A Project Activity Report
Submitted for
SOFTWARE ENGINEERING
(UCS 503)

III year, BE COE

EXPENSE TRACKER:
BroBroke

Submitted By:

RAVNEET KAUR - 102203202
HARDETYA GILL - 102203213
RADHIKA AGGARWAL - 102203226
SARA AGNIHOTRI - 102203248

Submitted To:
DR. PRAGYA MISHRA

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(A DEEMED TO BE UNIVERSITY),
PATIALA, PUNJAB, INDIA

Semester V: Jul - Dec 2024

TABLE OF CONTENTS

1.
PROJECT SELECTION PHASE

1.1 SOFTWARE BID
UCS 503- Software Engineering Lab

Group : 3C06 Dated:7/08/2024

Team Name: BroBroke
Team ID (will be assigned by Instructor):

Please enter the names of your Preferred Team Members:-

Name	Roll No	Project Experience	Programming Language used	Signature
Radhika Aggarwal	102203226	E-Commerce Website	MERN	
Sara Agnihotri	102203248	Resume Website, Flight Management	HTML,CSS, JS,SQL	
Ravneet Kaur	102203202	Directory Management, Elder Home Management, Pharmacy Management	Shell, C, SQL, Python	
Hardetya Gill	102203213	Elderly Management system	SQL, HTML, CSS, JS	

Programming Language / Environment Experience

List the languages you are most comfortable developing as a team in your order of preference. Many of the projects involve Java or C/C++ programming.

1.HTML, CSS , JS
2.REACT
3.PYTHON

Choices of Projects:

Please select **4 projects** your team would like to work on, by order of preference: *[Write at least one paragraph for each choice (motivation, reason for choice, feasibility analysis, etc.)]*

Choose your team members wisely. You will not be allowed to change teams.

	Project Name	Unique Selling Point
First Choice **Expense Tracker**	Expense Tracker web application project is designed to help users manage their finances by tracking expenses, categorizing transactions, and providing insightful reports to promote better budgeting and financial planning.	Users can log and categorize their expenses in real time, ensuring accurate financial records and making it easier for students struggling with expenses to keep them up-to-date.
Second Choice **Netflix Clone**	A Netflix clone project involves creating a web application that allows users to stream video content.	It features user authentication, content management, search and filter, recommendations, subscriptions, and multi-device support.
Third Choice **Diet and Fitness Check**	A diet and fitness check project involves creating an web application that helps users track their nutritional intake, exercise routines, and overall health metrics to achieve their fitness goals.	Empowers users to easily monitor and optimize their health and wellness by providing personalized nutritional insights, exercise tracking, and progress visualization tools.
Fourth Choice **Email Sorter**	An email sorter website project involves creating a platform that automatically categorizes and prioritizes incoming emails to enhance user productivity and organization.	Streamlines inbox management by automatically categorizing and tagging incoming emails. It helps users quickly identify important messages, reducing clutter and boosting productivity.

Additional Remarks/ Inputs

Please tell us about any other factors that we should take into consideration (e.g., if you really would like to work on a project for some particularly convincing reason).

Creating an expense tracker website for college students provides several key benefits:

1. **Financial Literacy**: It helps students develop essential money management skills by tracking their spending, setting budgets, and understanding their financial habits.
2. **Expense Awareness**: By visualizing their expenses, students can make informed decisions, avoid overspending, and better allocate their limited resources, leading to improved financial stability during their college years.

2.
FEASIBILITY REPORT

FEASIBILITY REPORT

Project Overview:

This project aims to develop an expense tracker using the MERN stack. The tracker will allow users to record and manage their expenses, categorize them, set budgets, generate reports, and visualize spending patterns through charts and graphs.

Purpose of the Report:

The purpose of this report is to assess the feasibility of developing the expense tracker, considering technical, economic, operational, and scheduling aspects.

1. Technical Feasibility:

· Technology Stack:
MongoDB (database), Express.js (backend API), React (UI), Node.js (runtime environment).
· Development Environment:
Tools: Git/GitHub (version control), VSCode (IDE), Jest (testing), AWS/Heroku (deployment).
· Security:
Features: JWT (authentication), data encryption.

2. Financial Feasibility:

· Development Costs: No Budget Constraints
Since the project is non-commercial and educational, most resources can be accessed for free or at a low cost, such as using open-source libraries, free development environments, and limited cloud hosting services like AWS Free Tier or GitHub Pages. The main expenses would likely be student time and effort, making the financial risk negligible. With no immediate need for paid resources, licenses, or complex infrastructure, the project is financially viable and can be completed within a tight budget.

3. Market Analysis:

· Market Trends:
Growth in personal finance management tools driven by digitalization and increasing financial literacy.
· Target Audience:
Demographics: Millennials, Gen Z, tech-savvy professionals.
Geographic Focus: Urban areas, developed markets (US, EU), emerging markets.
· Market Entry Strategy:
Soft launch, digital marketing, partnerships, and referral programs.
Scaling: Feature expansion, geographical growth.

4. Operational Feasibility:

User Experience:
· Assessing whether the intended users (students, individuals, etc.) will find the system easy to use and beneficial.
· Ensuring the interface is user-friendly and meets the needs of the target audience.
· Planning for post-launch activities, such as user support and system upgrades.
· Determining if the system integrates smoothly into existing workflows and doesn't disrupt current operations.

5. Legal Feasibility:

· User Data Collection:
If we collect and store user data (e.g., usernames, emails, transaction history), it is essential to comply with basic data privacy principles.
· Compliance with Regulations:
Obtain user consent before collecting data. Allow users to view, update, and delete their data. Implement basic data security measures like encryption.
· Data Security:
Ensure that sensitive user data (like passwords or financial information) is securely stored using encryption techniques.
· Our project simulates financial transactions (e.g., expense tracking), clarifying that no real monetary transactions are involved.

For our project, legal feasibility is mainly about adhering to best practices in data protection, respecting intellectual property rights, and understanding the basics of legal compliance in a real-world scenario.There are no significant legal hurdles, and no special licenses are needed.

3.
GANTT CHART

Expense Tracker

Gantt Chart

PROCESS	ANALYSIS PHASE			DESIGN PHASE		
	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
Planning						
Wireframing						
Set up Project Repository						
Front-end development						
Develop UI Components						
Set Up React.js Project Structure						

PROCESS	CODING PHASE				TESTING		DEPLOYEMENT
	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15
Back-end development							
Set Up Node.js and Express.js Server							
Design and Implement MongoDB Database Schema							
Connect Frontend with Backend APIs							
Mock API calls and test the interactions with the backend.							
Preparation of deployment strategy							

Fig 1- Gantt Chart

4.
ANALYSIS PHASE

4.1 USE CASE DIAGRAM

This use case diagram illustrates a comprehensive expense tracking system that caters to different types of users (students, influencers) and supports both individual and shared expense management. Core functionalities include user authentication, expense tracking income tracking, expense sharing (Splitwise), report generation, and wishlist management.

Actors:

· User - The main actor, representing anyone who uses the application. This includes:
· Student - A specific type of user, likely with access to certain features tailored for students.
· Influencer - Another specific type of user, perhaps with additional needs for tracking and managing expenses.

Use Cases:

1. Authentication:
· Sign-up - Allows users to create a new account.
· Login - Lets existing users access their account. This use case includes extensions like Forgot Password to help users recover their accounts.

2. Splitwise:
This component likely handles expense sharing among multiple users. Key use cases include:
· Add Member - Allows users to add other people for expense sharing.
· Edit Member - Permits editing details of existing members.
· Add Expense - Users can add expenses that can be shared among members.
· Edit Expense - Allows modifications to existing shared expenses.
· Split Amount - Distributes the expense amount among the members.

3. Tracker:
This core component focuses on managing and tracking personal expenses. It includes:
· Add Expense - Users can log their expenses.
· Add Income - Users can add records of their income.
· Update - Allows users to update any existing entries (expenses or income).
· Delete - Enables the deletion of any existing entries.
· Report - Generates reports on the user's spending and income over time. This may include data visualizations and PDFs.
· Get PDF - Provides an option to export the data as a PDF.
· Visualize - Helps users view their expense and income patterns visually (e.g., with charts).

4. Wishlist:
Allows users to create and manage a wishlist for future purchases. This might include:
· Add Product & Price - Users can add items they want to buy along with their prices.
· Edit Items - Permits modifications to the wishlist items.

Relationships Between Use Cases:

· Include Relationship -
Some features are shared across multiple use cases. For example, Sign-up and Login both include Authentication as a required step.

· Extend Relationship -
Some use cases extend others by adding additional functionality. For instance, Forgot Password extends Login as it's a secondary feature.

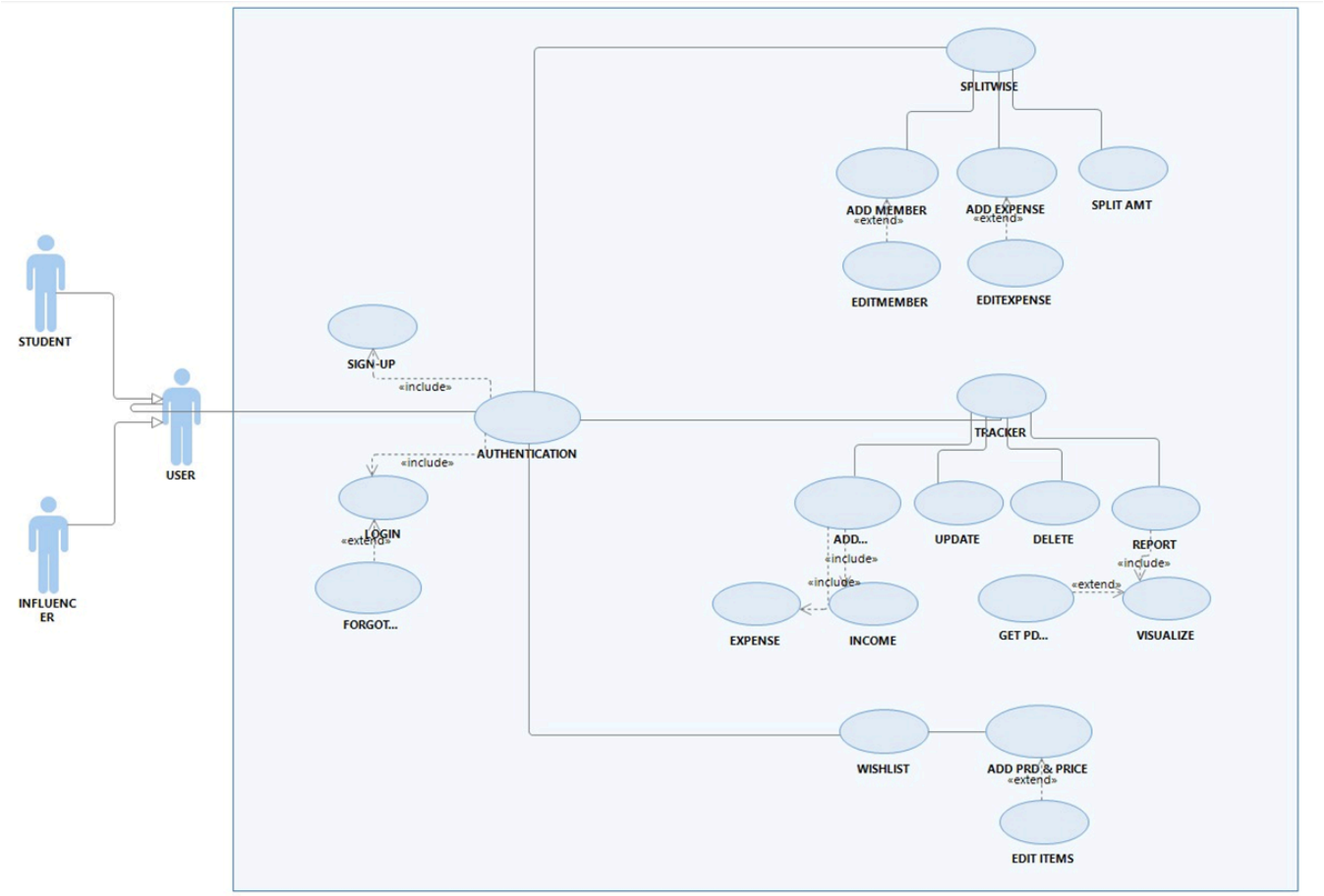


Fig 2- Use Case Diagram

Use Case Template:

1. Adding Transaction

1. Use Case Title	Add Transaction
2. Abbreviated Title	Add Transaction
3. Use Case Id	1
4. Actors	User
5. Description: This use case allows users to add a transaction (income or expense) to the expense tracker application. The transaction includes details such as amount, date, category, and description.	
5.1. Pre-Conditions: The user is authenticated and logged into the application. The user is on the "Add Transaction" page (Homepage).	
5.2.Task Sequence: • The user navigates to the "Add Transaction" page. • The user fills in the transaction details, including amount, date, category, and description. • The user selects the type of transaction (income or expense). • The user submits the form by clicking the "Add Transaction" button.	
5.3.Post-Conditions: • The transaction is successfully stored in the MongoDB database. • The user sees the newly added transaction reflected in the application in the transaction list.	
6. Modification History: 11th September 2024	

Table 1- Use Case for Add Transaction

2. Viewing Expense

1. Use Case Title	View Expenses (Graph/Pie Chart/Download as Excel)
2. Abbreviated Title	View Expenses
3. Use Case Id	2
4. Actors	User
5. Description: This use case allows users to visualize their expenses as a graph or pie chart and allows them to download the expense data in Excel format for offline analysis.	
5.1. Pre-Conditions: The user is authenticated and logged into the application. The user has previously added transactions (income and/or expenses) to the system.	
5.2.Task Sequence: • The user navigates to the "View Expenses" page. • The user selects the type of visualization (e.g., bar graph, pie chart) or chooses to download the data as an Excel file.	
5.3.Post-Conditions: • The user successfully views their expense data as a graph or pie chart. • The user is able to download their expense data as an Excel file, which can be used for offline analysis.	
6. Modification History: 11th September 2024	

Table 2- Use Case for Viewing Expense

3. Splitwise

1. Use Case Title	Split Expenses with Others
2. Abbreviated Title	Split Expenses
3. Use Case Id	3
4. Actors	User
5. Description: This use case allows the user to split expenses with others, enabling the user to manage shared expenses (e.g., splitting bills or purchases with friends or family). The system will keep track of who owes what and allows the user to settle balances.	
5.1. Pre-Conditions: • The user is authenticated and logged into the application. • The user has added transactions to the system. • The participants with whom the expense is split are either registered users of the application or have their contact information added by the user.	
5.2.Task Sequence: • The user navigates to the "Split Expenses" page. • The user selects an expense to split or adds a new expense. • The user chooses the participants with whom the expense will be split by either selecting existing contacts or entering new participant details. • The user decides how to split the expense (equally, by percentage, or by specific amounts). • The user submits the form by clicking the "Split Expense" button.	
5.3.Post-Conditions: • 1. The expense is successfully split between the selected participants and stored in the MongoDB database. • 2. Users and participants can view the split expense details and track who owes what.	
6. Modification History: 11th September 2024	

Table 3- Use Case for Splitwise

4.2 ACTIVITY DIAGRAM

The diagram shows a linear flow, ensuring users input correct data before saving, with options to manage expenses once they are in the system. It provides a straightforward approach to personal expense management with basic error handling and CRUD (Create, Read, Update, Delete) operations.

Steps in the Activity Diagram:

1.Start -
The process begins when a user initiates the expense management function.

2.Enter Username and Password -
The user inputs their login credentials.

3.Check Authentication -
The system verifies the user's credentials. If authentication fails, the user will be denied access (though this specific path isn't shown in the diagram; it's implied as part of security handling).

4.Add Personal Expense -
The user can add a personal expense after authentication.

5.Is the Input Valid? -
The system checks if the entered expense details are valid (e.g., correct format, positive values, etc.).
No (Invalid Input) - If the input is invalid, an error message is displayed to the user, prompting them to correct the information.
Yes (Valid Input) - The system proceeds to the next step if the input is valid.

6.Save Expense -
The valid expense is saved in the system.

7.View/Update/Delete Expense -
The user can view, update, or delete existing expenses.

8.Update or Delete Expense -
The user updates or deletes a specific expense.

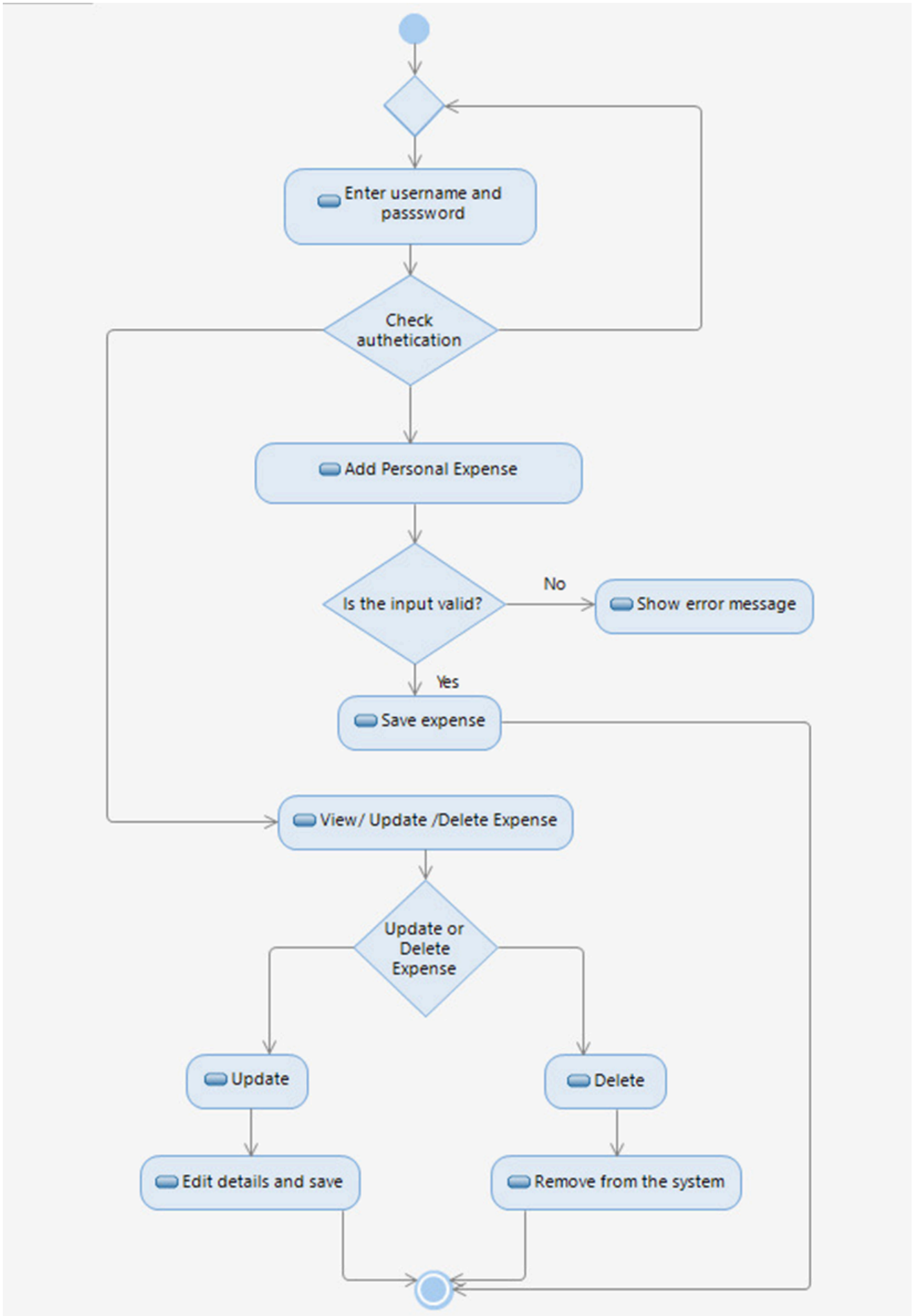


Fig 3- Activity Diagram

4.3 SWIMLANE DIAGRAM

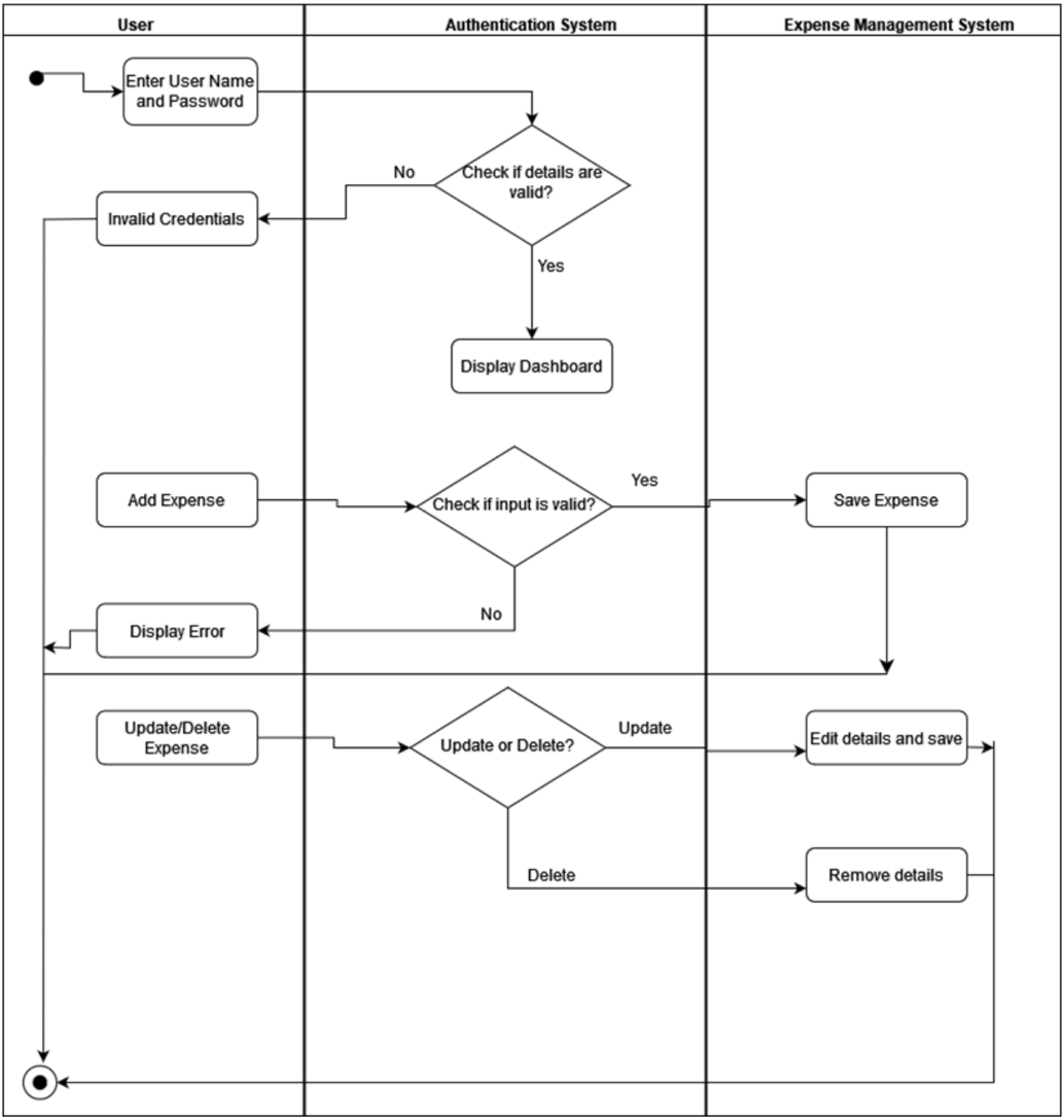


Fig 4- Swimlane Diagram

4.4 CLASS DIAGRAM

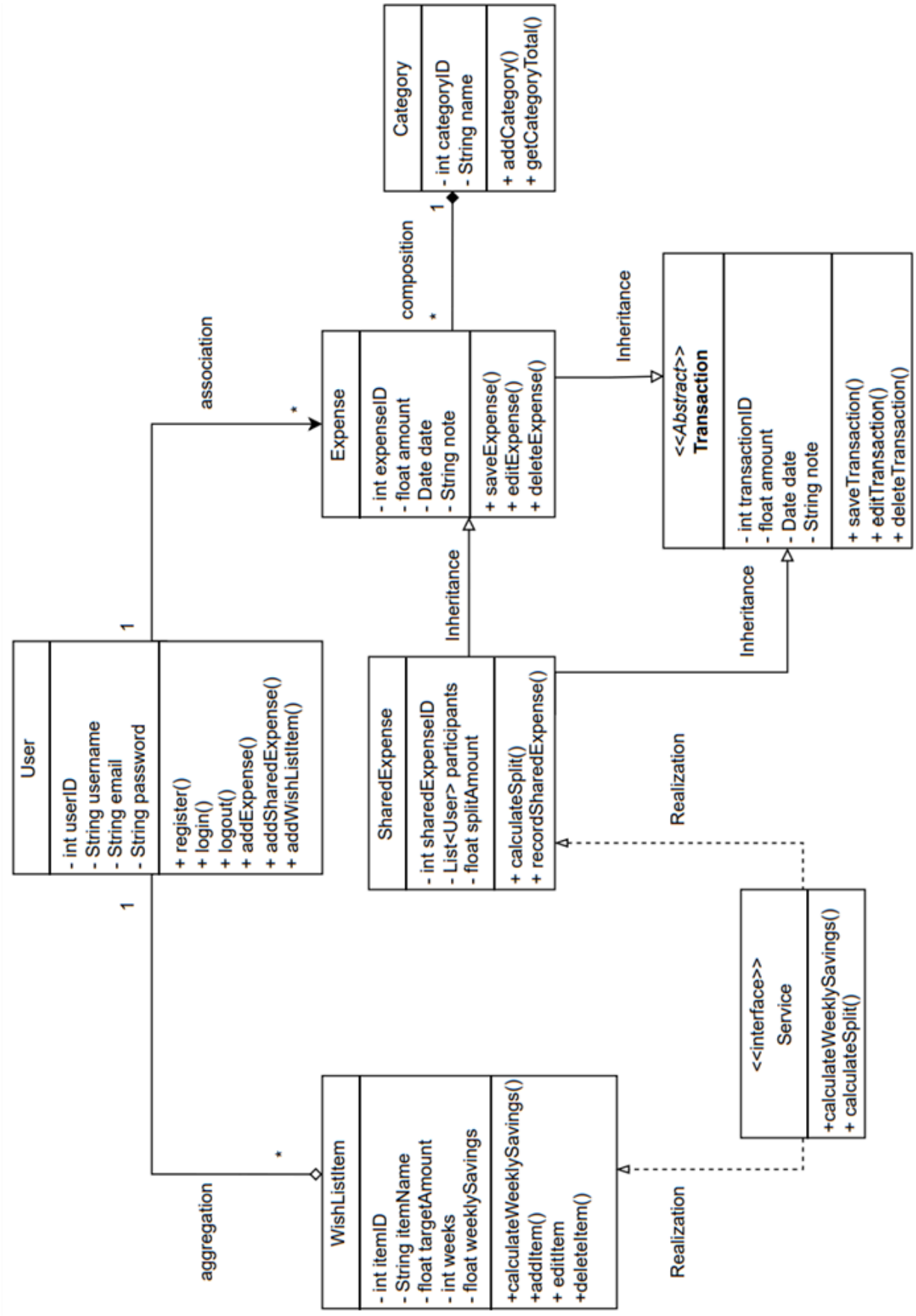


Fig 5- Class Diagram

5.
SOFTWARE REQUIREMENTS SPECIFICATIONS

Software Requirements Specification

for

BroBroke

Expense Tracker
Software Engineering (UCS503)

Version 1.0 approved

Prepared By:

Ravneet Kaur (102203202)
Hardetya Gill (102203213)
Radhika Aggarwal (102203226)
Sara Agnihotri (102203248)

Computer Science Department,
Thapar Institute of Engineering and Technology

25 August 2024

Table of Contents

1. Introduction

1.1 Purpose

This SRS document aims to deliver an in-depth description of our software product, covering its objectives, key specifications, and intended purpose. It specifies the target audience, the design of the user interface, and the required hardware and software configurations. Furthermore, the document provides a clear definition of the product's functionality from the perspectives of all key stakeholders, including the client, development team, and end-users.

1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement Specification Documents.

1.3 Intended Audience and Reading Suggestions

This expense tracker is designed to help students everywhere manage their finances by categorizing spending, setting budgets, and monitoring financial habits. Groups with limited budgets can also track their expenses and ensure funds are allocated according to plan. Anyone looking to improve their financial habits, save for specific goals, or pay off debts can benefit from better tracking and analysis of their spending. Independent workers and small business owners can monitor business expenses, track invoices, and manage budgets more efficiently too.

1.4 Project Scope

The scope of this development project focuses on creating an expense tracker application. This software will allow users to manage and track their daily expenses efficiently. The project encompasses features such as categorizing expenses, setting budgets, generating reports, and offering insights into spending habits. The goal is to deliver an intuitive, user-friendly interface compatible across various devices while ensuring secure data storage and smooth performance. The project also includes providing the necessary support and updates post-launch to ensure continued reliability and user satisfaction.

1.5 References

[1] http://expense-manager.com/how-expensesoftware/
[2] http://code.google.com/p/socialauth-android/wiki/Facebook
[3] http://code.google.com/p/socialauth-android
[4] https://developer.android.com
[5] http://www.appbrain.com/app/expensemanager/com.expensemanager
[6] https://www.xpenditure.com/en?
[7] http://expense-manager.com/how-expensesoftware/
[8] Donn Felker, "Android Application Developmentfor Dummies", published by For Dummies, 2010.
[9] Ed Burnette, "Hello, Android: Google's Mobile Development", by Pragmatic Bookshelf
[10] Lee, "Beginning Android Application Development", Published by WroxPress, 2011.

2. Product Description

2.1 Product Perspective

BroBroke can be accessed from any web browser, for a portable work environment. This web application usually is developed using React Js as the framework and uses its libraries like material UI to add and create the functionalities. ReactJS is a declarative, efficient, and flexible JavaScript frontend library for building reusable UI components. It uses virtual DOM (JavaScript object), which enhances the efficiency and performance of the app. The JavaScript virtual DOM is faster than the conventional DOM. We can use ReactJS on both client and server-side as well as with other available frameworks.

This software is an extrapolation and exhibits additional advanced features of already existing expense tracker applications like Splitwise. Here, the interactive interface helps us analyze the input of income and expense, which is taken separately, and further helps in better categorizing and budgeting.

Our income and expense tracker software acts as a helpful companion for managing your finances, whether you're a student, a worker, or part of a family. It allows you to set spending limits across various categories to help you save money more effectively. The software features vibrant charts and graphs that visually represent your spending patterns, making it easier to understand where your money goes. Plus, with our commitment to safeguarding your financial information, you can trust that your data remains private and secure. With our software, managing your money becomes straightforward and stress-free.

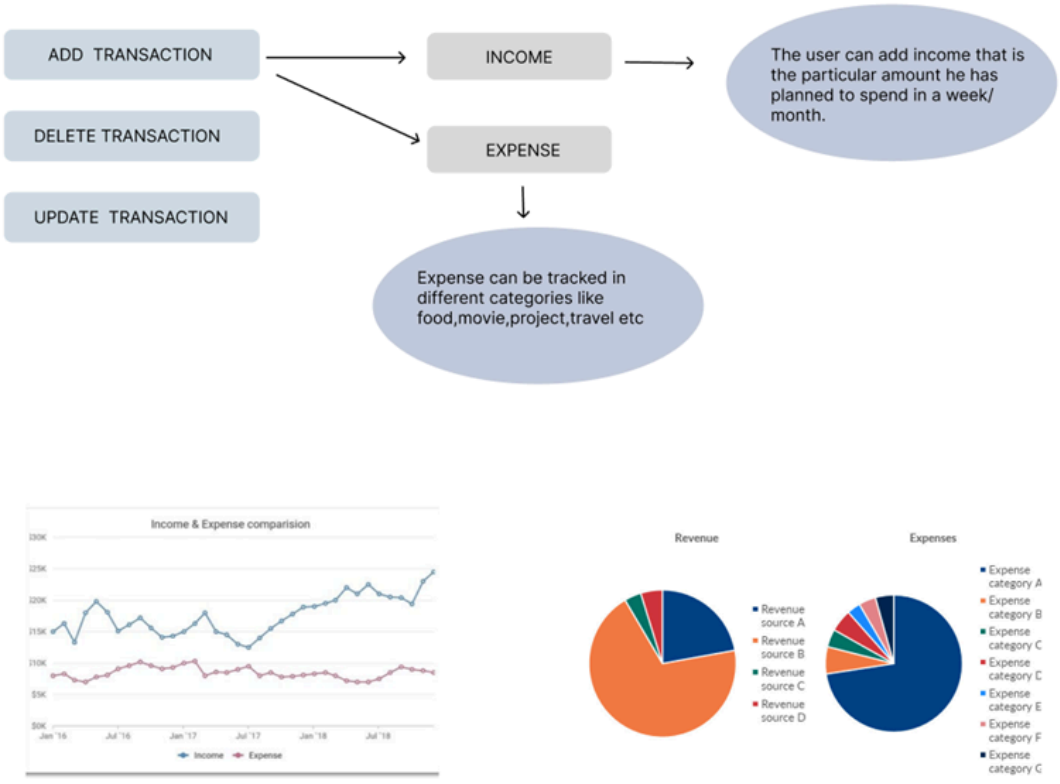


Fig 6- Basic Template of BroBroke, displaying transaction functionality and visualization of the expenses

2.2 Product Features

A. Login and Registration Module

The Login and Registration Module provides a secure entry point for users to access their personal expense management accounts. New users can create an account by registering with their email address, choosing a strong password, and verifying their identity through a confirmation link or code. Existing users can log in using their credentials. The module ensures a seamless onboarding process while maintaining security through encrypted password storage and session management.

B. Bill Split Module

The Bill Split Module facilitates the division of shared expenses among multiple users. Whether for group activities, family expenses, or shared bills, users can easily split costs by specifying the total amount and how it should be divided (equally, by percentage, or by specific amounts). The module calculates each person's share and tracks payments, ensuring transparency and accuracy in managing collective finances.

C. Budget Organizer Module

The Budget Organizer Module helps users plan and manage their financial goals by creating and organizing budgets across various categories. Users can set monthly or weekly budgets for different spending categories such as groceries, entertainment, or transportation. The module provides visual representations of budget usage, allowing users to monitor their spending against set limits, make adjustments, and stay on track with their financial goals.

D. Categorical Classification Module

The Categorical Classification Module automatically categorizes expenses into predefined categories like food, travel, utilities, and more. Users can also create custom categories to better suit their unique spending habits. By tagging each expense with a category, the module provides detailed insights into spending patterns, helping users understand where their money is going and identify areas for potential savings.

E. Authentication, Security and Password Hashing

The Authentication and Security feature ensures that user data is protected through robust security measures. This includes secure login processes, data encryption, and password hashing. Passwords are hashed and salted to protect against unauthorized access, while encryption safeguards sensitive user information during transmission and storage. The system also supports two-factor authentication (2FA) for an additional layer of security, ensuring that only authorized users can access their accounts.

F. Adding, Updating, and Deleting Expense and income

The Adding, Updating, and Deleting Expenses feature provides users with flexible control over their expense records. Users can easily add new expenses by entering details such as amount, date, and category. Existing expenses can be updated to reflect changes or corrections, while obsolete or incorrect expenses can be deleted. This module ensures that users can maintain accurate and up-to-date records of their spending, contributing to effective financial management and tracking.

G. Pictorial Representation of Expenses Module

The Pictorial Representation of Expenses Module offers users visual insights into their spending through interactive pie charts and graphs. This module generates graphical representations of expenses by category, time period, or budget, providing a clear and intuitive way to understand spending patterns. Users can view pie charts to see how expenses are distributed across different categories, and use bar or line graphs to track spending trends over time. These visual tools help users quickly grasp their financial situation and make informed decisions based on their spending habits.

H. Comprehensive Report Generation and Export Module

The Comprehensive Report Generation and Export Module enables users to create detailed financial reports and export them in PDF format. This module allows users to generate customized reports based on various criteria, such as date ranges, expense categories, and budget comparisons. Users can select the specific data they want to include in the report, such as total expenses, budget adherence, and spending trends. Once generated, reports can be easily downloaded as PDF files, providing users with a professional, shareable format for personal review or presentation. This feature is particularly useful for end-of-year financial reviews, tax preparation, and sharing financial summaries with family or financial advisors.

2.3 User Classes and Characteristics

We can broadly categorize our software's user classes based on their specific characteristics and requirements under the following labels:

I. Students: The expense tracker is designed to help students manage their finances by categorizing spending, setting budgets, and monitoring financial habits.
II. School Clubs & College Society: Groups with limited budgets can track their expenses and ensure funds are allocated according to plan.
III. NGOs: With already limited funds and resources, an expense tracker helps NGOs manage their finances more effectively, ensuring that more resources are directed towards their core mission.
IV. Budget-Conscious Individuals: Anyone looking to improve their financial habits, save for specific goals, or pay off debts can benefit from better tracking and analysis of their spending.

2.4 Operating Environment

- Windows 2000, Windows XP, Windows Vista
- Windows 7, 8, 10
- Mac OS X
- Linux

2.5 Design and Implementation Constraints

A. Platform Compatibility:

- Cross-Platform Support: The application might need to be compatible across different operating systems (e.g., Windows, macOS, Android, iOS) which may require using frameworks like React Native or Flutter.
- Web vs. Mobile: The design might need to cater to both web and mobile platforms, which could lead to responsive design challenges and require different implementation approaches.

B. Data Security and Privacy:

- Encryption: Sensitive user data such as financial transactions must be encrypted both at rest and in transit.
- User Authentication: Secure login mechanisms, such as OAuth or two-factor authentication, should be implemented to protect user accounts.

C. Scalability:

- Database Management: As the number of users grows, the database must efficiently handle increased data storage and retrieval demands.
- Backend Infrastructure: The application should be designed to scale horizontally or vertically to accommodate more users without degrading performance.

D. User Interface (UI) and Experience (UX):

- Simplicity: The UI must be intuitive and easy to navigate, as users may want to quickly input and track expenses.
- Accessibility: The app should be accessible to users with disabilities, which may require adhering to guidelines like WCAG.

E. Performance:

- Speed: The application should be optimized to ensure quick loading times and smooth user interactions, especially on lower-end devices.
- Resource Efficiency: The app should efficiently manage resources like CPU, memory, and battery usage, particularly on mobile devices.

F. Maintenance and Updates:

- Continuous Integration/Continuous Deployment (CI/CD): Implementing a CI/CD pipeline to facilitate regular updates and bug fixes can be challenging.
- Backward Compatibility: Ensuring that updates do not break existing functionality, especially for users on older versions of the app, requires thorough testing and version control.

G. Budget Constraints:

- Development Costs: The budget might limit the choice of technology stack, tools, or third-party services that can be used in the project. None in case of our project
- Maintenance Costs: Ongoing costs for hosting, database management, and third-party API usage must be considered within the project's financial constraints. None in case of our project.

2.6 User Documentation

Our software is designed with simplicity and ease of use in mind, ensuring that it is highly intuitive and user-friendly. With its straightforward interface and clear navigation, new users can seamlessly get started without the need for tutorials or extensive training. Its self-explanatory features and guided design make it accessible and efficient, allowing users to quickly grasp its functionality and integrate it into their daily routine with minimal effort.

2.7 Assumptions and Dependencies

There are no viable dependencies and/or assumptions taken into account for this software, as we are not using any external projects. All our features are user-operated, as it is an interactive user interface.

3. System Features

This project idea combines expense tracking with shared expense management, offering a robust tool for users to control their finances and collaborate on managing joint expenses. The goal is to develop a sophisticated expense tracking application that not only manages individual expenses but also integrates with Splitwise for shared expenses. This application will provide users with detailed insights into their spending habits, allow for collaborative expense management, and help users allocate funds towards specific financial goals.

3.1 System Feature 1

A. Weekly, Monthly, Yearly Expense Tracking:
- Track and log expenses on a weekly and monthly basis.
- Provide summary reports showing total expenditure and income.

B. Detailed Expense Analysis:
- Generate overall spending analysis for the week and highlight the days with the highest expenditure.
- Use color-coded indicators (green for under budget, red for over budget) to visualize spending trends, like trading charts.

C. Splitwise Integration:
- Seamlessly integrate with Splitwise to manage and track shared expenses among friends, family, or roommates.
- Automatically synchronize and update shared expenses and balances.

D. Bucket List Budgeting:
- Allow users to allocate funds towards bucket list items or specific financial goals.
- Track progress towards these goals and adjust budgets as needed.

E. Visual Spending Insights:
- Display spending data in pie charts to show the proportion of money spent across different categories.
- Include graphs to visualize trends and identify areas where spending is highest.

F. Categorization and Labeling:
- Categorize expenses with customizable labels (e.g., groceries, entertainment, transportation).
- Provide options to add, edit, or remove categories to tailor the tracker to individual needs.

G. Budget Visualization:
- Use color-coded indicators to show budget status.
- Implement a “green red” tracker similar to trading charts. Green indicates spending within or under the budget, while red signals overspending. This helps users quickly assess their financial status.

H. Pocket Money Distribution:
- Manage and allocate pocket money for various needs.
- Allow users to distribute pocket money into different categories or purposes (e.g., savings, leisure, necessities). Provide an overview of how this money is being used and adjust distributions as needed.

3.2 System Feature 2

A. User Interface (UI):
Develop an intuitive and user-friendly interface that makes it easy for users to navigate and interact with their financial data. Use clear visual elements to enhance user experience and accessibility.

B. Data Security:
Implement strong security measures to protect user data. Ensure encryption of sensitive information and provide options for secure authentication and data backup.

C. Cross-Platform Compatibility:
Ensure the application is accessible across various devices, including mobile phones, tablets, and desktops. Consider responsive design to provide a consistent experience on different screen sizes.

D. User Customization:
Allow users to personalize the application to fit their specific needs. This includes customizable budgets, categories, and alert settings.

E. Reporting and Exporting:
Provide options to generate and export financial reports in various formats (e.g., PDF, CSV). This feature is useful for users who need to review their finances in detail or share information with others.

F. Integration with Other Financial Tools:
Consider integrating with other financial tools or services (e.g., banking APIs, investment tracking) for a more comprehensive financial management experience.

Table 1:
Functional requirements for a person using BroBroke as an EXPENSE TRACKER

Purpose	Record all financial transactions, including purchases, bills, and other expenses.
Inputs	• Financial transactions: Data about purchases, bills, and other expenses, typically entered manually. • Budget information: Details about income, savings goals, and spending limits for different categories.
Processing	• Identify spending patterns: Analyze spending habits to uncover areas where money might be wasted or saved. • Set and manage budgets: Create budgets for different categories (e.g., housing, food, transportation) and monitor progress towards financial goals.
Outputs	• Expense reports: Summarized information about spending patterns, categorized by category or time period. • Visualizations: Charts, graphs, and other visual representations of financial data to aid understanding and decision-making.

Table 2:
Functional requirements for a person using BroBroke for making a WISHLIST

Purpose	The user wants to save money over time to purchase specific items on their wish list.
Inputs	• The user adds items to their wish list, specifying the desired item, its cost, and any savings goals. The user also tracks daily expenses and monitors how much money can be saved daily.
Processing	• The app calculates how much money is saved daily after expenses, tracks progress towards the wish list item, and suggests adjustments to spending or savings based on progress.
Outputs	• The user receives updates on their savings progress, notifications when they've reached their savings goal, and encouragement to save or adjust spending to achieve their wish list goals.

Table 3:
Functional requirements for a person using BroBroke as SPLITWISE

Purpose	The user wants to manage shared expenses with friends, family, or roommates, ensuring everyone pays their fair share.
Inputs	• The user inputs expenses that are shared with others, including details like the amount, participants, and who paid. spending limits for different categories.
Processing	• The app calculates how much each person owes or is owed based on the shared expenses, splits the total fairly among participants, and tracks balances.
Outputs	• The user receives a clear summary of who owes what to whom, along with notifications or reminders to settle balances.

4. External Interface Requirements

4.1 User Interface

Design and Usability-
Accessibility: Ensure the UI is accessible and user-friendly, with intuitive navigation & clear visual elements.
Responsiveness: The application must adapt to various screen sizes and devices (mobile, tablet, desktop).
Data Input-
Expense Entry: This allows users to enter expenses quickly with fields for amount, category, date, and notes.
Categorization: Include dropdown menus or tags for expense categories and labels.
Visual Feedback-
Charts and Graphs: Display spending data using pie charts, bar graphs, and line charts.
Alerts: Provide clear visual and/or auditory alerts for budget thresholds and financial notifications.

Web Interface-
Design: Responsive design supporting modern web browsers (Chrome, Firefox, Safari, Edge). Adapt for various screen sizes (desktops, laptops, tablets).
Features: Expense entry forms, dashboard with charts and graphs, budget tracking, alerts, and notifications.
Accessibility: Ensure high contrast, keyboard navigation, and screen reader compatibility.

Mobile Interface-
Design: Native or responsive design for iOS and Android devices. Optimize for touch interactions and varying screen resolutions.
Features: Expense logging, budget views, notifications, and visual reports accessible on the go.

Desktop Application Interface-
Design: Compatible with primary desktop operating systems (Windows, macOS). Ensure a consistent user experience across different platforms.
Features: Expense management, detailed reports, and budget analysis.

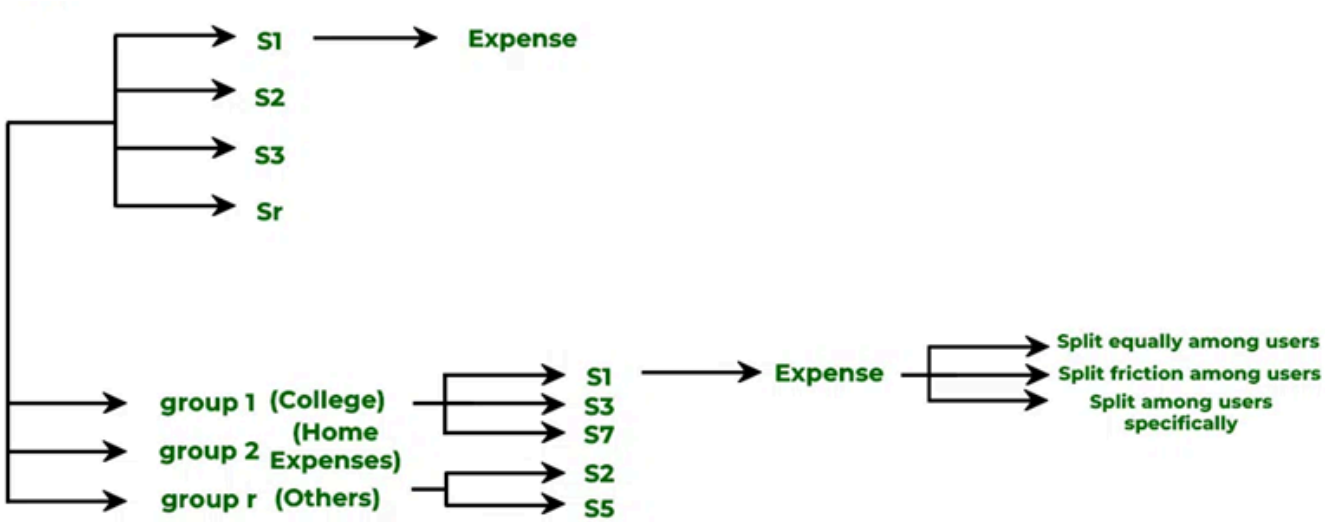


Fig 7- Flow diagram of the Splitwise functionality

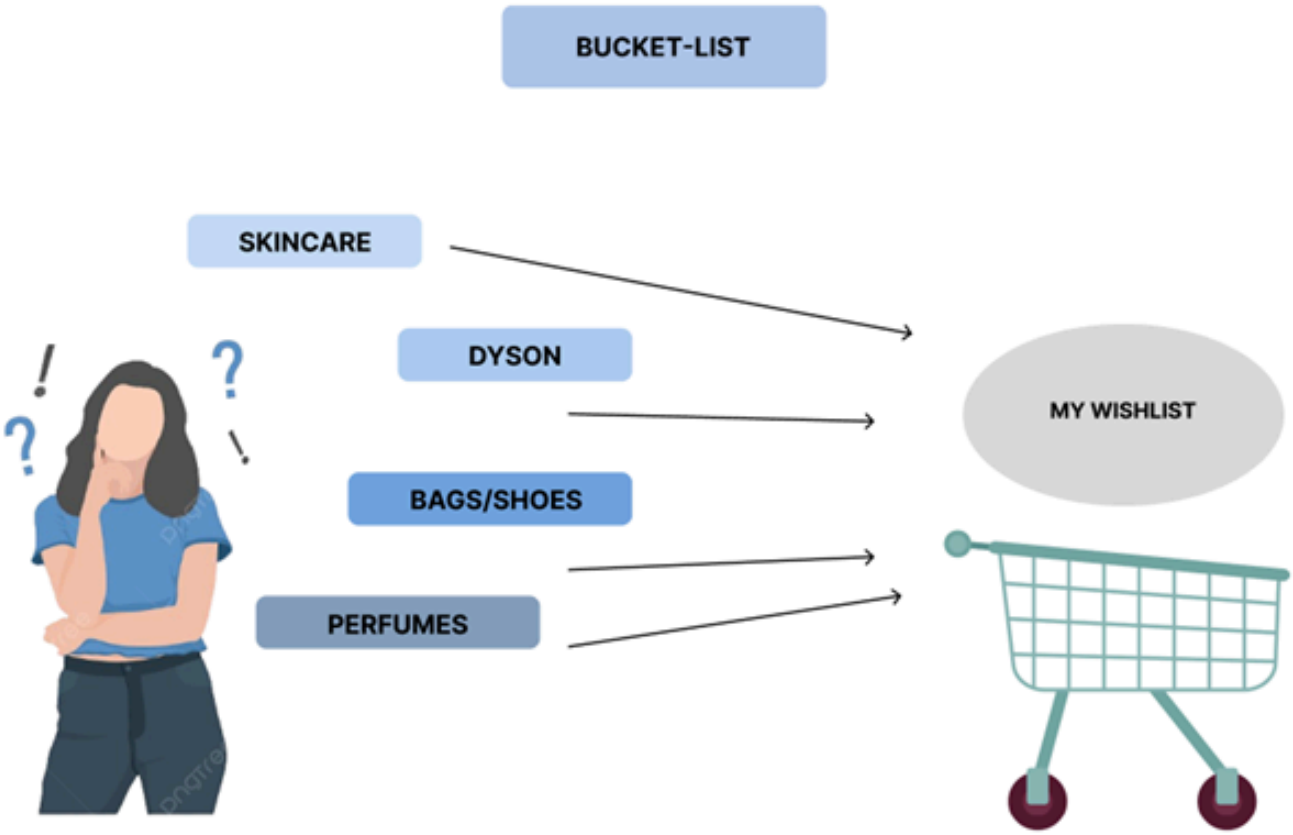


Fig 8- How a wish list functionality would look like

4.2 Hardware Interfaces

Mobile Devices-
Compatibility: Support for Android smartphones and tablets, iPhones, and iPads. Ensure functionality across various models and screen sizes.
Sensors: Utilize mobile sensors if needed (e.g., camera for receipt scanning).

Desktops and Laptops-
Compatibility: Ensure compatibility with common hardware configurations (e.g., various CPU and RAM configurations). Support for external input devices such as keyboards and mice.

4.3 Software Interfaces

Database Systems-
Requirements: Integration with relational databases (e.g., MongoDB) for secure user data storage.
Backup: Implement backup mechanisms to prevent data loss.

APIs-
Splitwise API: Integration for managing and syncing shared expenses. Use secure API keys and OAuth for authentication.

4.4 Communications Interfaces

Network Protocols-
HTTP/HTTPS: Use HTTPS for secure data transmission between client and server.

APIs for External Services-
Security: Ensure secure API communication through OAuth and encryption.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Response Time: The website should load within 2 seconds for 95% of the users under normal load conditions.
Data Processing: The system should process and categorize expenses instantly with submission.
Availability: The system should have an uptime of 99.9% annually.

5.2 Safety Requirements

Data Backup: Daily backups of all financial data should be maintained and stored securely to prevent loss in case of system failures.
User Safety: The system should ensure that any sensitive financial data is exposed through any user interface.

5.3 Security Requirements

Access Control: Only authorized users should have access to specific features and data.
Encryption: All data, both in transit and at rest, should be encrypted using industry-standard encryption protocols.
Authentication: Users must authenticate using a secure method before accessing their accounts.

5.4 Software Quality Attributes

Usability: The interface should be user-friendly, with intuitive navigation and clear instructions for all features.
Portability: The website should be compatible with all major web browsers and should adapt to various screen sizes, including mobile devices.
Maintainability: The codebase should be well-documented, modular, and easy to update or modify as needed.

6. Other Requirements

Appendix A: Glossary

1. API (Application Programming Interface): A set of protocols and tools for building and interacting with software applications. It allows the frontend and backend of the application to communicate.
2. Authentication: The process of verifying the identity of a user attempting to access the system.
3. Authorization: The process of determining if a user has permission to perform a particular action within the system.
4. Backend: The server-side part of the application that handles the business logic, database interactions, and API calls.
5. Budget: A financial plan for managing expenses over a specified period.
6. Category: A classification of expenses (e.g., Food, Rent, Entertainment) that allows users to organize their spending.
7. CRUD Operations: An acronym for Create, Read, Update, Delete operations, which are the basic operations of persistent storage in a database.
8. Dashboard: A visual interface within the application that displays key financial metrics, such as total expenses, income, and budget overviews.
9. Database: A structured collection of data, usually stored electronically in a system like MongoDB, used to store user data, transactions, and other application-specific data.
10. Encryption: The process of converting information or data into a code to prevent unauthorized access.
11. Expense: A financial transaction that reduces the available balance of a user's account, typically representing money spent.
12. Frontend: The client-side part of the application that users interact with, built using technologies like React in this case.
13. Income: Money received by a user, usually used to track against expenses to calculate available balance.
14. MongoDB: A NoSQL database used for storing application data, such as user details, expense records, and budgets.
15. Middleware: Software that acts as a bridge between the frontend and backend, often handling authentication, logging, and other application processes.
16. Recurring Expense: An expense that occurs on a regular basis (e.g., monthly rent or subscription fees).
17. Transaction: A record of an exchange or transfer of money, including details such as the amount, date, and category.
18. UI (User Interface): The part of the application that users interact with directly, typically involving buttons, forms, and other elements.
19. UX (User Experience): The overall experience a user has while interacting with the application, encompassing design, usability, and performance.
20. Session: A period of time where a user interacts with the system, which may include actions like logging in and out, adding transactions, and viewing reports.

Appendix B: Analysis Models

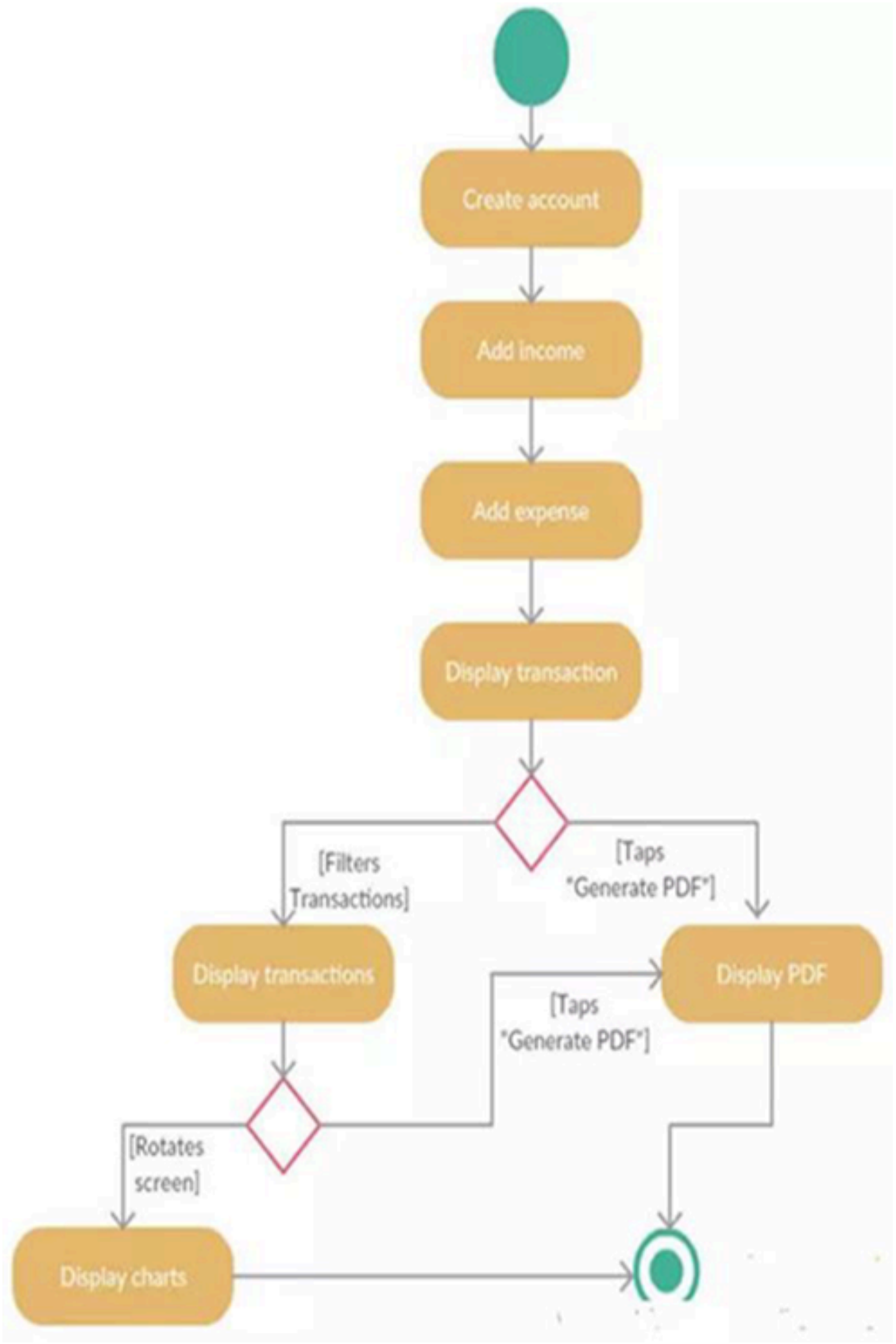


Fig 9- Flow Diagram of how BroBroke functions

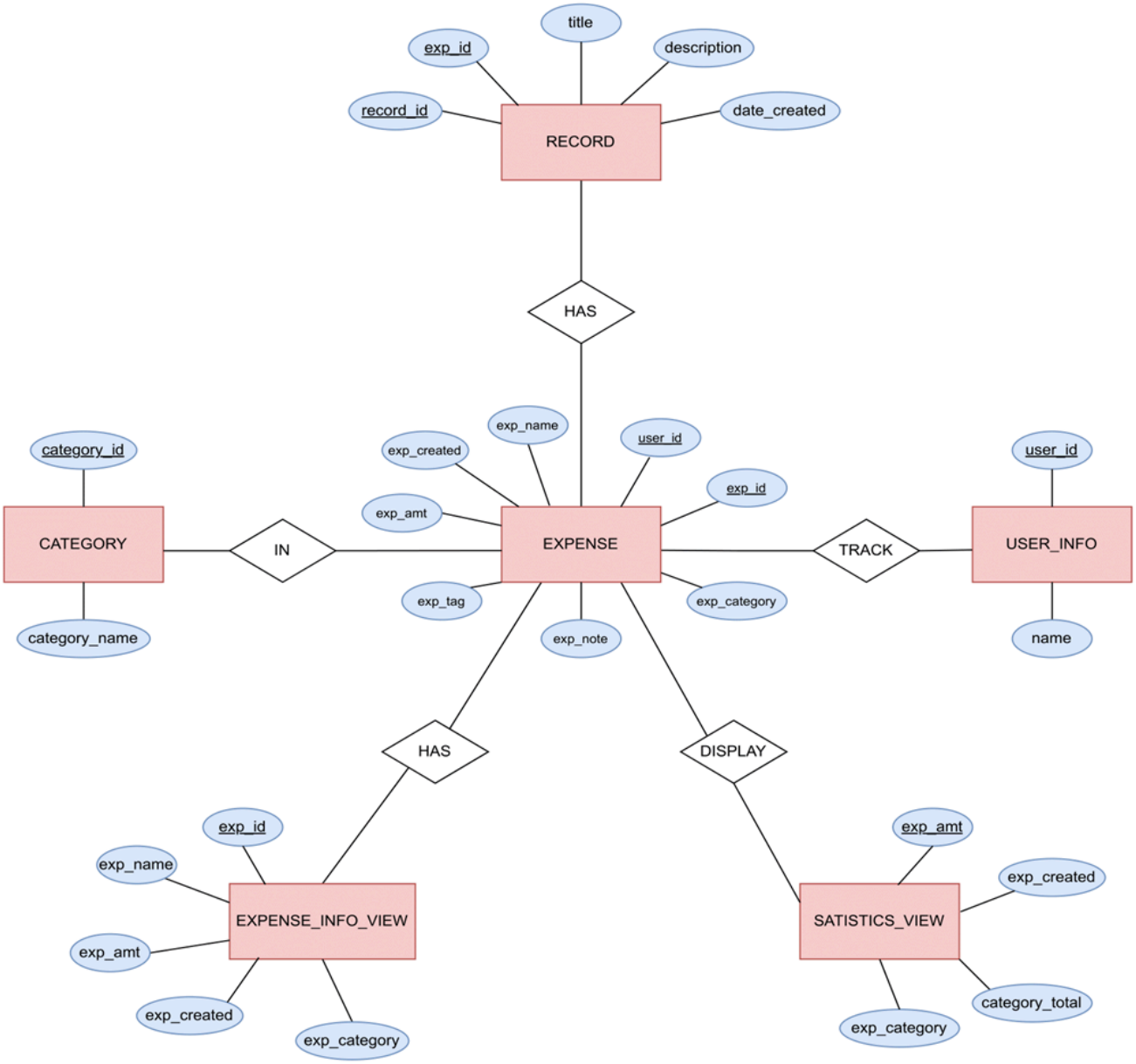


Fig 10- E-R Diagram (1)

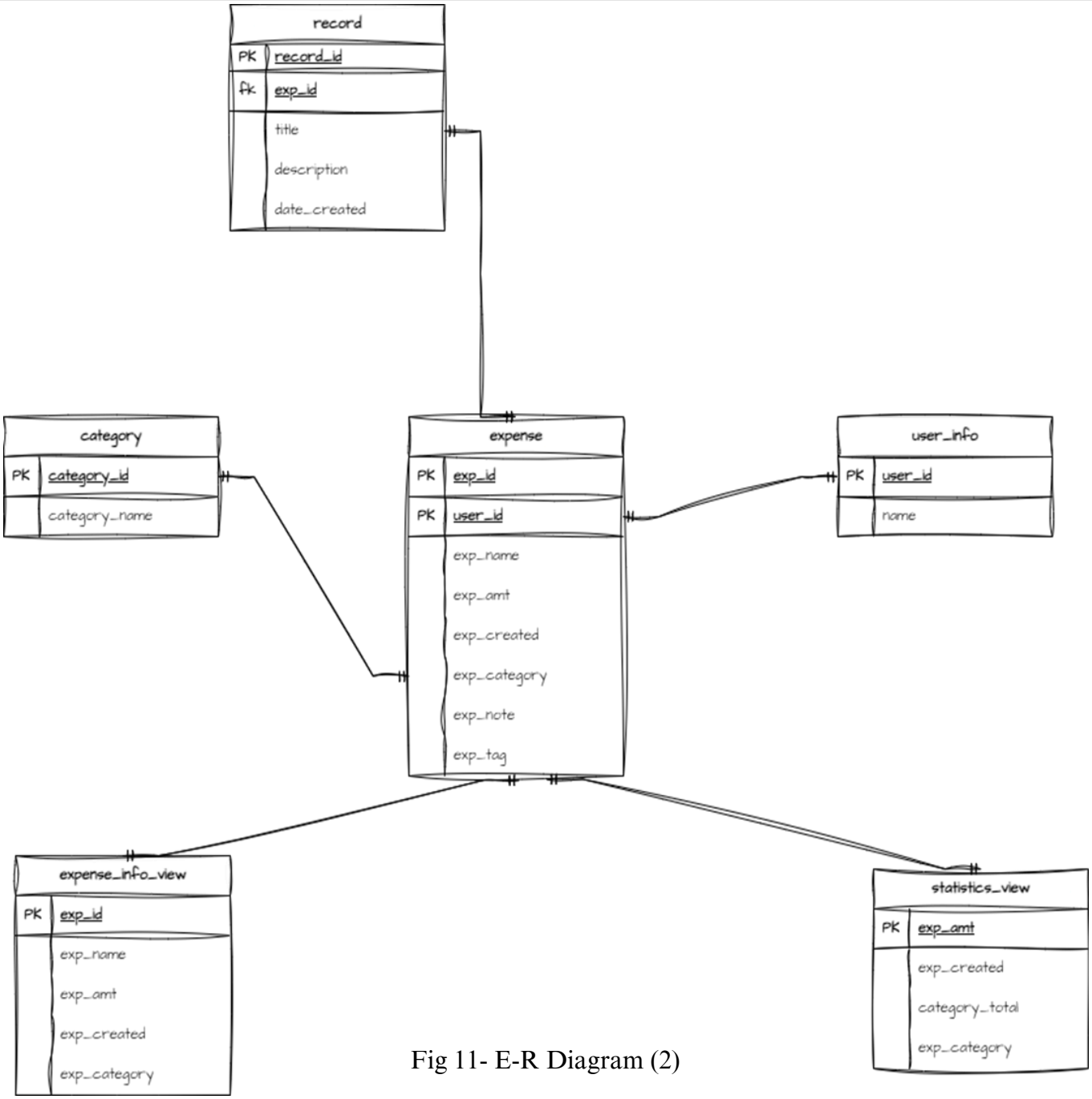


Fig 11- E-R Diagram (2)

Appendix C: Issues List

- Authentication and Authorization Issues: login and signup processes may fail or be insecure.
- Data Validation Issues: Incorrect or incomplete data input may lead to errors in transactions or incorrect expense tracking.
- API Route Issues: API endpoints may return incorrect data, fail to authenticate, or incorrectly handle errors.
- User Interface (UI) Responsiveness Issue: The UI may not be responsive on all devices, leading to a poor user experience.
- Deployment and Environment Configuration Issues: Inconsistent environment setups may cause deployment failures or runtime issues.

6.
DESIGN PHASE

6.1 Sequence Diagram

1. Login/Signup Process:

· Request Login/Signup: The User initiates the process by submitting login or signup details (username and password).

· Check Credentials: The System checks these credentials against the data stored in the Database.

· Return Validation: Based on the result (valid or invalid), the Database returns a validation response.

· Display Success/Failure: The System displays a success or failure message based on the validation result, granting or denying access.

2. Add Expense Workflow:

· Open the "Add Expense" Page: After a successful login, the User navigates to the page where they can add expenses.

· Display Expense Form: The System presents fields like Category, Amount, Notes, and Date for the User to fill in.

· Submit Expense Details: Once the User fills out the form, they submit the expense details.

· Store Expense: The system sends these details to the database for storage.

· Confirm Expense Storage: The Database confirms that the expense has been saved successfully.

· Calculate Updated Spending by Category: The System then calculates the updated spending for each category.

· Display Updated Summary: Finally, the System displays an updated summary of expenses by category to the User.

3. Add Shared Expense:

· Add Shared Expense Details: The User can add a shared expense by entering Amount, Participants, and other relevant details.

· Store Shared Expense: The System sends this information to the Database to save the shared expense.

· Confirm Storage and Display Split Amount: The Database confirms storage, and the System calculates and displays the split amount per participant.

4. Add Wishlist Item:

· Enter Wishlist Item Details: The User can add a wishlist item with Item Name, Target Amount, and Weeks to save up.

· Store Wishlist Item: This data is stored in the Database.

· Calculate Weekly Savings: The System calculates the weekly amount needed to reach the target based on the number of weeks.

· Display Weekly Savings: The System displays the calculated weekly savings required to meet the target.

5. Logout:

· Session Clear: The User can log out, clearing the session data, which ends their active session on the system.

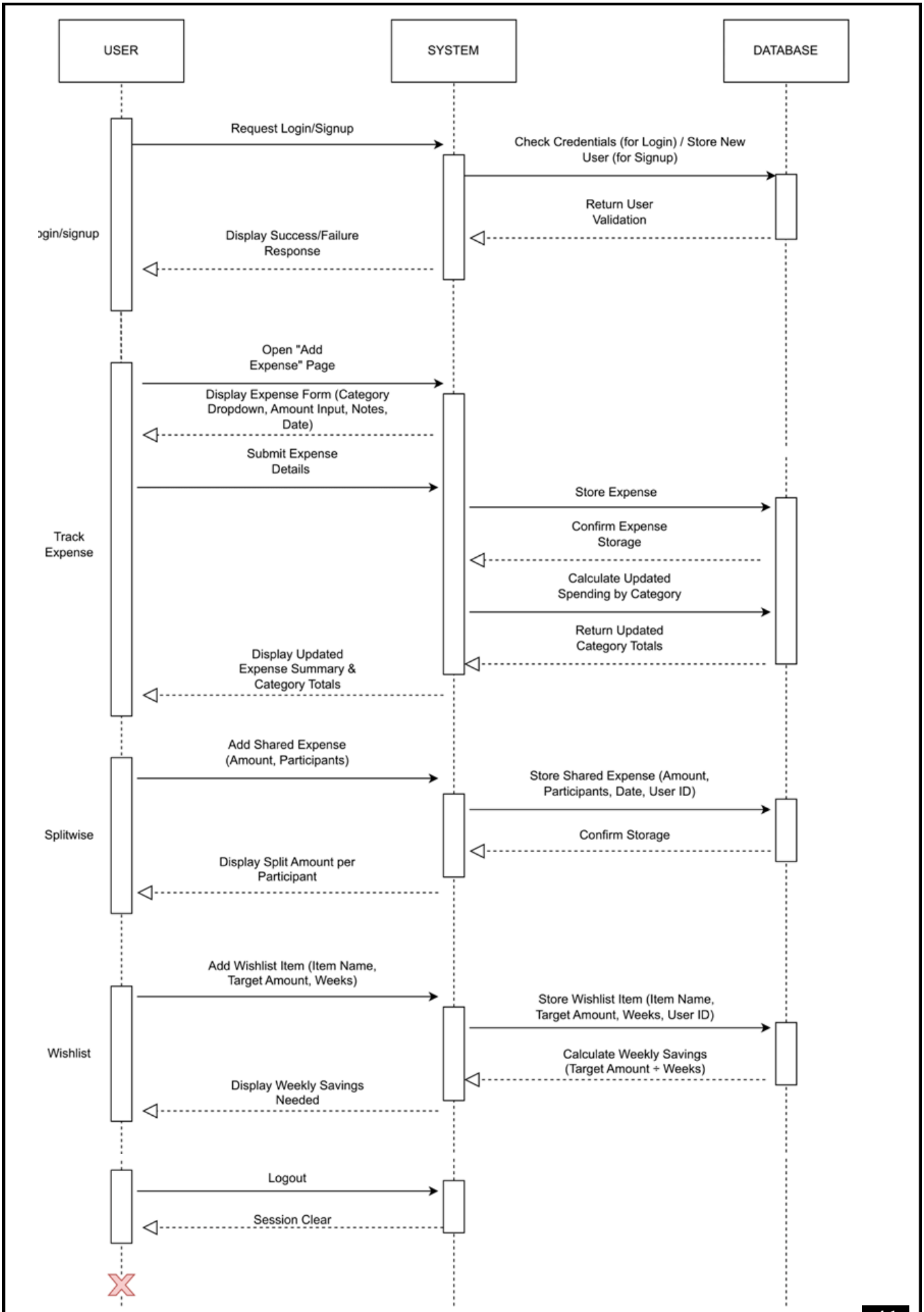


Fig 12- Sequence Diagram

6.2 Collaboration Diagram

This diagram illustrates the process flow of an expense tracker application, showing how the user, system, and database interact to manage user authentication, expense logging, shared expenses, wishlist items, and logout functionality. Each step represents a request from the user, processed by the system and stored in the database, resulting in real-time updates and guidance for effective expense tracking and savings.

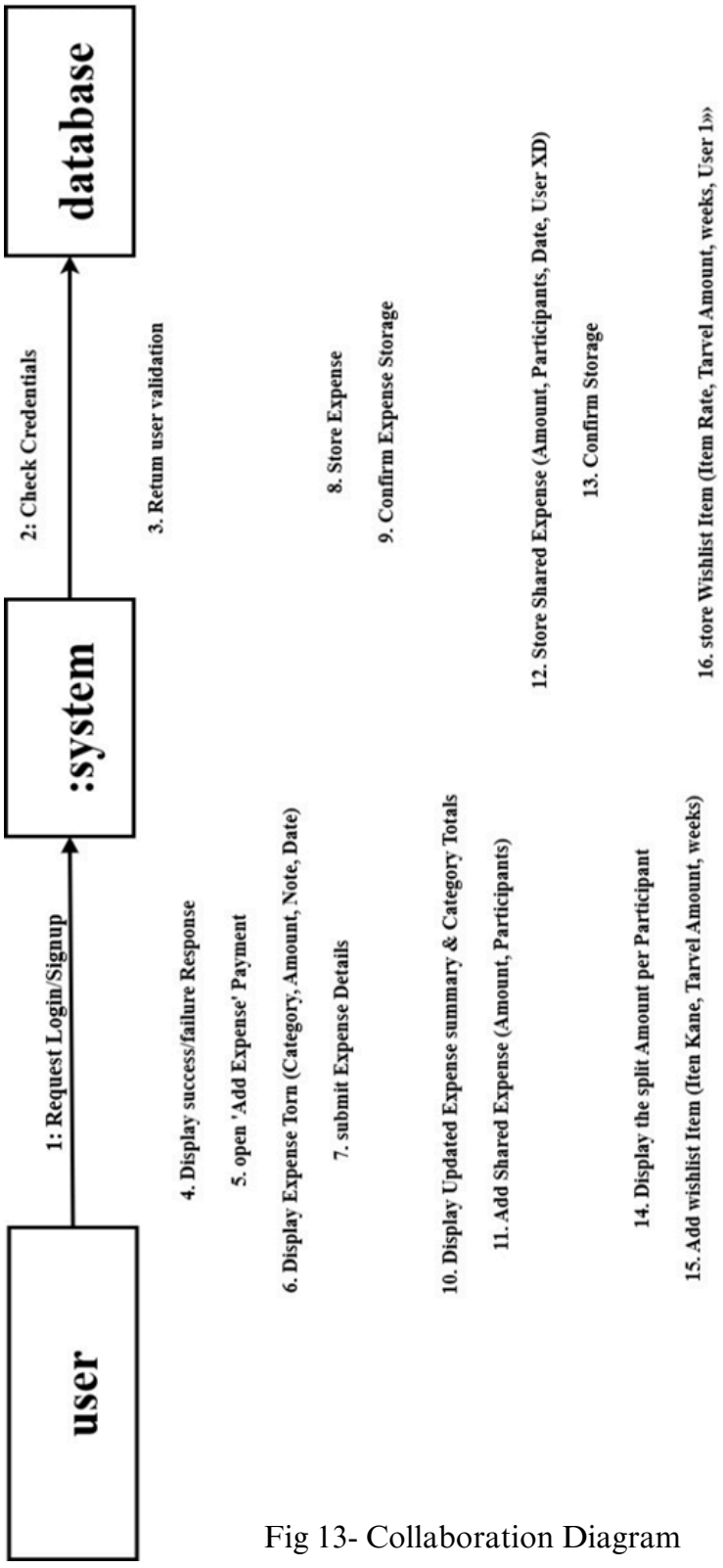


Fig 13- Collaboration Diagram

6.3 State Chart Diagram

This state chart diagram illustrates how the application manages various functional states related to user actions. Each primary function (adding expenses, shared expenses, wishlist management) has its substrates for processing and saving data. The diagram shows the flow between these states, from user authentication to expense management, shared expense calculations, and wishlist savings plan creation. After each operation, the system typically returns to the "View Summary" state, allowing users to continue managing their expenses and budget.

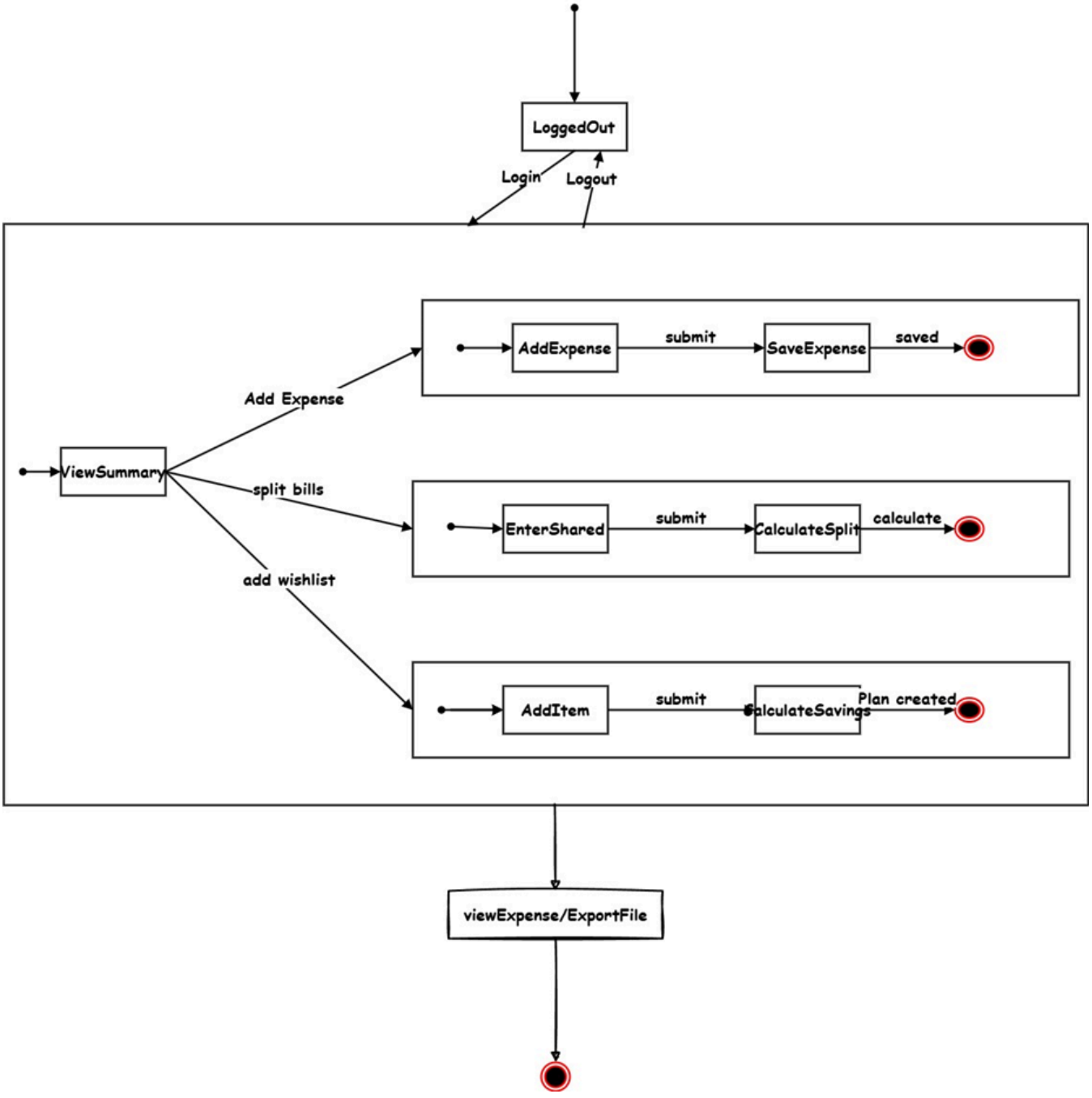


Fig 14- State Chart Diagram

7.
IMPLEMENTATION PHASE

7.1 Component Diagram

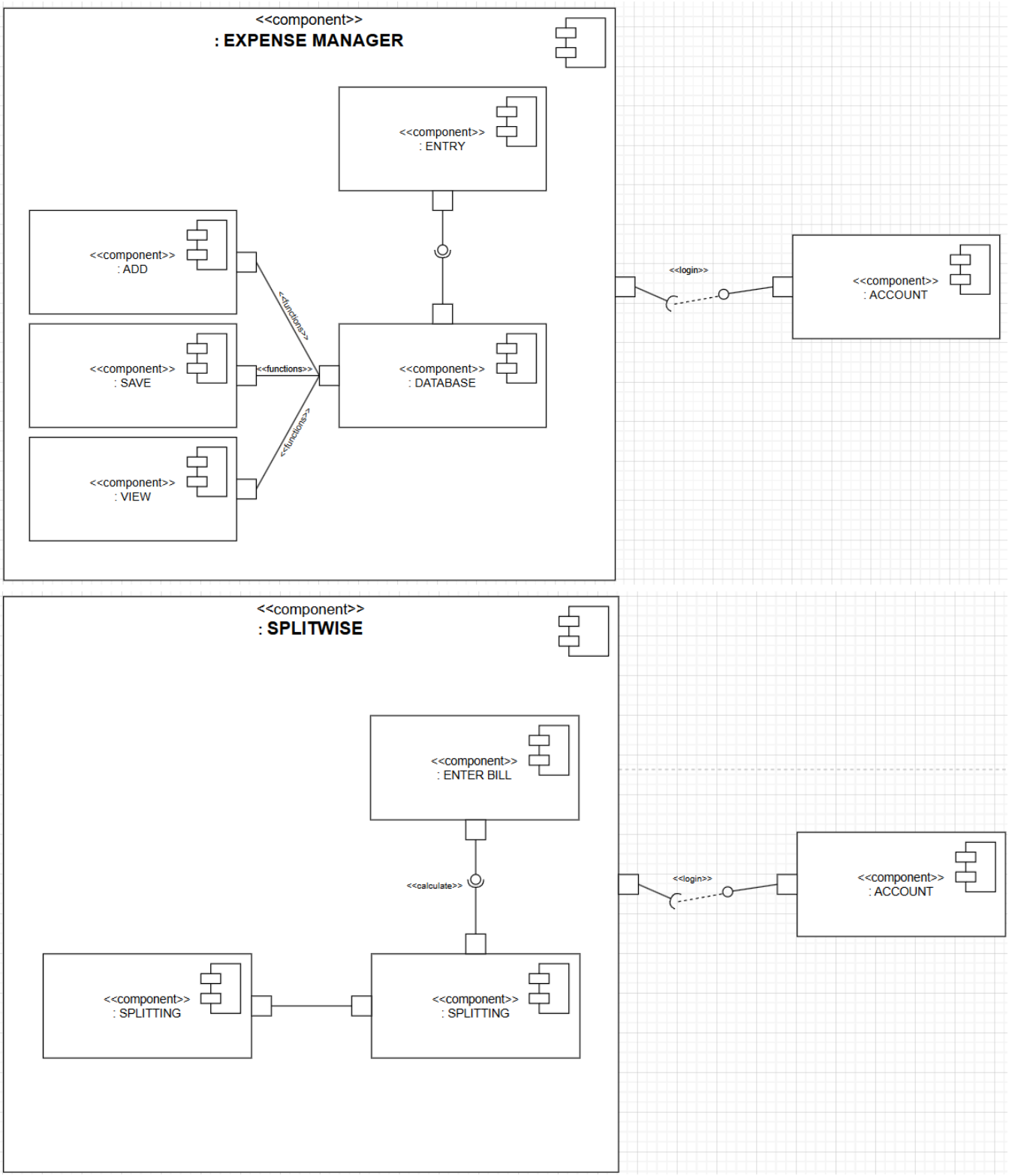


Fig 15- Component Diagram

7.2 Deployment Diagram

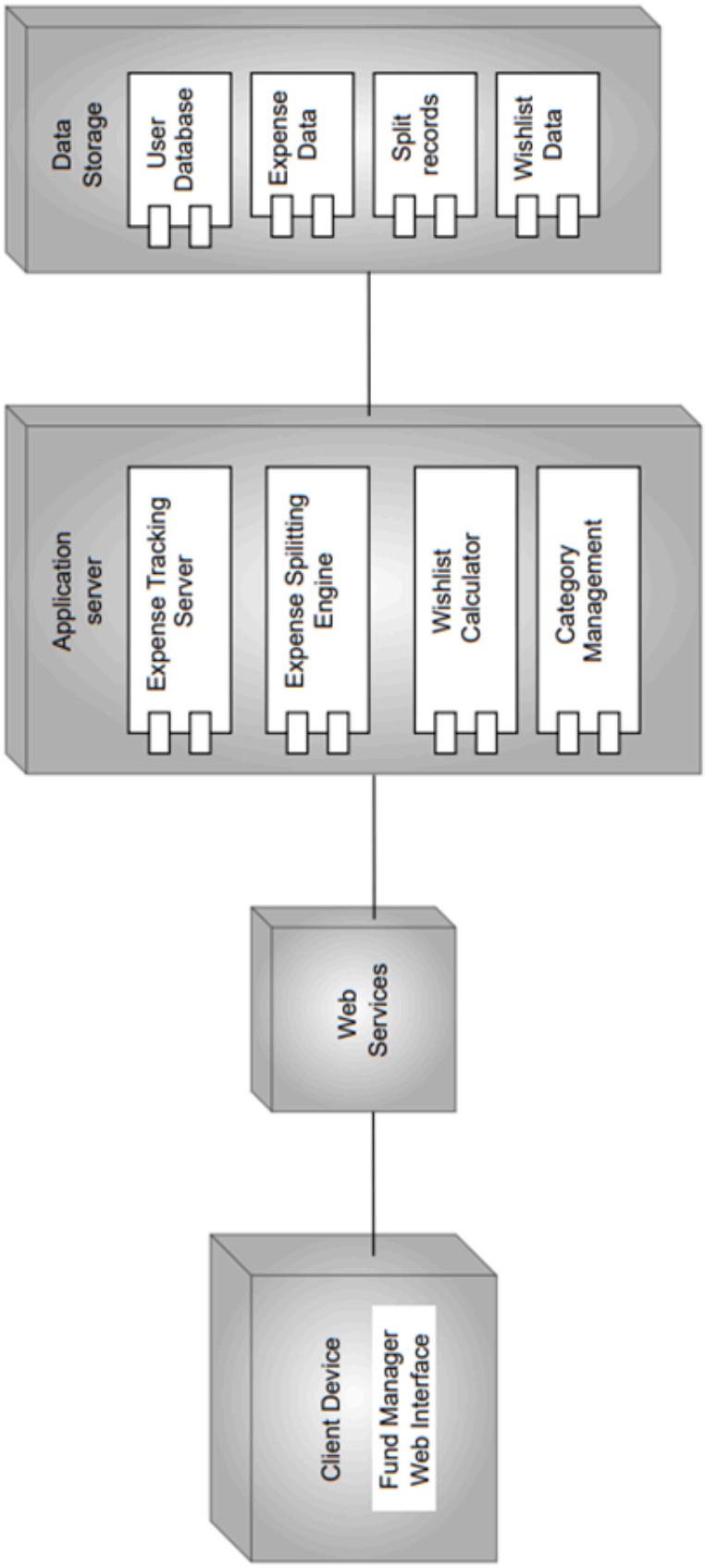
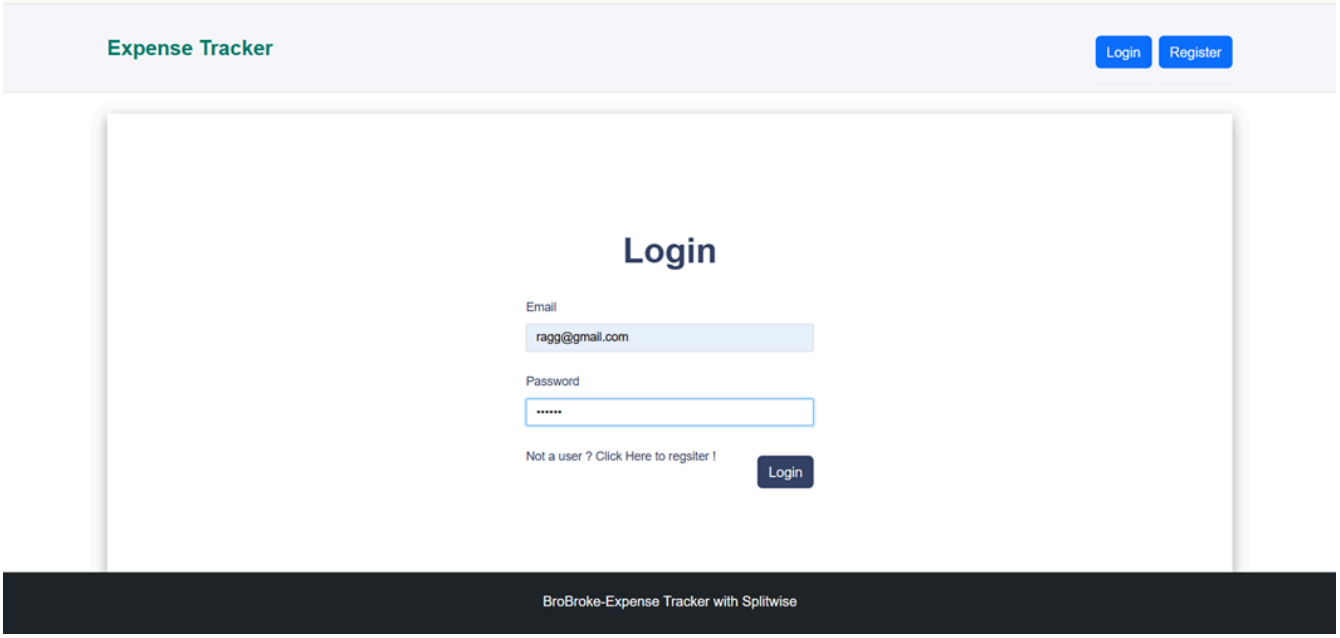
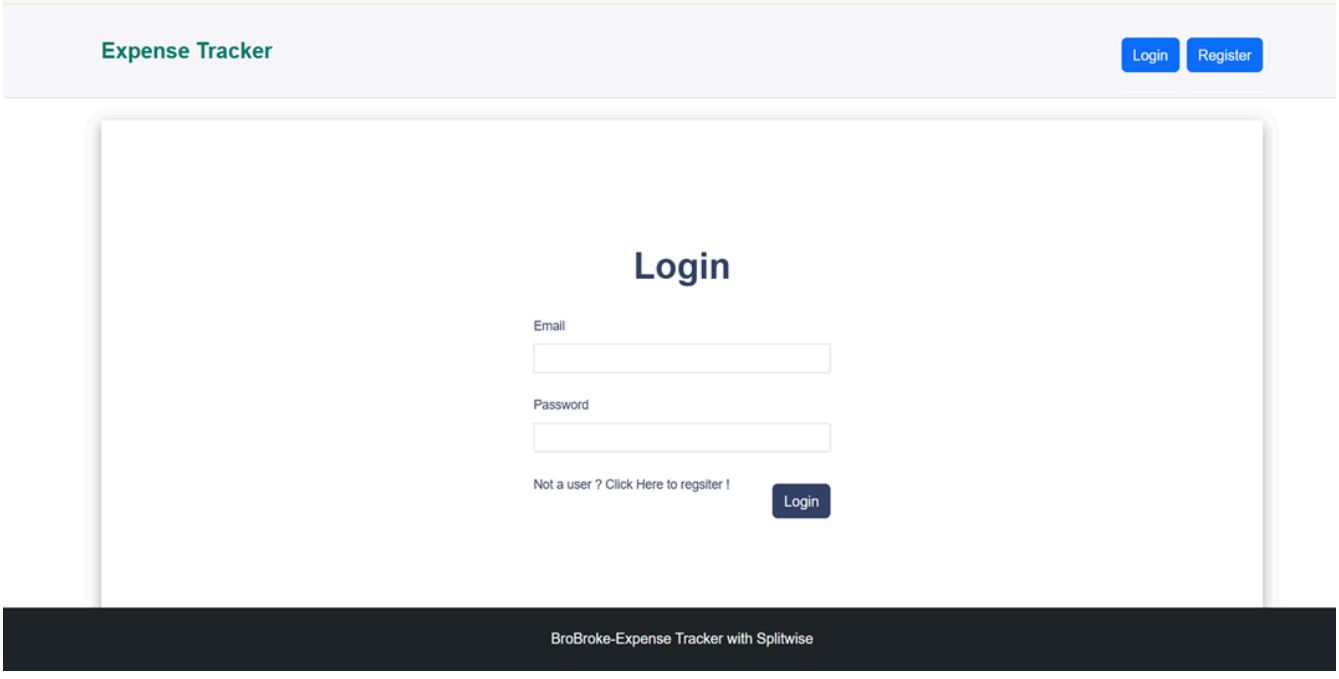


Fig 16- Deployment Diagram

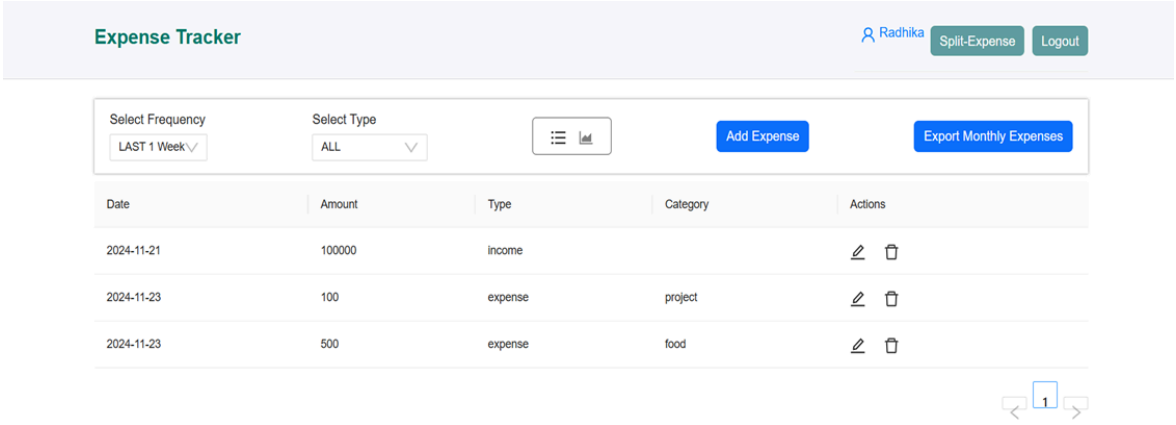
7.3 Screenshots

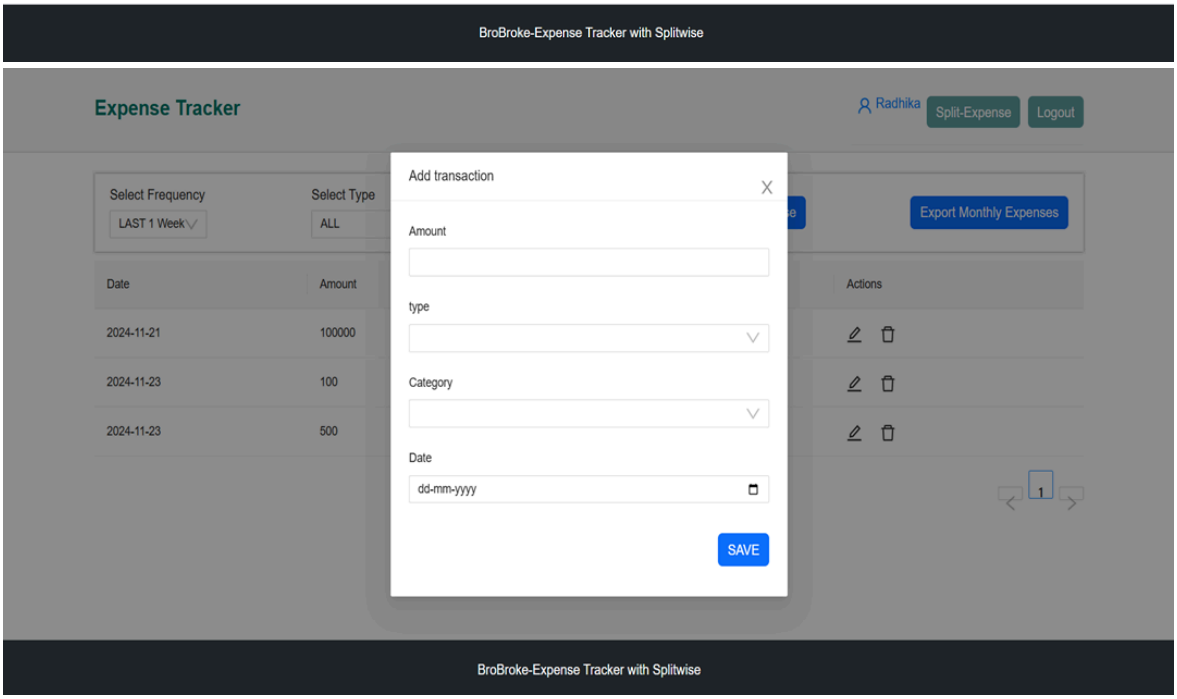
1. Login Page

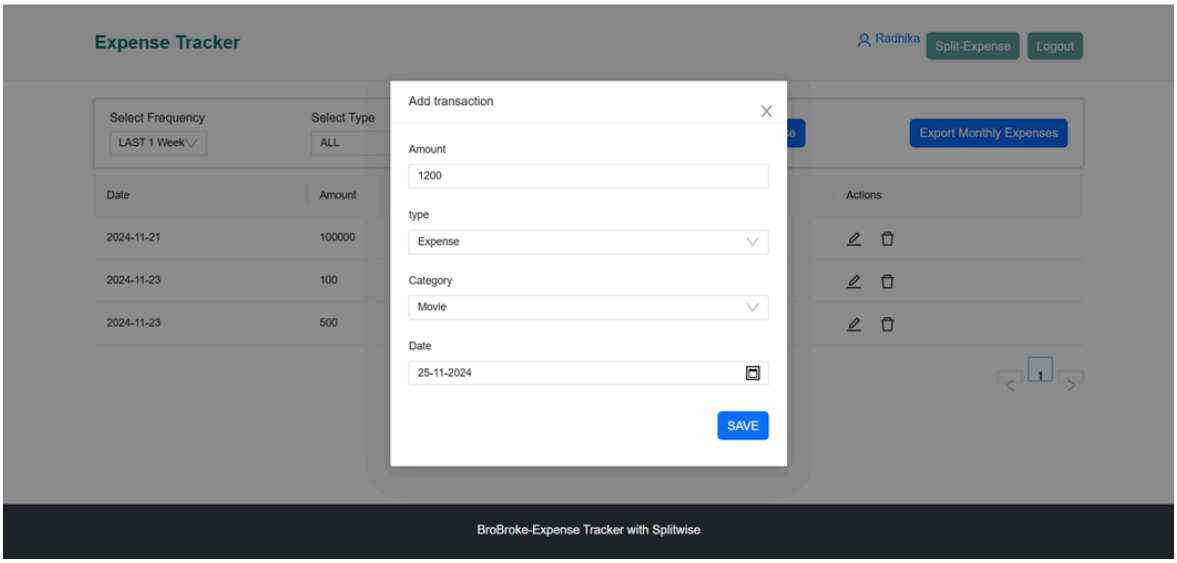




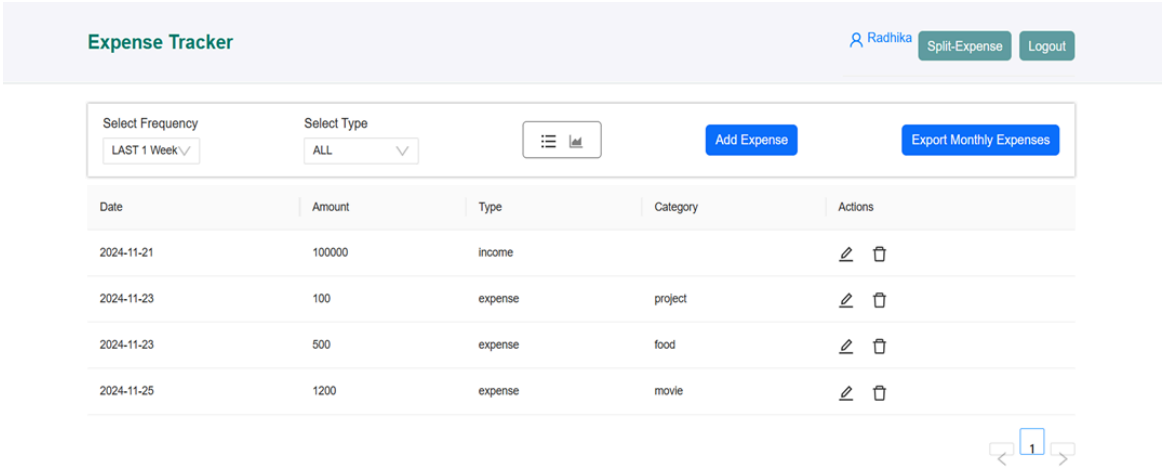
2. Expense Tracking

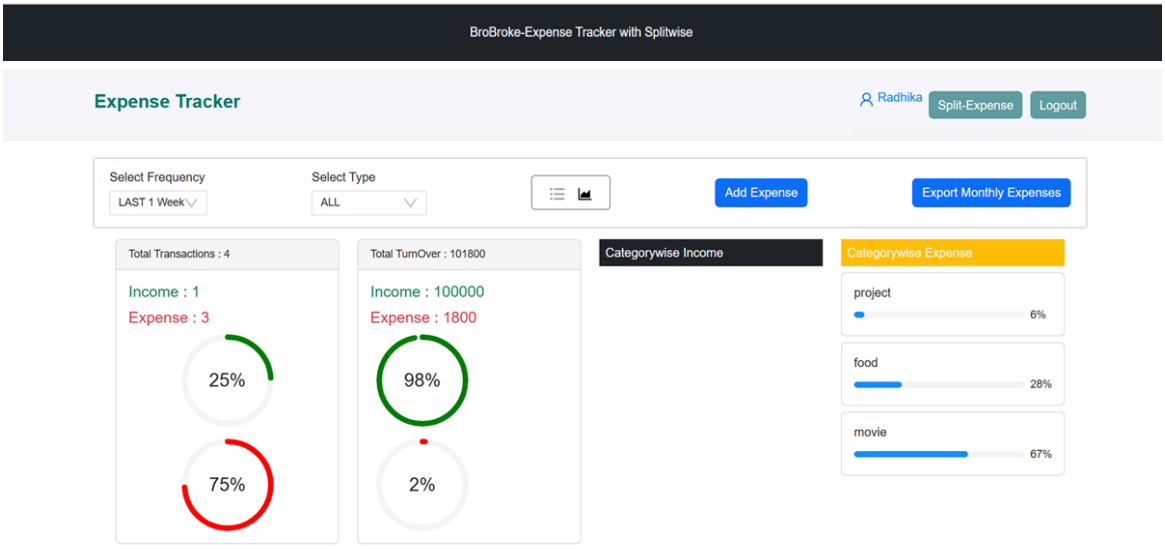


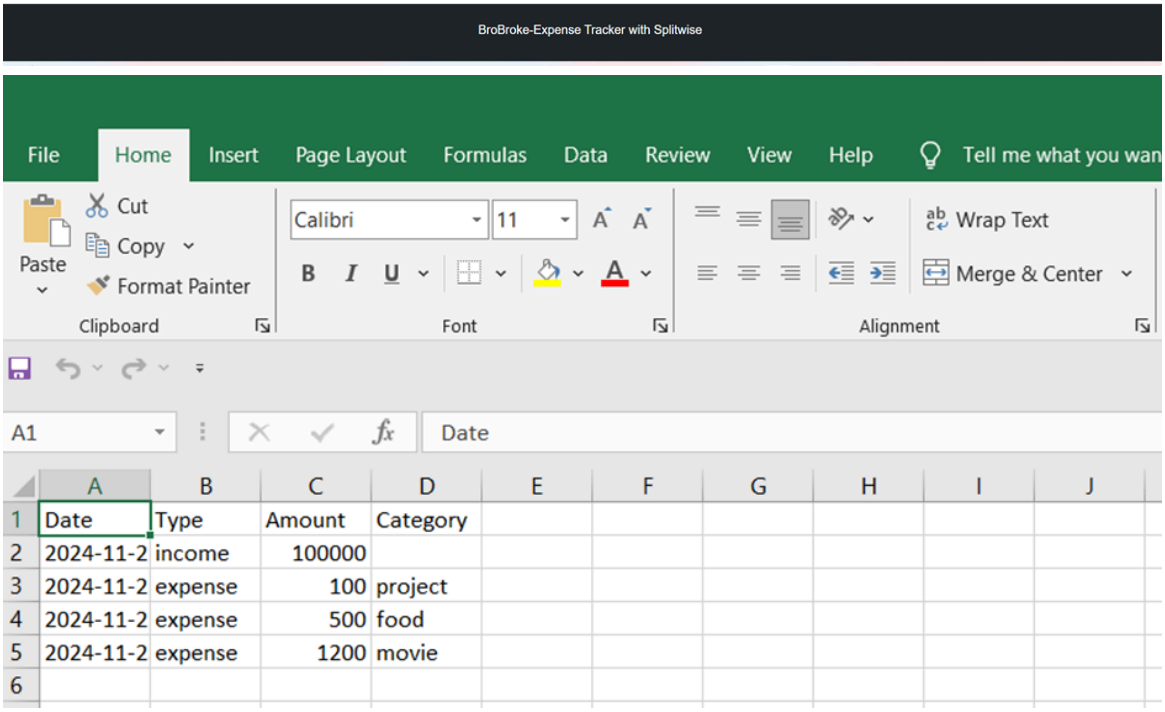




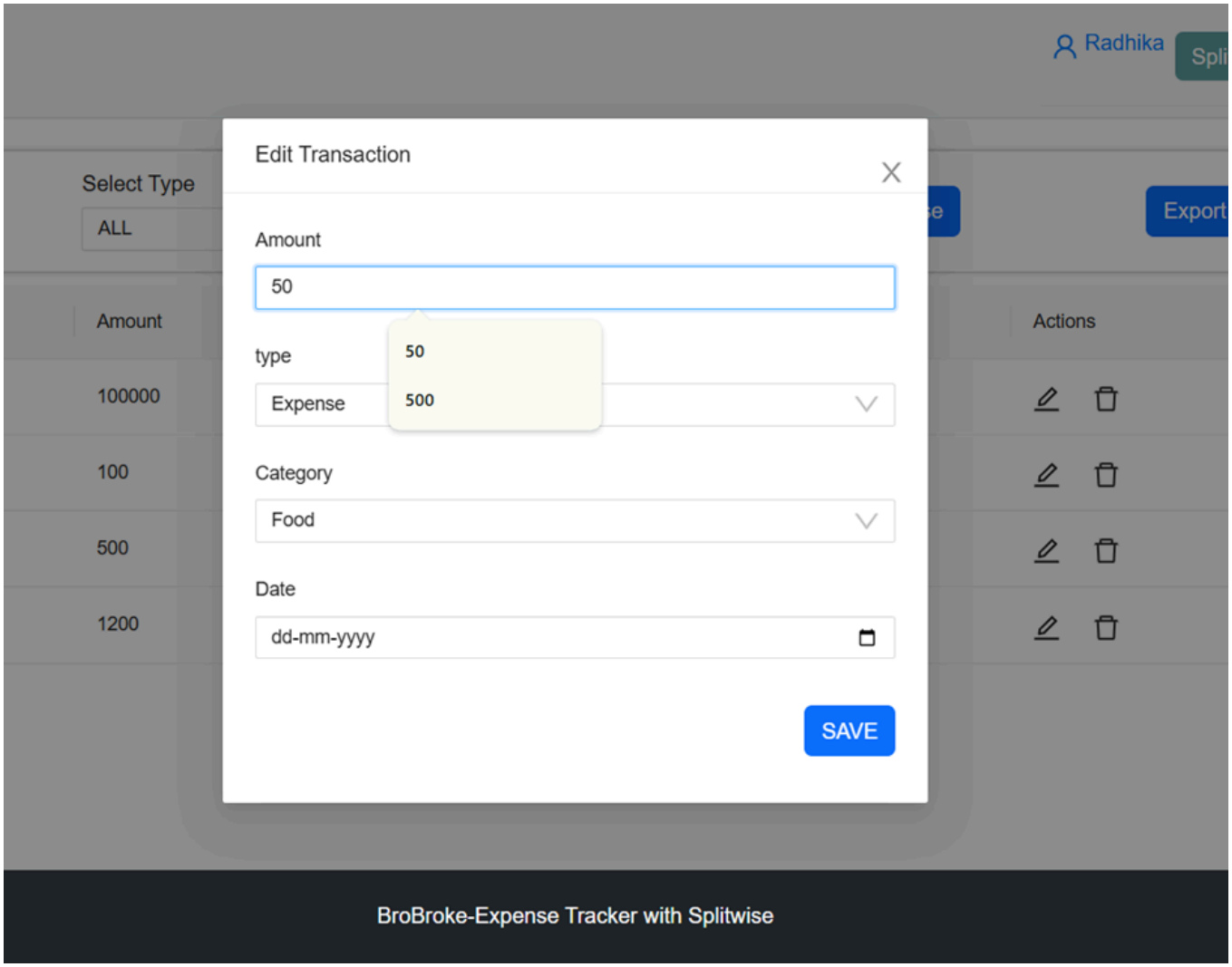
3. Displaying Data

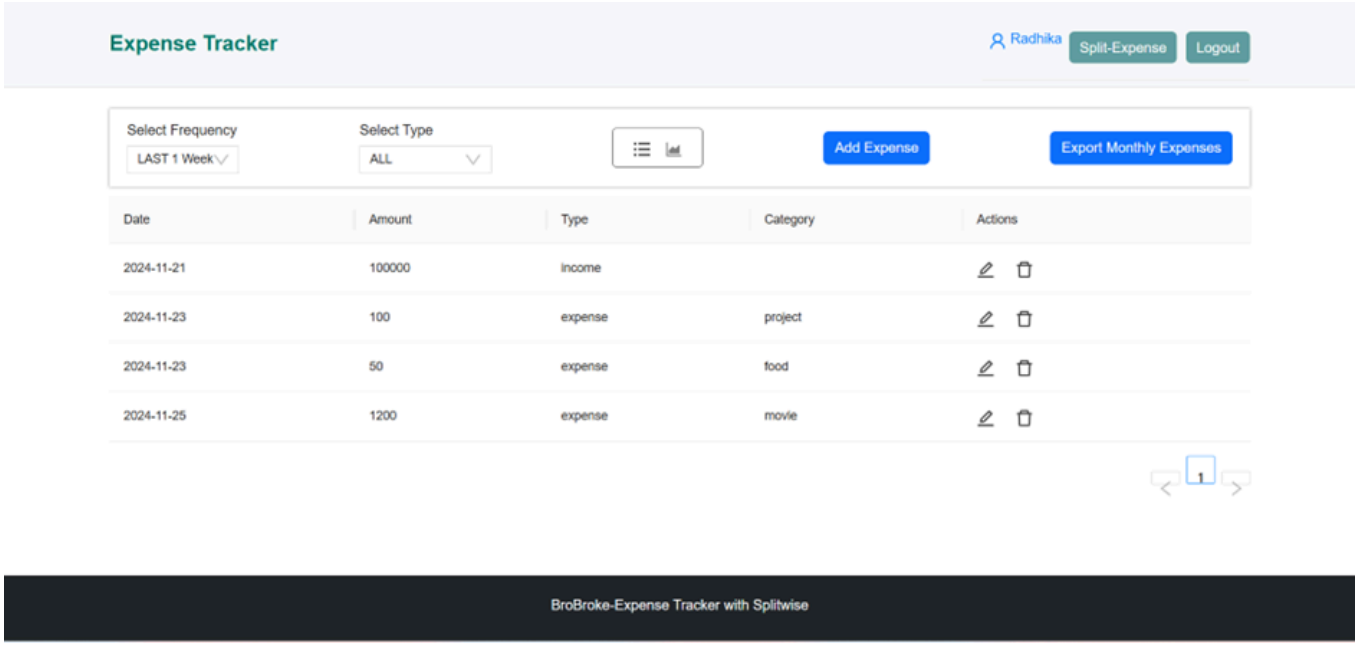




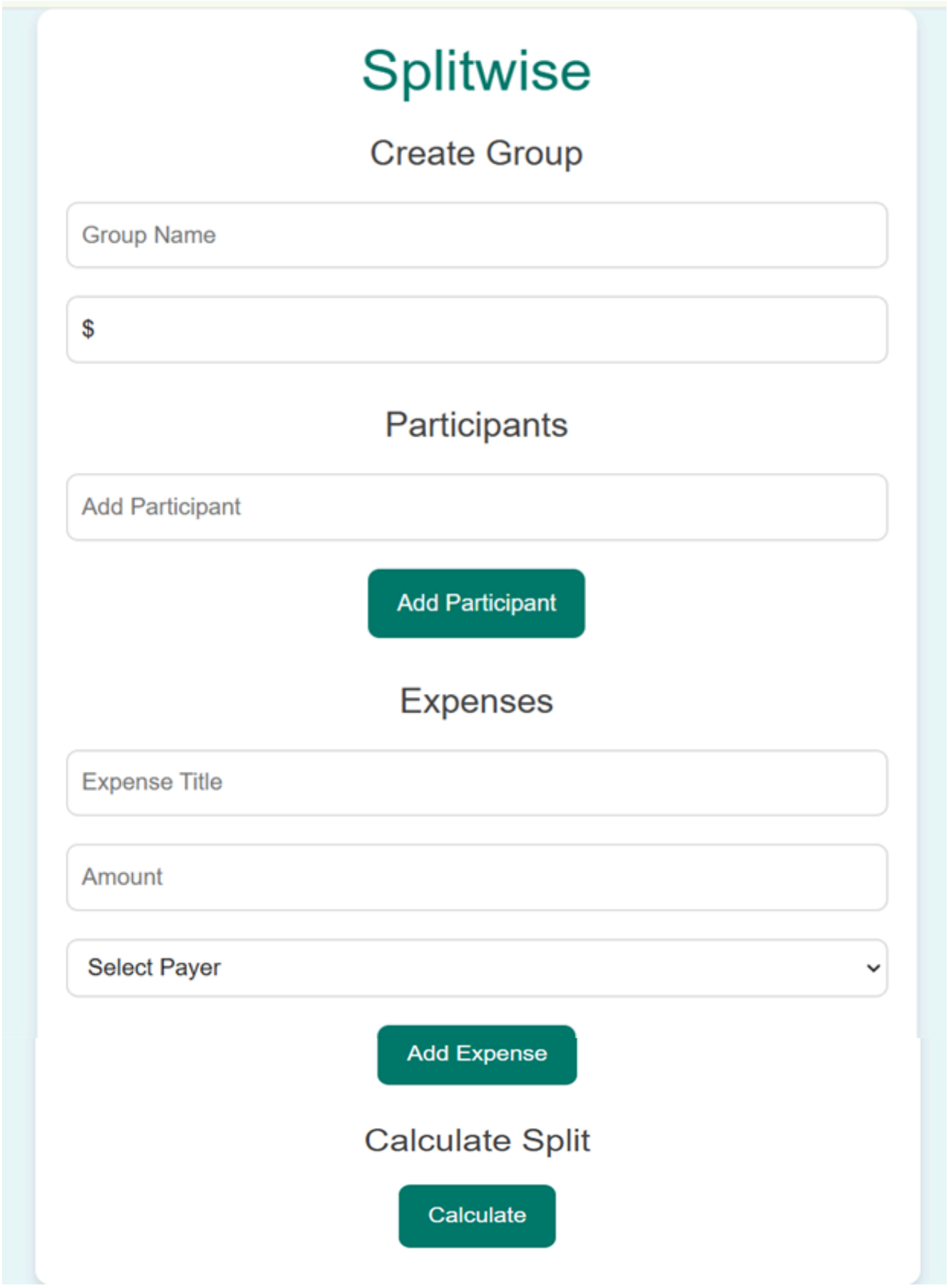


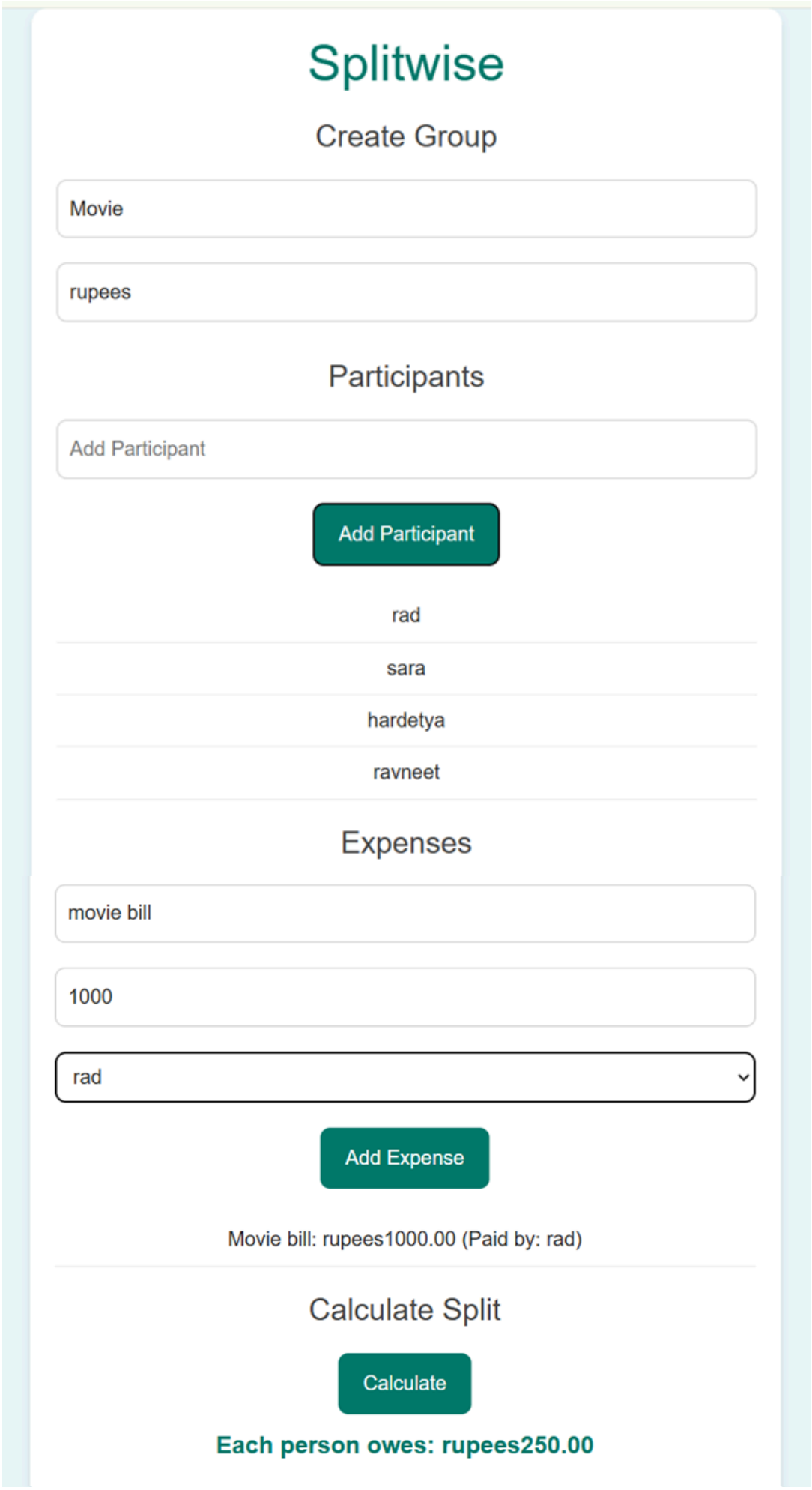
Adding an Expense





4. Splitwise


Splitwise
Create Group
Group Name
$
Participants
Add Participant
Add Participant
Expenses
Expense Title
Amount
Select Payer
Add Expense
Calculate Split
Calculate


Splitwise
Create Group
Movie
rupees
Participants
Add Participant
Add Participant
rad
sara
hardetya
ravneet
Expenses
movie bill
1000
rad
Add Expense
Movie bill: rupees1000.00 (Paid by: rad)
Calculate Split
Calculate
Each person owes: rupees250.00

8.
TESTING

Test Case 1

Test Case #: 1.0 **Test Case Name:** Login

System: Expense Management System **Subsystem:** User Authentication

Designed by: **Design Date:** 13/11/2024

Executed by: **Execution Date:**27/11/2024

Short Description: Verify that a user can log in from the Logged-Out state.

Pre-conditions

- The user is in the Logged-Out state.
- The application is launched and the login screen is displayed.
- The user has valid login credentials (username and password).

Step	Action	Expected System Response	Pass/ Fail	Comment
1	Select the "Login" option	The system should prompt the user to enter login credentials.		
2	Enter valid username	The system verifies username and proceeds to enter the password.		
3	Enter valid password	The system verifies password and proceeds to login.		
4	Confirm login action	The system transits to the Home Page.		

Post-conditions

Successful Login:

- The user is authenticated and is on the Home Page
- User session is initiated.

Unsuccessful Login:

- The user remains in the Logged-Out state.
- Appropriate error messages are displayed based on the failure reason.

Test Case 2

Test Case #: 2.0 **Test Case Name:** Add Expense

System: Expense Management System **Subsystem:** Expense Subsystem

Designed by: **Design Date:** 13/11/2024

Executed by: **Execution Date:**27/11/2024

Short Description: Verify that a user can add an expense.

Pre-conditions

- The user is in the Logged-Out state.
- The application is launched and the login screen is displayed.
- The user has valid login credentials (username and password).

Step	Action	Expected System Response	Pass/ Fail	Comment
1	Select the “Add Expense” option	The system should display an "Add Expense" form with required fields.		
2	Fill in the required expense details (e.g., amount, category, date, notes)	The system captures and displays the entered details for confirmation.		
3	Click “Submit” to save the expense.	The expense is displayed at the home page and is also saved at the backend server.		

Post-conditions

- The expense is successfully saved in the database system.
- The user is returned to the **Home Page** state.
- The new expense appears in the summary list.

Test Case 3

Test Case #: 3.0
Test Case Name: View Summary of Expense
System: Expense Management System
Subsystem: Expense Subsystem
Designed by:
Design Date: 13/11/2024
Executed by:
Execution Date:27/11/2024
Short Description: Verify that a user can view the expenses created and export it.

Pre-conditions

- The user is logged in.
- The user is at the Home Page and clicks on View Summary

Step	Action	Expected System Response	Pass/ Fail	Comment
1	Select the "Display" option	The system should display all the expenses, total transactions made, total turnover and categorical income and expense.		
2	Click on "Export Monthly Expense"	The expenses are downloaded in the form of an excel file and displays a message "Downloaded Successfully"		

Post-conditions

- The system remains in the **View Summary** state.
- The expense summary is accurately displayed.
- The excel sheet is displayed.

Test Case 4

Test Case #: 4.0
Test Case Name: Update/Delete Expense
System: Expense Management System
Subsystem: Expense Subsystem
Designed by:
Design Date: 13/11/2024
Executed by:
Execution Date:27/11/2024
Short Description: Verify that a user can update or delete an expense.

Pre-conditions

- The user is logged in.
- The user is in the **View Summary** state.
- At least one expense exists in the system.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	Locate an expense in the summary list that needs to be updated.	A summary view listing all the expenses with details and include action buttons like **Edit**, **Delete.**		
2	Click the **"Edit"** or **"Update"** button corresponding to that expense.	A form pre-filled with the selected expense's existing details.		
3	Modify the desired fields (e.g., amount, description, category, date).	Buttons or options for updating the fields: **Save/Submit** to confirm the changes or **Cancel** to go back without making changes.		
4	Click **"Save"** or **"Submit"** to apply the changes.	Reload and message is displayed "Expense updated successfully!".		

Post-conditions

- The updated expense details (e.g., amount, description, category, or date) are saved in the system/database.
- The user is returned to the **View Summary** state.

Test Case 5

Test Case #: 5.0

Test Case Name: Split Expenses

System: Expense Management System

Subsystem: Splitwise Subsystem

Designed by:

Design Date: 13/11/2024

Executed by:

Execution Date:27/11/2024

Short Description: Verify that a user can split expenses

Pre-conditions

- The user is logged in.
- The user is at the Home Page and clicks on Split Expenses

Step	Action	Expected System Response	Pass/ Fail	Comment
1	The user enters Splitwise mode	Enter Group Name and Currency.		
2	Enter Group Name and currency in which expenses are stored.	Enter Participants Name.		
3	The user enters participants among whom the expense is to be split.	Enter Expense Title, Amount and who paid for that particular expense		
4	The user enters all details	Click on Split Expense button		
5	User clicks on the button	The system displays the money split.		

Post-conditions

- The system successfully calculates the split and assigns the correct amount or percentage to each participant.

BIBLIOGRAPHY

1. https://www.geeksforgeeks.org/
2. https://fontawesome.com/
3. https://www.pexels.com/
4. https://app.diagrams.net

THANK YOU!