UE21CS352B - Object Oriented Analysis & Design using Java

Mini Project Report

Personal Finance Tracker

Submitted by:

Shashank Reddy Prakash	**PES2UG21CS491**
S V Gautham	**PES2UG21CS446**
Shishir B	**PES2UG21CS494**
Shree Hari Nadig B M	**PES2UG21CS499**

6th Semester H Section

Prof. K V N M Ramesh
Associate Professor

January - May 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

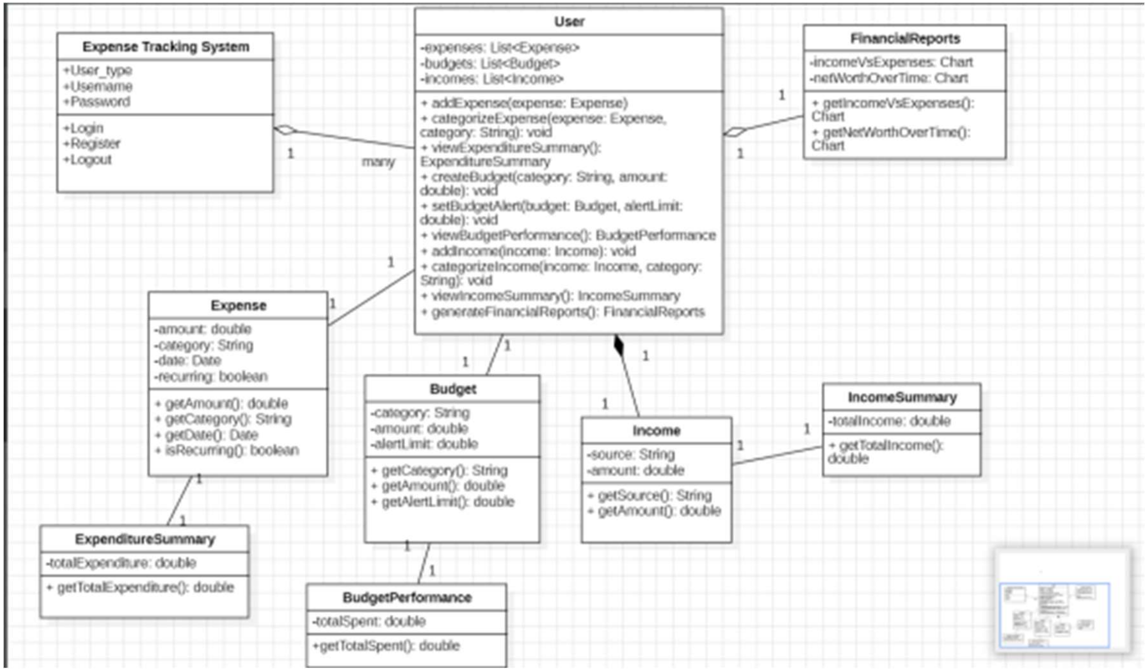
Problem Statement –

Managing personal finances effectively is crucial for achieving financial stability and meeting long-term goals, such as saving for retirement, paying off debt, or funding education. However, individuals often struggle with tracking their daily expenses, managing budgets, and planning for future financial needs due to a lack of tools that can integrate and simplify these tasks. Traditional methods of tracking expenses and income, such as using physical notebooks or basic spreadsheets, are time-consuming, prone to errors, and do not provide insights into spending patterns or financial health. Additionally, the process of manually managing and remembering payment due dates, categorizing expenses, and monitoring credit scores is cumbersome and inefficient.

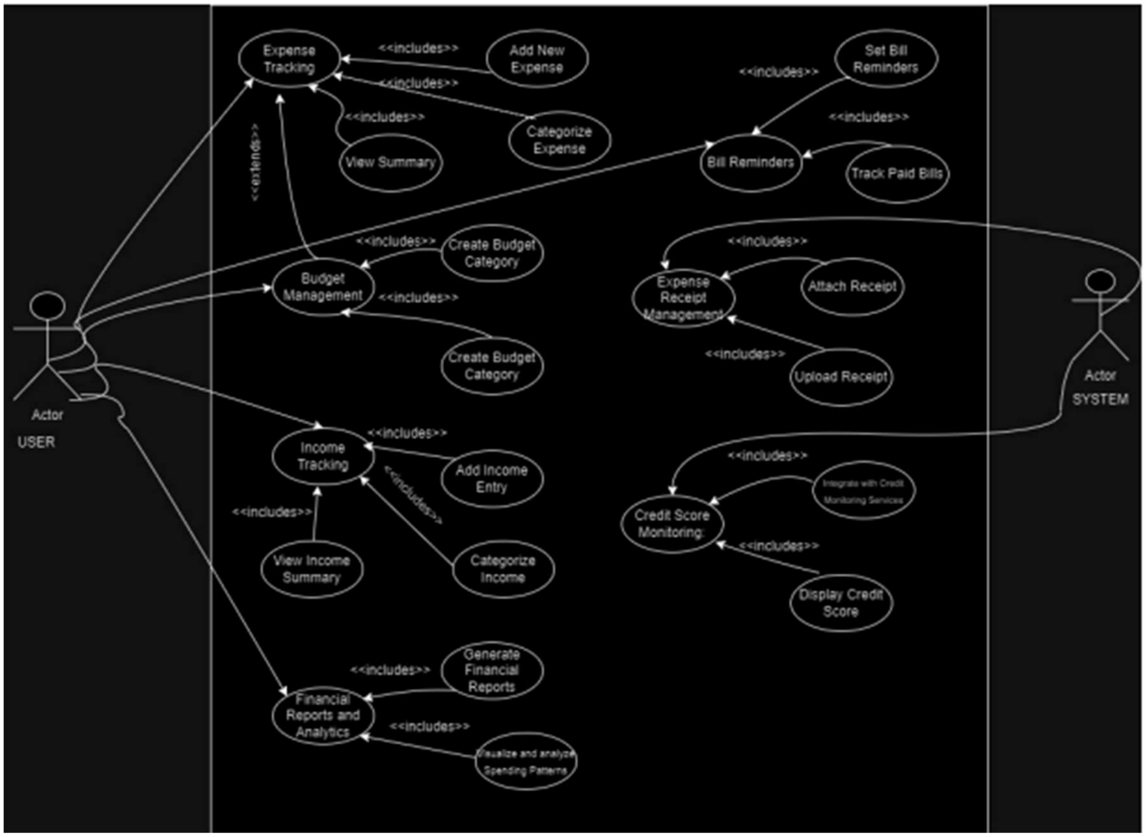
The goal of this project is to develop a comprehensive web-based personal finance management system that allows users to efficiently track their expenses, manage budgets, monitor income, and analyze their financial health through an integrated platform. This system will help users make informed financial decisions, improve their spending habits, and achieve their financial goals.

Models -

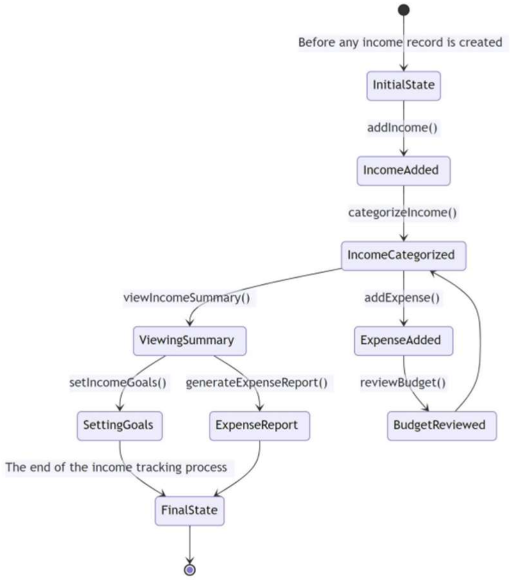
Class diagram –

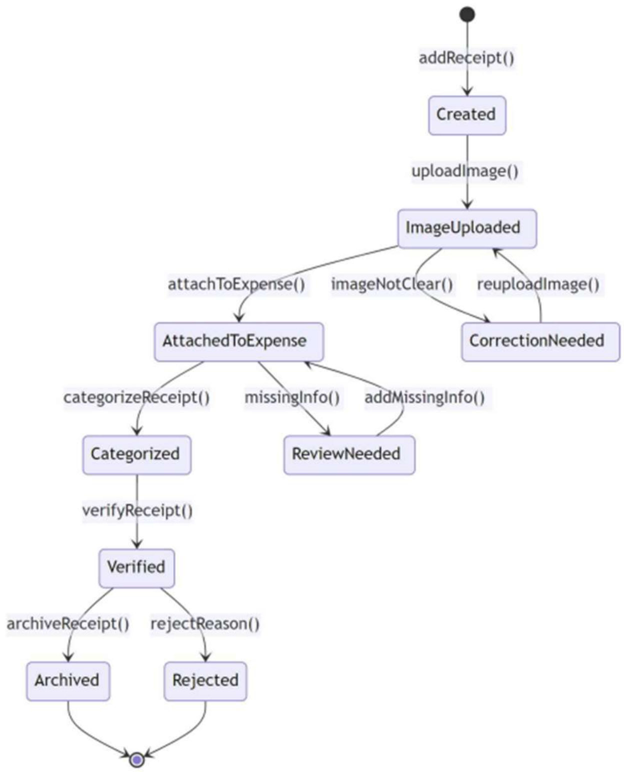


Use case diagram –

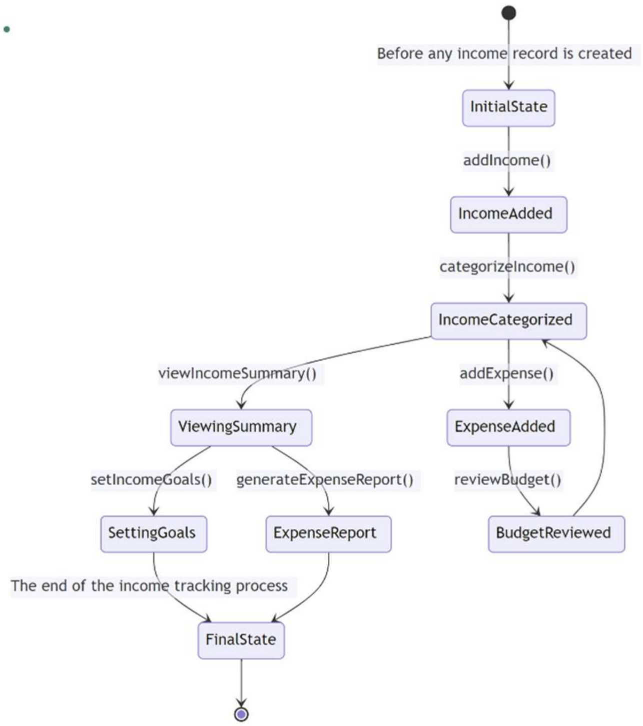


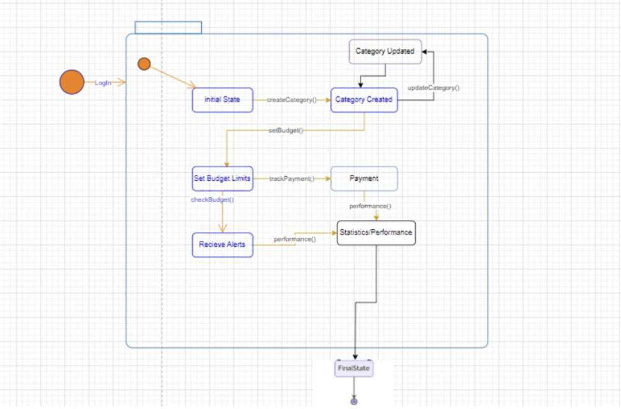
State diagram –

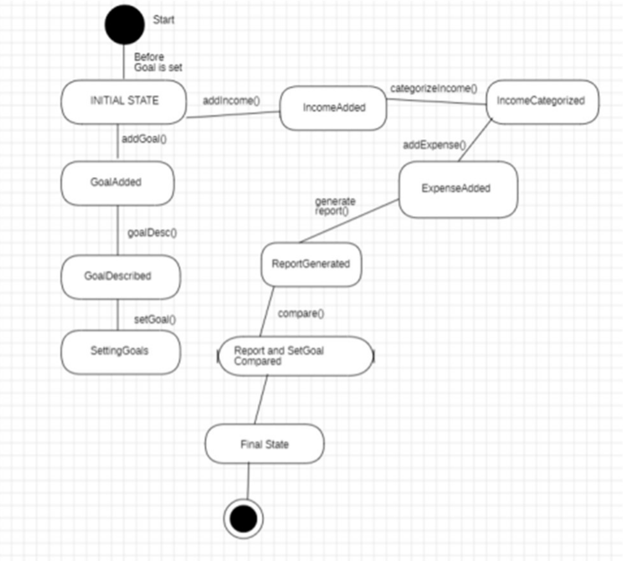



addReceipt()
Created
uploadImage()
ImageUploaded
attachToExpense()
imageNotClear()
reuploadImage()
AttachedToExpense
CorrectionNeeded
categorizeReceipt()
missingInfo()
addMissingInfo()
Categorized
ReviewNeeded
verifyReceipt()
Verified
archiveReceipt()
rejectReason()
Archived
Rejected

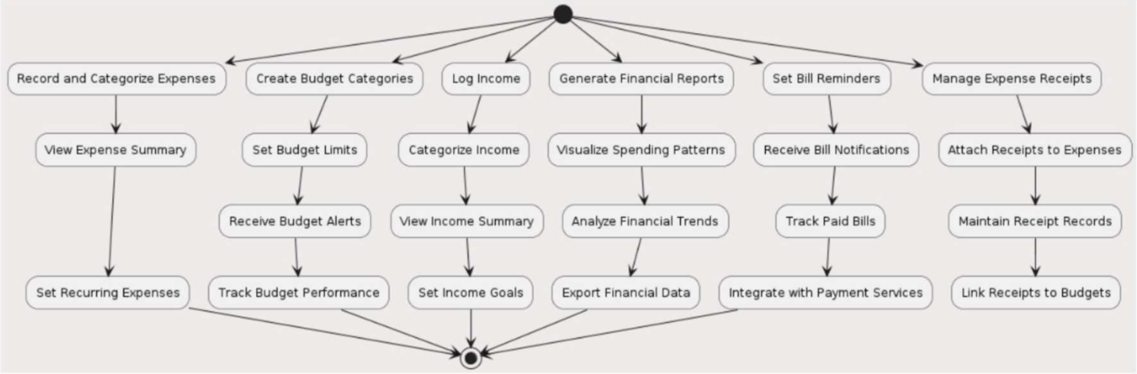
-


Before any income record is created
InitialState
addIncome()
IncomeAdded
categorizeIncome()
IncomeCategorized
viewIncomeSummary()
addExpense()
ViewingSummary
ExpenseAdded
setIncomeGoals()
generateExpenseReport()
reviewBudget()
SettingGoals
ExpenseReport
BudgetReviewed
The end of the income tracking process
FinalState





Activity Diagram –



Architectural Pattern

This project uses a MVC (Model, View, and Controller pattern). Each use case has 4 files, Model, Controller, Service and Repository.

Model - The Model component represents the domain-specific data and business logic in an application. It's responsible for retrieving, storing, and processing data, often through interaction with a database. In an MVC application, the model directly manages the data, logic, and rules of the application.

Repository - The Repository layer serves as an abstraction layer between the data access layer and the business logic layer of the application. It allows for a more flexible and decoupled way to access data stored in databases or other persistent stores.

Service - The Service layer is responsible for implementing the application's business logic. It serves as a bridge between the Controller and Repository, ensuring that business requirements are met by making calls to the repository for data.

Controller - The Controller acts as the interface between the user and the system. It handles user input and converts it to commands for the model or view.

Design Principles

The Design Principles used in the project is the single responsibility principle, every model / class has a single responsibility over a part of the functionality provided by the application. The OCP is also followed since the classes are open for extensions but closed for modifications as its existing functionality is not directly modified. The Dependency inversion principle is also followed in some places as some classes depend on the abstract implementation of ‘Date’

1) **Single Responsibility Principle** – Every component has a single responsibility over a part of the functionality provided by the application, Ex income class handles only adding income.
2) **Open Close Principle** – Is followed since the classes are open for extensions but closed for modifications as its existing functionality is not directly modified.
3) **Dependency Inversion Principle** – This principle is followed as the high-level models do not depend on the low-level models, but rather on abstractions, this being consistent with DIP.

4) **Dependency Injection principle** – The Dependency Injection principle is also followed as @Autowired is used in the service to inject the Repository into the service.

Design Patterns

The following design patterns have been implemented in the code -

1) **Builder pattern** – In our project, we used the Builder Pattern to manage the creation of Income and ExpenseAdd objects, which involve setting several fields. This pattern allows for a clear, step-by-step construction process, enhancing code readability and maintainability. Through a static inner Builder class in our Income and ExpenseAdd model, we enable chaining methods that set properties, culminating in a build() method that finalizes and returns the constructed object. This approach simplifies object creation and ensures clean and error-free code, especially beneficial when dealing with complex objects with multiple attributes.

2) **Facade Patter** - In our application, the FinancialGoalFacade and FinancialReportsFacade serves as a facade, abstracting the complexities of the financial goal management operations handled by the service. This facade simplifies the interaction for the controller by offering straightforward methods like findAllFinancialGoals() and addFinancialGoal(), which internally manage more complex operations such as database encapsulated in the service layer. By using the facade pattern, we ensure that the controller remains clean and focused only on handling incoming requests and responses, without being cluttered with the business logic and data access details.

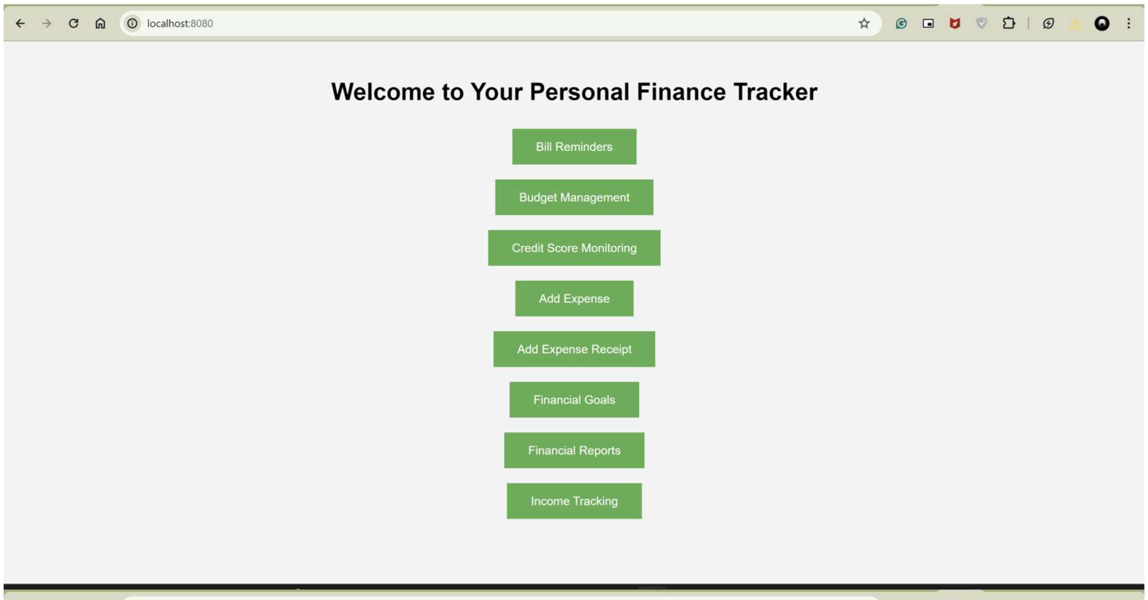
GitHub Link

https://github.com/anonymous0905/financetracker--OOADJ--Project

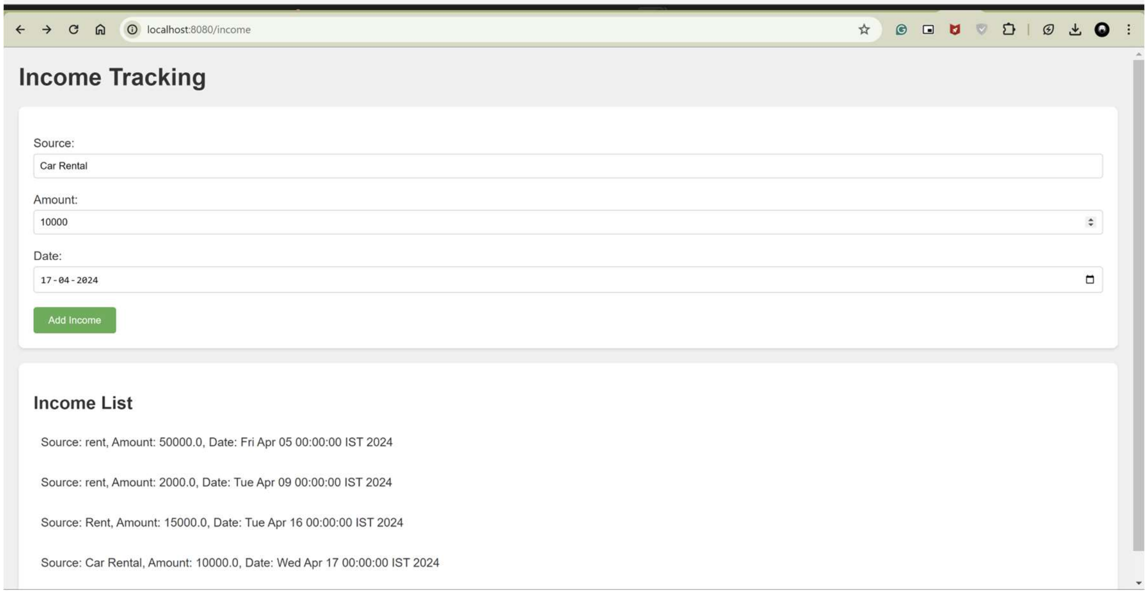
Individual Contributions

1) Shashank Reddy Prakash – Income and Expense Receipt Management
2) Shishir B – Expense and Bill reminder
3) Shree Hari Nadig B M – Budget and Financial Goal Management
4) S V Gautham – Credit Score and Financial Report

Screenshots



Welcome to Your Personal Finance Tracker



Income Tracking

Source: Car Rental

Amount: 10000

Date: 17-04-2024

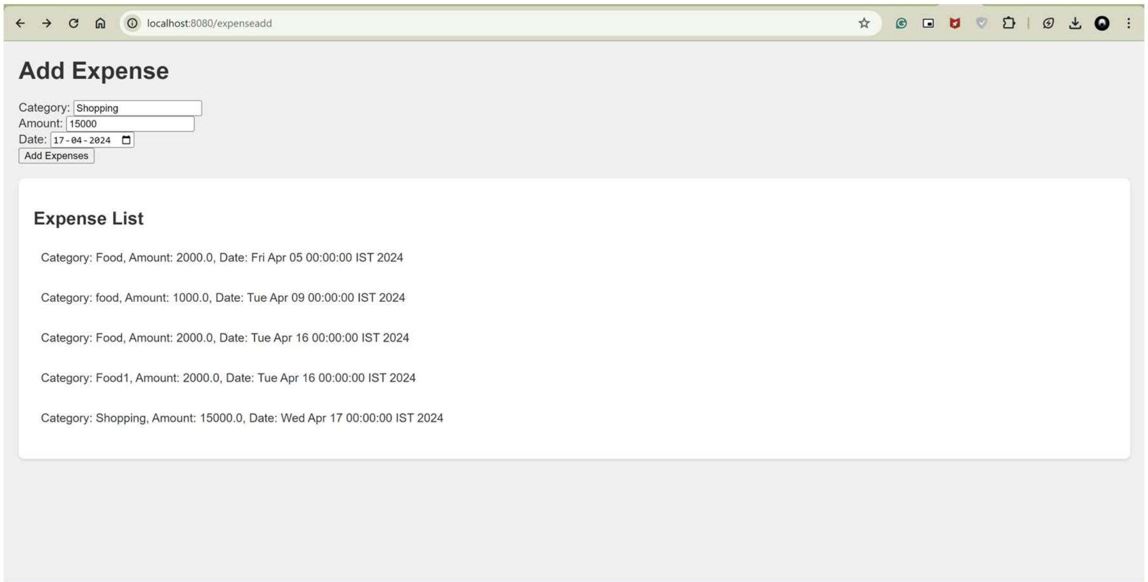
Income List

Source: rent, Amount: 50000.0, Date: Fri Apr 05 00:00:00 IST 2024

Source: rent, Amount: 2000.0, Date: Tue Apr 09 00:00:00 IST 2024

Source: Rent, Amount: 15000.0, Date: Tue Apr 16 00:00:00 IST 2024

Source: Car Rental, Amount: 10000.0, Date: Wed Apr 17 00:00:00 IST 2024



Add Expense

Category: Shopping
Amount: 15000
Date: 17-04-2024

Expense List

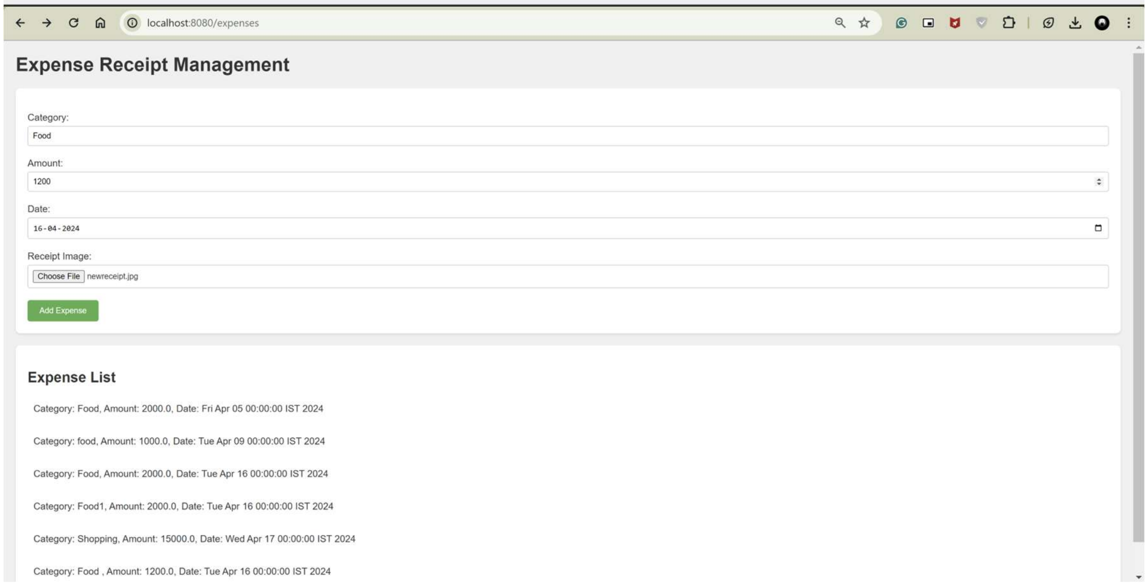
Category: Food, Amount: 2000.0, Date: Fri Apr 05 00:00:00 IST 2024

Category: food, Amount: 1000.0, Date: Tue Apr 09 00:00:00 IST 2024

Category: Food, Amount: 2000.0, Date: Tue Apr 16 00:00:00 IST 2024

Category: Food1, Amount: 2000.0, Date: Tue Apr 16 00:00:00 IST 2024

Category: Shopping, Amount: 15000.0, Date: Wed Apr 17 00:00:00 IST 2024



Expense Receipt Management

Category:
Food

Amount:
1200

Date:
16-04-2024

Receipt Image:
newreceipt.jpg

Expense List

Category: Food, Amount: 2000.0, Date: Fri Apr 05 00:00:00 IST 2024

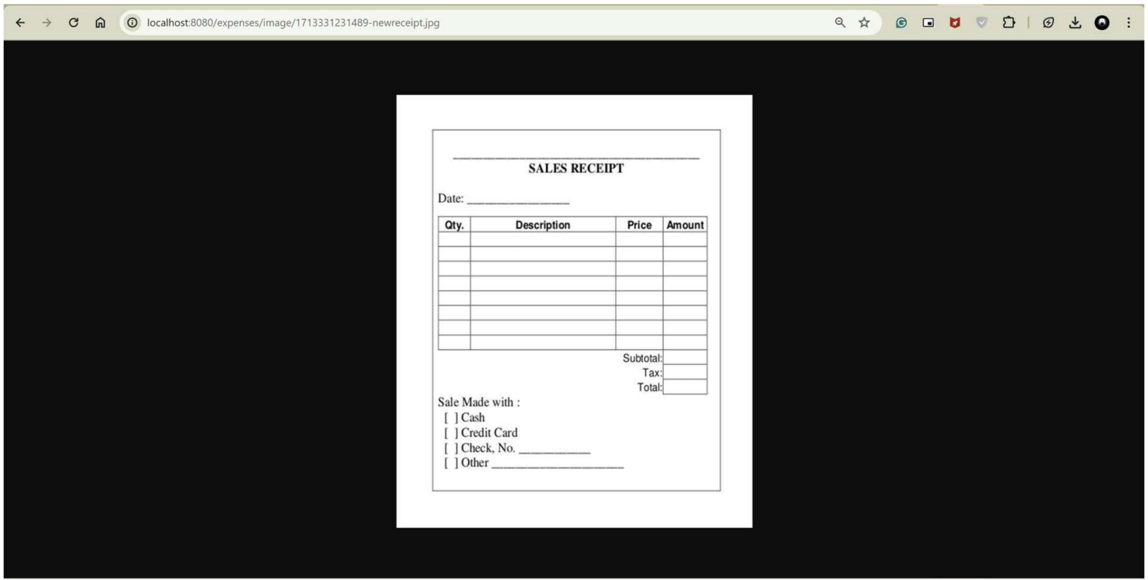
Category: food, Amount: 1000.0, Date: Tue Apr 09 00:00:00 IST 2024

Category: Food, Amount: 2000.0, Date: Tue Apr 16 00:00:00 IST 2024

Category: Food1, Amount: 2000.0, Date: Tue Apr 16 00:00:00 IST 2024

Category: Shopping, Amount: 15000.0, Date: Wed Apr 17 00:00:00 IST 2024

Category: Food , Amount: 1200.0, Date: Tue Apr 16 00:00:00 IST 2024



localhost:8080/expenses/image/1713331231489-newreceipt.jpg

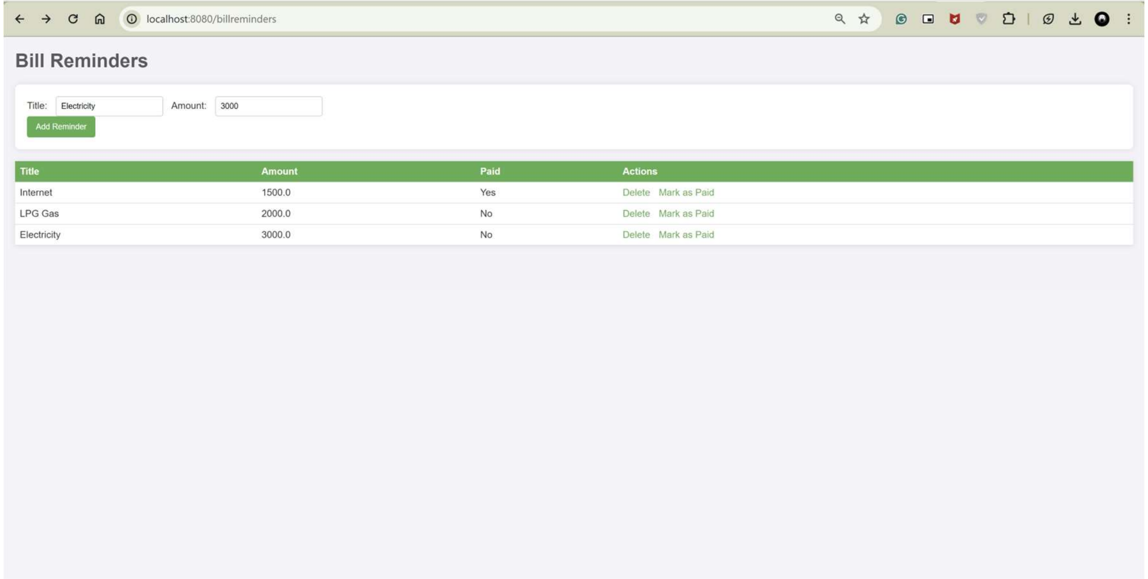
SALES RECEIPT

Date: ________________

Qty.	Description	Price	Amount

Subtotal:
Tax:
Total:

Sale Made with :
[ ] Cash
[ ] Credit Card
[ ] Check, No. ___________
[ ] Other ____________________

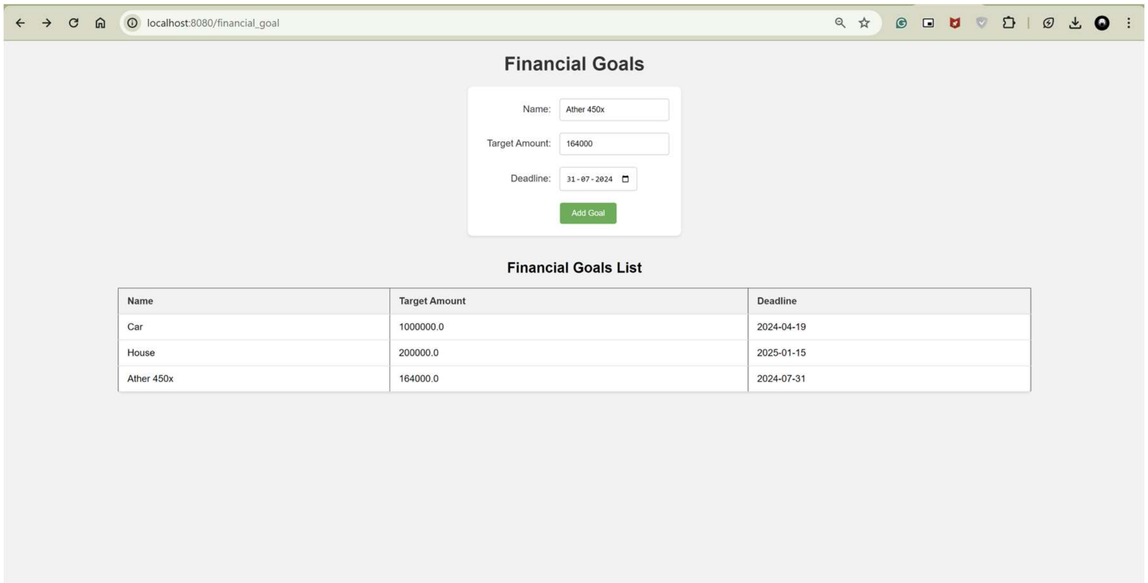


localhost:8080/billreminders

Bill Reminders

Title: Electricity Amount: 3000

Title	Amount	Paid	Actions
Internet	1500.0	Yes	Delete Mark as Paid
LPG Gas	2000.0	No	Delete Mark as Paid
Electricity	3000.0	No	Delete Mark as Paid



Financial Goals

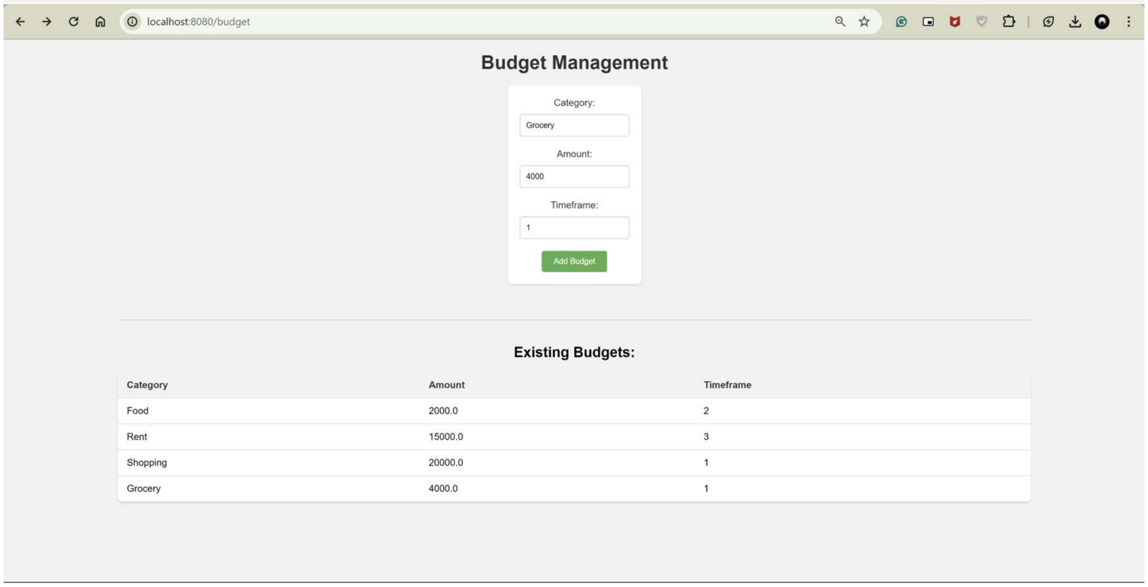
Name: Ather 450x

Target Amount: 164000

Deadline: 31-07-2024

Financial Goals List

Name	Target Amount	Deadline
Car	1000000.0	2024-04-19
House	200000.0	2025-01-15
Ather 450x	164000.0	2024-07-31



Budget Management

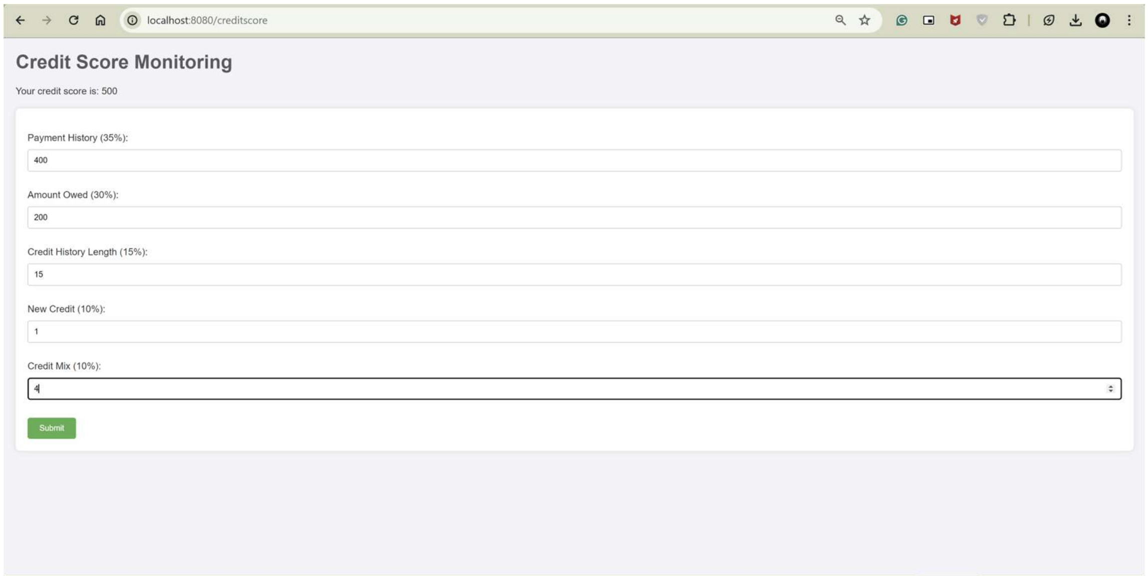
Category: Grocery

Amount: 4000

Timeframe: 1

Existing Budgets:

Category	Amount	Timeframe
Food	2000.0	2
Rent	15000.0	3
Shopping	20000.0	1
Grocery	4000.0	1



Credit Score Monitoring

Your credit score is: 500

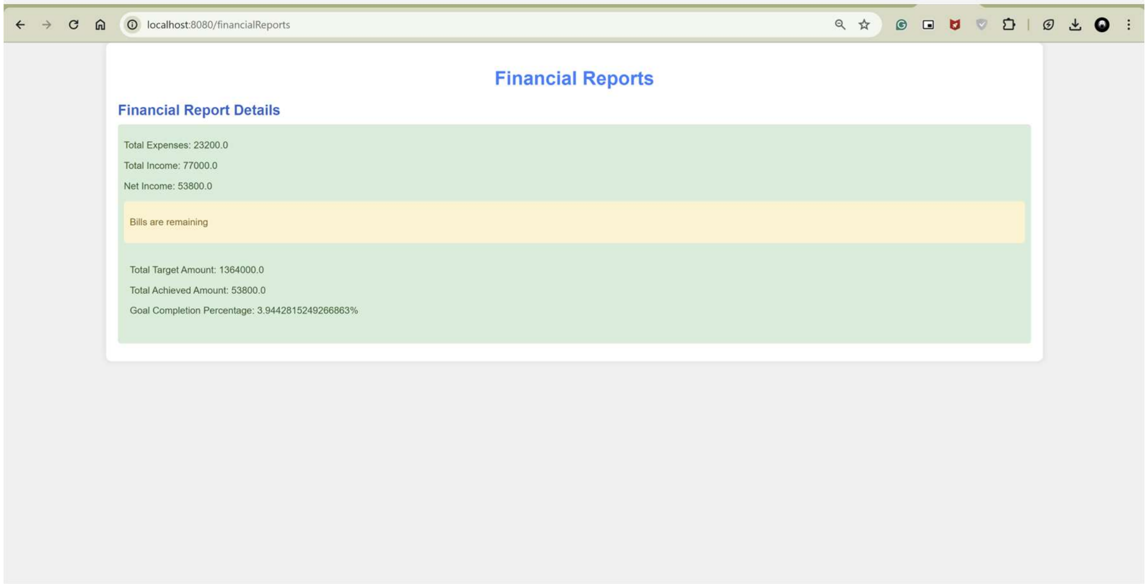
Payment History (35%): 400

Amount Owed (30%): 200

Credit History Length (15%): 15

New Credit (10%): 1

Credit Mix (10%): 4



Financial Reports

Financial Report Details

Total Expenses: 23200.0

Total Income: 77000.0

Net Income: 53800.0

Bills are remaining

Total Target Amount: 1364000.0

Total Achieved Amount: 53800.0

Goal Completion Percentage: 3.9442815249266863%