Tribhuvan University

Academia International College

Final Year Project Report

On

Personal Finance Tracking System

[CSC 412]

Under the supervision of

“Mr. Ananda Adhikari”

Submitted by

Bibek Thapa (T.U. Exam Roll No. 29007/078)

Sachana Maharjan (T.U. Exam Roll No. 29025/078)

Ushmita Basnet (T.U. Exam Roll No. 29034/078)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

September 2025

Tribhuvan University

Academia International College

Final Year Project Report

On

Personal Finance Tracking System

[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

Submitted by

Bibek Thapa (T.U. Exam Roll No. 29007/078)

Sachana Maharjan (T.U. Exam Roll No. 29025/078)

Ushmita Basnet (T.U. Exam Roll No. 29034/078)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

September 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision by Mr. Bibek Thapa (29007/078), Ms. Sachana Maharjan (29025/078) and Ms. Ushmita Basnet (29034/078) entitled "Personal Finance Tracking System" be accepted as fulfilling in partial requirements for the degree of Bachelor of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

……..

Mr. Ananda Adhikari

Project Supervisor

Department of Computer Science and Information Technology

Academia International College

Tribhuvan University

Department of Computer Science and Information Technology

Academia International College

Certificate of Approval

This is to certify that this project prepared by Mr. Bibek Thapa, Ms. Sachana Maharjan and Ms. Ushmita Basnet entitled "Personal Finance Tracking System" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. Mr. Ananda Adhikari Project Supervisor Department of Computer Science and IT Academia International College	…….. Mr. Bishwas Mathema HOD/Program Coordinator Department of Computer Science and IT Academia International College
…….. Internal Examiner Academia International College	…….. External Examiner Central Department of CSIT Tribhuvan University

Acknowledgement

We would like to thank Academia International College for giving us the opportunity to carry out this project as part of our course. Working on this project has helped us gain practical knowledge and skills that go beyond classroom learning. It gave us real experience in planning, developing, and completing a system as a team, which will be very useful in our future careers.

We would like to specially thank our supervisor, Mr. Ananda Adhikari, for his constant guidance, encouragement, and valuable suggestions throughout the project. His support at every stage, from the beginning to the final report, has been very important for us. We would also like to thank Mr. Bishwas Mathema who supported us during this project. His help, whether in sharing knowledge, giving advice, or providing resources, has played a big role in making this project possible.

Thanking You,

Bibek Thapa (29007/078)

Sachana Maharjan (29025/078)

Ushmita Basnet (29034/078)

Abstract

The Personal Finance Tracking System (PFTS) is a web-based application designed to help users manage and monitor their personal finances by recording income, expenses, and budgets. The primary objective of PFTS is to provide users with platform to record their earnings, spendings and provide clear, actionable insights into their financial behavior, helping them to make better decisions and achieve their financial goals.

PFTS offers essential features such as transaction entry, budget creation, and real-time feedback on spendings. A key innovation of the system is its budget pacing algorithm, which analyzes users spending patterns throughout the month and provides timely financial insight to help them stay on track. The application focuses on user privacy and simplicity by allowing manual data entry without requiring sensitive bank integrations. This project report details the design, development, and testing of the system, which is built using Node.js, Express.js, MongoDB with Mongoose, EJS templating and CSS for the frontend.

Keywords: Budget pacing, personal finance tracker, expense management, web application, financial insight

Table of Contents

Supervisor's Recommendation.....i
Certificate of Approval.....ii
Acknowledgement iii
Abstract.....iv
List of Figures.....vii
List of Tables..... viii
List of Abbreviations.....ix
Chapter 1: Introduction.....1
1.1 Introduction.....1
1.2 Problem Statement.....1
1.3 Objectives 2
1.4 Scope and Limitations.....2
1.4.1 Scope.....2
1.4.2 Limitations 2
1.5 Development Methodology 3
Chapter 2: Background Study and Literature Review 5
2.1 Background Study.....5
2.2 Literature Review.....6
Chapter 3: System Analysis 8
3.1 System Analysis 8
3.1.1 Requirement Analysis 8
3.1.2 Feasibility Analysis.....10
3.1.3. Object Modeling using Class and Object Diagram.....12
3.1.4. Dynamic modelling using state and sequence diagram 13
3.1.5 Process modelling using activity diagram 14
Chapter 4: System Design.....15
4.1. Design 15
4.1.1 Refinement of Class, Object and Activity Diagram.....15
4.1.2 Component Diagram.....15
4.2 Algorithm Details.....16
Chapter 5: Implementation and Testing 18

5.1 Implementation 18
5.1.1 Tools Used 18
5.1.2 Implementation Details for Modules 19
5.2 Testing 21
5.2.1 Test Cases for Unit Testing 21
5.3. Result Analysis 26
Chapter 6: Conclusion and Future Recommendation 27
6.1 Conclusion 27
6.2 Future Recommendation 27
References 28
Appendices 29

List of Figures

Figure 1.1: Agile Development Methodology 3
Figure 3.1: Use Case Diagram 9
Figure 3.2: Gantt Chart 11
Figure 3.3: Class Diagram 12
Figure 3.4: Sequence Diagram 13
Figure 3.5: Activity Diagram 14
Figure 4.1: Component Diagram 15

List of Tables

Table 5.1: Unit Test Cases for Authentication 21
Table 5.2: Unit Test Cases for Transaction Management 22
Table 5.3: Unit Test Cases for Budget Management 22
Table 5.4: Unit Test Cases for Insight System 23
Table 5.5: Unit Test Cases for Email Service 24
Table 5.6: System Test Cases for Authentication Flow 24
Table 5.7: System Test Cases for Transaction Workflow 25
Table 5.8: System Test Cases for Budget Management 25
Table 5.9: System Test Cases for Dashboard Integration 26

List of Abbreviations

CSS	Cascading Style Sheets
CSV	Comma-Separated Value
EJS	Embedded Javascript
MVP	Minimum Viable Product
PFTS	Personal Finance Tracking System
YNAB	You Need a Budget

Chapter 1: Introduction

1.1 Introduction

Managing personal finances is a challenge that nearly everyone faces, whether it is a student, a working professional, or managing a household. With many ways to spend and receive money such as cash, cards, online payments, and mobile wallets, it is easy to lose track of where the money goes. Many people try to keep records in notebooks or on their phones, but these methods are often disorganized, easy to forget, or too basic to provide a clear picture of their financial situation. This is where digital solutions become valuable.

The Personal Finance Tracking System (PFTS) is developed to make managing money simpler and smarter for everyone. The core idea behind PFTS is to offer users a straightforward, user-friendly web application where they can record their income and expenses, set budgets for categories such as food, rent, or transportation, and receive helpful advice to stay on track. Unlike many apps that only show numbers, PFTS uses a budget pacing algorithm to provide real-time feedback and suggestions. This means the system not only tracks how much users have spent but also evaluates whether users are spending at a healthy rate, sends warnings if users are overspending too quickly, and encourages users when they are managing their budget well. The entire system is designed to be easy to use, visually clear, and genuinely helpful for everyday users.

1.2 Problem Statement

Traditional ways of managing personal finance, such as notes or spreadsheets, are not enough for today's fast-moving world. People face many challenges when it comes to handling their money effectively. Tracking every income and expense accurately is often difficult, and there is usually no real-time feedback to show how spending habits are developing. Existing applications in the market are either too simple to be useful or too complicated for average users, making them hard to adopt. Similarly, there are no clear warnings before overspending occurs, which makes it harder for individuals to stay within their limits. Many users also struggle to stick to budgets and eventually lose motivation, which limits long-term financial discipline.

1.3 Objectives

The main objectives of the Personal Finance Tracking System are:

- To provide an easy way for users to record every income and expense.
- To allow users to set up and manage budgets across various categories.
- To implement budget pacing algorithm that monitors user budgets, spending and delivers real-time insights.

1.4 Scope and Limitations

1.4.1 Scope

The scopes of PFTS include:

- User registration and secure authentication.
- Adding, editing, and deleting income and expense transactions.
- Setting up and managing budgets for different categories (e.g., food, rent, travel, entertainment).
- Providing dashboard with summaries and simple charts for easy understanding.
- Giving real-time feedback using the budget pacing algorithm.
- Allowing users to export their data as CSV files.

1.4.2 Limitations

The limitations of PFTS include:

- Single-user system (no group or family budgeting).
- Does not connect directly to bank accounts; all entries must be made manually.
- No mobile app version (web only, but mobile-friendly).
- No support for investment or asset tracking beyond basic income/expense.

1.5 Development Methodology

The development of the Personal Finance Tracking System (PFTS) was carried out using an Agile methodology tailored to fit the needs of our small, three-member team. Instead of following formal Scrum roles such as: Scrum Master or Product Owner, our team maintained collaborative structure where all members shared responsibilities equally.

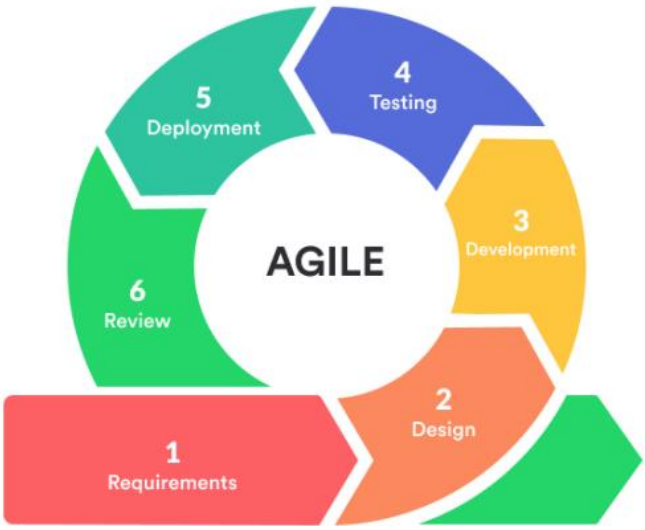

Figure 1.1: Agile Development Methodology

The figure above represents the stages of Agile development methodology. Here, we organized our work into weekly sprints, each consisting of a clear sequence of phases:

- **Sprint Planning:** The team met weekly with the project supervisor to define the objectives and scope for the upcoming sprint.
- **Design Phase:** Mockups and technical designs were created to guide the development of planned features.
- **Development Phase:** Features were implemented using Node.js and Express following the Model-View-Controller (MVC) architecture.
- **Testing and Review Phase:** Developed features were thoroughly tested, bugs were fixed, and progress was demonstrated to the supervisor for feedback.

Regular communication and coordination were maintained through informal daily meetings, enabling the team to quickly address issues and adapt to changing requirements. This iterative approach allowed for continuous improvement based on supervisor input and user feedback.

By sharing responsibilities and collaborating closely, the team was able to deliver functional software incrementally, ensuring that core features such as user authentication and transaction management were completed before advancing to more advanced features like budget tracking and data visualization.

1.6 Report Organization

This report is organized into six chapters as follows:

Chapter 1: Introduces the project by presenting the background, problem statement, objectives, scope and limitations, development methodology, and overall structure of the report.

Chapter 2: Provides the background study of fundamental concepts related to the project and reviews similar tools and related works to highlight the research gap.

Chapter 3: Focuses on system analysis, including requirement analysis (functional and non-functional), feasibility study (technical, operational, economic, and schedule), and system modeling through appropriate diagrams.

Chapter 4: Describes the system design, covering database design, interface and dialogue design, and the details of the algorithms used, such as the budget pacing algorithm.

Chapter 5: Explains the implementation process and testing of the system. It includes tools used, module implementation details, unit and system test cases, and analysis of results.

Chapter 6: Presents the conclusion of the project along with future insights for improvements and possible extensions.

Chapter 2: Background Study and Literature Review

2.1 Background Study

Managing money has always been a part of daily life. In the past, people used notebooks, ledgers, or mental notes to track their income and expenses. But as financial lives become more complex due to mobile payments, online shopping, and multiple income sources, these older methods are no longer enough [1].

Tracking finances on paper might feel familiar, but it often leads to errors, lost records, and inconsistent logging. There is no built-in calculation or visual feedback, and over time, it becomes harder to maintain. Spreadsheets solve some of these issues by allowing custom formulas and structured data entry. However, they come with their own problems. Most spreadsheets require a basic understanding of formulas, formatting, and logic. Research shows that over 70 percent of user-made budgeting spreadsheets contain serious errors that can affect financial decisions [2]. Moreover, they are not mobile-friendly, not intuitive for the average user, and offer limited automation or reminders.

In contrast, personal finance apps are built specifically to handle these issues. They offer quick data entry, visual insights, and access from any device. These tools are helping people build better financial habits, avoid overspending, and gain more awareness of their money [3].

What is surprising is how well the simplest apps perform. Applications that only let users manually log income and expenses and show basic bar or pie charts are downloaded by millions of people. For instance, the “Spending Tracker” app, with no advanced features, has over 1 million downloads on Android alone [4]. Clearly, the market values simplicity and control over complexity and automation. Studies in behavioral finance suggest that the easier it is to use a tool, the more likely users are to stick with it. Simple interfaces that offer immediate visual feedback encourage routine tracking, which is key to long-term financial awareness [5].

A global survey showed that 48 percent of personal finance app users prefer manual-entry apps over those that sync with bank accounts. Privacy concerns and a desire for hands-on awareness were the top reasons [6]. In country like Nepal, these trends are even more relevant. Limited access to digital banking, low mobile data speeds, and trust issues around

data sharing mean users are more likely to adopt apps that work offline, do not require account linking, and offer a straightforward experience [7].

The Personal Finance Tracking System (PFTS) was designed with these realities in mind. Instead of competing on complexity, it focuses on core features that matter to everyday users. It allows income and expense logging, shows visual summaries, and keeps everything private and under user control. In doing so, it serves the real needs of people who want clarity, simplicity, and reliability in managing their finances.

2.2 Literature Review

Many researchers have studied the way people handle their personal finances. A key insight from behavioral economics is that individuals often make mistakes when managing money. They may underestimate their spending, forget to record transactions, or fail to save regularly [1]. These behaviors are not only common but also predictable, which means tools can be designed to help users avoid them.

Financial literacy plays an important role in reducing these errors. Studies show that people who understand budgeting, interest rates, and debt are more likely to make better financial decisions [2]. But knowledge alone is not always enough. Without a simple way to track financial behavior, even well-informed users may struggle to stay on track [3].

This is where digital finance apps have had an impact. Tools like Mint, YNAB (You Need a Budget), and PocketGuard offer users an easy way to track spending, plan budgets, and receive reminders. Mint connects to bank accounts and categorizes transactions automatically [6]. YNAB focuses on forward planning by helping users assign a purpose to every dollar they earn [7]. PocketGuard shows how much money is available after accounting for bills and goals [8].

In addition to these well-known apps, several popular Android applications demonstrate the widespread demand for simple spending trackers. Apps like Spendee, Spending Tracker, and Monefy have millions of downloads worldwide. For example, Spendee has over 1 million downloads and offers easy manual entry, clear charts, and budgeting features [12]. Monefy, with more than 5 million downloads, is praised for its straightforward interface and quick expense logging. These applications thrive despite their relatively simple functionality compared to full-featured banking apps. Their success highlights that many users prefer tools that prioritize ease of use and quick insights over complexity.

Research indicates that people are more likely to maintain financial tracking habits when the app interface is simple, visual, and provides immediate feedback [9].

However, concerns around privacy and security remain a barrier for some users. Many hesitate to link bank accounts or share sensitive data with finance apps, especially in regions with less developed digital infrastructure [10] [11].

The Personal Finance Tracking System (PFTS) addresses these gaps by offering a simple, privacy-conscious tool focused on manual input and straightforward visual feedback. It avoids linking to financial institutions, reducing risk and appealing to users concerned about security. By focusing on essentials tracking income and expenses, displaying bar graphs, and providing budget pacing. PFTS meets the needs of users who want a practical, easy-to-use system for everyday money management.

Chapter 3: System Analysis

3.1 System Analysis

For the Personal Finance Tracking System (PFTS), this step was crucial to ensure that the final product would be both useful and user-friendly. The analysis phase included gathering requirements, evaluating feasibility, and planning for both technical and practical aspects of the project.

3.1.1 Requirement Analysis

i. Functional Requirements

For PFTS, the main functional requirements are:

- User registration and secure login/logout.
- Adding, editing, and deleting income and expense transactions.
- Setting up budgets for different categories (e.g., food, rent, travel).
- Viewing dashboard summaries of income, expenses, and remaining balance.
- Generating simple charts and reports for better understanding.
- Providing real-time feedback and insight using the budget pacing algorithm.
- Exporting financial data as CSV files.
- Managing user profiles and account settings.

The use case diagram represents the functionality of our project from the perspective of user. A user can register and verify their email, log in, and view an interactive dashboard to monitor finances. They can add income and expense transactions, allocate budgets monthly and by category, and receive budget insight for better financial planning. This use case diagram shows the user flow on personal finance tracking system:

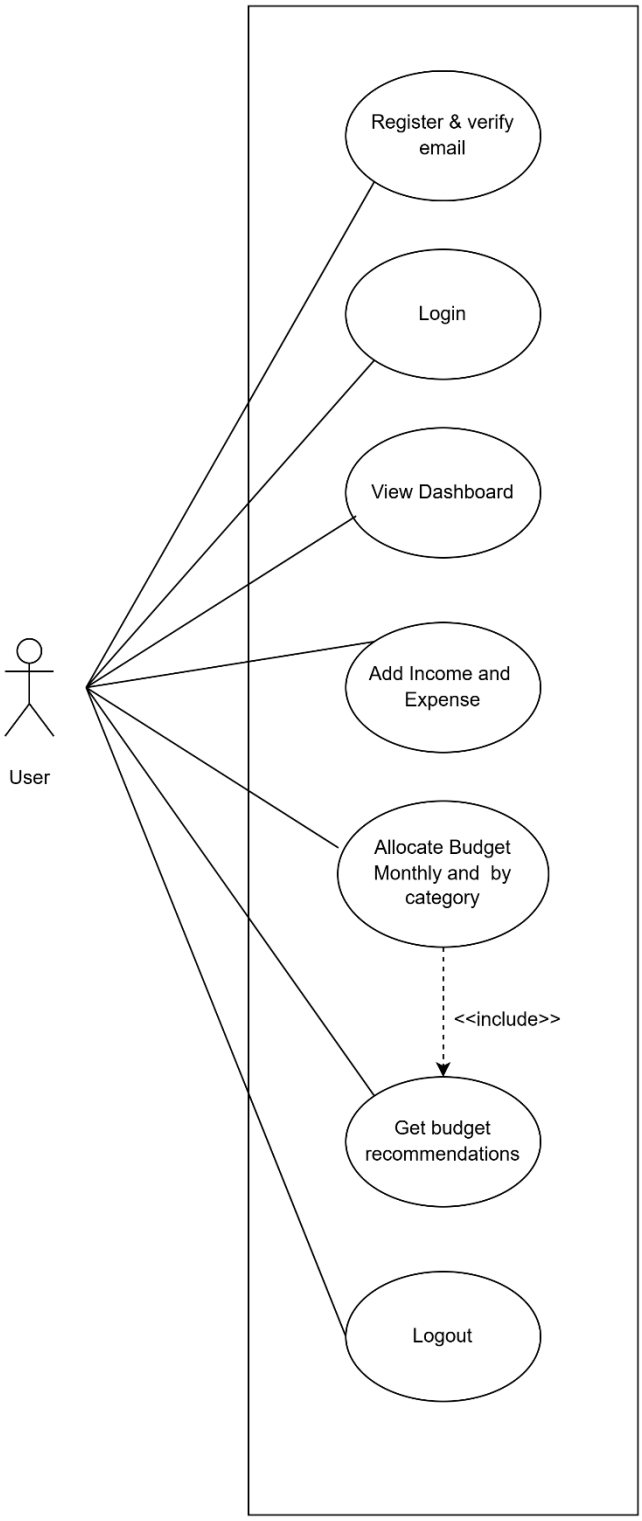

Figure 3.1: Use Case Diagram

ii. Non-Functional Requirements

For PFTS, the non-functional requirements include:

- The system is easy to use and understand, even for beginners.
- Data security and privacy are ensured at all times.
- The application is responsive and works well on both computers and mobile devices.
- The system provides fast performance with minimal loading times.
- User data is stored reliably with backup measures in place.
- The system is maintainable and can be updated easily in the future.
- Proper error messages and help options are available to guide users.

3.1.2 Feasibility Analysis

i. Technical Feasibility

The project is technically feasible due to its reliance on proven, open-source technologies requiring minimal hardware and software costs. The technology stack includes JavaScript for development, Node.js for the backend, EJS for the frontend, and MongoDB as the database server. These components can be hosted locally for development and demonstration purposes. EJS enables dynamic rendering of user interfaces, while MongoDB offers a flexible, NoSQL database solution for handling transaction data. These easily accessible technologies ensure that the project is technically viable.

ii. Operational Feasibility

The project is operationally feasible, as it can be achieved with the team's available resources within the given time constraints. The team comprises members with sufficient expertise in programming, web development, and UI/UX design. The chosen technology stack supports rapid development and prototyping, making it suitable for creating MVP. This ensures that the project is achievable with the team's skills and resources.

iii. Economic Feasibility

The project is economically feasible, as it uses free and open-source tools, eliminating the need for significant monetary investment. Development tools such as Visual Studio Code, Git, GitHub and Draw.io are freely available, and the software components can be hosted on existing hardware. This cost-effective approach ensures the project's economic viability.

iv. Schedule Feasibility

The project was divided into clear phases: planning, design, development, testing, documentation. Each phase was estimated and tracked using simple project management tools. The overall timeline was realistic, allowing for feedback and improvements at each stage.

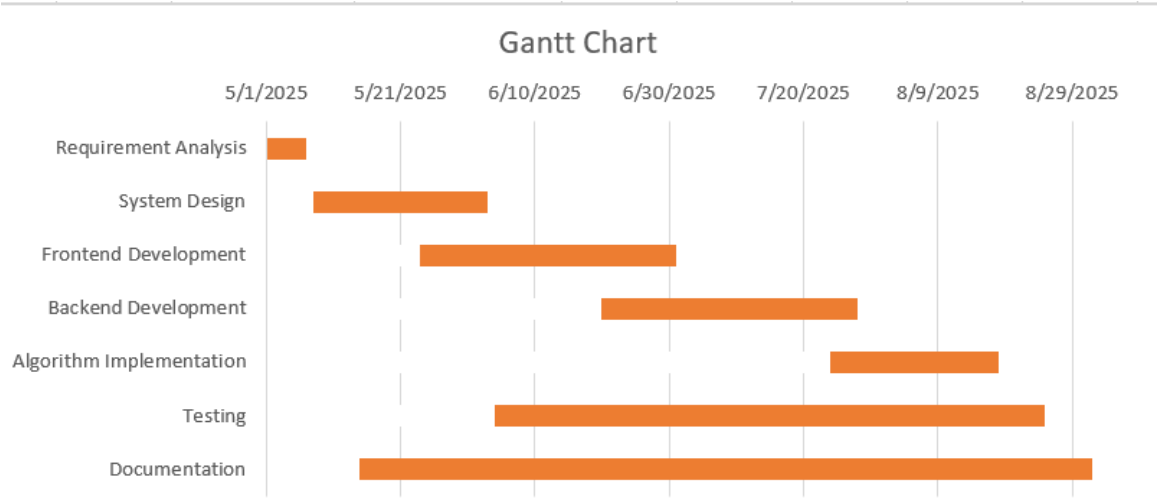

Figure 3.2: Gantt Chart

This Gantt chart provides a visual work schedule for our project spanning eleven weeks. Each phase progresses sequentially, ensuring focused execution. The timeline provides a clear roadmap from planning to completion. First, the initiation of the project was done with requirement gathering phase. Here, we gathered all the information and researched on what type of software to develop. Second, we designed the system which helped us to understand the software workings and user flows, then we moved forward with frontend development followed by backend development. Then, we tested the entire system with different levels of testing like unit testing and system testing. Along the way we documented the entire development process as shown in the diagram.

3.1.3. Object Modeling using Class and Object Diagram

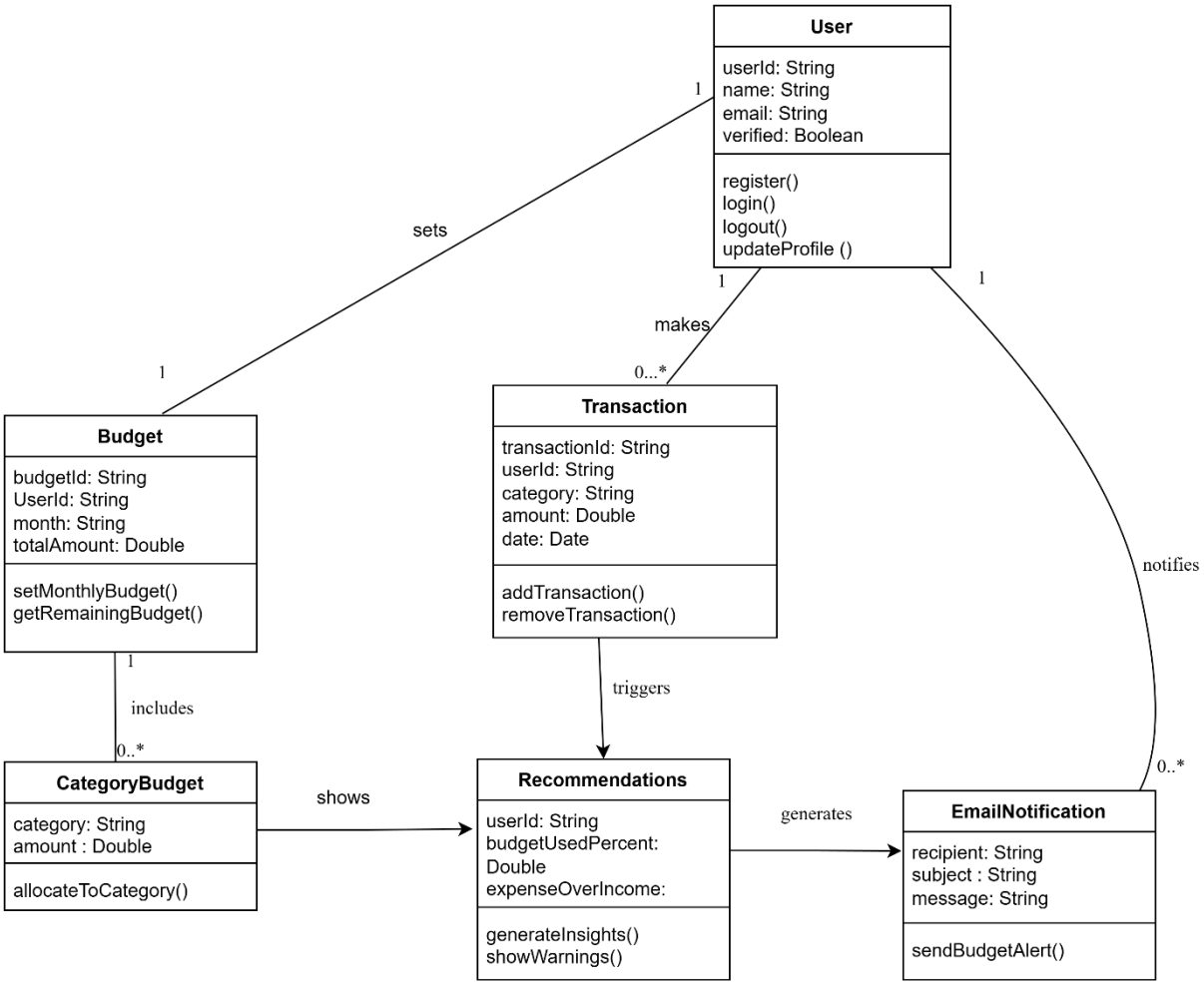

Figure 3.3: Class Diagram

This class diagram outlines the core components of a personal finance management system. The User with its attributes can perform different actions. Each user can make Transactions which in turn trigger personalized insight (generating insights and warnings based on spending patterns). Additionally, the system sends Email Notification alerts to users.

3.1.4. Dynamic modelling using state and sequence diagram

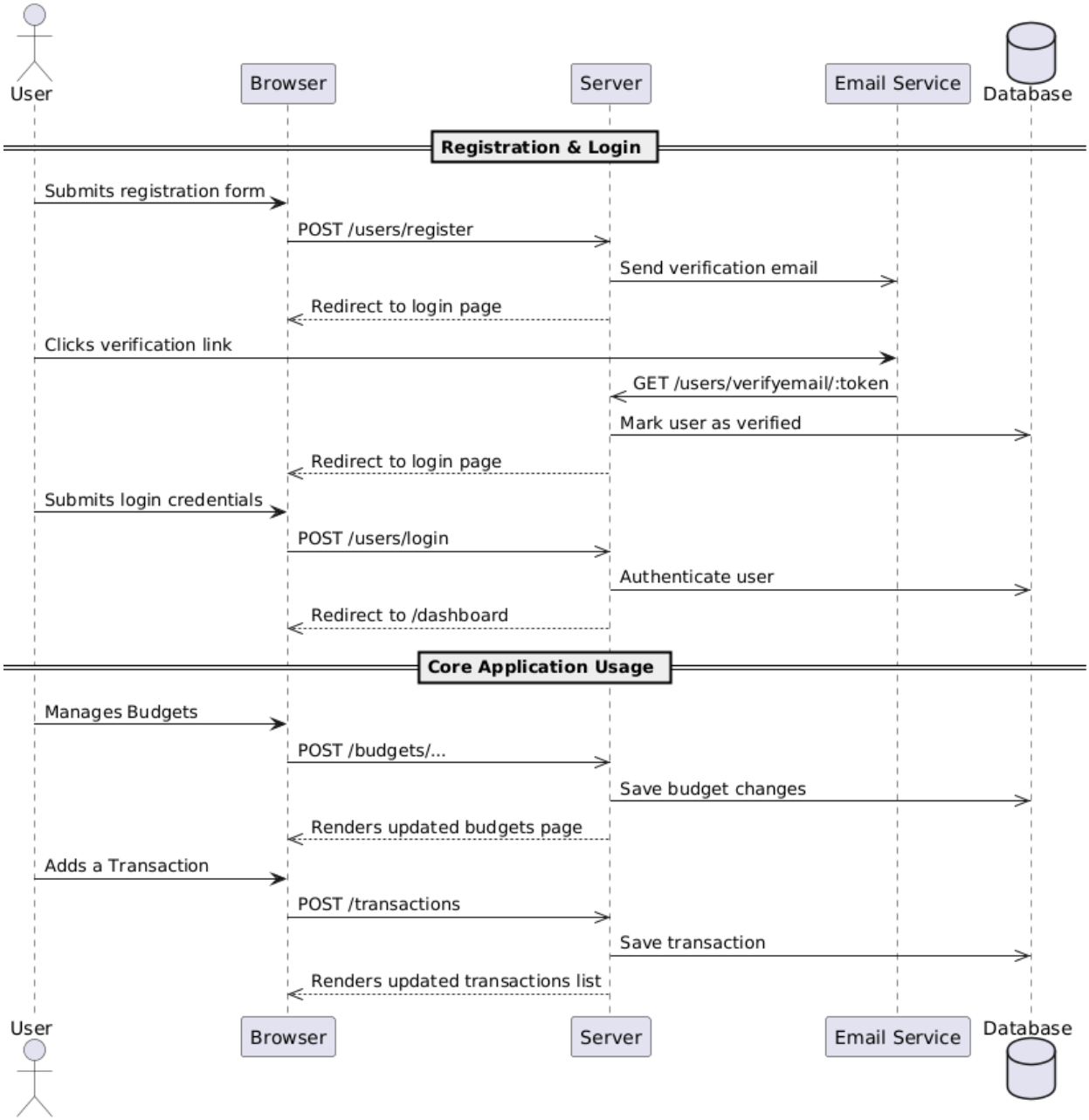

Figure 3.4: Sequence Diagram

This sequence diagram outlines a user's journey from registration to budget management: they sign up, verify via email, log in, then interact with the system to manage budgets and record transactions, with all actions securely processed through the server and database. The flow ensures data integrity and provides real-time updates back to the user's browser.

3.1.5 Process modelling using activity diagram

This activity diagram illustrates the user journey: from login and transaction management to budget tracking and all other functionalities in the application.

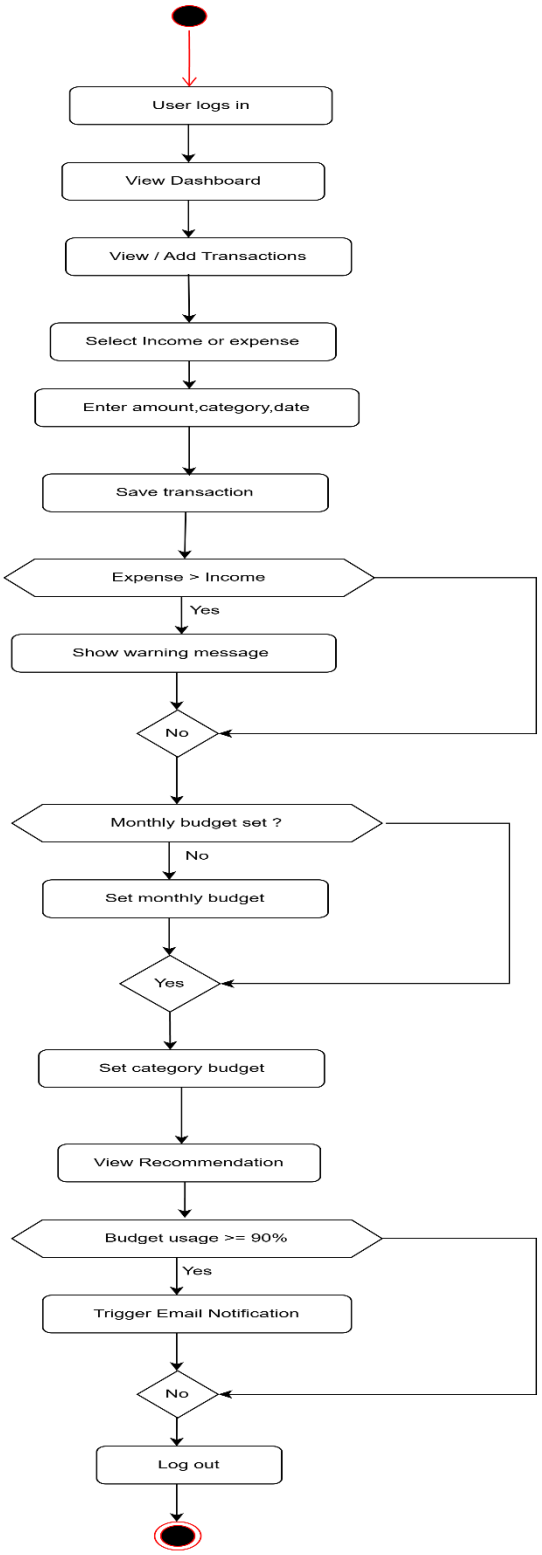

Figure 3.5: Activity Diagram

Chapter 4: System Design

4.1. Design

4.1.1 Refinement of Class, Object and Activity Diagram

As discussed in the analysis chapter, the system design also follows an object-oriented approach. Since, this is a simple project with not much complexity, the class diagrams, object diagrams and activity diagrams designed earlier are already refined enough to be applicable within the project.

4.1.2 Component Diagram

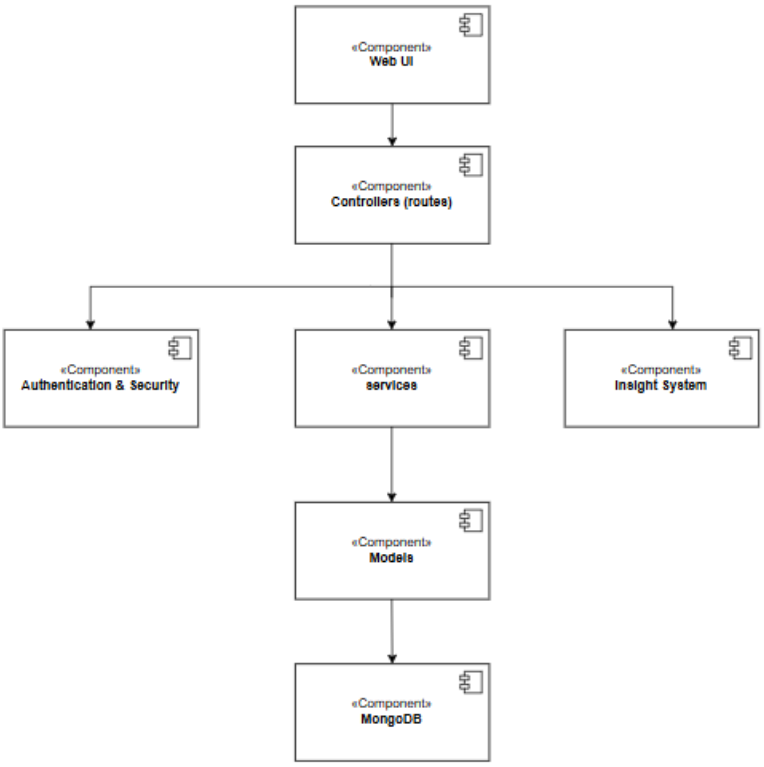

Figure 4.1: Component Diagram

This component diagram illustrates the architecture of the Personal Finance Tracking System (PFTS). Users interact with the Browser frontend, which communicates with an Express Web Server. The server routes requests to the Services layer containing core business logic for authentication, transactions, budget calculations, and notifications. These services interact with the database through a Data-Access layer using Mongoose models, with all data ultimately stored in MongoDB Atlas. The design maintains a clean separation of concerns, with data flowing from the presentation layer (UI) down to persistent storage and back.

4.2 Algorithm Details

Budget Pacing Algorithm:

The budget pacing algorithm in PFTS is designed to give users continuous, precise feedback about their spending patterns, helping them proactively manage budgets and avoid overspending. It not only checks if users are on track but also projects future spending, issues early warnings, and provides actionable insight.

Step-by-Step Process

1. Data Preparation and Time Calculation

- The algorithm starts by determining the user's total budget for a specific category and month.
- It calculates the total number of days in the month and identifies the current day.
- This allows the system to compute how much of the month has passed, as well as the days remaining.

2. Ideal Spending and Pace Calculation

- The algorithm calculates the "ideal" amount the user should have spent by today if spending were perfectly distributed:

  *IdealSpentByNow = (TotalBudget / DaysInMonth) * DaysPassed*

- It also computes the daily spending rate and projects the total spending for the month:

  DailyRate = Spent / DaysPassed
  *ProjectedSpending = DailyRate * DaysInMonth*

3. Pacing and Budget Health Analysis

- The algorithm determines the user's spending pace as a percentage:

  *PacePercentage = (Spent / IdealSpentByNow) * 100*

- It calculates the remaining budget and the daily budget for the rest of the month:

  Remaining = TotalBudget - Spent
  DailyBudget = (Remaining) / (DaysInMonth - DaysPassed)

4. Insight Generation

The algorithm evaluates the user's current status:

- **Overspending:** If projected spending exceeds the total budget, a strong warning is issued.
- **On Track:** If the pace is within a healthy range (e.g., 90–110%), positive feedback is provided.
- **Underspending:** If the user is well below the ideal pace, they are congratulated and encouraged to maintain discipline.
- **Critical Warning:** If the user has used more than 90% of the budget early in the month, a critical alert is shown.

Feedback is delivered as clear messages, and dashboard icons for instant recognition.

5. Real-Time and Personalized Updates

- The algorithm runs automatically whenever a transaction or budget is added, edited, or deleted.
- Insights are personalized for each category and updated in real time, ensuring users always see the latest guidance.

Chapter 5: Implementation and Testing

5.1 Implementation

5.1.1 Tools Used

The implementation of PFTS used a combination of programming languages, frameworks, libraries, and platforms to create a functional and user-friendly application. The main tools and technologies are explained below:

Programming Languages:

- **JavaScript:** Used for both backend and frontend logic, enabling dynamic functionality throughout the application.
- **HTML and CSS:** Used to structure web pages and apply styling for a clean and responsive user interface.

Frameworks and Libraries:

- **Node.js:** A JavaScript runtime environment used to run server-side code and handle backend operations.
- **Express.js:** A web framework for building RESTful routes and managing server requests efficiently.
- **EJS (Embedded JavaScript):** A templating engine that allows dynamic content to be rendered on HTML pages.

Database Platform:

- **MongoDB:** A NoSQL database used to store user information, transactions, budgets, and categories in a flexible document-based format.
- **Mongoose:** An Object Data Modeling (ODM) library for MongoDB that helps define schemas and ensures data validation.

Other Tools:

- **Git:** Version control software used to track changes and manage collaboration.
- **VS Code:** The main code editor for writing and managing the project code.
- **npm (Node Package Manager):** Used to manage and install project dependencies.

- **Draw.io:** Tool used to create all figures, diagrams, and illustrations for the project documentation.

5.1.2 Implementation Details for Modules

The implementation details for some of the core modules of the Personal Finance Tracking System (PFTS) are described as follows:

i. Authentication Module:

The authentication system handles user registration, login, and email verification. Registration requires users to enter their name, email, and password. The frontend validates input before sending it to the backend. On the backend, passwords are securely hashed using bcryptjs, and user data is stored in MongoDB. The system includes email verification functionality. For login, the backend verifies credentials using bcrypt comparison and establishes sessions for authenticated users.

ii. Transaction Management Module:

Users can add, view, edit, and delete income or expense transactions through a comprehensive interface. Each transaction includes amount, type (income/expense), category, description, date, and optional recurring settings. The system supports recurring transactions with daily, weekly, or monthly intervals, automatically processed by a cron job scheduler running at midnight. When transactions are added or modified, the frontend sends data to the backend, which updates the MongoDB database and triggers real-time budget insight updates. All transactions are user-specific and can be filtered by date ranges, categories, and types. The system also includes CSV export functionality for transaction data analysis.

iii. Budget Management Module:

Users can create and manage monthly budgets for specific categories with comprehensive tracking features. The system supports both category-specific budgets and overall monthly budget limits. Budget creation forms validate input and ensure only one budget per category per month per user. The module includes visual progress indicators, spending vs. budget comparisons, and automated budget alerts sent via email when spending reaches 90% of the allocated amount.

iv. Insight Module (Budget Pacing Algorithm):

This advanced module implements a deterministic budget pacing algorithm that continuously analyzes spending patterns against set budgets. The system calculates daily spending rates, projected monthly totals, and pace percentages to generate personalized insight. Real-time updates occur whenever transactions or budgets are modified, providing immediate feedback through color-coded messages and actionable advice. The algorithm considers current day of month, days remaining, and spending rate to offer specific guidance such as daily spending limits.

v. Email Service Module:

Integrated email functionality using Nodemailer sends automated notifications for account verification and budget alerts. The service handles email verification tokens during registration and sends budget warning emails when spending approaches limits. Email templates provide professional formatting with user-specific data and actionable insight.

vi. Dashboard Module:

The comprehensive dashboard provides users with real-time financial summaries including total income, expenses, budget utilization, and pacing feedback. Visual elements utilize Chart.js for interactive charts showing spending breakdowns by category, income vs. expense trends, and budget progress indicators.

5.2 Testing

5.2.1 Test Cases for Unit Testing

Table 5.1: Unit Test Cases for Authentication

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
UT-A1	Register New User	Enter valid name, email, password; submit registration	User registered, verification email sent	As expected	Pass
UT-A2	Register with Existing Email	Enter email already in use, submit registration	Error: "Email already exists"	As expected	Pass
UT-A3	Email Verification	Click verification link from email	Account verified, redirect to login page	As expected	Pass
UT-A4	Login with Verified Account	Enter valid email/password for verified user	User logged in	As expected	Pass
UT-A5	Login with Unverified Account	Enter valid credentials for unverified user	Error: "Please verify your email"	As expected	Pass
UT-A6	Login with Wrong Password	Enter valid email, incorrect password	Error: "Invalid password"	As expected	Pass
UT-A7	Password Hashing	Register user, check database	Password stored as bcrypt hash	As expected	Pass

Table 5.2: Unit Test Cases for Transaction Management

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
UT-T1	Add Income Transaction	Enter amount, select income type, category, description	Transaction saved with correct type	As expected	Pass
UT-T2	Add Expense Transaction	Enter amount, select expense type, category, description	Transaction saved, budget insight updated	As expected	Pass
UT-T3	Edit Transaction	Select transaction, modify details, save	Transaction updated, insights recalculated	As expected	Pass
UT-T4	Delete Transaction	Select transaction, confirm deletion	Transaction removed, budget calculations updated	As expected	Pass
UT-T5	Add Recurring Transaction	Set transaction as recurring with interval	Template created with nextRecurringDate	As expected	Pass
UT-T6	CSV Export	Request transaction export	CSV file generated with all user transactions	As expected	Pass

Table 5.3: Unit Test Cases for Budget Management

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
UT-B1	Set Category Budget	Enter budget amount for specific category/month	Budget saved and visible on page	As expected	Pass
UT-B2	Set Monthly Budget	Enter overall monthly budget limit	Monthly budget saved and tracked	As expected	Pass

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
UT-B3	Update Existing Budget	Edit budget amount, save changes	Budget updated, insights recalculated	As expected	Pass
UT-B4	Budget Deletion	Delete existing budget	Budget removed, related insights cleared	As expected	Pass

Table 5.4: Unit Test Cases for Insight System

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
UT-R1	Budget Pace Calculation	Add transactions, check pace percentage	Accurate pace calculation based on spending	As expected	Pass
UT-R2	Overspending Warning	Exceed budget limit	Warning insight with email alert	As expected	Pass
UT-R3	On-track Insight	Spend within normal pace	Info. insight with appropriate message	As expected	Pass
UT-R4	Under-budget Positive	Spend well below budget pace	Positive insight	As expected	Pass
UT-R5	Real-time Updates	Add/edit transaction	Insights immediately recalculated	As expected	Pass

Table 5.5: Unit Test Cases for Email Service

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
UT-E1	Verification Email	Register new user	Verification email sent with valid token	As expected	Pass
UT-E2	Budget Alert Email	Reach 90% of budget	Alert email sent to user	As expected	Pass
UT-E3	Forget Password Email	Click on Foreget Password in Login Page	Reset password email sent	As expected	Pass

5.2.2 Test Cases for System Testing

Table 5.6: System Test Cases for Authentication Flow

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
ST-A1	Complete Registration Flow	Register new user, verify email, login with credentials	Full authentication cycle works	As expected	Pass
ST-A2	Session Management	Login with valid credentials, navigate between pages, logout	Session maintained across pages, cleared on logout	As expected	Pass

Table 5.7: System Test Cases for Transaction Workflow

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
ST-T1	Complete Transaction Cycle	Add new transaction, view in list, edit details, delete transaction	All operations work, dashboard updates	As expected	Pass
ST-T2	Recurring Transaction Processing	Set transaction as recurring, wait for cron job execution	New transactions automatically created	As expected	Pass
ST-T3	Transaction Filtering	Apply filters by description, category, and transaction type	Correct transactions displayed	As expected	Pass
ST-T4	Data Export	Navigate to transactions page, export data to CSV format	Complete data exported accurately	As expected	Pass

Table 5.8: System Test Cases for Budget Management

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
ST-B1	Budget Creation and Tracking	Set category budget, add related transactions, view progress	Budget progress accurately tracked	As expected	Pass
ST-B2	Budget Alert System	Add expenses to exceed 90% of budget, check email inbox	Alert email received automatically	As expected	Pass
ST-B3	Multi-category Budgets	Set budgets for multiple categories in same month	All budgets tracked independently	As expected	Pass

Table 5.9: System Test Cases for Dashboard Integration

Test ID	Test Scenario	Test Steps	Expected Result	Observed Result	Test Status
ST-D1	Dashboard Data Accuracy	Add various transactions and budgets, view dashboard summary	All data accurately reflected	As expected	Pass
ST-D2	Chart Visualization	Navigate to dashboard, view spending breakdown charts	Interactive charts display correctly	As expected	Pass
ST-D3	Responsive Design	Access application on different screen sizes and devices	Layout adapts appropriately	As expected	Pass

5.3. Result Analysis

Result analysis evaluates the outcomes of the implemented features and the results of testing to verify whether the system meets all expected requirements. Both unit and system testing were conducted, with all test cases passing successfully. Unit testing included 25 test cases, all of which passed, and system testing included 12 test cases, all of which also passed. No critical bugs or failures were observed during the testing phase.

All key functionalities of the PFTS, such as user registration, secure login, adding and managing transactions, setting and tracking budgets, real-time budget pacing insights, and email notifications, were implemented and tested successfully. Minor issues identified during initial testing, such as input validation and clarity of error messages, were resolved before final deployment. The system's fixed category management and CSV export features also performed as expected.

The system successfully meets its stated objectives. All major functionalities described in the project objectives, including easy transaction entry, budget management, real-time feedback, secure authentication, and user-friendly reporting, were fulfilled and verified through comprehensive testing. The application is stable, responsive, and provides users with clear, actionable financial insights, confirming that the project goals have been achieved.

Chapter 6: Conclusion and Future Recommendation

6.1 Conclusion

The Personal Finance Tracking System (PFTS) was developed to address the practical difficulties individuals face in managing their personal finances. By offering a user-friendly web application, PFTS allows users to record income and expenses, define budgets across multiple categories, and receive real-time feedback through a budget pacing algorithm. Key features such as email notifications, data export, and a visually intuitive dashboard were designed to enhance usability and encourage consistent financial tracking.

Throughout the project, emphasis was placed on building a secure and responsive system. The application was thoroughly tested using both unit and system-level test cases. Core functionalities including user authentication, transaction logging, budgeting and personalized insights were validated, and all identified issues were addressed effectively. The system successfully meets its defined objectives by delivering a practical and accessible tool for everyday users. More importantly, it demonstrates that a simple, well-targeted solution can promote financial awareness, support better spending habits, and provide users with a greater sense of control over their money.

6.2 Future Recommendation

While PFTS fulfills its current objectives, there are several areas where the system could be further improved or expanded in future versions:

- Integrate with bank APIs to automatically import transactions, reducing manual entry for users.
- Develop a dedicated mobile application for Android and iOS to enhance accessibility and convenience.
- Develop features for multi-user or family budgeting to allow shared management of finances.
- Support multiple languages and regional formats to make the system accessible to a wider audience.
- Implement enhanced notifications, such as push alerts or SMS reminders, for important updates and warnings.

References

[1] R. H. Thaler, “Mental accounting matters,” *Journal of Behavioral Decision Making*, vol. 12, no. 3, pp. 183–206, 1999.

[2] R. Panko, “What we know about spreadsheet errors,” *Journal of End User Computing*, vol. 10, no. 2, pp. 15–21, 1998.

[3] A. Lusardi and O. S. Mitchell, “The economic importance of financial literacy: Theory and evidence,” *Journal of Economic Literature*, vol. 52, no. 1, pp. 5–44, 2014.

[4] Google Play Store, “Spending Tracker,” [Online]. Available: https://play.google.com/store/apps/details?id=com.troypoint.spendingtracker. Accessed: Jul. 26, 2025.

[5] S. Shrestha and S. Shakya, “Adoption of personal finance apps in Nepal: Opportunities and challenges,” *Nepal Journal of Management*, vol. 8, no. 2, pp. 45–58, 2021.

[6] Statista, “Personal finance app usage worldwide 2021,” [Online]. Available: https://www.statista.com/statistics/1207127/personal-finance-app-usage-worldwide/. Accessed: Jul. 26, 2025.

[7] M. Tamang and R. Joshi, “Designing for adoption: Challenges of financial app usability in South Asia,” *Asian Journal of UX Design*, vol. 2, no. 3, pp. 22–37, 2023.

[8] Intuit Inc., “Mint: Budget tracker & planner,” [Online]. Available: https://mint.intuit.com/. Accessed: Jul. 26, 2025.

[9] G. T. Foster, L. Samuels, and B. Wu, “Digital budgeting interfaces and saving behavior: An experimental study,” *Journal of Consumer Research*, vol. 47, no. 4, pp. 765–781, 2021.

[10] S. H. Schwartz, M. Gupta, and D. Klein, “Trust and user retention in fintech apps: The role of perceived surveillance,” *Journal of Financial Technology*, vol. 3, no. 1, pp. 12–26, 2023.

[11] S. Shrestha, “Financial technology adoption challenges in Nepal,” *International Journal of Finance and Banking Studies*, vol. 9, no. 1, pp. 1–15, 2020.

[12] Google Play Store, “Spendee: Budget and money manager,” [Online]. Available: https://play.google.com/store/apps/details?id=com.cleevio.spendee. Accessed: Jul. 26, 2025.

Appendices

Screenshots of PFTS:

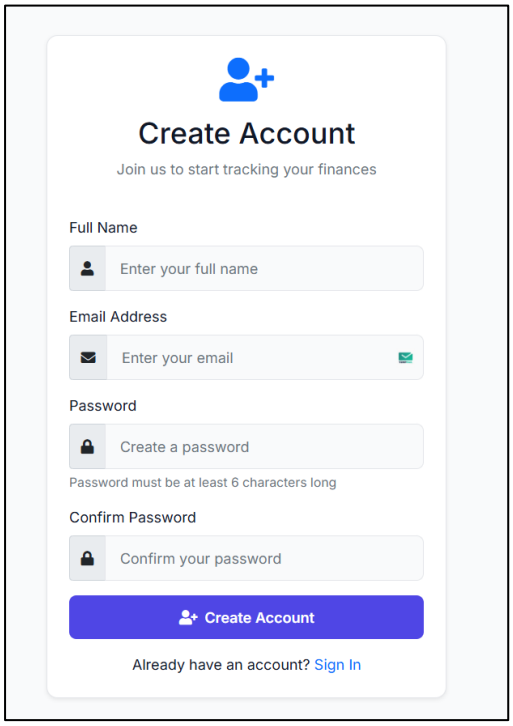

Figure 1: Register Page

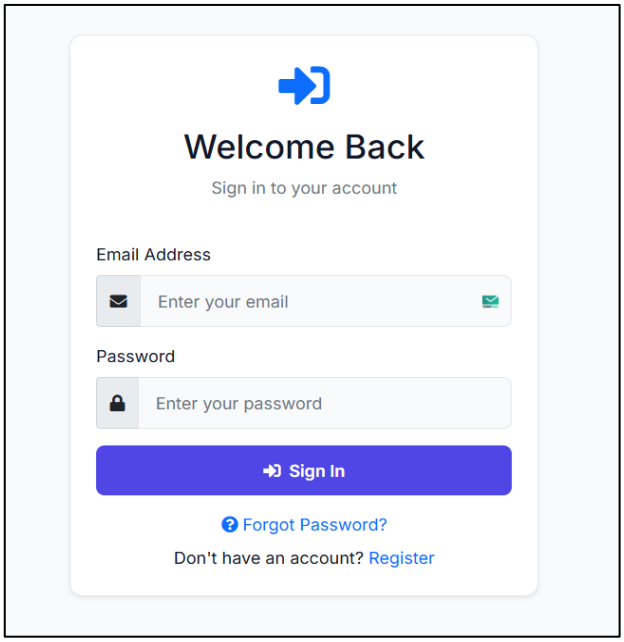

Figure 2: Login Page

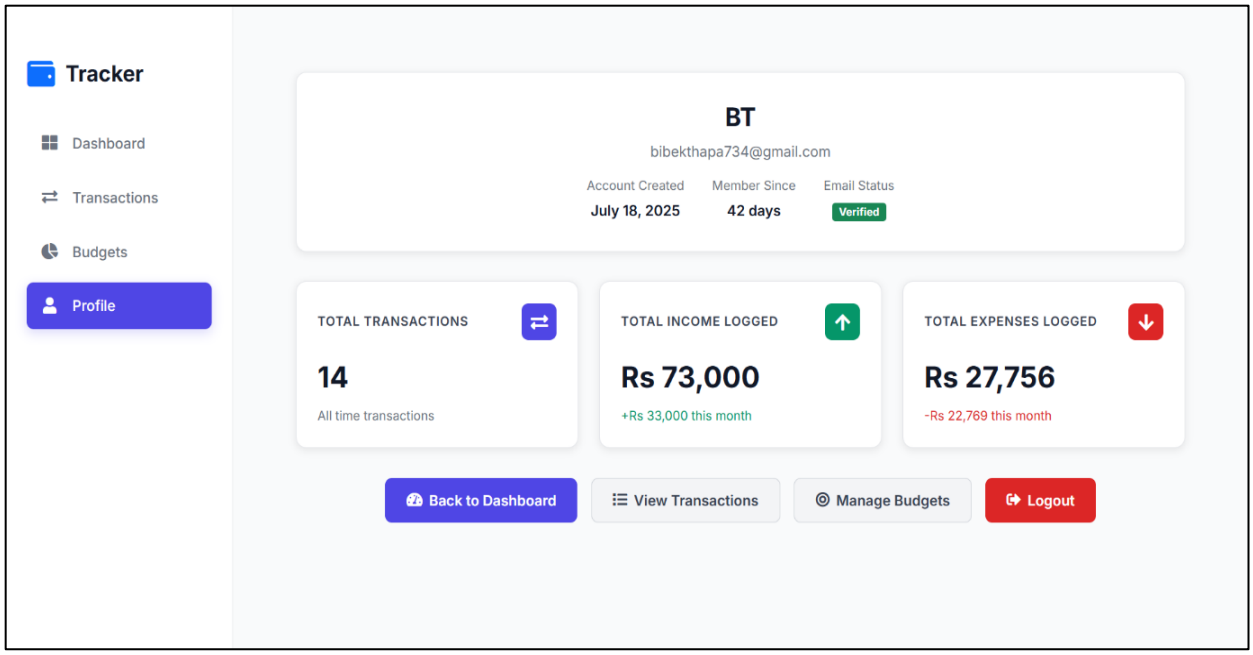

Figure 3: Profile Page

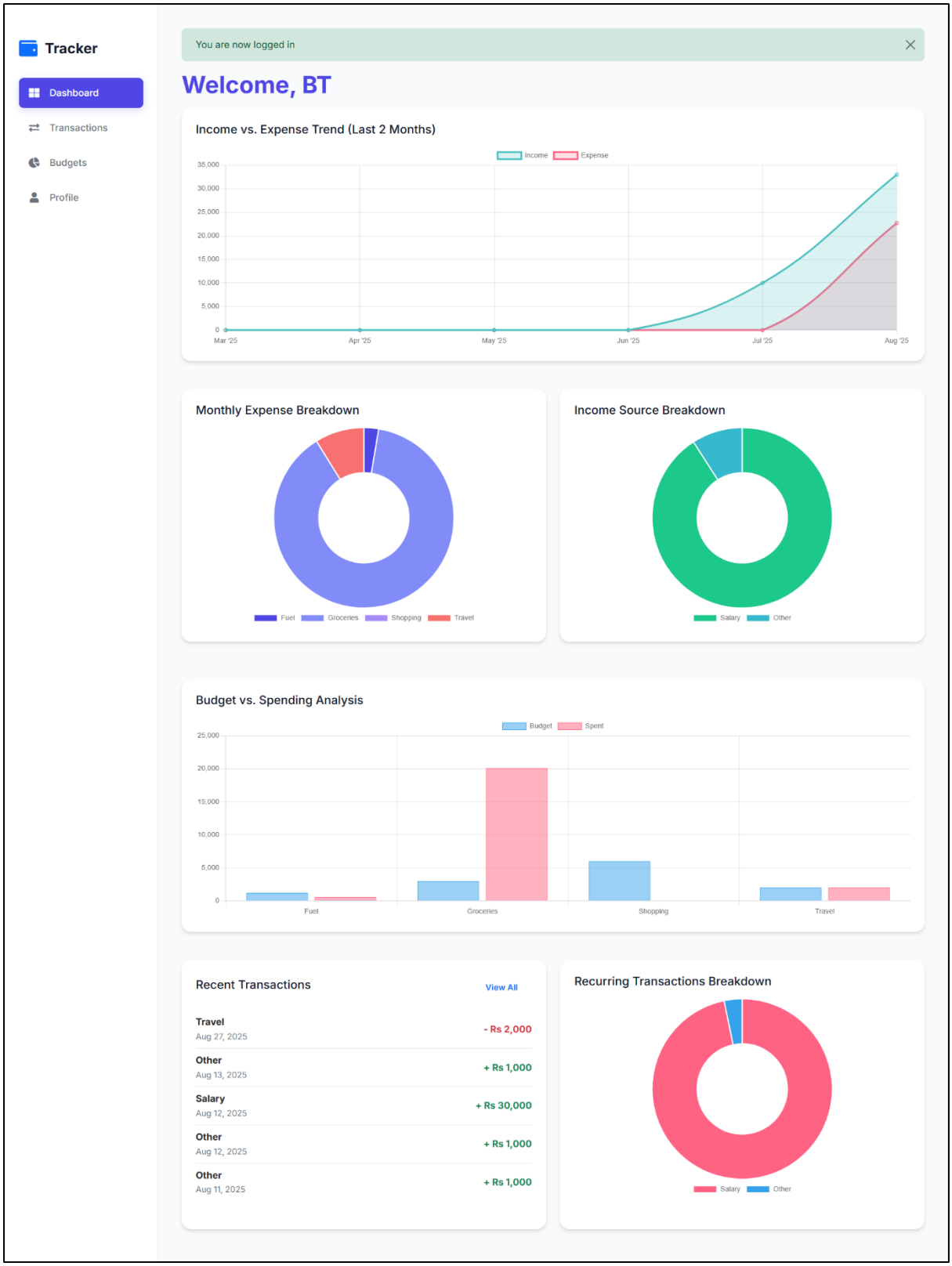

Figure 4: Dashboard Page

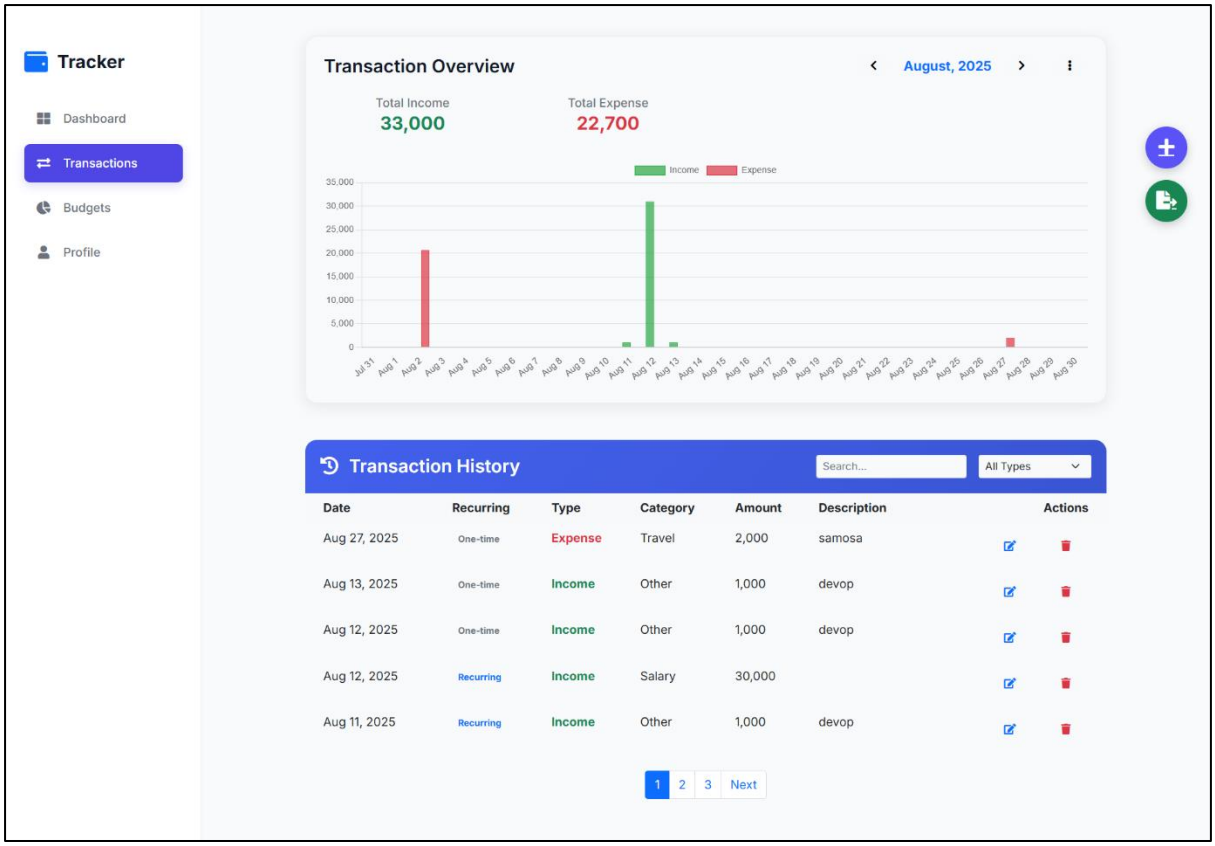

Figure 5: Transaction Page

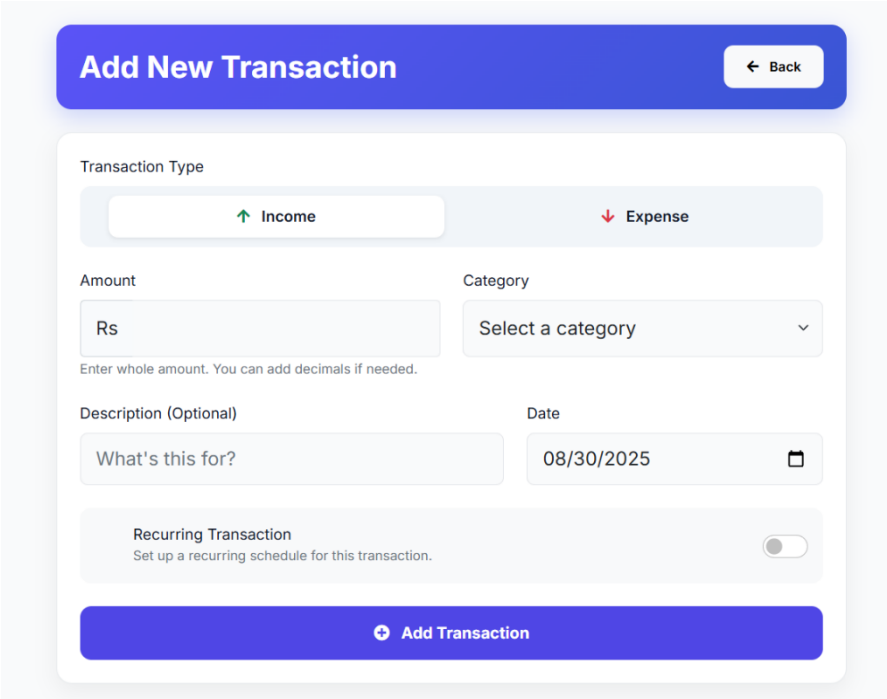

Figure 6: Add Transaction (Income /Expense)

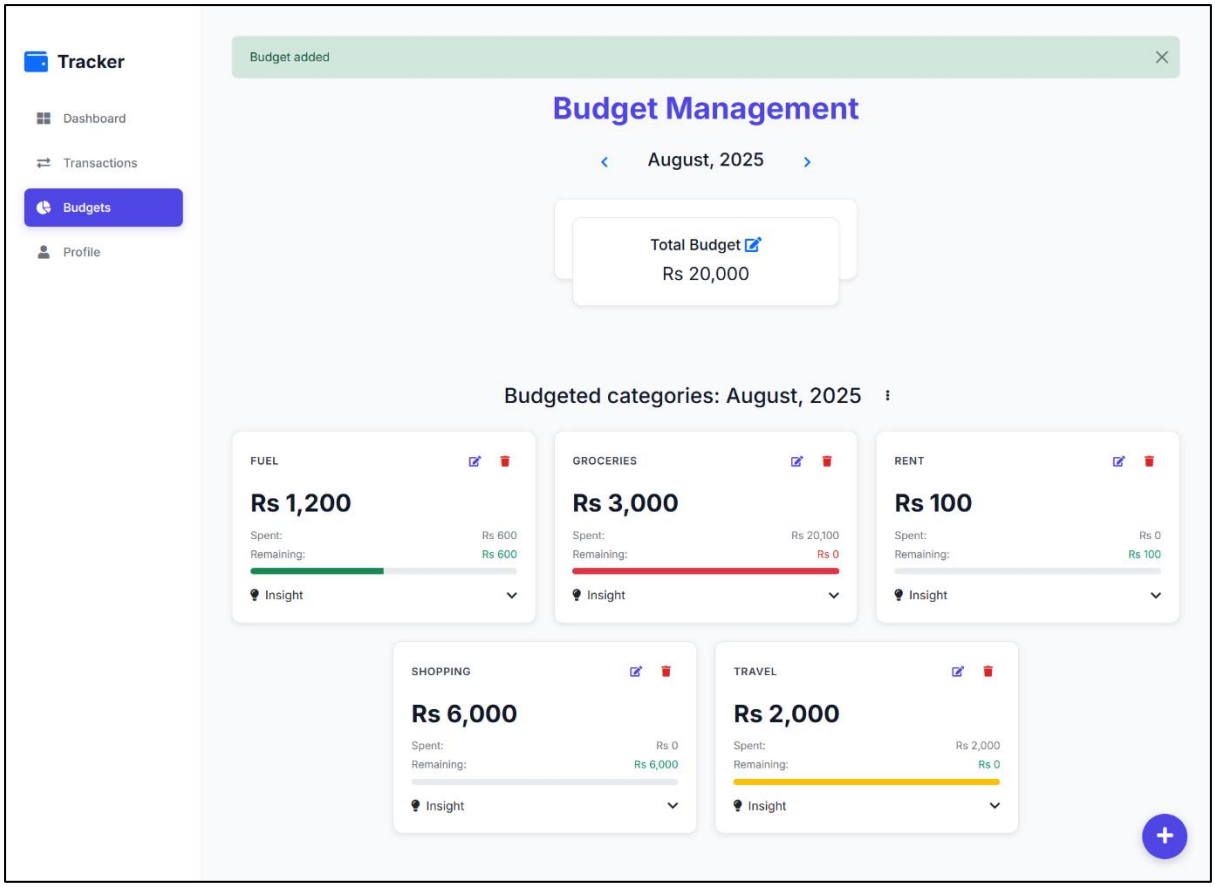

Figure 7: Budget Page

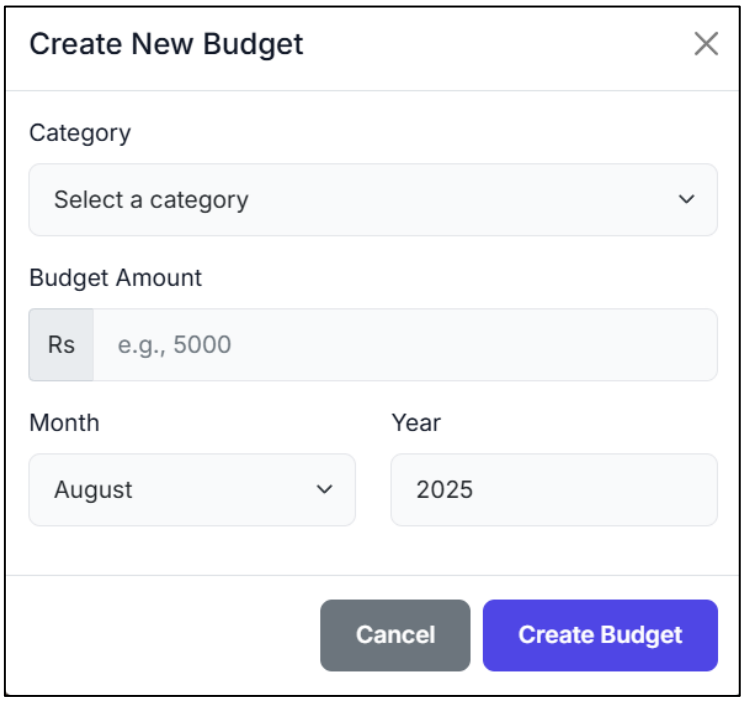

Figure 8: Create Budget

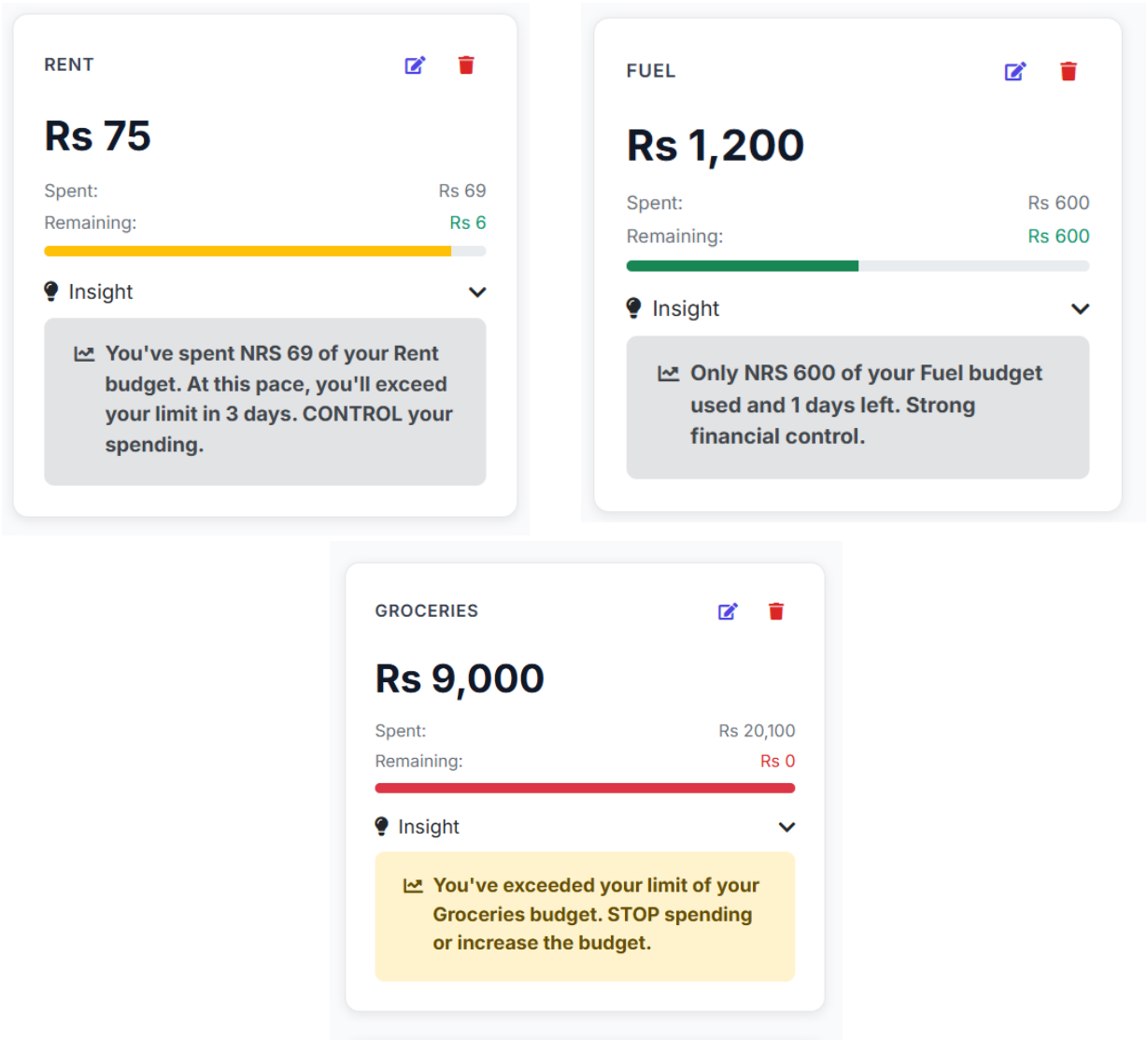

Figure 10: Budget Insights

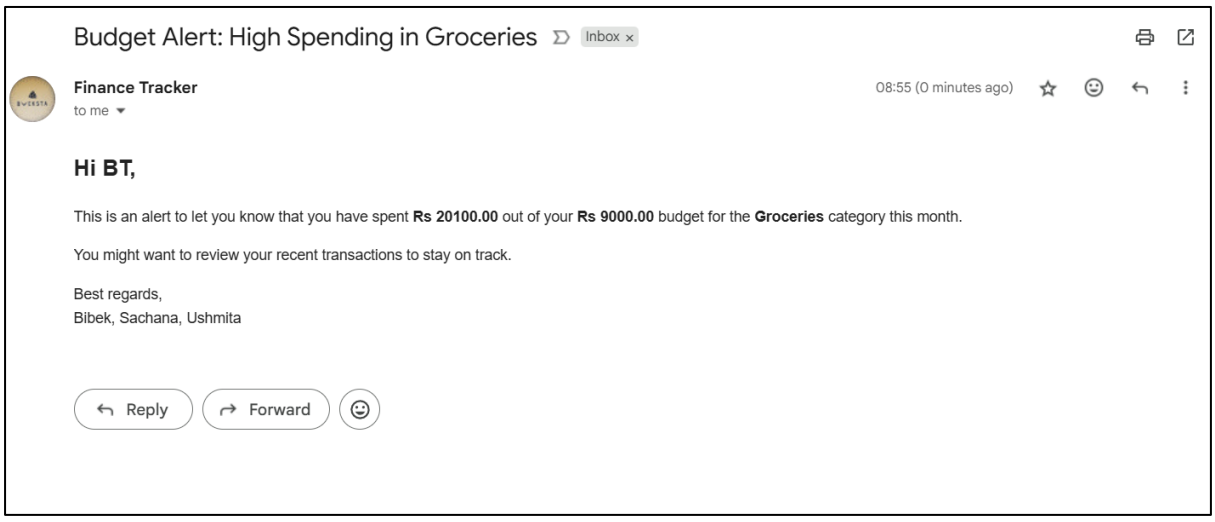

Figure 11: Budget Email Alert

Code Implementation of Budget Pacing Algorithm:

```
// Calculate budget pacing metrics
static calculateBudgetPace(budget, spent, currentDay, daysInMonth) {
  const budgetAmount = parseFloat(budget.amount) || 0;
  const daysPassed = Math.max(1, currentDay);
  const idealSpentByNow = (budgetAmount / daysInMonth) * daysPassed;
  const pacePercentage =
    idealSpentByNow > 0 ? (spent / idealSpentByNow) * 100 : spent > 0 ? Infinity : 0;
  const dailyRate = spent / daysPassed;
  const projectedSpending = dailyRate * daysInMonth;
  const remaining = budgetAmount - spent;
  const remainingDays = daysInMonth - daysPassed;
  const dailyBudget = remainingDays > 0 ? remaining / remainingDays : 0;

// Generate insight based on budget pace
static getPaceRecommendation(category, paceData) {
  const { spent, pacePercentage, projectedSpending, remaining, dailyRate, budget } =
paceData;
  let message, type;
  const budgetAmount = parseFloat(budget.amount) || 0;

  const formatCurrency = (amount) => {
    const num = parseFloat(amount);
    if (isNaN(num)) {
```

```
    return new Intl.NumberFormat("en-NP", { style: "currency", currency: "NRs", maximumFractionDigits: 0 }).format(0);
  }
  return new Intl.NumberFormat("en-NP", { style: "currency", currency: "NRs", maximumFractionDigits: 0 }).format(num);
};

const today = new Date();
const daysInMonth = new Date(today.getFullYear(), today.getMonth() + 1, 0).getDate();
const daysLeft = daysInMonth - today.getDate();

if (pacePercentage > 100) {
  type = "warning";
  if (remaining === 0) {
    message = `You have spent the entire budget for the ${category} category. Be careful with any further spending.`;
  } else if (remaining < 0) {
    message = `You've exceeded your limit of your ${category} budget. STOP spending or increase the budget.`;
  } else {
    const overBudgetAmount = projectedSpending - budgetAmount;
    message = `You've spent ${formatCurrency(spent)}. At this pace, you are projected to go over budget by ${formatCurrency(overBudgetAmount)}.`;
  }
} else if (pacePercentage >= 90) {
  type = "caution";
```

```
    const daysUntilExceeded = dailyRate > 0 ? Math.ceil(remaining / dailyRate) : Infinity;
    message = `You've spent ${formatCurrency(spent)} of your ${category} budget. At this pace, you'll exceed your limit in ${daysUntilExceeded} days. CONTROL your spending.`;
  } else if (pacePercentage <= 60) {
    type = "positive";
    message = `Only ${formatCurrency(spent)} of your ${category} budget used and ${daysLeft} days left. Strong financial control.`;
  } else {
    type = "info";
    message = `You've used ${formatCurrency(spent)}, with ${daysLeft} days remaining. You're on track.`;
  }
  return { message, type };
}

// Generate personalized insight for a user
static async generateRecommendations(userId, budgets, transactions) {
  const model = await this.getOrCreateModel(userId);
  model.recommendationHistory = model.recommendationHistory.filter((rec) => {
    const recDate = new Date(rec.date);
    const now = new Date();
    return now - recDate > 30 * 24 * 60 * 60 * 1000;
  });
  const recommendations = [];
```

```
  const today = new Date();
  const currentMonth = today.getMonth() + 1;
  const currentYear = today.getFullYear();
  const currentDay = today.getDate();
  const daysInMonth = new Date(currentYear, currentMonth, 0).getDate();


    if (budget.month === currentMonth && budget.year === currentYear) {
      if (spent === 0) {
        recommendations.push({
          category: budget.category,
          message: `No spending recorded for ${budget.category} yet. Your full budget of ${formatCurrency(budget.amount)} is available.`,
          type: "info",
          icon: "fas fa-info-circle",
        });
        continue;
      }


const overallPace = totalBudget > 0 ? (totalSpent / ((totalBudget / daysInMonth) * currentDay)) * 100 : 0;
    let overallMessage, overallType;
    if (overallPace > 110) {
      overallMessage = "Looking at the big picture, you might want to slow down your spending a bit this month.";
      overallType = "warning";
```

```
    } else if (overallPace < 90) {
      overallMessage = "Overall, you're doing great with your finances this month! Nice work saving money.";
      overallType = "positive";
    } else {
      overallMessage = "Your overall spending is perfectly on track this month. You're balancing things well!";
      overallType = "positive";
    }
    recommendations.push({ category: "overall", message: overallMessage, type: overallType });
  }
  if (recommendations.length < 3) {
    recommendations.push({ message: "Having clear savings goals can help keep you motivated with your budget.", type: "general", category: "general" });
    recommendations.push({ message: "Many people find the 50/30/20 approach helpful: 50% for needs, 30% for wants, and 20% for savings.", type: "general", category: "general" });
  }
  await model.save();
  return recommendations;
}
```