ExpenseMate: A Mobile Personal Finance and Job Opportunity Assistant for Students

Abstract

University students often struggle to manage their daily expenses, subscriptions, and part-time job opportunities in a structured way. Many existing personal finance applications target working adults, and typically do not consider the specific financial patterns and needs of students. At the same time, information about student part-time jobs is scattered across group chats, posters, and generic job websites. This project presents ExpenseMate, an Android mobile application that acts as a small but complete finance assistant for students. ExpenseMate allows users to record and categorize expenses, manage subscriptions and monthly budgets, visualize their spending trends with charts, and browse student part-time job posts. The app is implemented using Flutter on the client side and Firebase (Firebase Authentication and Cloud Firestore) as Backend-as-a-Service. To protect privacy, financial data is stored under per-user paths (users/{userId}), while student job posts are shared through a global jobs collection with an admin-only posting rule. The goal of this report is to describe the motivation, design, implementation, and preliminary evaluation of ExpenseMate, and to discuss limitations and possible future extensions.

The full project source code is available on GitHub:
https://github.com/hongweiluhappy/ExpenseMate.git

1.Introduction

Managing a limited budget is a common and important challenge for university students. Many students rely on a combination of monthly allowances, scholarships, and part-time job income. However, they often lack the habit and tools to regularly track expenses, distinguish between necessary and optional spending, and monitor how close they are to their monthly budget limit. As a result, they may run out of money at the end of the month, or feel that their money “disappears” without knowing where it went.

At the same time, students frequently look for part-time jobs such as campus assistant, tutor, or service work in cafes and restaurants. Information about such opportunities is typically spread informally, for example via messaging groups or posters on campus. This makes it difficult to have a central and up-to-date view of relevant job opportunities. Moreover, there is usually no connection between the job information and the student’s personal financial planning.

Existing personal finance applications, such as general expense trackers or budgeting apps, are usually designed for working adults with regular salaries. They rarely combine expense tracking with student-oriented features such as small, flexible part-time jobs and student-specific spending categories. As a result, students often need to use separate tools for financial tracking and job search, and there is no integrated view of “how I spend” and “how I can earn.”

To address these problems, this project proposes ExpenseMate, a mobile application designed specifically for students. The main objectives of ExpenseMate are:

- To help students record and categorize expenses in a simple and fast way, including one-time expenses and recurring subscriptions.
- To provide monthly budget management with alerts when spending approaches or exceeds the budget.
- To offer visualizations such as charts of spending over time and by category, so that students can better understand their spending habits.
- To provide a simple student part-time job board integrated into the same app, where only admin users can post jobs but all students can browse them.
- To respect privacy by isolating financial data per user, while sharing job posts globally.

The rest of this report is organized as follows. Section 2 introduces background and design goals. Section 3 presents the system architecture. Section 4 describes the data model and storage design. Section 5 explains key implementation details of the main modules. Section 6 discusses evaluation and observations. Section 7 describes limitations and future work, and Section 8 concludes the report.

2.Background and Design Goals

2.1 Background

Personal finance management tools have become popular in recent years. Typical applications allow users to enter expenses, categorize them, and view summaries or charts. Some also support advanced features such as automatic bank account synchronization, bill reminders, and investment tracking. However, many of these tools assume a relatively stable monthly income, and may be too complex or not tailored to the lifestyle of students.

On the other hand, student job platforms and campus job boards focus mainly on listing available positions, often without any integration into students' daily planning or budgeting tools. For example, a student may find a part-time job on one website and then use an unrelated app to track the income and spending. The lack of integration makes it harder to answer questions like "How much do I need to work to cover my current spending?" or "If I accept this job, how will it affect my budget?"

In the context of a university software engineering course, this project aims to explore how modern mobile technologies such as Flutter and Firebase can be used to quickly prototype an integrated tool that covers both personal expense management and exposure to part-time job opportunities.

2.2 Design Goals

The main design goals of ExpenseMate are:

- Student-focused: Support typical student spending categories (such as food, transportation, entertainment, and study materials) and scenarios like small, irregular income from part-time jobs.
- Simplicity and usability: Provide a clear and minimal user interface so that students can

quickly add expenses and browse information without complex configuration.

- Data privacy: Ensure that each student's financial data is private, while non-sensitive information such as job postings is shared.
- Lightweight architecture: Use a simple client–cloud architecture without maintaining a custom backend server, in order to reduce development and deployment complexity.
- Extensibility: Design the data model and architecture so that new features, such as more advanced analytics or richer job filtering, can be added later.

3.System Architecture

ExpenseMate follows a client–cloud architecture. The client is a Flutter application running on Android devices. The backend uses Firebase services, mainly Firebase Authentication for user management and Cloud Firestore as a NoSQL document database.

On the client side, the app is built with a layered structure:

- The user interface (UI) layer is implemented with Flutter and follows Material Design 3 guidelines. It includes screens such as the home page, expense form, subscriptions page, charts page, and jobs page.
- Application state is managed by a lightweight singleton class called AppState. This class maintains in-memory lists of expenses, subscriptions, job posts, and budget information for the currently logged-in user.
- A separate FirestoreService class encapsulates all interactions with Cloud Firestore, such as loading and saving expenses, subscriptions, budget settings, and jobs.

On the backend side, Firebase Authentication manages user accounts and provides secure user IDs that are used as keys in Firestore. Cloud Firestore stores all persistent data. Optional notification mechanisms can be used to deliver local or push notifications for budget and subscription alerts.

Figure 1 shows an overview of this architecture. The Flutter UI interacts with AppState, which delegates data access to FirestoreService. FirestoreService communicates with Firebase Authentication and Cloud Firestore using the official SDK. This structure keeps the business logic and data access logic separated from the UI and makes it easier to test and extend.

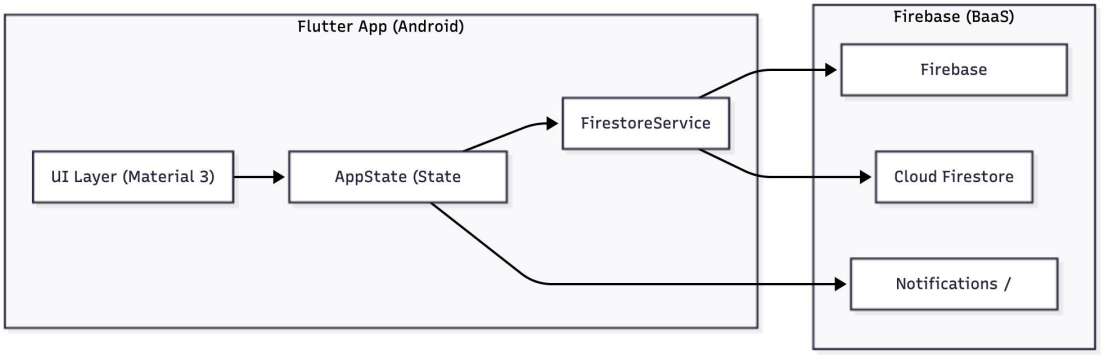

Figure 1. Overall architecture and tech stack.

4.Data Model and Storage Design

The central design decision of the data model is to clearly separate per-user financial data from globally shared job data, while still keeping everything in a single Firestore database.

The Firestore structure is organized as follows:

- users/{userId}: One document per user. Besides basic metadata, this document contains a boolean field isAdmin that indicates whether this user is allowed to post new jobs.
- users/{userId}/expenses: A subcollection of expense documents under each user. Each expense includes attributes such as amount (typically stored in cents to avoid floating point issues), category, spending date, optional note, and timestamps.
- users/{userId}/subscriptions: A subcollection of subscription documents representing recurring payments such as gym memberships or online services. Each subscription has a name, amount, billing cycle (e.g., monthly), billing day, category, optional note, active flag, and timestamps.
- users/{userId}/budget: A document or small collection storing budget-related settings, including the monthly budget amount in cents and information about which alert thresholds have already been sent.
- jobs: A top-level collection of job post documents shared by all users. Each job includes a title, pay information, location, contact details, description, and timestamps.

With this structure, each student's expenses, subscriptions, and budget are kept under their own users/{userId} path, and other users cannot access them. At the same time, the jobs collection is shared so that all students see the same list of student part-time jobs. Admin privileges (isAdmin) are stored on the user document so that the app can determine whether to show job posting features.

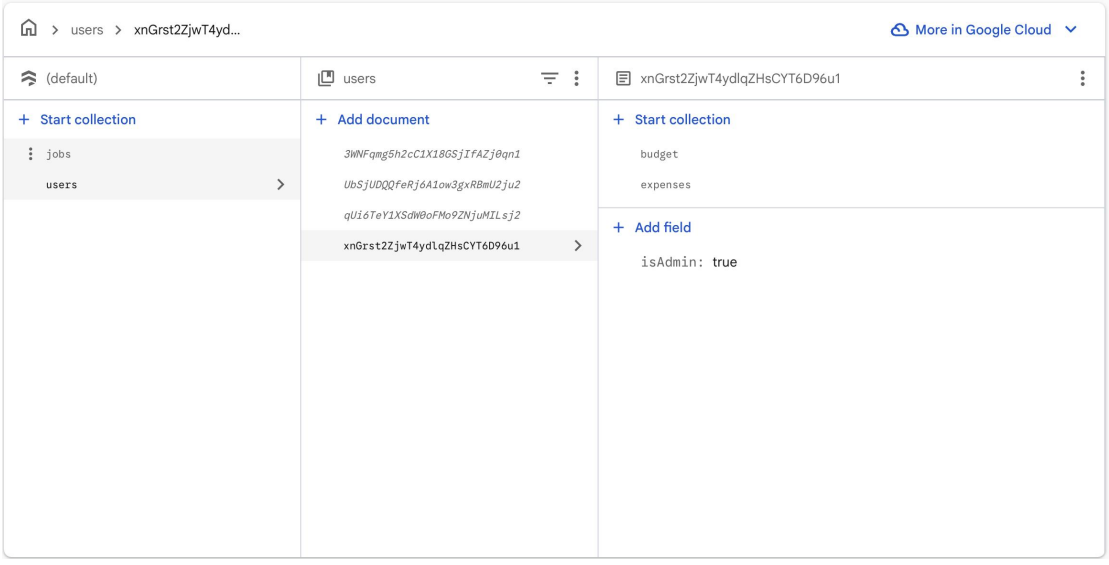

Figure 2. Firestore data model and collection structure.

5.Implementation

5.1 Authentication and Role Management

Authentication in ExpenseMate is implemented using Firebase Authentication with email and

password. A small AuthService wrapper class is used to abstract away the details of the Firebase SDK and provide methods such as sign-up, sign-in, sign-out, and retrieval of the current user ID.

When a user signs in, the app obtains their user ID from Firebase Authentication. The FirestoreService then loads the corresponding users/{userId} document to check whether the isAdmin field is set to true. This result is stored in AppState as a boolean flag isAdmin.

In the user interface, particularly on the jobs page, the app uses the isAdmin value to control access to job posting functionality. Only admin users see the “Post Job” button and can access the job posting form. All users, including non-admins, can view the list of job posts and open job details. This simple role mechanism supports basic moderation without requiring a complex permission system.

5.2 Expense Tracking and Budget Management

Expenses are represented as Expense objects in the Flutter app and as documents in the users/{userId}/expenses subcollection in Firestore. The home page shows important aggregated values such as monthly spending and today’s spending, as well as a list of recent expenses.

When the user taps the button to add a new expense, the app opens a form where the user can enter the amount, choose a category (e.g., food, transport, entertainment), select the date, and optionally write a note. After the user saves the form, the app sends the new expense to Firestore via FirestoreService and then reloads or updates the in-memory expense list in AppState.

Budget management is implemented by storing a monthly budget amount (in cents) under users/{userId}/budget. When expenses are added or modified, the app calculates the total spending for the current month and compares it to the budget. If the spending crosses certain thresholds, such as 80% or 100% of the budget, the app can trigger alerts. To avoid sending repeated alerts for the same threshold, the app stores markers (for example, which thresholds have already been alerted) in the budget settings.

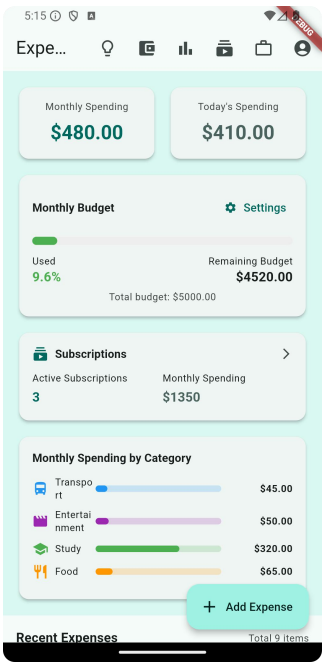

Figure 3. Home page showing monthly and today spending with the add-expense flow.

5.3 Subscription Management

Many student expenses are recurring, such as gym memberships, public transport passes, and streaming services. To better support these, ExpenseMate includes a subscription management module. Subscriptions are stored in the users/{userId}/subscriptions subcollection in Firestore and represented as Subscription objects in the app.

Each subscription has attributes such as name, amount, billing cycle (for example monthly), billing day in the month, category, whether the subscription is active, and an optional note. The subscription page allows users to view, add, edit, and deactivate subscriptions.

The app can automatically create expense entries for subscriptions on their billing day. For example, when the billing date for a gym membership arrives, the app may generate a corresponding expense record in the expenses collection. In addition, the app can display upcoming subscription charges and optionally send reminders a few days before the billing date, helping students plan their cash flow.

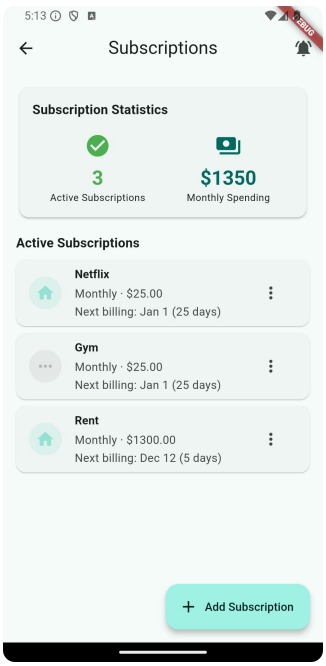

Figure 4. Subscription management page.

5.4 Charts and Insights

To help students move from raw numbers to understanding and action, ExpenseMate provides a charts and insights module. Based on the stored expense data, the app aggregates amounts by time period and by category to generate visualizations.

Typical charts include:

- Time-based charts such as weekly or monthly total spending, which show how spending evolves over time and help identify months or weeks with unusually high expenses.
- Category allocation charts, which visualize how spending is distributed among categories such as food, transport, rent, entertainment, and education.

These charts are implemented on the client side using Flutter chart widgets. AppState prepares the aggregated data, and the UI components display it in the form of bar charts, line charts, or pie charts. By observing these charts, students can quickly see which categories dominate their spending and whether there are opportunities to adjust their behavior.

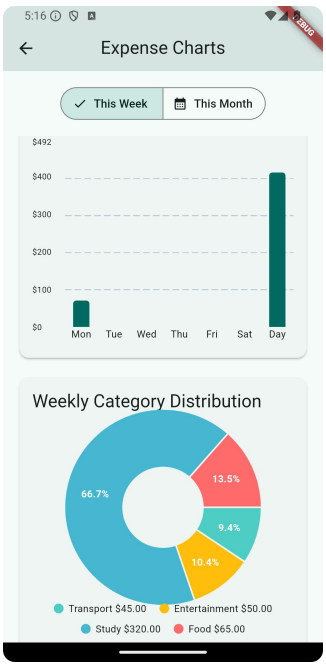

Figure 5. Charts showing spending trends and category allocation.

5.5 Student Part-time Job Module

The student part-time job module is centered around the global jobs collection in Cloud Firestore. A JobPost model represents each job with fields such as title, pay description, location, contact information, and a detailed description of responsibilities and requirements.

The jobs page shows a list of job cards. Each card usually displays the job title, pay, and possibly location. When the user taps on a job card, the app opens a detailed view, for example as a modal bottom sheet, showing full information and contact details required to apply for the job.

Only admin users can create or edit job posts. For admin users, the jobs page includes an additional button that opens a job posting form. In this form, the admin can enter title, pay, location, contact information, and description, and then save the new job to the jobs collection via FirestoreService. Non-admin users do not see this button and cannot directly create job posts, which helps maintain a basic level of moderation.

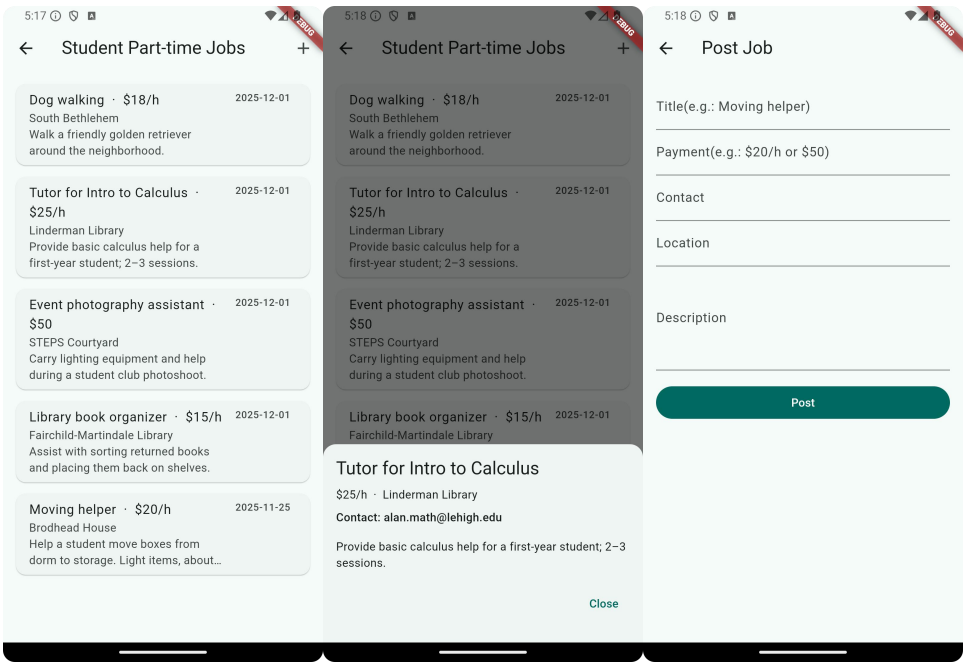

Figure 6. Jobs module: list screen (left), job detail view (middle), and admin job posting form (right).

6.Evaluation

As this project is developed in the context of a course, evaluation focuses on functional correctness, basic performance, and informal usability observations rather than a large-scale user study.

From a functional perspective, the main workflows were tested manually:

- User registration and login through Firebase Authentication.
- Adding, viewing, and deleting expenses, and checking that the monthly and daily totals update correctly.
- Setting and updating the monthly budget, and verifying that alerts are triggered when spending crosses thresholds.
- Adding, editing, and deactivating subscriptions, and (where implemented) generating corresponding expense entries on billing days.
- Viewing charts of spending over time and by category, and checking that they reflect the underlying expense data.
- Browsing the list of jobs as a normal user and opening job details.
- Logging in as an admin user and verifying that the job posting button is visible and that new job posts appear correctly for all users.

Performance tests on a typical Android device show that Fetching data from Firestore and updating the UI are sufficiently fast for the expected scale of a student's personal data (hundreds or thousands of expense records). Caching data in AppState reduces the need for repeated network requests, and Firebase's real-time capabilities can be leveraged for live updates if needed.

Informal usability feedback from a small number of peers suggests that the interface is understandable and that integrating job information into the same app as expense tracking is

perceived as useful. However, a formal user study with more participants and controlled tasks has not been conducted within the scope of this course project.

7.Limitations and Future Work

Despite its functionality, ExpenseMate has several limitations.

First, role management is currently very simple. The isAdmin flag is set manually in Firestore, and there is no user interface for changing roles or handling more complex permissions. In the future, a richer role and moderation system, possibly combined with server-side logic or Firebase security rules, could improve robustness.

Second, analytics features are basic. The charts and alerts focus on simple aggregations and thresholds, without personalized recommendations or predictions. Future work could include automatic category suggestions for new expenses based on past data, detection of unusual spending patterns, or customized saving advice.

Third, job search capabilities are limited. Currently, jobs are displayed as a simple list without advanced filtering or ranking. Future work could introduce search by keyword, filters by location or pay, and bookmarking or application-tracking features.

Fourth, evaluation is limited to manual testing and informal feedback. A more rigorous evaluation could involve deploying the app to a group of students for a longer period, collecting anonymized usage data and survey responses, and analyzing whether the app leads to measurable improvements in budgeting behavior.

8. Conclusion

This report has presented ExpenseMate, a mobile application that combines personal finance management and student part-time job information in a single tool. Built with Flutter and Firebase, the app demonstrates how a lightweight client–cloud architecture can support key features such as expense tracking, subscription and budget management, charts and insights, and a moderated job board.

By storing financial data under per-user paths and sharing job posts through a global collection, ExpenseMate balances privacy and usefulness. The addition of an admin-only job posting role provides basic moderation without complicating the user experience for normal students.

Although the current version has limitations and has only been evaluated at a small scale, it shows the potential of integrating financial tracking and job information for students. Future work could enrich analytics, extend job features, improve role management, and conduct larger-scale user studies to better understand how such tools can support students in managing both their spending and income.