M.E.,Ph.D., **HEAD OF THE DEPARTMENT**

PROFESSOR

Department of CSE

K Ramakrishnan College of Technology (Autonomous)

Samayapuram – 621 112

**SIGNATURE**

Mrs.V.Kalpana, M.E.,(Ph.D)

**SUPERVISOR**

Assistant Professor

Department of CSE

K Ramakrishnan College of Technology (Autonomous)

Samayapuram – 621 112

Submitted for the viva-voice examination held on ………………

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# DECLARATION

We jointly declare that the project report on **“PERSONAL EXPENDITURE TRACKER”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of Bachelor of Engineering. This project report is submitted on the partial fulfillment of the requirement of the award of Degree of Bachelor of Engineering.

|  |
| --- |
| Signature |
| MUBINA BEGUM M |
| OVIYA S |
| RANJANI S |

Place: TRICHY

Date:

# ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtedness to our institution **“K RAMAKRISHNAN COLLEGE OF TECHNOLOGY”**, for

providing us with the opportunity to do this project.

We are glad to credit and praise our honorable and respected chairman sir **Dr. K.. RAMAKRISHNAN, B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering adequate duration to complete it.

We would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project with full satisfaction.

We heartily thank **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.,** Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing her support to pursue this project.

We express our deep and sincere gratitude and thanks to our project guide **Mrs. V. Kalpana, M.E., (Ph.D.),** Department of **COMPUTER SCIENCE AND ENGINEERING,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry our this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

# ABSTRACT

The Personal Expense Trackeris an Android application designed using Jetpack Compose, aimed at helping users effectively manage and monitor their financial activities. This digital tool serves as a modern alternative to traditional manual expense tracking methods such as maintaining diaries or spreadsheets. The application allows users to log their daily income and expenditures, categorize transactions, and set monthly savings or spending limits. If the user exceeds the predefined financial limits, the app instantly notifies them, promoting better budgeting habits.

In addition to real-time expense tracking, the application provides users with detailed financial reports and summaries that offer insights into their spending patterns over days, weeks, or months. This helps users make informed financial decisions and achieve their savings goals more efficiently. With user-friendly design and intuitive navigation powered by Jetpack Compose, the application ensures a seamless and responsive user experience.

By automating the process of recording, calculating, and analyzing financial data, the Personal Expense Trackersignificantly reduces the burden of manual tracking. The app not only promotes financial discipline but also empowers users to take full control over their money. Overall, this application acts as a smart financial assistant that simplifies expense management and contributes to a more organized financial life.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE** |
|  | **ABSTRACT** | **iv** |
|  | **LIST OF FIGURES** | **viii** |
|  | **LIST OF ABBREVIATIONS** | **ix** |
| **1** | **INTRODUCTION** |  |
|  | 1.1 Background | 1 |
|  | 1.2 Overview | 1 |
|  | 1.3 Problem Definition | 3 |
|  | 1.4 Objective | 3 |
|  | 1.5 Implication | 3 |
| **2** | **LITERATURE SURVEY** | **4** |
| **3** | **SYSTEM ANALYSIS AND DESIGN** |  |
|  | 3.1 Existing System | 8 |
|  | 3.1.1 Disadvantages | 8 |
|  | 3.2 Proposed System | 9 |
|  | 3.2.1 Advantages | 9 |
|  | 3.3 System Configuration | 11 |

* + 1. [Hardware Requirements 11](#_TOC_250021)
    2. Software Requirements 11
  1. Block Diagram 12

1. MODULES
   1. [Module Description 13](#_TOC_250020)
      1. [Secure Authentication 13](#_TOC_250019)
      2. [Expense Management 14](#_TOC_250018)
      3. [Smart Budgeting 14](#_TOC_250017)
      4. [Data Insights 15](#_TOC_250016)
      5. Protective Alerts 15
2. SOFTWARE DESCRIPTION
   1. Operating system 17
   2. Development Environment 17
   3. Required Libraries 18
   4. Permission 18
   5. [Notification Permission 19](#_TOC_250015)
3. TEST RESULT AND ANALYSIS
   1. [Testing 20](#_TOC_250014)
   2. [Test Objectives 21](#_TOC_250013)
   3. [Program Testing 21](#_TOC_250012)
   4. [Testing and Correctness 22](#_TOC_250011)
      1. [Unit Testing 22](#_TOC_250010)
      2. [Integration Testing 22](#_TOC_250009)
      3. [Functional Testing 22](#_TOC_250008)
      4. [White Box Testing 23](#_TOC_250007)
      5. [Black Box Testing 23](#_TOC_250006)
   5. [Analysis 23](#_TOC_250005)
   6. [Feasibility Study 24](#_TOC_250004)
4. RESULT AND DISCUSSION
   1. [Result 25](#_TOC_250003)
   2. [Conclusion 26](#_TOC_250002)
   3. [Future Enhancement 27](#_TOC_250001)

APPENDIX A (SOURCE CODE) 29

APPENDIX B (SCREENSHOTS) 43

[REFERENCES 45](#_TOC_250000)

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **LIST OF FIGURES**  **FIGURE NAME** | **PAGE NO** |
| 1.1 | Flow of control | 1 |
| 3.1 | Use Case diagram | 10 |
| 3.2 | Flow chart Diagram | 10 |
| 4.1 | ER Diagram | 14 |
| 2.1 | Execution of code | 27 |

**LIST OF ABBREVIATIONS**

##### Abbreviation Full Form

**API** Application Programming Interface

**CNN** Convolutional Neural Network

**UAT** User Acceptance testing

**UI** User Interface

**IDE** Integrated Development Environment

**PDF** Portable Document Format

**CSV** Comma-Separated Values

**MVVM** Model-View-View Model

**UAT** User Acceptance Testing

**EMG** Electromyography

**OCR** Optical Character Recognition

**SDK** Software Development Kit

**AI** Artificial Intelligence

**HCI** Human-Computer Interaction

#### CHAPTER 1

#### INTRODUCTION

* 1. **BACKGROUND**

Managing personal finances is an essential aspect of modern life, yet many individuals struggle to maintain accurate records of their income and expenses. Traditional methods, such as writing in diaries or using spreadsheets, are often prone to human error, lack real-time updates, and can be tedious to maintain consistently. As digital solutions become more integrated into daily routines, there is a growing need for smart applications that simplify financial tracking and provide meaningful insights.

The Personal Expense Trackerapplication addresses this need by offering a user-friendly, automated platform to monitor income, expenses, and savings. Developed using Jetpack Composefor Android, the app enables users to input transactions, categorize expenses, and set monthly spending limits. It enhances financial discipline by sending alerts when spending exceeds the predefined threshold. This project aims to reduce the burden of manual financial tracking while promoting efficient money management through an intuitive and responsive mobile interface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Authentication | Dashboard Overview | Add Transaction | Set Budget & Alerts | Generate Reports |

Fig 1.1: Flow of Control

#### OVERVIEW

Managing personal finances has always been a critical part of everyday life, yet many individuals still rely on traditional methods such as handwritten logs or spreadsheets, which can often lead to errors, missed entries, and

difficulty tracking spending habits over time. To address these limitations, the Personal Expense TrackerAndroid application offers an efficient and intelligent solution for automated financial management. Designed using JetpackCompose, this application provides a modern and intuitive interface for users to record, monitor, and analyze their daily income and expenses directly from their smartphones.

With the advancement of mobile technology and widespread use of Android devices, maintaining a digital record of financial transactions has become more convenient and accessible. The application functions as a smart assistant by offering features such as budget planning, spending alerts, income tracking, and detailed report generation. Users can set custom budget limits and receive instant notifications if their expenses exceed those limits, promoting better money management and financial discipline.

Instead of manually calculating monthly balances, users can view real-time financial summaries, categorized expense data, and graphical reports with ease. The Personal Expense Tracker eliminates the need for physical books, calculators, or separate financial planning tools. All data is securely stored and updated dynamically to reflect the user’s financial activity.

By simplifying financial tracking, the app enhances personal accountability and encourages savings. It supports both static (manual entry) and dynamic (real-time alerts and reports) interaction modes, ensuring that all types of users—from beginners to budget-conscious professionals—can benefit from its functionality. As automation becomes more integrated into everyday life, this application stands out as a user-friendly, efficient, and highly practical tool for digital money management. It not only saves time but also transforms the way users engage with their financial goals.

#### PROBLEM STATEMENT

Traditional methods of tracking personal finances—such as using paper records or spreadsheets—are often inefficient, prone to errors, and difficult to maintain consistently. These manual approaches make it challenging for individuals to keep a clear view of their income, expenses, and overall financial health. Without real-time monitoring and alerts, users may overspend or fail to meet their savings goals. The lack of automated tools for budgeting and report generation creates barriers to effective financial planning. The Personal Expense Tracker application addresses these challenges by offering a digital, user-friendly solution that ensures accurate tracking, timely alerts, and insightful financial reporting.

#### OBJECTIVE

The objective of the Personal Expense Tracker is to provide users with a simple and efficient way to monitor their income, expenses, and savings. It aims to promote better financial management through real-time tracking, budget alerts, and detailed reports, helping users make informed spending decisions and achieve financial goals.

#### IMPLICATION

The implementation of the Personal Expense Tracker can significantly improve individual financial habits by reducing dependency on manual record-keeping and increasing financial awareness. It encourages users to set realistic budgets, track expenses in real-time, and avoid overspending. By providing insights through reports and alerts, the application supports smarter financial decisions, leading to better savings, reduced financial stress, and enhanced long-term financial stability.

#### CHAPTER 2 LITERATURE SURVEY

##### Designing an Android Application for Bills Segregation - Ruifeng Guo (2016)

This paper introduces an Android application aimed at simplifying bill splitting among individuals sharing expenses. The app allows users to create events, invite participants, and set payment rules, facilitating transparent and efficient financial interactions. A dashboard provides an overview of events and includes a chat feature for seamless communication. This approach addresses the common challenges faced by roommates or groups in managing shared expenses, offering a practical solution through mobile technology.

##### Financial Studio: Android Based Application for Computing Tax

##### Muhammad Zubair Asghar et al. (2020)

This paper Studio application integrates multiple financial calculators into a single Android platform, enabling users to compute taxes, pensions, zakat, and loans. Developed using MIT App Inventor, the app provides a user-friendly interface for performing diverse financial calculations. Statistical analysis indicates the application's effectiveness in facilitating various financial computations, demonstrating its potential as a comprehensive financial tool for individuals and organizations.

##### Mobile App Privacy in Software Engineering Research: A Systematic

##### Mapping Study - Fahimeh Ebrahimi et al. (2020)

This paper mapping study examines 54 primary studies on mobile app privacy within software engineering research. The research categorizes existing literature into privacy policy, requirements, user perspective, and leak detection.

Findings reveal an imbalance, with a majority focusing on privacy leak detection tools, while areas like privacy requirements and user perspectives are underrepresented. The study highlights gaps in current research and suggests directions for future investigations to enhance mobile app privacy practices.

##### A Systematic Survey on Android API Usage for Data-Driven Analytics with Smartphones - Hansoo Lee et al. (2021)

This paper investigates the utilization of Android APIs for data-driven analytics in smartphone applications. It categorizes research purposes and data types, providing a hierarchical classification structure. The study discusses limitations and future research directions, emphasizing the impact of Android API version changes on research, privacy and data quality issues, and the need for standardized data typology to mitigate reproducibility risks in mobile usage and sensor data-driven analytics.

##### Ekspensify: Personal Finance Tracking App with Jetpack Compose –

**Dilip Suthar (2021)**

This paper is an Android application designed to simplify personal finance management. Built with Kotlin and Jetpack Compose, it offers features like real-time income and expense tracking, budget management, and visual analytics. The app supports multiple accounts, automatic transaction tracking via SMS, and provides options to export data in PDF and CSV formats. Its modern UI and integration with cloud storage enhance user experience, making financial tracking accessible and efficient.

##### Building a Budget Tracker with Jetpack Compose - Victor Brandalise (2022)

This paper gives details about the development of a budget tracking application using Jetpack Compose. The author shares insights into building synchronized UI components, handling state management, and implementing features like expense categorization and budget alerts. The project emphasizes the advantages of Jetpack Compose in creating responsive and maintainable Android applications, offering practical guidance for developers interested in modern UI development.

##### Developing an Expense Tracking App: A Case Study of Pocket Planner –

**Davie Kim (2022)**

This paper explores the development process of Pocket Planner, an expense tracking application. Utilizing Agile methodologies, the project emphasizes iterative development, user feedback integration, and performance optimization. Key features include budget and expense management, financial data visualization, and offline functionality. The study highlights the importance of choosing appropriate algorithms and data structures to ensure efficient performance and a user-friendly experience.

##### xWalletWise: A Personal Finance Management App - Baohuy Van Ba (2022*)*

This paper is a personal finance management application developed using Kotlin and Jetpack Compose. The app allows users to track income and expenses, set budgets, and generate financial reports. Initially a university group project, it evolved into a comprehensive tool for exploring modern Android development practices. The project serves as a practical example of implementing MVVM architecture and leveraging Jetpack Compose for building intuitive user interfaces.

##### Expense Tracker Android App - Furqanullah (2023)

This paper facilitates daily expense tracking, offering features like adding, viewing, and analyzing expenses through intuitive statistics and charts. Developed with Kotlin, Jetpack Compose, Hilt, MVVM architecture, and Room Database, the app emphasizes modern Android development practices. It aims to provide users with a seamless experience in managing their finances, highlighting the integration of contemporary tools and frameworks in application development.

##### An Android-Based Mobile Application for Tracking Daily Expenses –

**ResearchGate Publication (2023)**

This study addresses the challenges individuals face in managing personal finances due to manual recording methods. It proposes an Android-based application to automate expense tracking, aiming to improve financial control and planning. The application allows users to record income and expenses, categorize transactions, and generate reports, thereby enhancing financial awareness and decision-making. The study underscores the potential of mobile applications in facilitating effective personal finance management

#### CHAPTER 3 SYSTEM ANALYSIS

* 1. **EXISTING SYSTEM**

Currently, several personal finance management applications like Mint, YNAB (You Need A Budget), Money Manager, and PocketGuard are available in the market. These applications help users track expenses, categorize transactions, and create budgets. They often integrate with bank accounts to automatically fetch transactions and provide visual insights through graphs and charts.

##### Features of the Existing Systems:

* + - * Automatic bank account syncing
      * Expense categorization and recurring transaction tracking
      * Budget setting and financial goal planning
      * Report generation with pie charts and bar graphs
      * Cloud synchronization across multiple devices
      * Notifications for bill reminders and overspending

##### Disadvantages of Existing Systems:

1. **Internet Dependency:**

Most existing apps require constant internet access for full functionality,

while our system works offline with local Room DB storage.

##### Complex UI:

Existing systems often have cluttered or complex interfaces. Our app uses Jetpack Compose for a simple, modern UI that’s beginner-friendly.

##### Limited Free Features:

Many features in apps like YNAB and Mint are locked behind subscriptions. Our app offers core features completely free.

#### PROPOSED SYSTEM

The proposed system, Personal Expense Tracker, is a mobile application developed using Kotlin in Android Studio, leveraging Jetpack Compose for building a modern, responsive user interface and Room Database for efficient local data storage. This application enables users to manage their personal finances by adding, editing, and deleting income and expense entries. Users can categorize their spending, set monthly budgets, and receive real-time alerts when their expenses exceed the predefined limits. The system supports user authentication, secure login, and password recovery, with future scalability for social login integration. Financial data is visualized through reports and summaries, helping users understand spending patterns over time. The app functions both online and offline, with data synchronization when connectivity is restored. By combining Kotlin’s concise syntax with Jetpack Compose’s UI capabilities, the system ensures a seamless and intuitive user experience. This project aims to reduce manual tracking efforts and promote better financial decision-making through a smart and accessible solution.

##### Advantages of Proposed System

1. **Offline Functionality**

Unlike most existing apps that require constant internet connectivity, your application works offline using Room Database. Users can access and manage their financial data without relying on network access, making it ideal for all environments.

##### Simple and Intuitive UI (Jetpack Compose)

Existing apps often have cluttered interfaces that can confuse new users. Your app uses Jetpack Compose, offering a clean, modern, and responsive UI designed for smooth user interaction on any Android device.

##### Full Feature Access Without Subscription

Many popular apps lock important features like report exports and budget limits behind paywalls. Your app provides all core features for free, including expense tracking, budget setting, real-time alerts, and report generation.

##### Enhanced Data Privacy and Security

Unlike apps that sync with banks (posing potential privacy risks), your app stores all user data locally and securely using Room DB. This ensures better control and protection of sensitive financial information.

##### Real-Time Budget Alerts

Your app includes a Proactive Alerts module that notifies users immediately when spending exceeds budget limits, helping maintain financial discipline. This feature is often delayed or limited in other systems.

##### Lightweight and Performance-Optimized

Built with Kotlin and Android Studio, your app is optimized for performance and battery efficiency, offering faster load times and smoother operation than many bulkier finance apps.

* 1. **System Configuration**

##### Hardware Requirements

* Processor: 2.5 GHz quad-core CPU or equivalent
* RAM: 4 GB or more
* Storage: 500 MB or more

##### 3.3.2 Software Requirements

* Operating System: Android 10.0 or later
* Development Environment: Android Studio with the latest stable

version by Kotlin

* Required Libraries: Latest stable version of Jetpack Compose
* Permissions: Full Storage access for optimal performance and cloud synchronization via Room

#### BLOCK DIAGRAM OF PROPOSED SYSTEM

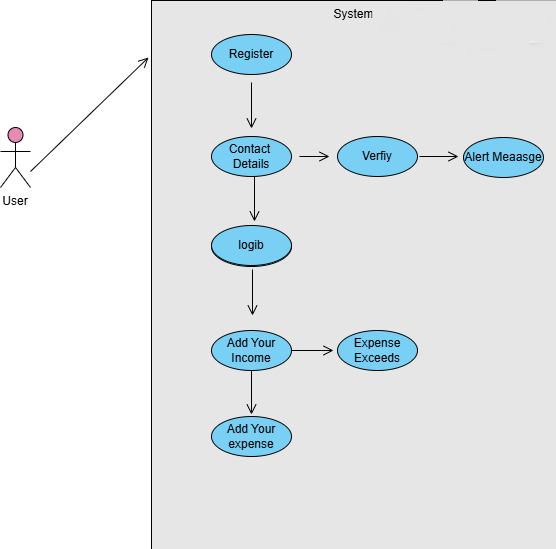
****

Fig 3.1: Usecase Diagram

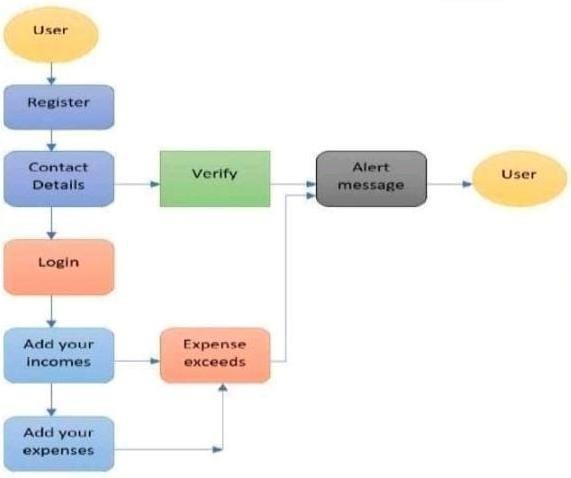


Fig 3.2: Flow Diagram

#### CHAPTER 4 MODULES

#### MODULE DESCRIPTION

* Secure Authentication
* Expense Management
* Smart Budgeting
* Data Insights
* Proactive Alerts

## Secure Authentication

The Secure Authenticationmodule is responsible for user registration, login, and account management. Developed using RoomDatabase and Kotlin, this module ensures that user credentials are securely stored and validated during login. It supports basic security features such as password protection, password recovery, and validation checks. The authentication module allows only authorized users to access their financial data, protecting sensitive information from unauthorized access. In the future, this module can be extended to support multi-factor authentication and social logins (Google or Facebook). The system securely stores login details using encryption, ensuring user privacy. On successful login, users are directed to the dashboard where they can manage their expenses and budgets. The simplicity and security of this module make the app suitable for daily use without compromising user data. This foundation provides a reliable entry point for users and keeps their personal finance records private and protected.

## Expense Management

The Expense Management module is the core functionality of the application. It allows users to add, edit, and delete income and expense entries. Each transaction includes details such as the amount, category, date, and description. Users can assign expenses to categories like Food, Travel, Bills, etc., for better tracking. All transaction data is stored locally using Room Database, allowing the app to function even when offline. When the user returns online, the data can sync automatically. This module ensures data consistency and provides a seamless experience for users to update their financial records anytime. The system also displays a categorized list of transactions in the dashboard, making it easy to review and manage past spending. Through Kotlin and Jetpack Compose, the user interface for this module is kept intuitive, ensuring smooth navigation. The Expense Management module forms the backbone of the tracker, helping users gain control over their day-to-day financial activities.

## Smart Budgeting

The Smart Budgeting module allows users to define and manage monthly or category-wise spending limits. Users can set personal budgets for various expense categories like Groceries, Entertainment, Utilities, and more. Once a budget is set, the system continuously tracks the total spending against these limits. This module helps users gain a clear picture of how much they are spending in each area and encourages better money management. If expenses approach the defined limit, the system triggers alerts via the Proactive Alerts module. Budgets can be adjusted or reset monthly, and the system can provide suggestions based on past spending patterns. The module is integrated with the Expense Management and Alert systems, allowing real-time tracking and

feedback. By using Kotlin and Room DB, the budgeting data remains persistent and secure. This feature supports financial discipline and allows users to plan ahead, ensuring that they do not exceed their spending goals.

## Data Insights

The Data Insights module focuses on providing users with visual and analytical summaries of their financial activity. It generates graphs, charts, and reports based on income, expenses, and savings over different time periods (daily, weekly, monthly). Users can view pie charts showing category-wise spending or bar graphs showing savings trends. These insights help users understand their spending behavior and make informed decisions. The reports can be viewed within the app and exported in PDF or CSV formats for record-keeping or sharing. This module retrieves data from the Room Database and processes it dynamically using Kotlin. Jetpack Compose is used to render beautiful and responsive data visualizations. The main goal of this module is to convert raw financial data into actionable insights. By analyzing trends and patterns, users can identify overspending areas, adjust their budgets, and improve their financial planning. Overall, this module empowers users with the knowledge needed for smarter money management.

## Proactive Alerts

The Proactive Alerts module keeps users informed and in control of their finances by sending real-time notifications when certain financial thresholds are met or exceeded. For example, when the user’s expenses exceed the monthly or category-wise budget, the system automatically triggers an alert. These alerts are generated by continuously monitoring data changes in the Room Database. Jetpack Compose is used to present the alerts in a non-intrusive but noticeable manner through in-app banners or Android push notifications. The module helps users take immediate corrective actions to avoid overspending.

Additionally, reminders for daily summaries, missed entries, or report generation can also be implemented in this module. This feature promotes financial discipline and reduces the chances of neglecting important transactions or budget goals. It integrates seamlessly with the Smart Budgeting and Expense Management modules, ensuring that users receive accurate and timely information about their financial activities. The Proactive Alerts module adds intelligence and responsiveness to the entire application.

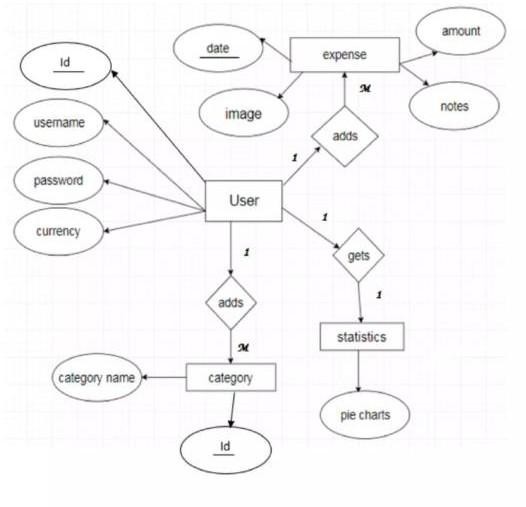


Fig 4.1: ER Diagram

#### CHAPTER 5 SOFTWARE DESCRIPTION

##### Operating System: Android 10.0 or Later

The Personal Expense Tracker is designed to run on Android 10.0 or later, ensuring compatibility with modern smartphones and security protocols. Android 10 offers enhanced privacy controls, background activity limits, and system-wide dark mode support—all of which contribute to improved app performance and user experience. Utilizing Android’s robust APIs, the app functions smoothly on devices running newer versions, taking full advantage of Jetpack components and optimized background services. This requirement ensures stability, efficient resource management, and support for features like in-app permissions, real-time alerts, and seamless background synchronization using Room DB.

##### Development Environment: Android Studio with Latest Kotlin

The application is developed in Android Studio, the official IDE for Android app development. Using the latest stable version of Kotlin, the project benefits from concise syntax, null safety, and seamless integration with Jetpack libraries. Android Studio provides advanced tools for UI design, performance profiling, and emulator testing, making development efficient and error-free. Kotlin’s interoperability with Java and its support for coroutine-based asynchronous tasks ensure smooth background data operations, such as syncing expense records and updating budget status. This environment ensures that the application is built using current industry standards with future scalability in mind.

##### Required Libraries: Jetpack Compose and Room SDK

Jetpack Compose is used to design the app's UI, providing a declarative and responsive user interface with minimal boilerplate code. This enhances performance and simplifies UI updates based on state changes. Room SDKacts as the local database, allowing structured data storage with compile-time checks and efficient data queries. Room supports LiveData and Flow, enabling real-time updates to the user interface when database content changes. Together, these libraries create a powerful foundation for building robust, maintainable, and user-friendly financial applications that can function offline and sync seamlessly when connectivity returns.

##### Permissions: Full Storage Access

The app requires full storage access to manage user-generated financial reports, such as exporting PDF or CSV files. This permission enables the app to read and write report data, cache images or graphs used in visual summaries, and store offline user data for later synchronization. Full storage access is essential for users who wish to keep backup files locally or share reports via email or other apps. Ensuring this permission is granted enhances the app's flexibility and ensures that the user can maintain and manage their financial data securely across different formats**.** Granting full storage access also empowers users with greater control over their financial data. It allows them to create local backups, share reports via email or other applications, and access their records anytime, across multiple platforms. This not only enhances the app's functionality but also ensures data is handled securely and conveniently, making financial management more flexible and user-friendly.

##### Notification Permission

The notification permission is crucial for enabling real-time alerts and reminders in the Personal Expense Tracker. These notifications inform users when their spending exceeds set budgets, prompt daily summaries, and provide timely reminders to log expenses. This improves user engagement and encourages consistent financial tracking. Using Android’s native notification manager, the app delivers clear, actionable messages without disturbing the user experience. This module works in sync with the budgeting and expense modules, ensuring that users are always updated about their financial activities. Notification permission ensures proactive financial monitoring, even when the app is not actively in use. Beyond just alerts, the notification system operates in close coordination with the budgeting and expense tracking modules. It ensures users receive proactive updates and reminders even when the app is running in the background or not actively in use. This always-on functionality helps users stay in control of their finances, avoid overspending, and make informed decisions in real time. Granting notification permission allows the app to act as a smart financial assistant, offering personalized, timely nudges that support responsible money management.Users can customize the type and frequency of notifications based on their preferences, ensuring relevance and avoiding overload. Smart notifications can be tailored to highlight specific spending categories, savings milestones, or upcoming bill payments. This level of personalization helps users stay focused on their financial goals. The app also supports silent notifications for subtle reminders that don't interrupt workflow. Overall, notification access enhances user accountability and promotes long-term financial discipline.

#### TESTING

**CHAPTER 6**

**TEST RESULT AND ANALYSIS**

The Personal Expense Tracker application underwent comprehensive testing to ensure functionality, usability, and reliability across different Android devices. The testing process included unit testing, integration testing, and user acceptance testingto validate all components.

Unit testingfocused on individual modules such as transaction entry, budget limit settings, and alert notifications. Each feature was tested for accuracy, such as verifying that expense entries correctly updated the balance and that alerts triggered at the appropriate thresholds.

Integration testingensured that all modules worked seamlessly together. For instance, after entering expenses, the app was tested to verify that summaries and reports reflected the changes immediately and correctly. The interaction between the database and user interface was also validated to maintain data consistency and smooth user experience.

User acceptance testing (UAT)involved real users interacting with the app to assess usability and overall satisfaction. Feedback highlighted the app’s intuitive design and the effectiveness of budget alerts. Minor usability improvements were made based on this feedback to enhance navigation and report readability.

Additionally, the application was tested for performance, ensuring quick loading times and minimal battery consumption. Error handling was also tested to confirm the app’s stability during invalid inputs or unexpected scenarios.

Overall, the testing process confirmed that the Personal Expense Tracker is a robust, user-friendly application capable of delivering reliable financial management support.

#### TEST OBJECTIVES

Testing is the systematic process of executing the Personal Expense Tracker application with the goal of identifying defects or errors. Effective testing aims to verify that all features, including transaction recording, budget setting, alert notifications, and report generation, function correctly according to the design specifications. A good test case should have a high likelihood of uncovering hidden errors or inconsistencies in the software.

The primary objectives of testing the Personal Expense Tracker are to ensure the accuracy of financial calculations, the reliability of real-time alerts, and the usability of the user interface. Testing also aims to validate data integrity between user inputs and stored records, as well as confirm that performance requirements—such as quick response times and smooth navigation—are met.

Successful testing demonstrates that the application operates as intended, providing users with an efficient and error-free experience. It also helps identify and fix potential issues before deployment, improving overall software quality. Ultimately, these objectives ensure the Personal Expense Tracker delivers a reliable and user-friendly tool for managing personal finances.

## PROGRAM TESTING

Program testing for the Personal Expense Tracker application was conducted to verify that all modules function correctly and efficiently under various conditions. The testing process included unit testing, integration testing, and user acceptance testing (UAT). Each core feature—such as transaction

entry, budget setting, and alert notifications—was tested individually to ensure accurate performance. Integration testing verified smooth data flow between modules like authentication, database, and report generation. UAT involved real users evaluating usability, with feedback guiding UI improvements. The app was also tested for performance, ensuring fast loading, smooth navigation, and proper handling of invalid inputs without crashes.

## TESTING AND CORRECTNESS

### Unit Testing

Unit testing was performed on individual components such as login authentication, transaction entry, and budget setting. Each function was tested in isolation to ensure it performed as expected. This helped identify and fix bugs early in development, ensuring that each module of the app behaves correctly and independently before integration.

### Integration Testing

Integration testing was carried out to verify data flow between modules like the authentication system, Room Database, and report generation. It ensured seamless interaction between components such as user input, transaction updates, and real-time alerts. The focus was on validating that all interconnected parts work cohesively to deliver accurate financial tracking.

### Functional Testing

Functional testing validated that the Personal Expense Tracker performs all expected features—adding expenses, generating reports, setting budgets, and sending alerts. Each function was tested against defined requirements to ensure correct outputs. This testing confirmed the app’s

usability and reliability in delivering its core purpose of personal finance management.

### White Box Testing

White box testing involved examining the internal logic, code structure, and flow of the application. Developers tested loops, conditions, and data processing functions in the Kotlin codebase. This helped ensure that the system’s logic was implemented correctly and optimized for performance, especially in budget calculations and alert mechanisms.

### Black Box Testing

Black box testing focused on testing the application from the user’s perspective without looking at internal code. Features like user login, budget alerts, and report exports were tested by providing various inputs and observing outputs. This ensures that the system responds correctly to user actions and handles edge cases gracefully.

##### ANALYSIS

The Personal Expense Tracker project aims to provide users with a tool to manage their finances effectively. The primary goal is to track income, expenses, and budgets while offering insightful data visualizations. The system will enable users to set budgets, categorize spending, and view spending trends through graphical reports. Key features include transaction management, recurring expense handling, and real-time notifications for budget overages.From a technical analysis perspective, the system requires seamless integration with APIs (such as Plaid) to fetch bank data automatically, and strong backend architecture to handle user data securely. The project needs a clean, intuitive UI to ensure ease of use for non-technical users, making data entry, categorization,

and financial analysis as straightforward as possible. Challenges might include handling security (for sensitive financial data), ensuring accurate budgeting algorithms, and creating a responsive design that works well across devices. Testing will be crucial to ensure data integrity, UI consistency, and proper transaction handling.

##### FEASIBILITY STUDY

A feasibility study for the Personal Expense Tracker project indicates high viability in terms of technical, operational, and economic feasibility. Technically, the project can be developed using widely available tools like React, Node.js, and PostgreSQL. Integration with APIs such as Plaid is feasible for automatic transaction imports. Operationally, the app’s design will be intuitive, ensuring accessibility for non-technical users, and it can be developed in stages for better scalability. Economically, the project can be built with relatively low cost, utilizing open-source libraries and cloud services. Monetization strategies, such as premium features, offer a strong revenue potential.

#### 

**CHAPTER 7**

**RESULT AND DISCUSSION**

**7.1 RESULT**

The development and deployment of the Personal Expense Tracker application have demonstrated significant benefits in simplifying personal financial management. Users reported that the app provided a clear and organized platform to record all income and expenses, reducing the need for manual tracking and minimizing errors commonly found in traditional methods like handwritten logs or spreadsheets. The intuitive interface developed using Jetpack Compose enhanced user experience by offering seamless navigation and quick access to key features such as transaction entry, budget setting, and report viewing.

The real-time alert system effectively notified users when their spending approached or exceeded predefined limits, encouraging better financial discipline and helping prevent overspending. Many users found this feature particularly valuable as it provided immediate feedback and promoted awareness of their spending habits. The ability to categorize expenses and view detailed summaries allowed users to identify areas of excessive spending and adjust their budgets accordingly.

The reporting functionality, which included graphical charts and summaries, empowered users with visual insights into their financial behavior over time. This made it easier to understand trends, track savings progress, and make informed decisions. The app’s capability to generate reports in various timeframes (daily, weekly, monthly) was highlighted as a useful tool for continuous financial monitoring.

Overall, the Personal Expense Tracker proved to be an effective tool that not only simplified expense management but also fostered responsible financial

behavior among users. It demonstrated how automation and smart design can reduce the cognitive load of managing money and make budgeting accessible to everyone. The positive feedback indicates strong potential for adoption by a wider user base, particularly as future enhancements like cloud syncing and bank integration are implemented to further improve functionality and convenience.

#### CONCLUSION

The Personal Expense Tracker application is an innovative and practical solution designed to simplify the complex task of managing personal finances. By replacing traditional manual methods with a digital platform, the app empowers users to effortlessly track their income, expenses, and savings. The integration of Jetpack Compose technology ensures a smooth, intuitive, and responsive user interface that caters to users of all levels, from beginners to those more experienced in budgeting.

This application addresses common financial challenges, such as difficulty in maintaining accurate records, unawareness of spending habits, and the risk of exceeding budgets. By providing real-time monitoring and instant alerts when spending limits are approached or exceeded, the app promotes financial discipline and encourages responsible money management. Furthermore, its ability to generate detailed reports and graphical summaries offers users clear insights into their financial behavior, enabling informed decisions and better planning.

The Personal Expense Tracker is more than just an expense logging tool; it acts as a financial assistant that reduces the burden of manual calculations and record-keeping. It helps users set realistic budget limits and provides continuous feedback on their spending patterns. This not only aids in preventing

overspending but also motivates users to save consistently and achieve their financial goals.

Looking ahead, the app’s potential for future enhancements, such as AI-driven recommendations, cloud synchronization, and bank integration, positions it as a forward-thinking tool that can evolve with users’ needs. By embracing technology, the Personal Expense Tracker paves the way for a smarter, more organized approach to personal finance management.

In conclusion, this application offers a comprehensive, user-friendly, and effective solution that empowers individuals to take control of their finances, leading to improved financial health and peace of mind.

#### FUTURE ENHANCEMENT

As technology evolves and user needs grow, the Personal Expense Tracker application has the potential for several future enhancements to improve functionality, user experience, and financial intelligence. One of the key upgrades would be the integration of AI-powered financial insights. By analyzing user spending patterns, the app can provide personalized suggestions, detect anomalies, and predict future expenses, helping users make more informed financial decisions.

Another promising enhancement is cloud synchronization, allowing users to access their financial data across multiple devices securely. This would also enable automated backups, ensuring users don’t lose their data even if they change or reset their device. Additionally, multi-user supportcould be introduced, allowing families or roommates to track shared expenses under one account with individual access control. Bank integrationis another vital feature planned for future versions. By securely linking with bank accounts, the app could automatically import transaction data, categorize it, and update financial

summaries without requiring manual input. This would save time and increase accuracy.

The addition of voice command support and OCR (Optical Character Recognition) for reading receipts and bills would make data entry even more effortless. Users could speak their expenses or scan receipts directly to log transactions.

A goal-based savings module is also a planned enhancement, where users can set specific savings targets (e.g., vacation, emergency fund) and get updates and suggestions on how to reach them efficiently.

To enhance user engagement, gamification elements such as badges, progress bars, and rewards for achieving financial milestones could be introduced. Lastly, incorporating data visualization tools like advanced graphs, pie charts, and dashboards will help users better understand their financial behavior.

These enhancements aim to transform the Personal Expense Tracker into a comprehensive financial companion, blending simplicity with smart features for modern money management.

#### APPENDIX – 1 SOURCE CODE

##### Login Environment.java

package com.wisnu.kurniawan.wallee.features.login.data; import

com.wisnu.kurniawan.wallee.foundation.datasource..PreferenceManager; import com.wisnu.kurniawan.wallee.foundation.datasource.server.ServerManager; import dagger.internal.DaggerGenerated;

import dagger.internal.Factory;

import dagger.internal.QualifierMetadata; import dagger.internal.ScopeMetadata;

import javax.annotation.processing.Generated; import javax.inject.Provider; @ScopeMetadata

@QualifierMetadata @DaggerGenerated @Generated(

value = "dagger.internal.codegen.ComponentProcessor", comments = "https://dagger.dev"

)

@SuppressWarnings({ "unchecked", "rawtypes", "KotlinInternal", "KotlinInternalInJava", "cast",

"deprecation"

})

public final class LoginEnvironment\_Factory implements Factory<LoginEnvironment> {

private final Provider<ServerManager> serverManagerProvider;

private final Provider<PreferenceManager> preferenceManagerProvider; public LoginEnvironment\_Factory(Provider<ServerManager>

serverManagerProvider,

Provider<PreferenceManager> preferenceManagerProvider) { this.serverManagerProvider = serverManagerProvider; this.preferenceManagerProvider = preferenceManagerProvider;

}

@Override

public LoginEnvironment get() {

return newInstance(serverManagerProvider.get(), preferenceManagerProvider.get());

}

public static LoginEnvironment\_Factory create(Provider<ServerManager> serverManagerProvider,

Provider<PreferenceManager> preferenceManagerProvider) {

return new LoginEnvironment\_Factory(serverManagerProvider, preferenceManagerProvider);

}

public static LoginEnvironment newInstance(ServerManager serverManager, PreferenceManager preferenceManager) {

return new LoginEnvironment(serverManager, preferenceManager);

}

}

##### Account Details Environment.kt

package com.wisnu.kurniawan.wallee.features.account.detail.data

import com.wisnu.kurniawan.wallee.features.balance.summary.data.Account import com.wisnu.kurniawan.wallee.foundation.datasource.local.LocalManager import com.wisnu.kurniawan.wallee.foundation.extension.isAmountChanged import com.wisnu.kurniawan.wallee.foundation.extension.isChanged

import com.wisnu.kurniawan.wallee.foundation.wrapper.DateTimeProvider import com.wisnu.kurniawan.wallee.foundation.wrapper.IdProvider

import com.wisnu.kurniawan.wallee.model.Account import com.wisnu.kurniawan.wallee.model.AccountRecord import com.wisnu.kurniawan.wallee.model.CategoryType import com.wisnu.kurniawan.wallee.model.Transaction

import com.wisnu.kurniawan.wallee.model.TransactionType import javax.inject.Inject

import kotlinx.coroutines.flow.Flow import kotlinx.coroutines.flow.flowOf import kotlinx.coroutines.flow.map import kotlinx.coroutines.flow.onEach import kotlinx.coroutines.flow.take

class AccountDetailEnvironment @Inject constructor( private val localManager: LocalManager,

private val dateTimeProvider: DateTimeProvider, private val idProvider: IdProvider

) : IAccountDetailEnvironment {

override fun getAccount(id: String): Flow<Account> { return if (id.isBlank()) {

localManager.getDefaultAccount().take(1)

} else {

localManager.getAccount(id).take(1)

}

override suspend fun saveAccount(account: AccountBalance, changeReason:

AdjustBalanceReason): Flow<Boolean> { return if (account.id.isBlank()) {

val newAccount = Account( id = idProvider.generate(),

currency = account.currency, amount = account.amount, name = account.name,

type = account.type,

createdAt = dateTimeProvider.now(), updatedAt = null,

transactions = listOf()

)

localManager.insertAccount(newAccount) flowOf(true)

} else {

val newAccount = Account( id = account.id currency =account.currency, amount = account.amount,

name = account.name, type = account.type,

createdAt = account.createdAt, updatedAt = dateTimeProvider.now(), transactions = listOf()

) localManager.getAccount(account.id).take(1)

.onEach { oldAccount ->

record(oldAccount, newAccount, changeReason)

update(oldAccount, newAccount)

}

.map { true } }

private suspend fun update(account: Account, newAccount: Account) { if (account.isChanged(newAccount)) {

localManager.updateAccount(newAccount)

}

private suspend fun record(account: Account, newAccount: Account,

changeReason: AdjustBalanceReason) {

if (account.isAmountChanged(newAccount)) { when (changeReason) {

AdjustBalanceReason.FORGOT\_FOR\_UPDATE -> {

val transaction = buildNewTransaction(account, newAccount) localManager.insertTransaction(

accountId = account.id, transferAccountId = null, transaction = transaction

)

}

AdjustBalanceReason.LONG\_TIME\_NOT\_UPDATE -> { val accountRecord = AccountRecord(

id = idProvider.generate(), accountId = account.id, amount = account.amount,

createdAt = dateTimeProvider.now()

)

localManager.insertAccountRecord( accountRecord = accountRecord

)

}}

private fun buildNewTransaction(account: Account, newAccount: Account): Transaction {

return if (account.amount < newAccount.amount) { Transaction(

id = idProvider.generate(), currency = account.currency,

categoryType = CategoryType.INCOME, amount = newAccount.amount - account.amount, type = TransactionType.INCOME,

date = dateTimeProvider.now(), createdAt = dateTimeProvider.now(), updatedAt = null,

note = ""

)

} else { Transaction(

id = idProvider.generate(), currency = account.currency,

categoryType = CategoryType.FINANCIAL, amount = account.amount - newAccount.amount, type = TransactionType.EXPENSE,

date = dateTimeProvider.now(), createdAt = dateTimeProvider.now(), updatedAt = null,

note = ""

)

}}

override suspend fun deleteAccount(id: String) {

localManager.deleteAccount(id)

}}

##### Transaction.java

package com.wisnu.kurniawan.wallee.features.transaction.all.data; import com.wisnu.kurniawan.wallee.foundation.datasource.local.LocalManager; import dagger.internal.DaggerGenerated;

import dagger.internal.Factory;

import dagger.internal.QualifierMetadata; import dagger.internal.ScopeMetadata;

import javax.annotation.processing.Generated; import javax.inject.Provider; @ScopeMetadata

@QualifierMetadata @DaggerGenerated @Generated(

value = "dagger.internal.codegen.ComponentProcessor", comments = "https://dagger.dev"

)

@SuppressWarnings({ "unchecked", "rawtypes", "KotlinInternal", "KotlinInternalInJava", "cast",

"deprecation"

})

public final class AllTransactionEnvironment\_Factory implements Factory<AllTransactionEnvironment> {

private final Provider<LocalManager> localManagerProvider;

public AllTransactionEnvironment\_Factory(Provider<LocalManager> localManagerProvider) {

this.localManagerProvider = localManagerProvider;

}

@Override

public AllTransactionEnvironment get() {

return newInstance(localManagerProvider.get());

}

public static AllTransactionEnvironment\_Factory create( Provider<LocalManager> localManagerProvider) {

return new AllTransactionEnvironment\_Factory(localManagerProvider);

}

public static AllTransactionEnvironment newInstance(LocalManager

localManager) {

return new AllTransactionEnvironment(localManager);

}

}

**TopExpense Environment.java**

package com.wisnu.kurniawan.wallee.features.transaction.topexpense.ui; import com.wisnu.kurniawan.wallee.features.transaction.topexpense.data.ITopExpense Environment;

import dagger.internal.DaggerGenerated; import dagger.internal.Factory;

import dagger.internal.QualifierMetadata; import dagger.internal.ScopeMetadata;

import javax.annotation.processing.Generated; import javax.inject.Provider; @ScopeMetadata

@QualifierMetadata @DaggerGenerated @Generated(

value = "dagger.internal.codegen.ComponentProcessor", comments = "https://dagger.dev"

)

@SuppressWarnings({ "unchecked", "rawtypes", "KotlinInternal", "KotlinInternalInJava", "cast",

"deprecation"

})

public final class TopExpenseViewModel\_Factory implements Factory<TopExpenseViewModel> {

private final Provider<ITopExpenseEnvironment> topExpenseEnvironmentProvider;

public TopExpenseViewModel\_Factory( Provider<ITopExpenseEnvironment> topExpenseEnvironmentProvider) { this.topExpenseEnvironmentProvider = topExpenseEnvironmentProvider;

}

@Override

public TopExpenseViewModel get() {

return newInstance(topExpenseEnvironmentProvider.get());

}

public static TopExpenseViewModel\_Factory create( Provider<ITopExpenseEnvironment> topExpenseEnvironmentProvider) {

return new

TopExpenseViewModel\_Factory(topExpenseEnvironmentProvider);

}

public static TopExpenseViewModel newInstance(ITopExpenseEnvironment topExpenseEnvironment) {

return new TopExpenseViewModel(topExpenseEnvironment);

}

}

##### Logout Environment.java

package com.wisnu.kurniawan.wallee.features.logout.data;

import com.wisnu.kurniawan.wallee.foundation.datasource.PreferenceManager; import dagger.internal.DaggerGenerated;

import dagger.internal.Factory;

import dagger.internal.QualifierMetadata; import dagger.internal.ScopeMetadata;

import javax.annotation.processing.Generated; import javax.inject.Provider;

@ScopeMetadata @QualifierMetadata @DaggerGenerated @Generated(

value = "dagger.internal.codegen.ComponentProcessor", comments = "https://dagger.dev"

)

@SuppressWarnings({ "unchecked", "rawtypes", "KotlinInternal", "KotlinInternalInJava", "cast",

"deprecation"

})

public final class LogoutEnvironment\_Factory implements Factory<LogoutEnvironment> {

private final Provider<PreferenceManager> preferenceManagerProvider; public LogoutEnvironment\_Factory(Provider<PreferenceManager>

preferenceManagerProvider) {

this.preferenceManagerProvider = preferenceManagerProvider;

}

@Override

public LogoutEnvironment get() {

return newInstance(preferenceManagerProvider.get());

}

public static LogoutEnvironment\_Factory create( Provider<PreferenceManager> preferenceManagerProvider) {

return new LogoutEnvironment\_Factory(preferenceManagerProvider);

}

public static LogoutEnvironment newInstance(PreferenceManager

preferenceManager) {

return new LogoutEnvironment(preferenceManager);

}

}

##### Setting screen .kt

package com.wisnu.kurniawan.wallee.features.setting.ui import androidx.compose.foundation.layout.Spacer import androidx.compose.foundation.layout.height import androidx.compose.foundation.lazy.items

import androidx.compose.runtime.Composable import androidx.compose.runtime.getValue import androidx.compose.ui.Modifier

import androidx.compose.ui.res.stringResource import androidx.compose.ui.unit.dp

import androidx.lifecycle.compose.collectAsStateWithLifecycle import com.wisnu.kurniawan.wallee.R

import com.wisnu.kurniawan.wallee.foundation.uicomponent.PgModalCell import com.wisnu.kurniawan.wallee.foundation.uicomponent.PgModalLayout import com.wisnu.kurniawan.wallee.foundation.uicomponent.PgModalTitle import com.wisnu.kurniawan.wallee.runtime.navigation.SettingFlow

@Composable fun SettingScreen(

viewModel: SettingViewModel,

onClick: (String) -> Unit,

) {

val state by viewModel.state.collectAsStateWithLifecycle() PgModalLayout(

title = {

PgModalTitle(stringResource(R.string.setting\_title))

},

content = {

items(state.items) { item -> SettingItem(

onClick = {

val route = when (item) {

is SettingItem.Logout -> { SettingFlow.Logout.route

}

is SettingItem.Theme -> { SettingFlow.Theme.route

}

is SettingItem.Language -> { SettingFlow.Language.route

}

}

onClick(route)

},

stringResource(item.title)

)

Spacer(Modifier.height(8.dp))

}}

)

@Composable

private fun SettingItem( onClick: () -> Unit, title: String,

) {

PgModalCell( onClick = onClick, text = title,

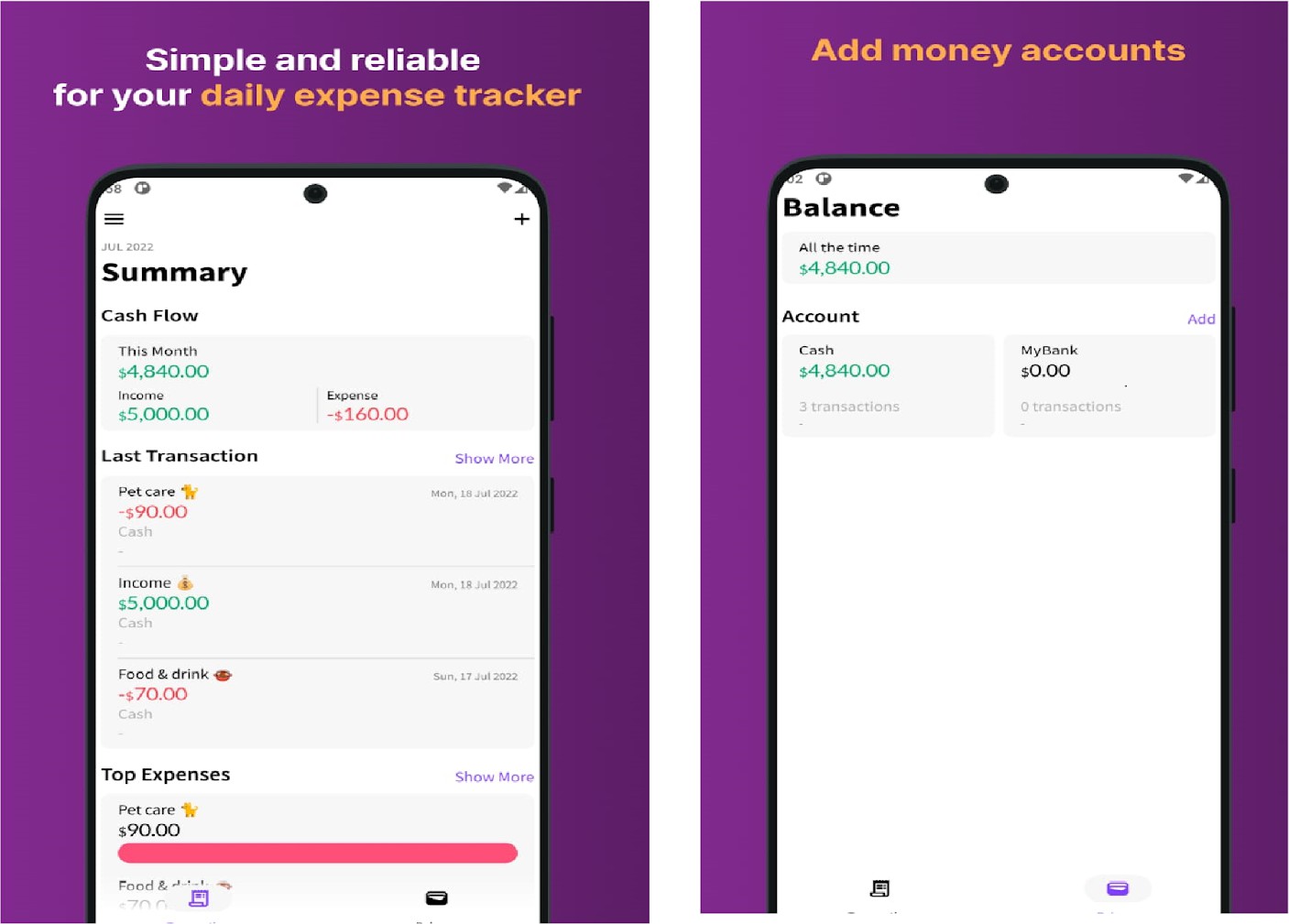
)

}

#### APPENDIX – 2

**SCREENSHOTS**

##### Sample Output

****

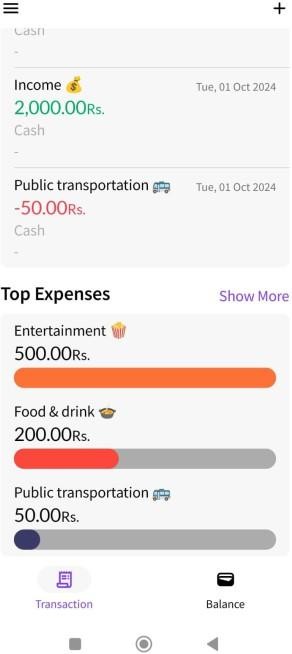
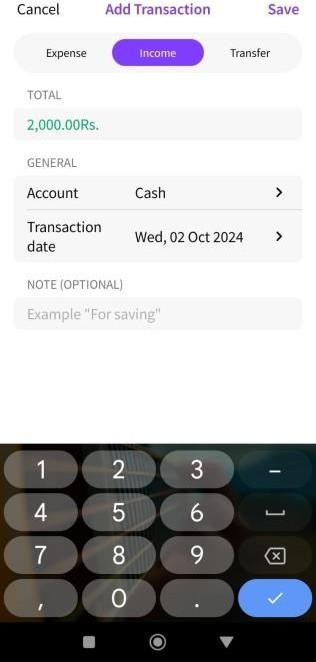
 

Fig 2.1: Execution of code

#### REFERENCES

#### Jetpack Compose Documentation: Android Developers. "Jetpack Compose." [Online]. Available: <https://developer.android.com/jetpack/compose>

#### 2 . ChatGPT: OpenAI. "Assistance with Technical Documentation and Learning Jetpack Compose." [Online]. Available: <https://chat.openai.com/>

#### Gemini AI: Gemini AI. "Assistance in Learning Jetpack Compose and Solving Errors in the Coding." [Online]. Available: <https://gemini.google.com/>

#### Google Developers. "Room Persistence Library." [Online]. Available: <https://developer.android.com/jetpack/androidx/releases/room>

#### Android Developers. "Building a Budget App with Jetpack Compose." [Online]. Available: <https://developer.android.com/codelabs/jetpack-compose-budget-app>

#### Material Design Guidelines: Google. "Material Design Components for Android." [Online]. Available: <https://material.io/develop/android>

#### Stack Overflow. "Best Practices for Android Expense Tracker Apps." [Online]. Available: <https://stackoverflow.com/questions/tagged/android-expense-tracker>

#### Ebrahimi, F., Yu, Y., Yu, Y., & Li, L. (2019). Mobile App Privacy in Software Engineering Research: A Systematic Mapping Study. arXiv preprint arXiv:1910.03622.

#### Lee, H., Kim, J., Kim, Y., & Kim, H. (2021). A Systematic Survey on Android API Usage for Data-Driven Analytics with Smartphones. arXiv preprint arXiv:2104.11271.

#### 10. Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011). Android Permissions: User Attention, Comprehension, and Behavior. Proceedings of the Eighth Symposium on Usable Privacy and Security (SOUPS '12), pp. 3:1–3:14.