Tribhuvan University

Institute of Science and Technology

Academia International College

Project Report

On

"Personal Finance Management System (PFMS)"

CSC 412

Under the supervision of

Mr. Rabin Maharjan

Submitted By

Jivan Parajuli (26490/077)

Sandesh Humagain (26513/077)

Submitted To

Department of Computer Science and Information Technology

Academia International College

January, 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Project Report

On

"Personal Finance Management System (PFMS)"

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

Jivan Parajuli (26490/077)

Sandesh Humagain (26513/077)

Submitted To

Department of Computer Science and Information Technology

Academia International College

January. 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision by Mr. Jivan Parajuli (26490/077), Mr. Sandesh Humagain (26513/077) entitled "Personal Finance Management System (PFMS)" be accepted as fulfilling in partial requirements for the degree of Bachelors of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

……..

Mr. Rabin Maharjan

Project Supervisor

Department of Computer Science and Information Technology

Academia International College

Tribhuvan University

Institute of Science and Technology

Academia International College

Certificate of Approval

This is to certify that this project prepared by Mr. Jivan Parajuli (26490/077), Mr. Sandesh Humagain (26513/077) entitled "Personal Finance Management System (PFMS)" 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. Rabin Maharjan Project Supervisor Department of Computer Science and IT Academia International College	…….. 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 owe our most profound appreciation to Academia International College for giving us a chance to work on this project as part of our syllabus.

Special thanks to our supervisor, Mr. Rabin Maharjan for his consistent guidance, support, and feedback throughout the report's creation. We are generously obligated to him for providing this excellent opportunity to expand our knowledge.

We would like to express our sincere gratitude to all those individuals for supporting and helping us a lot for the successful completion of this project.

Thanking You,

Jivan Parajuli (26490/077)

Sandesh Humagain (26513/077)

Abstract

PFMS is a web-based application which helps users to manage and track their financial transactions by keeping record of their income, expenses, and savings. The major objective of the application is to provide users a clear picture of their financial habits, so that they can make informed decisions and achieve their financial goals.

With the inclusion of fundamental features of finance management systems like adding finances and tracking them, PFMS provides an advanced approach to personal finance management by providing users the feature to categorize expense automatically and view predictions of their spending. The project report provides in-depth information about the development of the application while the core technologies in development being: PostgreSQL, Python Flask and ReactJS.

Keywords: Expense categorization, finance tracker, finance management, personal finance management system

Table of Contents

List of Figures

List of Tables

List of Abbreviations

PFMS	Personal Finance Management System
YNAB	You Need A Budget
UI	User Interface
REST	Representational State Transfer
API	Application Programming Interface
MVP	Minimum Viable Product
ER	Entity Relationship
DFD	Data Flow Diagram
BERT	Bidirectional Encoder Representations from Transformers
NLP	Natural Language Processing

Chapter 1: Introduction

1.1. Introduction

In today's time, people are concerned about the management of their income and expenses. Regardless of one being freshly employed with minimal income or an experienced professional with handsome paycheck, the need of managing income and expenses can be visibly seen. (Personal Finance Management System) PFMS is a web-based application which helps users to manage and track their financial transactions by keeping record of their income, expenses, and savings. The major objective of the application is to provide users a clear picture of their financial habits, so that they can make informed decisions and achieve their financial goals.

Traditional digital finance tracking and budgeting systems have centers on basic tasks like recording transactions. On the other hand, PFMS takes a more advanced approach providing additional features like automatic expense categorization and future expense prediction. Besides, PFMS allows users to organize their expenses and makes it easier for users to see where their money is going. It provides insights that can help users plan their budget and improve financial habits. Additionally, users can set financial goals (saving for vacation, paying off debt) and track the progress towards these goals.

1.2. Problem Statement

In the world where expenses, income streams, and financial goals are complex, managing personal finances can be challenging for individuals. When one struggles to manage personal finances, it leads to overspending, bad budgeting and has no success to achieve their financial goals.

Existing solutions for finance management can be observed as far too complex and fail to provide help to the individuals who seek personalized insights to their financial habits.

PFMS aims to address these gaps and enable users to:

- Record and track expense and income streams with option to receive assistance of automatic expense categorization
- Set and monitor budget and financial goals.
- Visualize their spending habits to make informed financial decisions.

1.3. Objectives

PFMS majorly aims to provide a platform with a user-friendly experience to manage a user's personal finance. It aims to provide an intuitive, user-friendly tool for tracking and categorizing income and expenses, offering insights that enable better financial decision-making. A much more specific set of objectives are:

- To design and develop a web-based application that allows users to track their income and expenses.
- To implement an automated system for categorizing expenses based on the user's description.
- To provide future spending predictions and insights based on spending patterns.
- To develop interactive financial reports and visualizations that help users understand their cash flow.
- To empower users to set financial goals and monitor progress toward achieving them.
- To offer a secure and reliable platform for managing personal finances

1.4. Scope and Limitation

1.4.1. Scope

The major focus areas and coverage of PFMS is highlighted as in the scope below:

- A secured user registration and login functionalities for users to use and access the application.
- Ability for users to add and track their income/expense data.
- Automatic categorization of expenses into predefined categories with an option for users to manually adjust or re-categorize transactions.
- A goal-setting feature that allows users to define financial goals.
- Expenditure prediction based on past data.
- Visual representation of transactions, spending, goals and other.
- Support for setting monthly or budgets for different expense categories.

1.4.2. Limitation

Limitations of PFMS are:

- The application does not provide the service of multiple currencies or currency conversion.
- The application lacks integration with external financial services like online payment platforms, digital wallets and rely on manual entry of financial data.
- The application is only accessible via web browsers.
- The application requires active internet connection for all functionalities.
- The application does not include features like multi-language support or other accessibility options.

1.5. Development Methodology

For the development of PFMS, agile methodology of software development was applied. The core development process involved building system iteratively in multiple phases. Technically, these phases are referred to as sprints. Each sprint focused on specific set of features.

Figure 1: Agile Development Methodology

For each sprint, the process of agile were performed. Each sprint started with planning for the sprint. Then the required mockup design was prepared for the features in the sprint which was followed by its development. The outcome was then thoroughly tested and any bugs or errors were noted.

1.6. Report Organization

The project report was prepared by complete fulfilment of the guidelines included in the syllabus for Project Work. The report begins with a standard cover and title page. Then certificate consisting supervisor recommendation and letter of approval are included. It is then followed by the acknowledgement page, abstract page, contents table and lists for abbreviations, figures and tables. The main report consists of:

i. Chapter 1: Introduction
   This chapter covers the topics: Introduction, Problem Statement, Objectives, Scope and Limitation, Development Methodology, Report Organization
ii. Chapter 2: Background Study and Literature Review
   This chapter includes background study and literature review done for the project.
iii. Chapter 3: System Analysis
   System analysis covers: Requirement analysis, feasibility analysis, and system analysis
iv. Chapter 4: System Design
   This chapter includes: Design and algorithm details
v. Chapter 5: Implementation and Testing
   This chapter covers the topics: Implementation, testing and result analysis
vi. Chapter 6: Conclusion and Future Recommendations
   Finally, chapter 6 includes conclusion and future recommendations

The main report is followed by the references section, bibliography and appendices section.

Chapter 2: Background Study and Literature Review

2.1. Background Study

A personal finance management system is used to keep track of income, expenses and savings of an individual. Studies highlight that user often lack awareness of their spending patterns, making it difficult for them to set and achieve financial goals, leading to poor budgeting and impulsive expenditures.[1] It is a complex and challenging task to keep track of personal finances using traditional, manual record-keeping methods such as notebooks or spreadsheets. Keeping track of personal finance manually is time-consuming and error prone. Keeping track of minute day-to-day transactions on pen and paper can be very tedious and result in miscalculations. It also fails to provide analytical insights required make informed financial decisions. Assigning and managing various categories of income and expense transactions can be very useful for analysis, but it is very difficult to accomplish using traditional finance tracking methods. Without proper tools, it is difficult to visualize the areas of expenses, detect spending patterns or areas where costs can be reduced. Additionally, the prediction and forecast of expenses for future based on past spending pattern is not possible in traditional methods. This can lead to financial-stress, mismanagement of money and over-spending. Many people struggle to maintain discipline in saving for goals, as they lack a structured system to track contributions or monitor progress.

The rise in digital solutions in various sectors has highlighted the need for a digital transformation in personal finance management. A comprehensive web application that leverages features such as income/expense tracking, managing multiple accounts, expense categorization, expense prediction and goal-based savings can help simplify personal finance management.

2.2. Literature Review

As part of literature review, academic papers and relevant existing system were thoroughly studied. It was observed that users with complex financial portfolios face challenges in tracking diverse categories, such as groceries and entertainment, without real-time analytics and personalized recommendations. [1]

Mint (moved to Credit Karma) is a very popular free personal finance app that automatically links with user's bank accounts to track cash flow. It has features to show user's expenses and account balances in one place, track net worth, monitor categorized monthly cash flow and provides budgeting tools, along with visualizations of spending patterns. [2]

Wallet by BudgetBakers is another popular finance tracking app that offers expense tracking, budgeting, and financial planning features. It allows users to manually input the transactions or sync their bank accounts for automatic recordkeeping. It provides features such as expense categorization and budgeting. It requires a premium subscription for advanced features. [3]

YNAB (You Need A Budget) is a more rigid budgeting and finance management application which focuses on zero-based budgeting approach. Every count of money is assigned a specific purpose. While YNAB is highly effective for disciplined budgeting, it may not be suitable for all users because of its rigid budgeting feature. It is a paid software with no free version. Because of its paid model, it is ad-free, very secure and has good support. It links to user's bank accounts to automate recordkeeping. It has web application, mobile application and browser application as well. So, it can be used from various devices and the user data is synced across devices. The accounts of family members can be linked together to track family finances and allow for family budgeting. It also has other features such as goal tracking, loan calculator, spending and net worth reports etc. [4]

Chapter 3: System Analysis

3.1. System Analysis

System analysis of Personal Finance Management System was performed before starting the project, which resulted in a comprehensive task list, requirements list, project schedule and project documentation.

3.1.1. Requirement Analysis

The conclusions from the requirements analysis are outlined below in the form of functional and non-functional requirements.

i. Functional Requirements

The functional requirements of the PFMS are as follows:

- The user should be able to register an account in application.
- The user should be able to log in to the application using the credentials registered during account creation.
- The user should be able to input and track their expenses and incomes.
- The system should automatically categorize expenses into predefined categories based on expense description.
- The user should be able to set monthly spending budgets for various categories.
- The system should give predictions of future spending on the basis of past data.
- The system should visualize spending trends with charts or graphs.
- The user should be able to set financial goals, contribute to these goals and track progress of goals.

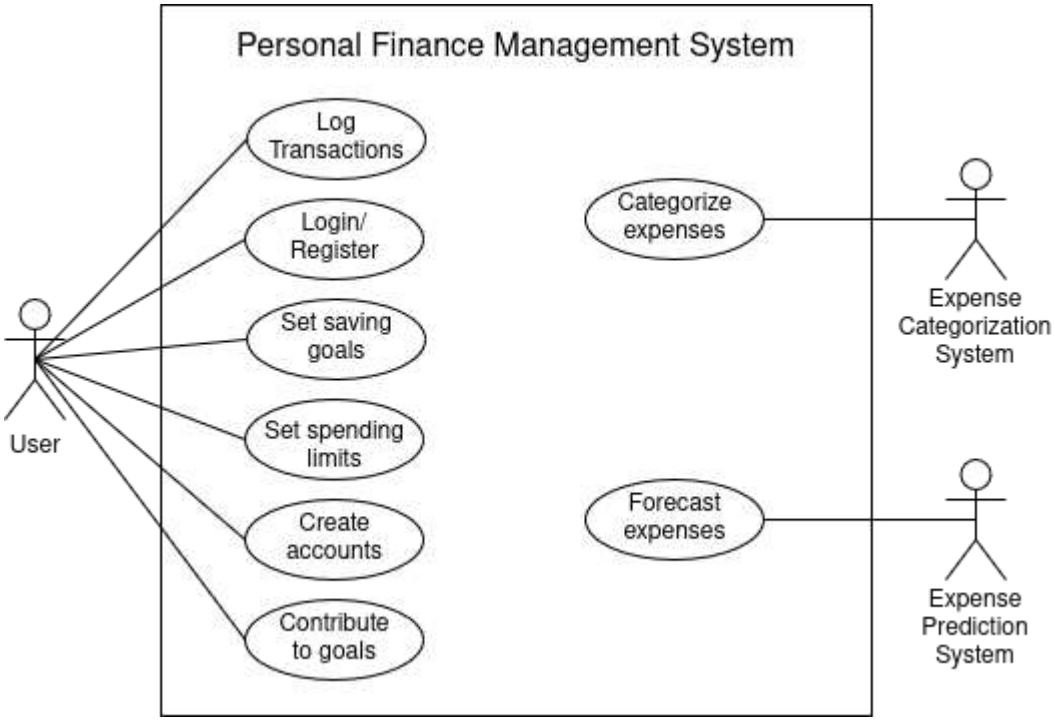


Figure 2: Use Case Diagram

ii. Non-Functional Requirements

The non-functional requirements of the PFMS are as follows:

- The system should have a simple, intuitive, and user-friendly interface.
- The UI should be responsive, ensuring smooth performance across devices (mobile, tablet, desktop).
- The system should handle the user's data efficiently, with minimal loading times for generating reports or charts.
- The app should be able to handle a large amount of transaction data without significant performance degradation.
- The system must ensure that all data entries and calculations are accurate and consistent across different parts of the application.

3.1.2. Feasibility Analysis

The feasibility of the Personal Finance Management System (PFMS) was evaluated through technical, operational, economic and schedule-based perspectives.

i. Technical Feasibility

The project is technically feasible because it requires minimal hardware and software cost. The tools and technology that will be used in developing PFMS PFMS are ReactJS and

Chart.js for developing user interface, Python/Flask for developing the backend and business logic, PostgreSQL for database server and scikit-learn, transformers libraries for machine learning tasks. The frontend and backend software along with database server can be hosted on localhost for development and demo purposes. All these tools and technologies are completely free or open source.

ReactJS and Graph.js are capable and dependable frameworks and libraries which have been used to develop various data related UIs. The backend will be developed using Python Flask, a lightweight and flexible web framework. Flask is ideal for creating scalable and secure REST APIs that can handle user data and process financial transactions efficiently. Developing using these technologies is not technically intensive. Thus, the above-mentioned facts prove that this software is technically feasible.

ii. Operational Feasibility

The project is also operationally feasible as it is achievable with team's human resources within time constraints. The team consists of two members with adequate knowledge of programming, web development, machine learning and UI/UX design. The technology stack used in the project allows fast and simple development process, suitable for prototyping and MVP development. Thus, developing this project should be viable with the team's skills and resources.

iii. Economic Feasibility

The proposed system doesn't require any kind of monetary investment as the technology and tools used are mostly available for free or are open source. The development of project is also economically viable because open-source and free build tools such as Visual Studio Code, Git, GitHub are used. The software components are hosted on already available hardware. Thus, the project is deemed economically feasible.

iv. Schedule Feasibility

A fair amount of timeframe was scheduled for successful completion of the project. The dive into a rather newly explored technology of machine learning for expense categorization and prediction was a challenge to be implemented in schedule. The established schedule for the project is outlined in the table below with a Gantt chart to better visualize the timeline.

Table 1: Project Timeline

Task	Start Date	End Date	Duration (Days)
Planning	5-Aug	25-Aug	20
System Analysis and Design	26-Aug	15-Sep	20
Implementation (Coding)	16-Sep	16-Oct	30
Testing and Quality Assurance	17-Oct	6-Nov	20
Documentation and Reporting	7-Nov	17-Nov	10

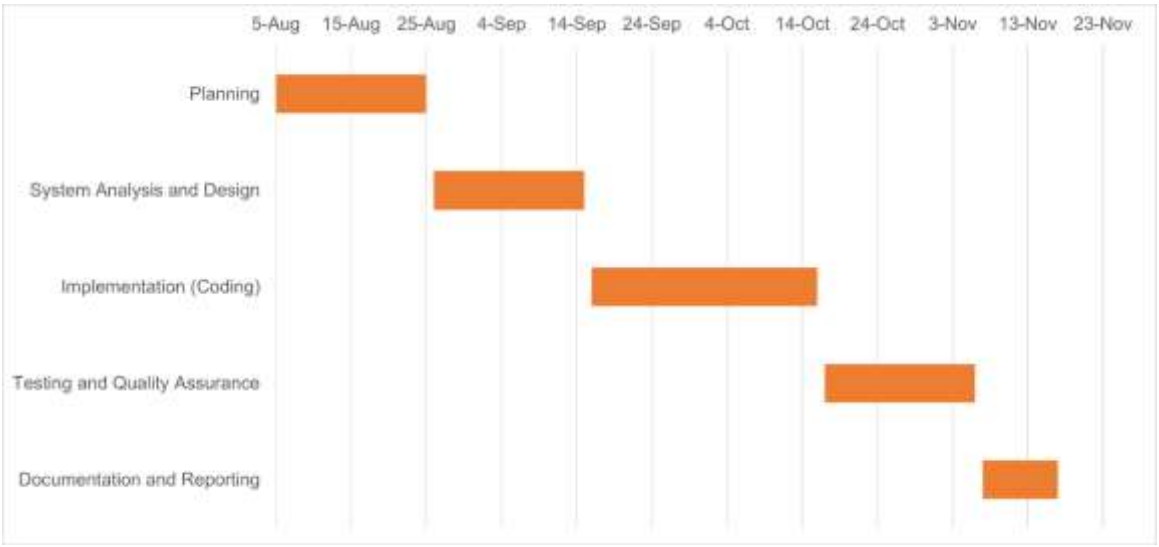


Figure 3: Gantt Chart

3.1.3. Analysis

i. Data modelling using ER Diagrams

The Entity-Relationship (ER) diagram for the Personal Finance Management System (PFMS) visually represents the relationships between various entities of system and their relationships.

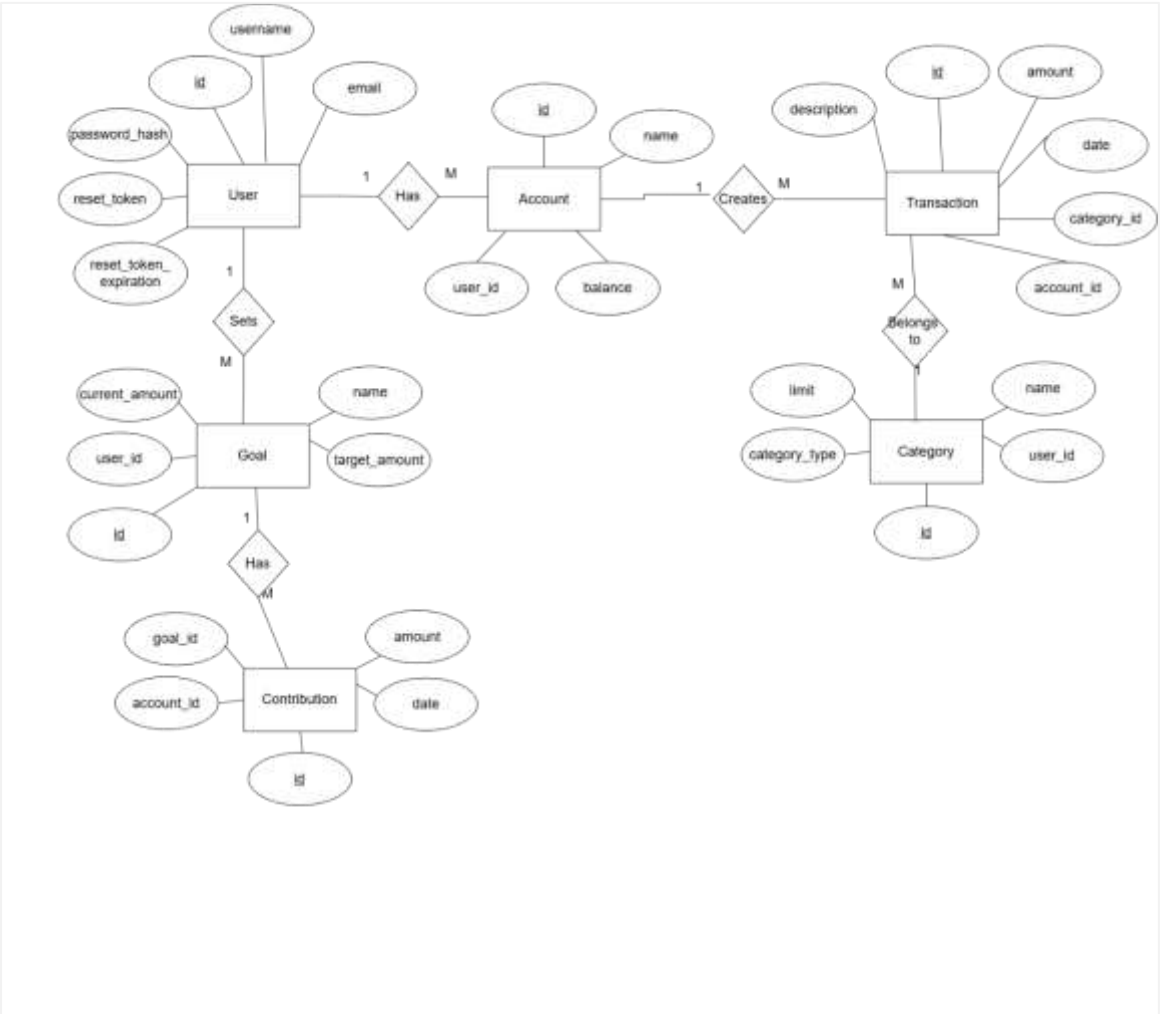


Figure 4: ER Diagram

ii. Process modelling using DFD

The Data Flow Diagram (DFD) for the Personal Finance Management System (PFMS) visually represents the flow of data in the system or processes of the system.

Figure 5: Level-0 DFD (Context Diagram)

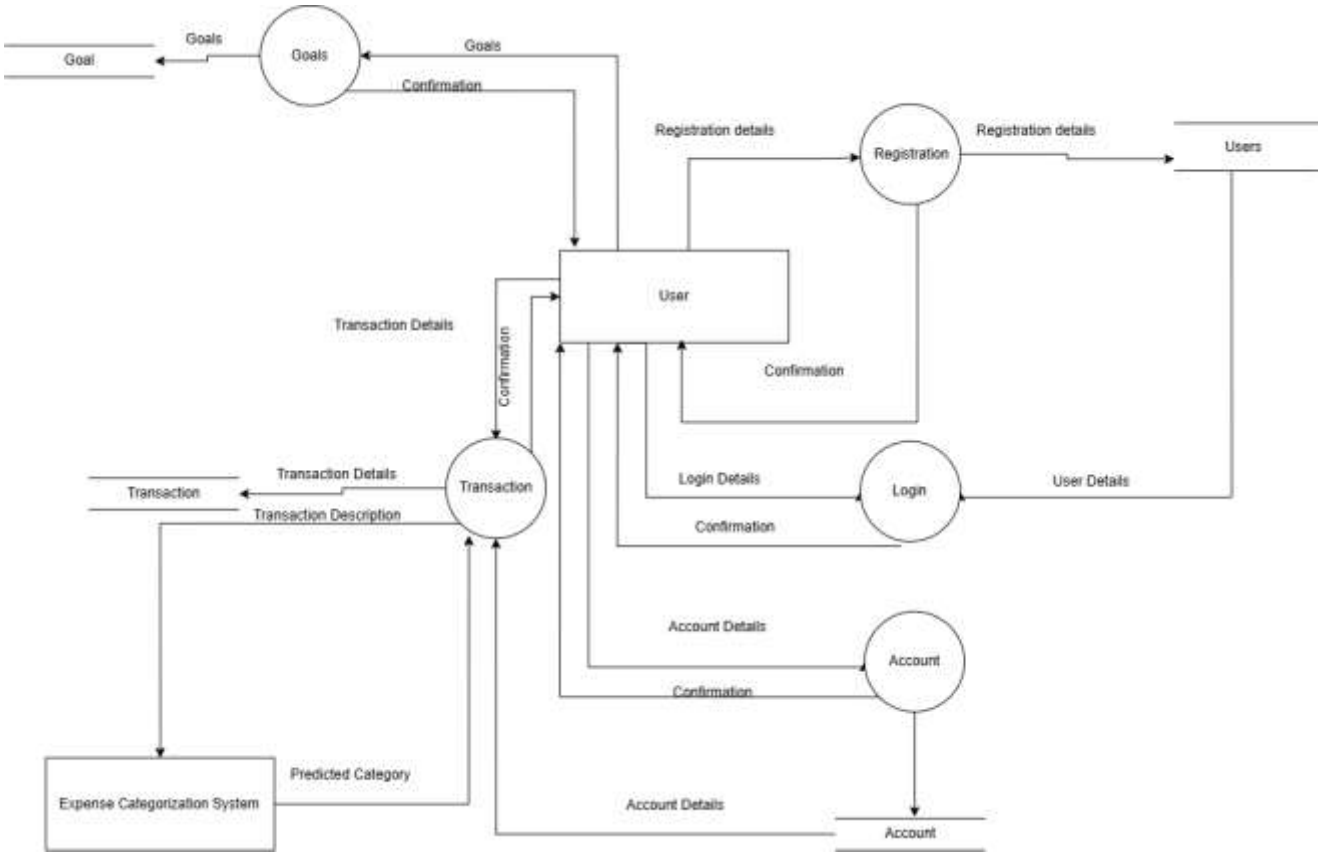


Figure 6: Level-1 DFD

Chapter 4: System Design

4.1. Design

The design process of PFMS involved detailing various components of the system. The major focus was to create efficient and reliable designs that would support on developing a robust and user-friendly system. System's design is divided into following components.

4.1.1. Architectural Design

The Personal Finance Management System (PFMS) is developed on three-tier architecture. The client, application server and database are separated on three different tiers. The client tier consists of client devices and frontend application written in ReactJS, through which users can interact with the application server to perform finance management. The application server is written in Python with Flask web framework. It handles business logic of transactions and communication with the database. The Data layer/tier is based on PostgreSQL database server.

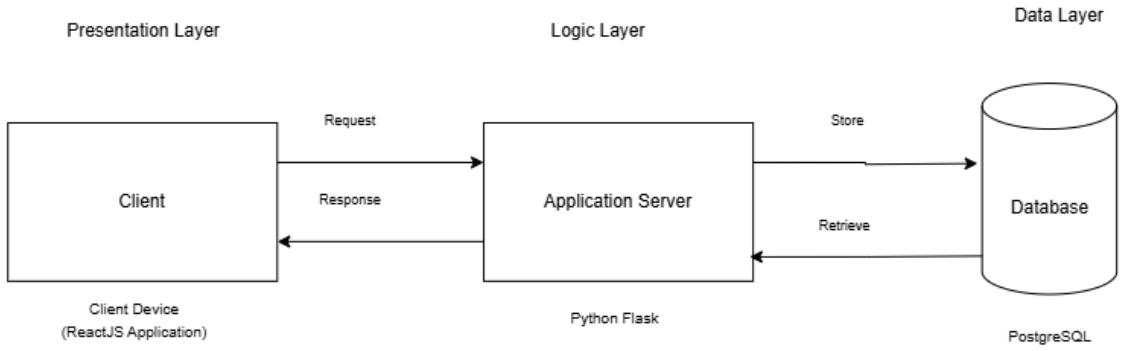


Figure 7: Three-tiered Architecture of PFMS

4.1.2. Database Design

The database was designed keeping in mind of the various functionalities of the system like transaction management, goals management, accounts management and other components of the system.

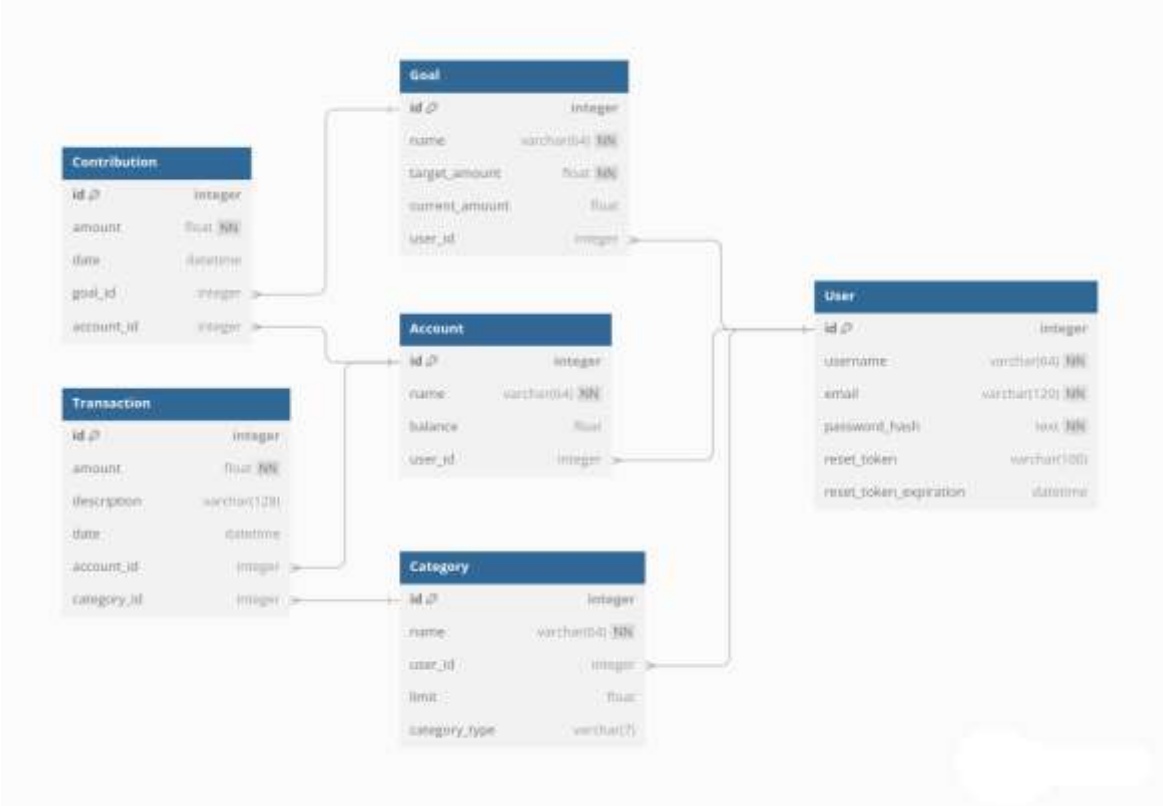


Figure 8: Database Schema Design

4.1.3. Forms and Interface Design

Figure 9: Login Page UI

Figure 10: Signup Page UI

Figure 11: Dashboard page UI

Figure 12: Transaction page UI

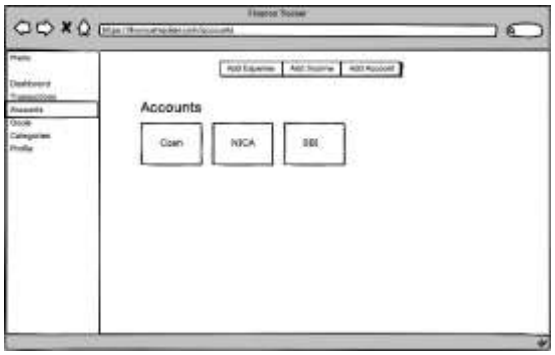


Figure 13: Accounts page UI

Figure 14: Goals page UI

Figure 15: Categories page UI

Figure 16: Profile page UI

4.2. Algorithm Details

4.2.1 Expense Categorization using BERT and Random Forest Classifier

In this project, expense categorization is performed to categorize expenses into one of predefined categories based on the expense prediction. The process involves converting the text descriptions into numerical embeddings using BERT (Bidirectional Encoder Representations from Transformers) and training a Random Forest classifier on these embeddings.

i. BERT Embeddings

BERT (Bidirectional Encoder Representations from Transformers) is an open-source machine learning framework used to represent human language in a vector space for NLP tasks. In this project, it is used to create embeddings for descriptions for semantic categorization of expenses.

ii. Random Forest Classifier

Random Forest algorithm is a decision tree-based classification algorithm that makes predictions from multiple decision trees and does voting of all the trees to make final prediction.

iii. Hyperparameter Tuning

Hyperparameter tuning is the process of finding optimal set of parameters for a machine learning algorithm. In this project, it is used to find optimal set of parameters for the Random Forest classifier to improve its performance. GridSearchCV is used to explore a set of hyperparameters to find the best ones.

The expense categorization using BERT embeddings and Random Forest classifier was achieved by following steps:

i. Load dataset: Prepare the labeled dataset in csv format. Load it into program using pandas library. A sample dataset is:

Table 2: Sample Dataset for Expense Categorization

Description	Category
School fees	Education
Fill petrol in bike	Transport
Dinner with sara	Dining

ii. Generate embeddings: Encode the descriptions into numeric embeddings using BERT tokenizer. An example of such embedding is:
    - Description: School fees
    - Embedding: [[2.16032907e-01, 3.61940801e-01, ...,1.74844861e-01]]
iii. Split dataset: Split the dataset into two sets. One set is for training the model and another set is for testing the model.
iv. Tune hyperparameters: Tune the parameters using GridSearchCV to find best hyperparameters for Random Forest classifier. The following hyperparameters are tuned:
    - n_estimators: Number of trees in the forest
    - max_depth: Maximum depth of the tree
    - min_samples_split: Minimum number of samples required to split a node
    - min_samples_leaf: Minimum number of samples required at each leaf node
    - max_features: Number of features to consider at each split
v. Train the Random Forest classifier: Train the Random Forest Classifier using the training dataset and selected hyperparameters.
vi. Generate confusion matrix: Generate the confusion matrix for model to evaluate the performance, accuracy of the model. The confusion matrix for dataset with total 100 descriptions (80:20 split) and 20 testing data is:

True \ Predicted	Clothing	Dining	Education	Entertainment	Groceries	Health	Other Expenses	Rent/Mortgage	Transportation
Clothing	2	0	0	0	0	0	0	0	0
Dining	0	4	0	0	0	0	0	0	0
Education	0	0	1	0	0	0	0	0	0
Entertainment	0	0	0	2	0	0	0	0	0
Groceries	0	0	0	0	3	0	0	0	0
Health	0	0	0	0	1	0	0	0	0
Other Expenses	0	0	0	0	0	0	2	0	0
Rent/Mortgage	0	0	0	0	0	0	0	2	0
Transportation	0	0	0	0	0	0	1	0	2

Figure 17: Confusion Matrix

vii. Prepare classification report: Prepare the classification report to better evaluate the results using measures such as precision, recall, accuracy, F1-score, support etc. The observed classification report is as:

Class	Precision	Recall	F1-Score	Support
Clothing	1.00	1.00	1.00	2
Dining	1.00	1.00	1.00	4
Education	1.00	1.00	1.00	1
Entertainment	1.00	1.00	1.00	2
Groceries	0.75	1.00	0.86	3
Health	0.00	0.00	0.00	1
Other Expenses	0.67	1.00	0.80	2
Rent/Mortgage	1.00	1.00	1.00	2
Transportation	1.00	0.67	0.80	3
Accuracy			0.90	20
Macro Avg	0.82	0.85	0.83	20
Weighted Avg	0.88	0.90	0.88	20

Figure 18: Classification Report

viii. Compute accuracy: Compute accuracy of model by finding ratio of correct predictions to total number of predictions.

Accuracy = Total Number of Predictions / Number of Correct Predictions

The accuracy of the model was calculated to be 90%.

4.2.2 Expense prediction using ARIMA model

The feature of prediction or forecasting of expense in this project is implemented using the ARIMA (AutoRegressive Integrated Moving Average) model. ARIMA is a widely used time series forecasting method. The past 30 days daily expense records are analyzed using ARIMA model to predict future expense trends.

ARIMA is a statistical model used for time series forecasting. It is particularly useful for non-stationary time series data. A non-stationary time series data has statistical properties that are not constant. It consists of three components:

- AutoRegressive (AR): It indicates that the model uses past values of the time series to predict future values.
- Integrated (I): It refers to differencing the time series to make it stationary. Differencing removes trends and seasonality.

- Moving Average (MA): It means that the model uses past error terms (residuals) to predict future values.

The ARIMA model is defined by three parameters: (p, d, q), where:

- p: Number of lag observations included in the model (AR term).
- d: Number of times the data is differenced to make it stationary (I term).
- q: Size of the moving average window (MA term).

The expense forecasting using the ARIMA model was achieved by following steps:

i. Load dataset: Prepare a dataset of historical expense data in a structured format. Load the prepared dataset into program using pandas library. The dataset is retrieved from database of user created transactions. A sample dataset is:

Table 3: Sample Dataset for expense prediction

Date	Amount
2024-10-01	100
2024-10-02	150
2024-10-03	200

ii. Handle missing data: Create a continuous date range to ensure there are no missing dates and values in time series. Fill missing values using interpolation.
iii. Make the series stationary: Apply differencing to make the time series stationary.
iv. Fit ARIMA model: Define the ARIMA model with parameters (p, d, q) = (1, 0, 1) and fit it to differenced time series.
v. Forecast future expenses: Use the fitted ARIMA model to forecast expenses for specified time period. Reverse the differencing to obtain actual expense predictions.

Chapter 5: Implementation and Testing

5.1. Implementation

The implementation of PFMS involved writing programs to develop and test the database, backend and frontend components of the system. During implementation, the system was developed in multiple iterations with each iteration having a clear objective and deliverables. The major effort was put to develop the system with defined requirements and objectives.

5.1.1 Tools Used

The tools and technologies used during the implementation phase are listed as:

Table 4: Implementation Tools

Tool	Description
Python	Programming language for backend
JavaScript	Frontend programming language
PostgreSQL	Relational database system
Flask	Framework for backend implementation
ReactJS	Framework for frontend implementation
Trello	Agile project management tool.
Google Docs, Microsoft Word	Collaborative documentation
Draw.io	Diagram creation tool
Balsamiq	UI Wireframing

5.1.2. Implementation Details for Modules

The implementation details for some of the core modules of the application are described are follows:

- Authentication Module: The authentication system has two parts: registration and login. Both are handled by the 'auth.py' file in backend. For registration, user is required to enter email, username and password. The input is first validated by a frontend logic and then sent to the backend via an API. The API returns a success message in case of successful registration and displays error in case of any error. For login, the user input for username and password is verified by backend. The user is logged in case of valid credentials and appropriate error message is displayed for any error occurrence.
- Transaction Management Module: For managing transactions, users can add income and expenses from frontend. The transactions get inserted into the database from the business logic of backend. Use can view, edit and delete a transaction.

- Account Management Module: Similar to transactions, user can create, view, edit and delete an account. Account can be added from the frontend. The account data is then stored in the database by the backend logic.
- Goal Management Module: Goal management allows user to add a financial goal and add contributions to the goal. The goal can be viewed, edited or deleted from the goals page in the application.
- Budget Limiting Module: User can set limit to a set of predefined expense categories from the categories page. The analytics for spending on this category are projected in the dashboard.
- Expense Categorization Module: For automatic expense category, the description entered by the user while adding an expense is used. The description is sent to the backend system via an API. The description text is processed by the categorization algorithm to classify the text into a category type. It is then sent as a response to frontend.
- Expense Prediction Module: The future expense of a user is predicted using the prediction algorithm and projected in the dashboard analytics. For this, the records of past transactions (30 days) of a user are analyzed using the algorithm.

5.2. Testing

5.2.1. Test Cases for Unit Testing

Table 5: Unit test cases for authentication

ID	Test Case	Test Data	Expected Outcome	Actual Outcome	Status
Auth1	Verify user registration with valid credentials	username: testuser, email:test@example.com password: testpassword'	User should be registered successfully	As expected	Pass
Auth2	Verify user login with valid credentials	Username: testuser Password: testpassword	User should be registered successfully	As expected	Pass

Table 6: Unit test cases for account

ID	Test Case	Test Data	Expected Outcome	Actual Outcome	Status
Account1	Test if user can create an account	Name: Test Account, Balance: 1000.0	Account can be successfully created	As expected	Pass
Account2	Test if user can create account with existing name	Name: Test Account, Balance: 1000.0	Account cannot be created	As expected	Pass
Account3	Test if accounts can be updated	Name: Update Account, Balance: 2000.0	Account should be updated	As expected	Pass
Account3	Test if accounts can be deleted		Account should be deleted	As expected	Pass

Table 7:Unit test cases for transaction

ID	Test Case	Test Data	Expected Outcome	Actual Outcome	Status
Transaction 1	Test if user can create a transaction	Type: Expense, Amount: 500, Category: Food, Date: 2025-01-01	Transaction can be successfully created	As expected	Pass
Transaction 2	Test if user can create a transaction with invalid data	Type: Expense, Amount: -100, Category: Food, Date: 2025-01-01	Transaction cannot be created	As expected	Pass
Transaction 3	Test if a transaction can be updated	Type: Income, Amount: 1000, Category: Salary, Date: 2025-01-15	Transaction should be updated successfully	As expected	Pass

Transaction 4	Test if a transaction can be deleted		Transaction should be deleted successful	As expected	Pass

Table 8: Unit test cases for goal

ID	Test Case	Test Data	Expected Outcome	Actual Outcome	Status
Goal1	Test if user can create a goal	Name: Vacation Fund, Target Amount: 5000, Start Date: 2025-01-01, End Date: 2025-12-31	Goal can be successfully created	As expected	Pass
Goal2	Test if a goal can be updated	Name: Emergency Fund, Target Amount: 8000, End Date: 2026-01-01	Goal should be updated successful	As expected	Pass
Goal3	Test if a goal can be deleted		Goal should be deleted successfully	As expected	Pass
Goal4	Test if contribution can be added to a goal	Amount: 200	Contribution should be added	As expected	Pass

5.2.2 Test Cases for System Testing

Table 9: Test Cases for Authentication

Test Case ID	Description	Steps	Test Data	Expected Outcome	Actual Outcome	Status
REG_01	Register with valid details	1. Go to registration page. 2. Enter valid details. 3. Click "Register".	Name: John Doe Email: john@example.com Password: John@123 Confirm Password: John@123	Registration successful. User redirected to login page or dashboard.	As Expected	Pass
REG_02	Register with an existing email	1. Go to registration page. 2. Enter an existing email. 3. Click "Register".	Email: john@example.com Password: John@123 Confirm Password: John@123	Error message: ""Email already exists.""	As Expected	Pass
REG_03	Register with mismatched passwords	1. Go to registration page. 2. Enter mismatched passwords. 3. Click "Register".	Password: John@123 Confirm Password: John@456	Error message: ""Passwords do not match.""	As Expected	Pass
REG_04	Register with missing required fields	1. Go to registration page. 2. Leave required fields blank. 3. Click "Register".	Name: Email: Password: Confirm Password:	Error message: ""Please fill in all required fields.""	As Expected	Pass

REG_05	Register with a weak password	1. Go to registration page. 2. Enter a weak password. 3. Click "Register".	Password: 123 Confirm Password: 123	Error message: ""Password must be at least 8 characters long and include special characters. ""	As Expected	Pass
LOGIN_01	Login with valid credentials	1. Go to login page. 2. Enter valid credentials. 3. Click "Login".	Email: john@example.com Password: John@123	Login successful. User redirected to dashboard.	As Expected	Pass
LOGIN_02	Login with invalid email	1. Go to login page. 2. Enter invalid email. 3. Click "Login".	Email: invalid@example.com Password: John@123	Error message: ""Invalid email or password. ""	As Expected	Pass
LOGIN_03	Login with invalid password	1. Go to login page. 2. Enter invalid password. 3. Click "Login".	Email: john@example.com Password: Wrong@123	Error message: ""Invalid email or password. ""	As Expected	Pass
LOGIN_04	Login with empty fields	1. Go to login page. 2. Leave fields blank. 3. Click "Login".	Email: Password:	Error message: ""Please enter your email and password. ""	As Expected	Pass
LOGIN_05	Click on ""Forgot Password"" link	1. Go to login page. 2. Click "Forgot Password".	N/A	User redirected to password reset page.	As Expected	Pass

Table 10: Test Cases for account

ID	Description	Steps	Test Data	Expected Outcome	Actual Outcome	Status
ACC_01	Add a new account with valid details	1. Go to Accounts page. 2. Click "Add Account". 3. Enter valid details. 4. Click "Save".	Account Name: Savings Account Type: Bank Balance: 1000	Account added successfully. Confirmation message displayed.	As Expected	Pass
ACC_02	Add an account with missing required fields	1. Go to Accounts page. 2. Click "Add Account". 3. Leave required fields blank. 4. Click "Save".	Account Name: Account Type: Balance:	Error message: ""Please fill in all required fields.""	As Expected	Pass
ACC_03	Update an existing account	1. Go to Accounts page. 2. Select an account. 3. Click "Edit". 4. Update details. 5. Click "Save".	Account Name: Savings Updated Account Type: Bank Balance: 1500	Account updated successfully. Confirmation message displayed.	As Expected	Pass
ACC_04	Delete an existing account	1. Go to Accounts page. 2. Select an account. 3. Click "Delete". 4. Confirm deletion.	N/A	Account deleted successfully. Confirmation message displayed.	As Expected	Pass
ACC_05	View details of an existing account	1. Go to Accounts page.	N/A	Account details displayed correctly	As Expected	Pass

Table 11: Test Cases for transaction

ID	Description	Steps	Test Data	Expected Outcome	Actual Outcome	Status
TRANS_01	Add a new transaction with valid details	1. Go to Transactions page. 2. Click "Add Transaction". 3. Enter valid details. 4. Click "Save".	Date: 2023-10-01 Amount: 100 Category: Food Account: Savings	Transaction added successfully. Confirmation message displayed.	As Expected	Pass
TRANS_02	Add a transaction with missing required fields	1. Go to Transactions page. 2. Click "Add Transaction". 3. Leave required fields blank. 4. Click "Save".	Date: Amount: Category: Account:	Error message: ""Please fill in all required fields.""	As Expected	Pass
TRANS_03	View details of an existing transaction	1. Go to Transactions page. 2. Select a transaction. 3. Click "View Details".	N/A	Transaction details displayed correctly (e.g.	As Expected	Pass
TRANS_04	Update an existing transaction	1. Go to Transactions page. 2. Select a transaction. 3. Click "Edit". 4. Update details. 5. Click "Save".	Date: 2023-10-01 Amount: 150 Category: Groceries Account: Checking	Transaction updated successfully. Confirmation message displayed.	As Expected	Pass
TRANS_05	Delete an existing transaction	1. Go to Transactions page. 2. Select a transaction. 3. Click "Delete". 4. Confirm deletion.	N/A	Transaction deleted successfully. Confirmation message displayed.	As Expected	Pass

5.3. Result Analysis

Result analysis aims to evaluate the outcomes of the implemented features and the testing outcomes to verify if the system meets the expected requirements.

5.3.1. Test Result Analysis

The results for unit and system testing are as follows:

Table 12: Test Result Analysis

Test Type	Total Test Cases	Passed	Failed
Unit Testing	14	14	0
System Testing	20	20	0

5.3.2. Analysis of Fulfillment of Requirements

All key functionalities for the, such as user registration, login, expense tracking, and budgeting, were tested successfully. Minor issues that were identified were resolved. The expense categorization was successfully implemented with the accuracy of 90%.

5.3.3. Comparison with Objectives

The system successfully meets its desired objectives. All the expected functionality are fulfilled and verified with proper test coverage.

Chapter 6: Conclusion and Future Recommendations

6.1. Conclusion

PFMS aims to provide its user the independence of managing their personal finances. The carefully designed system architecture lays the solid foundation for robust application. Thoughtful design of database schemas provides the system the ability to handle edge cases for different use case scenario. The backend and frontend system work seamlessly with each other using REST API service. The user interface design provides a sleek, intuitive and friendly platform for the users to access the application.

Alongside, the features of the system are more than enough to provide optimal satisfaction to the users. Each functionality is designed and developed keeping the user's perspective in mind. Besides transaction management, account management and goals management, the expense categorization feature provides the user an advanced way to log their expense into the system.

To sum up, the system demonstrates the potential to simplify personal financial management for users, empowering them to make informed financial decisions.

6.2. Future Recommendations

While the system aims to fulfil all of its desired requirements, below mentioned are some of the future recommendations.

- Integration of functionality to handle multiple currencies.
- Integration with third-party financial services for automatic synchronization of transactions.
- Development of mobile version of the application.
- Customizable dashboards to allow users to customize their dashboard.
- Expanding usability by providing multi-language support.
- Including accessibility options to make the application inclusive for all users.

References

[1] A. Pooja Bhatt, U. Kiran Kondapally, and H. Lokineni, “Expense Tracker: A Smart Approach to Track Daily Expense,” 2024.

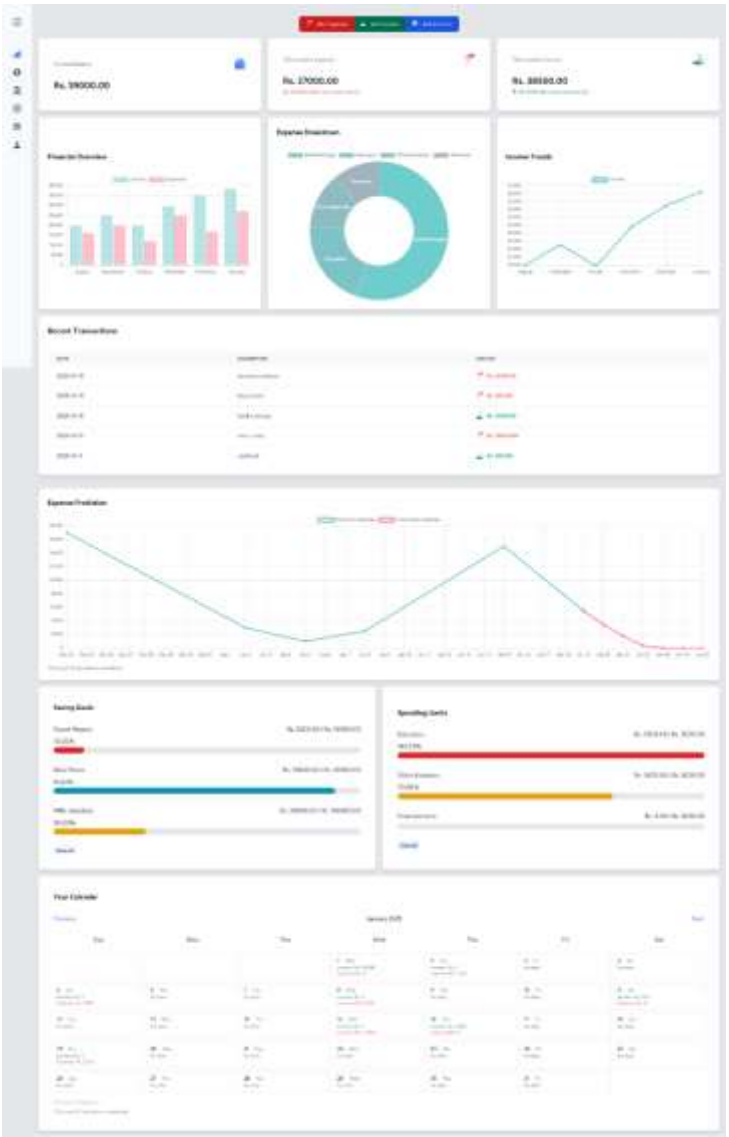
[2] Credit Karma, “mint-to-credit-karma-net-worth-signup.” [Online]. Available: https://www.creditkarma.com/lp/mint-to-credit-karma-net-worth-signup

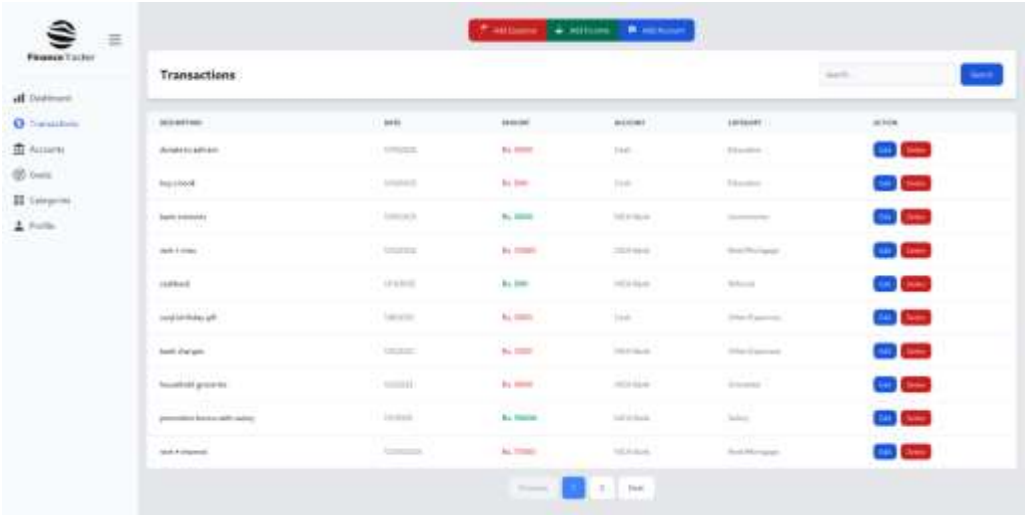
[3] BudgetBakers, “What is wallet.” [Online]. Available: https://budgetbakers.com/what-is-wallet/

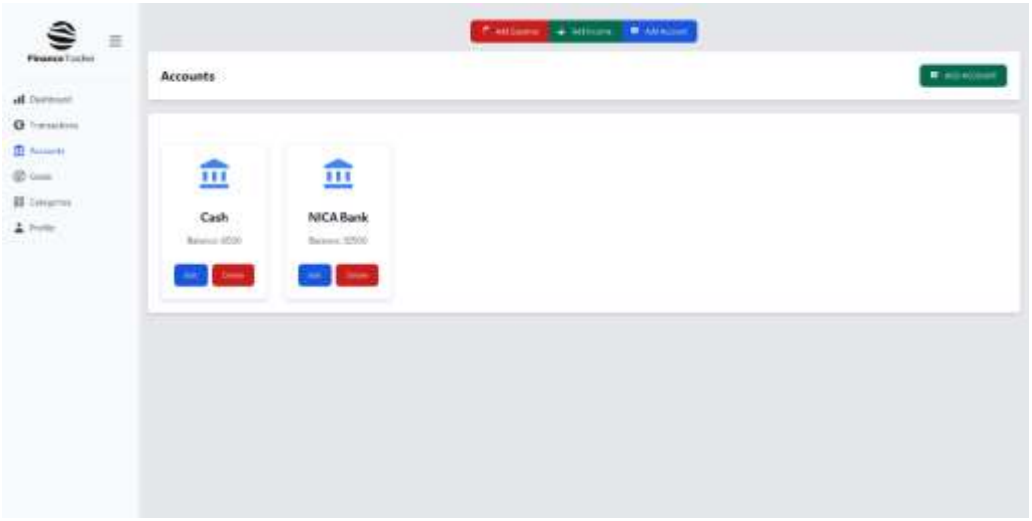
[4] YNAB, “You Need a Budget.” [Online]. Available: https://www.ynab.com/features

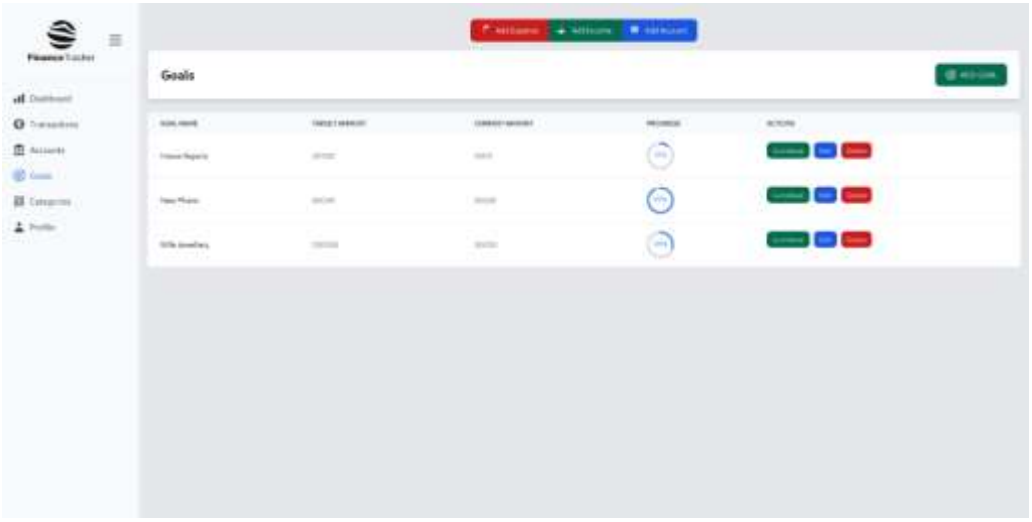
Appendices

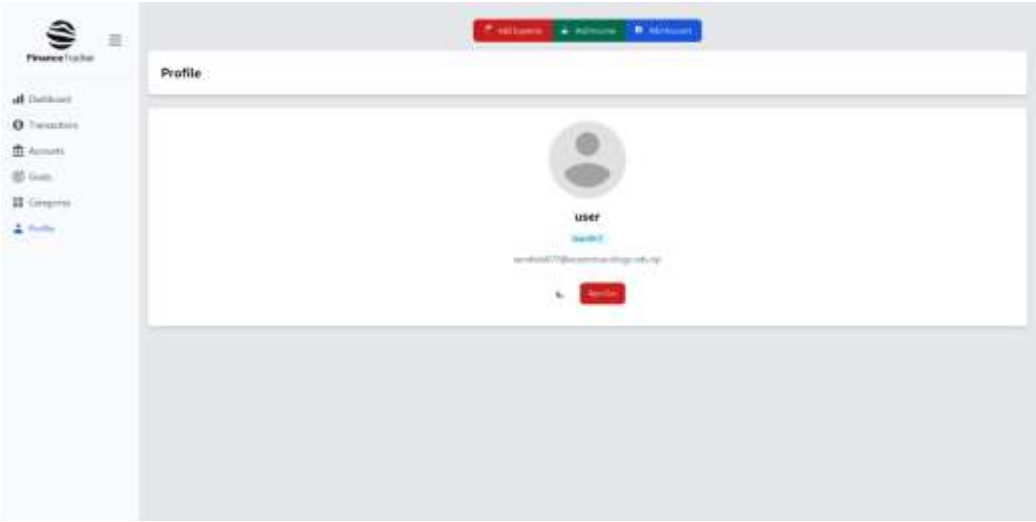
Screenshots of PFMS

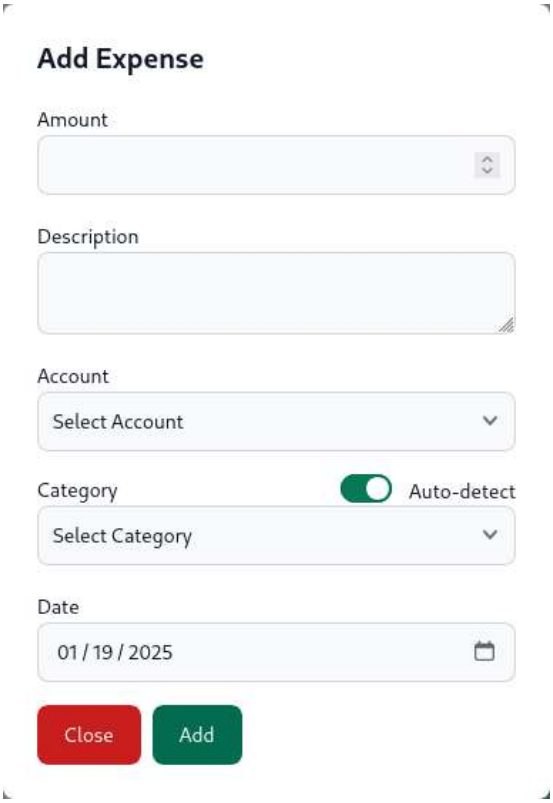


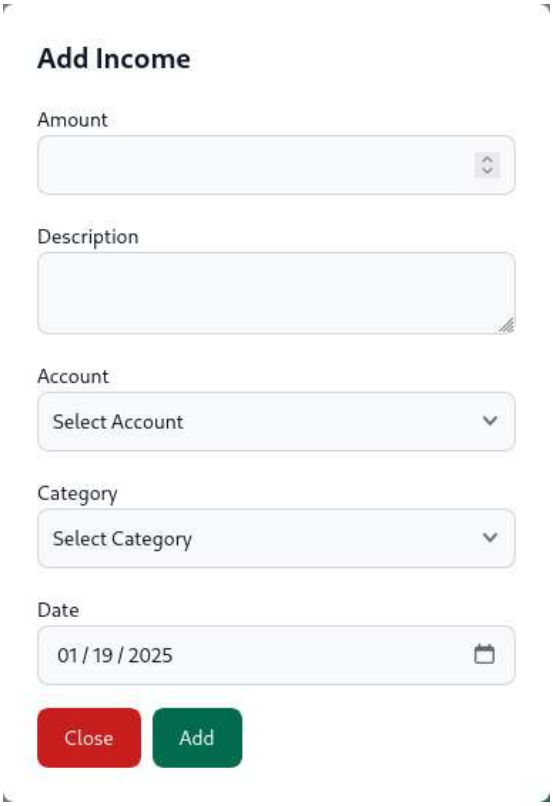

Transactions

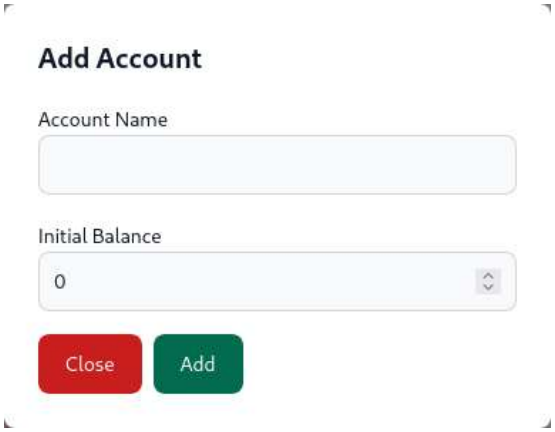

Accounts
Cash
NICA Bank

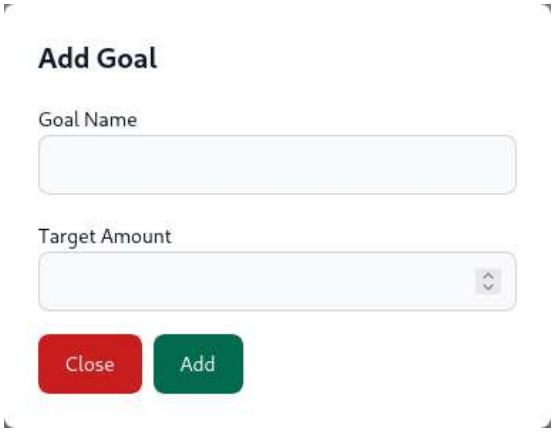

Goals

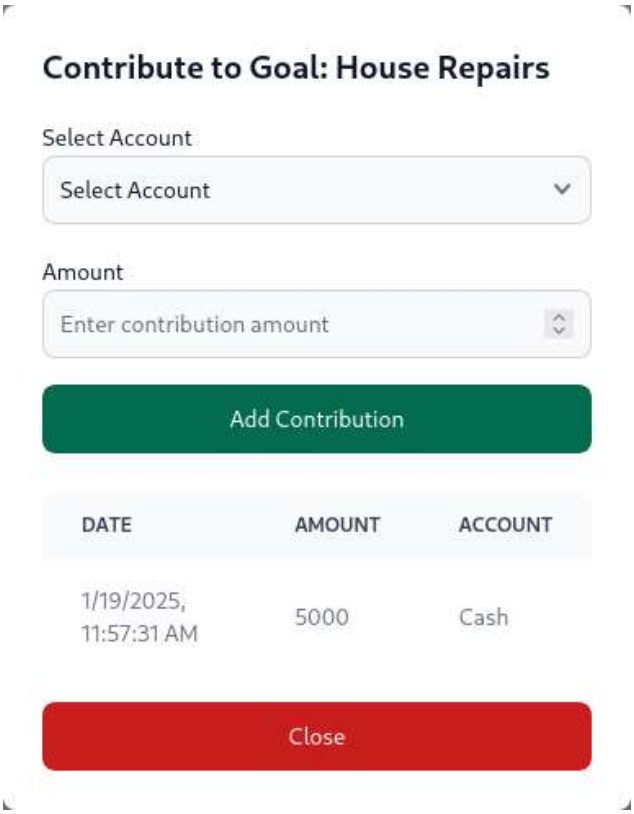

Profile
user


Add Expense
Amount
Description
Account
Select Account
Category
Auto-detect
Select Category
Date
01 / 19 / 2025
Close
Add


Add Income
Amount
Description
Account
Select Account
Category
Select Category
Date
01 / 19 / 2025
Close
Add


Add Account
Account Name
Initial Balance
0
Close
Add


Add Goal
Goal Name
Target Amount
Close
Add


Contribute to Goal: House Repairs
Select Account
Select Account
Amount
Enter contribution amount
Add Contribution
DATE
AMOUNT
ACCOUNT
1/19/2025, 11:57:31 AM
5000
Cash
Close

Code snippet for Random Forest

```
# Load JSON data into a pandas DataFrame
with open('data.json') as f:
    data = json.load(f)
df = pd.DataFrame(data)
# Load pre-trained BERT tokenizer and model
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
model = BertModel.from_pretrained("bert-base-uncased")
# Function to convert text description to BERT embedding
def description_to_bert_embedding(description):
    # Tokenize the description and prepare inputs for BERT
    inputs = tokenizer(description, return_tensors="pt", truncation=True, padding=True)
    # Get BERT model outputs
    outputs = model(**inputs)
    # Return the mean of the last hidden state as the embedding
    return outputs.last_hidden_state.mean(dim=1).detach().numpy()
# Convert all descriptions in the DataFrame to BERT embeddings
X = np.array([description_to_bert_embedding(desc) for desc in df["description"]])
# Reshape embeddings to 2D array for machine learning models
X = X.reshape(X.shape[0], -1)
# Extract target labels
y = df["category"]
```

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Define hyperparameter grid for RandomForestClassifier
param_grid = {
    "n_estimators": [50, 100, 200],  # Number of trees in the forest
    "max_depth": [None, 10, 20, 30],  # Maximum depth of the tree
    "min_samples_split": [2, 5, 10],  # Minimum samples required to split a node
    "min_samples_leaf": [1, 2, 4],  # Minimum samples required at each leaf node
    "max_features": ["sqrt", "log2"],  # Number of features to consider for the best split
}
# Initialize RandomForestClassifier
rf_model = RandomForestClassifier(random_state=42)
# Perform grid search with cross-validation to find the best hyperparameters
grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    cv=3,  # 3-fold cross-validation
    scoring="accuracy",  # Evaluation metric
    n_jobs=-1,  # Use all available CPU cores
    verbose=2,  # Print progress during grid search
)
# Fit the grid search to the training data
grid_search.fit(X_train, y_train)
```

```
# Retrieve the best hyperparameters found
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)
# Use the best model from the grid search
rf_model = grid_search.best_estimator_
# Predict on the test set
y_pred = rf_model.predict(X_test)
# Print classification report (precision, recall, F1-score, etc.)
print(classification_report(y_test, y_pred))
# Print confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
# Calculate and print accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

Code snippet for ARIMA

```
def arima_predict(expenses: list, time_period=30):
    df = pd.DataFrame(expenses)
    df['date'] = pd.to_datetime(df['date'])
    df.set_index('date', inplace=True)
    df.sort_index(inplace=True)
    # Create a continuous date range
    date_range = pd.date_range(
        start=df.index.min(), end=df.index.max(), freq='D')
    # Reindex the DataFrame to include all dates
    df_reindexed = df.reindex(date_range)
    # Forward fill the missing values (you can use other methods like interpolation)
    df_reindexed['amount'] = df_reindexed['amount'].interpolate(
        method='linear').fillna(0)
    # Now df_reindexed has continuous dates with filled values
    daily_expenses = df_reindexed['amount']
    # Make the series stationary by differencing
    differenced_expenses = daily_expenses.diff().dropna()
    p, q = 1, 1
    model = ARIMA(differenced_expenses, order=(p, 0, q))
    model_fit = model.fit()
    forecast_diff = model_fit.forecast(time_period)
    last_known_value = daily_expenses.iloc[-1]
```

```
    last_known_series = pd.Series([last_known_value], index=[
                                daily_expenses.index[-1]])
    forecast_index = pd.date_range(
        start=daily_expenses.index[-1] + pd.Timedelta(days=1), periods=time_period,
freq='D')
    forecast_diff_series = pd.Series(
        forecast_diff.values, index=forecast_index)
    # Combine the last known value and the forecasted differences
    combined_series = pd.concat([last_known_series, forecast_diff_series])
    # Reverse the differencing to get the actual expense predictions
    forecast_actual = combined_series.cumsum()
    # Convert the index to formatted date strings
    forecast_actual.index = forecast_actual.index.strftime('%Y-%m-%d')
    # Convert the Series to a dictionary with formatted date strings as keys
    forecast_dict = forecast_actual.to_dict()
    return forecast_dict
```