Software Requirements Specification

for

Brokeo

Version 1.0

Prepared by

Group Number:17 **Group Name: Runtime3rror**

Anjali Patra	**230148**	**anjalip23@iitk.ac.in**
Aujasvit Datta	**220254**	**aujasvitd22@iitk.ac.in**
Bhavnoor Singh	**230293**	**bhavnoor23@iitk.ac.in**
Darshan Sethia	**220326**	**darshands22@iitk.ac.in**
Dhriti Barnwal	**230364**	**dhritib23@iitk.ac.in**
Marisha Thorat	**230637**	**marishajt23@iitk.ac.in**
Rudransh Verma	**230881**	**rudranshv23@iitk.ac.in**
Sanjna S	**230918**	**sanjnas23@iitk.ac.in**
Solanki Shrey Jigneshbhai	**231017**	**shreyjs23@iitk.ac.in**
Suryansh Verma	**231061**	**suryanshv23@iitk.ac.in**

Course: CS253

Mentor TA: Mr. Paras Ghodeshwar

Date: 24 January 2025

Table of Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1 Introduction

1.1 Product Scope

Managing personal finances can often be challenging, leading to difficulties in tracking expenses, maintaining budgets, and saving effectively. Brokeo is a personal finance tracking application designed to address these issues by offering users a straightforward platform to manage their financial activities efficiently. It provides easy-to-use tools to help users make informed financial decisions and improve their money management habits.

The app features a reliable expense tracking system that helps users record and categorize their spending across various categories such as food, travel, and rent. By implementing an SMS detection system, it reduces the need for manually entering transactions. It enables users to set budgets and monitor spending through alerts. Visual tools like charts and graphs help users gain a more comprehensive understanding of their spending habits.

Brokeo also sends alerts to users about upcoming bill payments, subscriptions, and other outstanding dues. For shared expenses, the app offers a bill-splitting feature, making it easier for users to manage shared costs such as group trips and parties.

By combining expense tracking, budgeting tools, and features for shared expenses, Brokeo provides a comprehensive solution for effective financial management. Our team is committed to designing the app to be user-friendly, promoting better money management habits, and supporting users in achieving their financial goals.

1.2 Intended Audience and Document Overview

This software requirements document is written for the following audiences:

- **Software Development Team and Individual Contributors** will design and develop the software based on the outlined requirements for a personal finance management application.
- **Product Managers** (including Teaching Assistants and the Course Instructor) will oversee the planning, development, and execution of the application.
- **Testers** are expected to perform product and quality checks, carry out extensive testing, identify bugs, assess the app's functionality, and provide feedback on usability, interface, and areas for improvement.
- **End-Users of Brokeo** include individuals such as students, working professionals, or anyone who is looking for a simple and effective tool to manage their finances.

The document is divided into the following sections:

Section 1:
This section introduces key terminology and essential information to help users understand the Software Requirements Specification (SRS). If users are already familiar with the terminology, this section can be skipped. It provides an overview of the entire document.

Section 2:
This section provides clear and concise details about the app's functionality, interface, assumptions, and dependencies. It is an essential section for familiarizing users with the app. All users are encouraged to read this section for a comprehensive understanding of the SRS document.

Section 3:
This section presents a thorough description of the software, its functionality, and its operation by breaking the entire user experience into various use cases. It outlines the basic requirements for using the app and explains how each use case supports users in achieving their goals. End-users are required to review this section to fully understand the requirements for using the app.

Section 4:
This section outlines important non-functional requirements that users must follow to ensure a smooth and effective experience with the app. It is a crucial section for all users to read.

1.3 Definitions, Acronyms and Abbreviations

- **API** - Application Programming Interface
- **CSV** - Comma Separated Value
- **DDoS** - Distributed Denial of Service
- **FAQ** - Frequently Asked Questions
- **GB** - GigaBytes
- **IEEE** - Institute of Electrical and Electronics Engineers
- **MS** - IEEE Micro Software
- **OTP** - One-Time Password
- **RAM** - Random Access Memory
- **SD** - Secure Digital
- **SMS** - Short Message Service
- **SRS** - Software Requirement Specification
- **UI** - User Interface

1.4 Document Conventions

While preparing this document, to make readability user-friendly and important parts clear, we have used the following conventions*:*

- The headings of all the sections are written in bold using Arial font size 14.
- The headings of subsections are bold and written in the same font as that of content using font size 12.
- The content of the section is written in Arial font size 11.
- Any important term or short form is written in bold.
- The alignment of the whole content is justified.
- The text has been indented wherever required to highlight the content hierarchy.
- The document follows IEEE formatting, indenting, and numbering conventions. Any deviations from the same will be explicitly specified.

1.5 References and Acknowledgments

Style Guide (for this document)

W. P. Rogers, "A best practices approach to software requirements," in IEEE Software, volume 17, number 2, pp. 101-102, March-April 2000, doi: 10.1109/MS.2000.841701

Acknowledgements

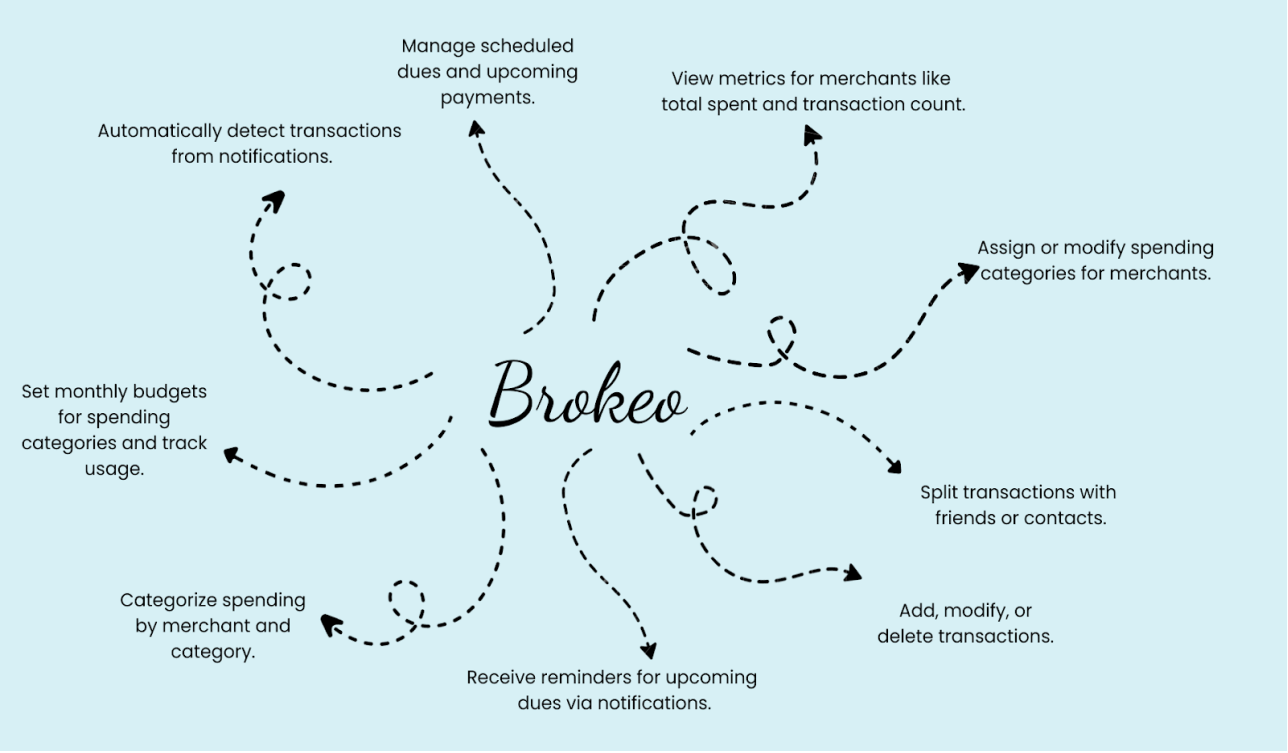
The fundamental principles of our system design were established through the lecture notes and guidance provided by Professor Dr Indranil Saha, along with the invaluable assistance of our Teaching Assistant, Paras Ghodeshwar.

2 Overall Description

2.1 Product Overview

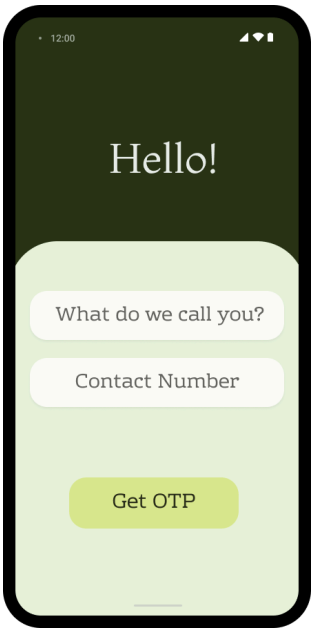
Brokeo is designed to help users develop better spending habits by providing a platform to manage, monitor, and analyze their expenses. The application offers detailed insights into spending patterns, empowering users to make more informed financial decisions. It automatically detects transactions from linked bank accounts via SMS detection and allows users to categorize these transactions or log additional expenses manually.

Brokeo delivers comprehensive analytics on yearly, monthly, and weekly trends, enabling users to effectively evaluate their financial activities. Shared expenses are easily managed using the "split" feature, which tracks contributions and amounts owed by others. Additionally, Brokeo includes tools for setting and tracking budgets, scheduling recurring transactions, and sending reminders for bill payments and subscription renewals, ensuring users stay financially organized.



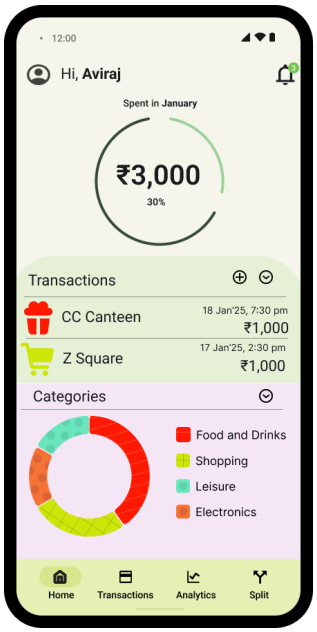
2.2 Product Functionality

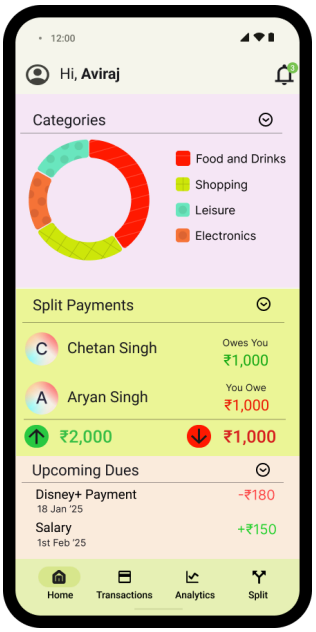
- **Login Page :**



The login page allows users to access their accounts by entering their name, phone number and password, ensuring they can use the app smoothly across multiple devices at the same time.

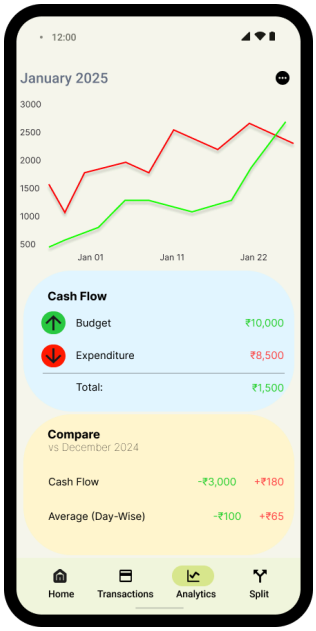
- **User Dashboard:**





The user dashboard is divided into six sections to ensure better clarity and organization. The first section allows users to monitor their spending and check their remaining balance. The second section allows manually adding transactions and reviewing the last five transactions. The third section displays the user's top spending categories. The fifth section offers access to the history of shared expenses. Lastly, the fourth section outlines details of upcoming scheduled payments.

- **Analytics Page :**

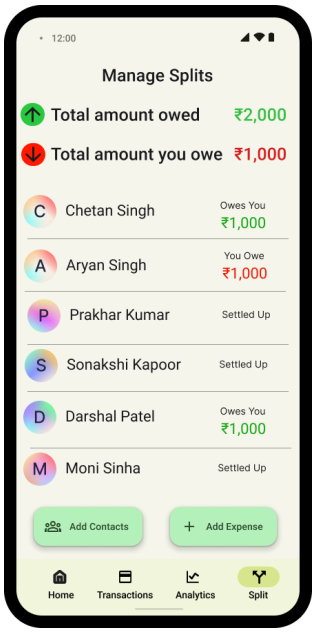


The Analytics page offers three key features: it enables users to monitor their spending patterns over weekly, monthly, quarterly, and yearly intervals; provides detailed insights such as total spending, average spending, and spending comparisons; and allows users to export transaction data for a specified custom time period.

- **Transactions Page :**

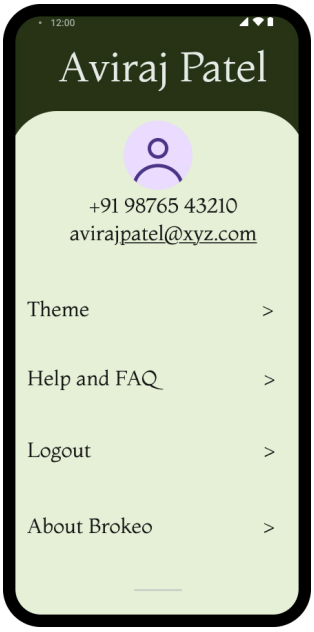
The Transactions page shows all recent transactions categorised into categories, provides a detailed breakdown of spending by merchant, organizes expenses into categories, and shows total expenditure both by individual categories and overall.

- **Split Transactions page :**



The split transactions page enables users to manage their shared expenses effectively. It allows them to view their net cash flow, create new transactions to be split among individuals, and see (and add) a list of people with whom expenses have been shared. Additionally, it provides a detailed view of the balance (debit/credit) for each transaction, records payments, and updates the status accordingly.

- **Profile Section :**



The Profile section consists of five subsections. The first subsection allows users to input their details, including name, mobile number, and email ID. The other subsections provide options to customize app themes, find help or FAQs for additional support, log out, and access information about the app.

2.3 Design and Implementation Constraints

- The application must be optimized to perform efficiently on devices with limited resources, such as lower-end smartphones with minimal processing power and memory (e.g., 2GB RAM).
- The central database is a crucial component of the application, storing all information about users and transactions. It must have sufficient memory and scalability to handle all users efficiently.
- Each user will have a local database that ensures seamless offline usage of the application. This local database must be optimized for quick access and periodic synchronization with the central database when an internet connection is available.
- The server must provide adequate resources and computational power to manage requests from all users and maintain smooth synchronization between local and central databases without delays.

2.4 Assumptions and Dependencies

- The maximum number of users that can register on the application will be 10,000, with scalability options for future growth if required.
- The number of users allowed to split a specific transaction will not exceed a configurable limit (e.g., 100).
- Transaction records and spending categories must be editable and maintain consistency across both central and local databases.
- The application is designed to accommodate offline usage seamlessly, and all updates to local databases will synchronize with the central database upon reconnection.
- Updates to the application or adoption of new technology should not disrupt user data or system functionality.

3 Specific Requirements

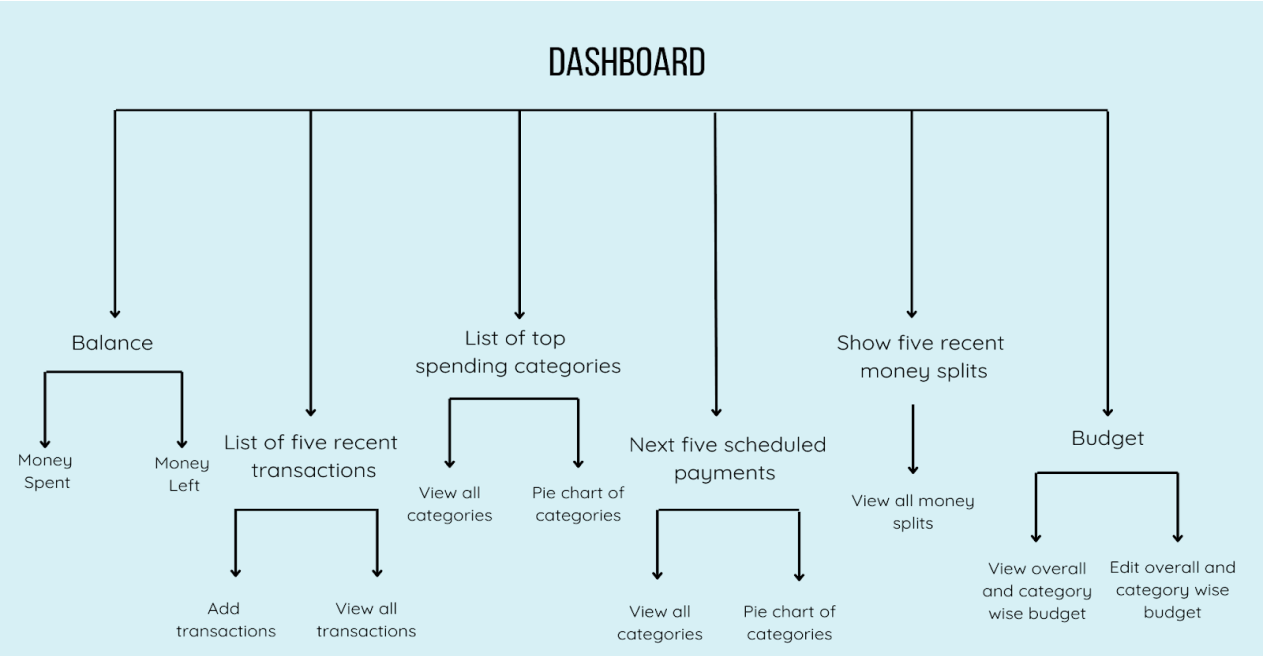
3.1 External Interface Requirements

3.1.1 User Interfaces

The application will have the following user interfaces

- The application will allow a single type of user to log in using their phone number, with authentication done through a one-time password (OTP).

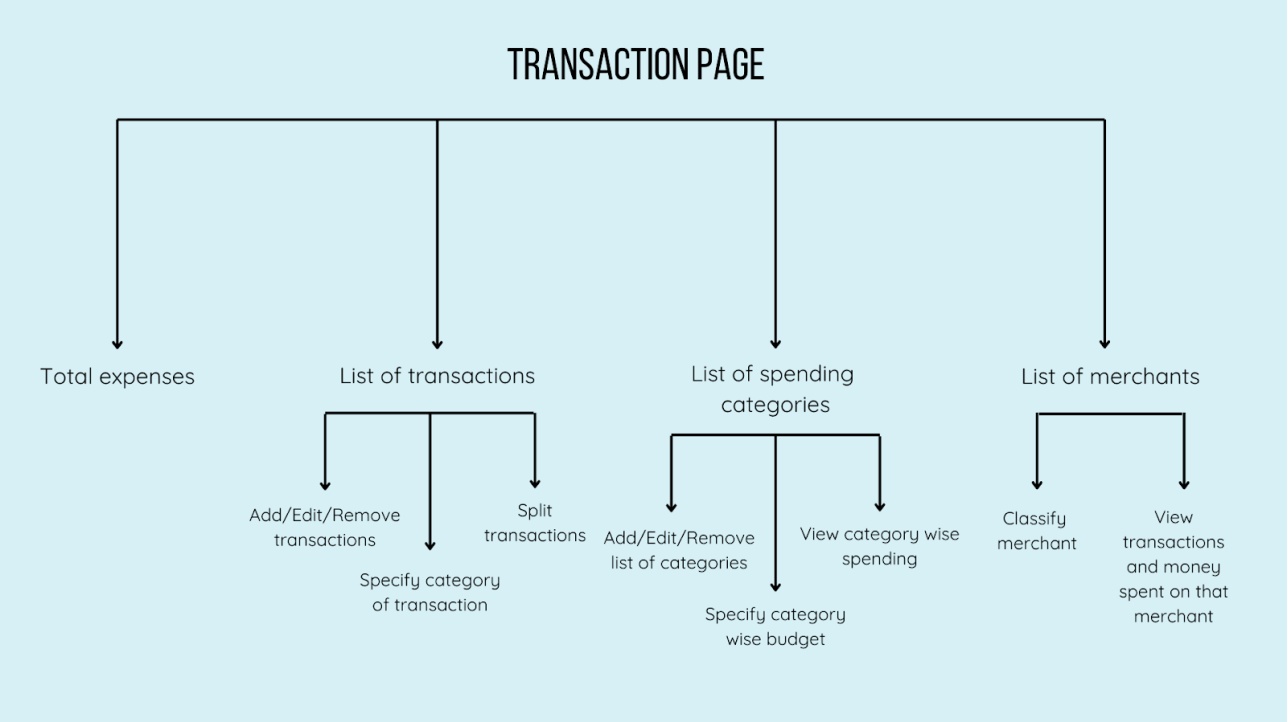
- The application will include a navigation bar at the bottom of the screen that allows users to switch between **four different pages**.
    - The first page will serve as the dashboard, displaying cards that provide various types of information as outlined below.



- The **first card** will display the available balance. Clicking on it will toggle to show how much money has been spent this month, alternating between these two views.
- The **second card** will show a list of transactions, highlighting the five latest transactions. It will include a button to add a transaction and an expand button that redirects to a page containing all transactions.
- The **third card** will list the top categories based on the amount of money spent in the current month. It will also include a doughnut chart showing this distribution. Clicking on this card will redirect to a page listing all categories.

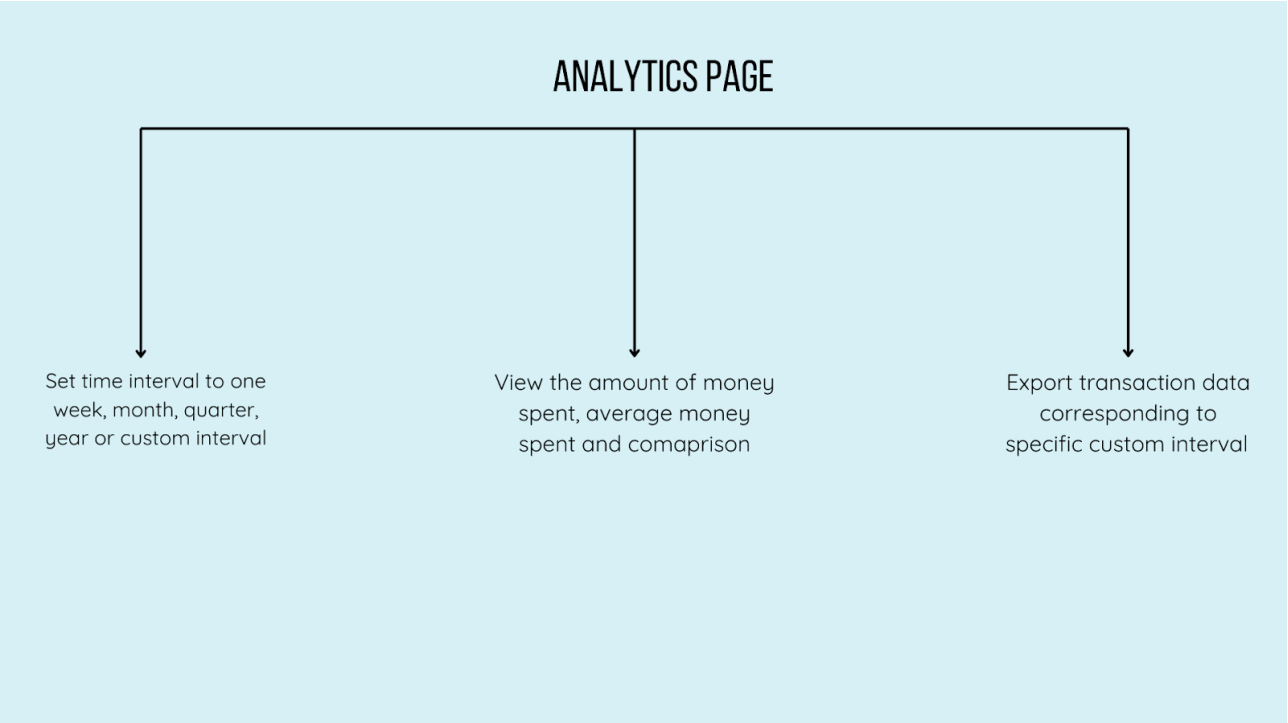
- The **fourth card** will display the recent split payments and indicate how much the user needs to pay or how much they are owed.
- The **fifth card** will display the five upcoming dues that need to be paid or will be deducted. Clicking on this card will show a list of all dues, including both past and future ones.

- The second page will be the transactions page. It will feature a tracker at the top for this month's expenses and include three subpages, as described below.

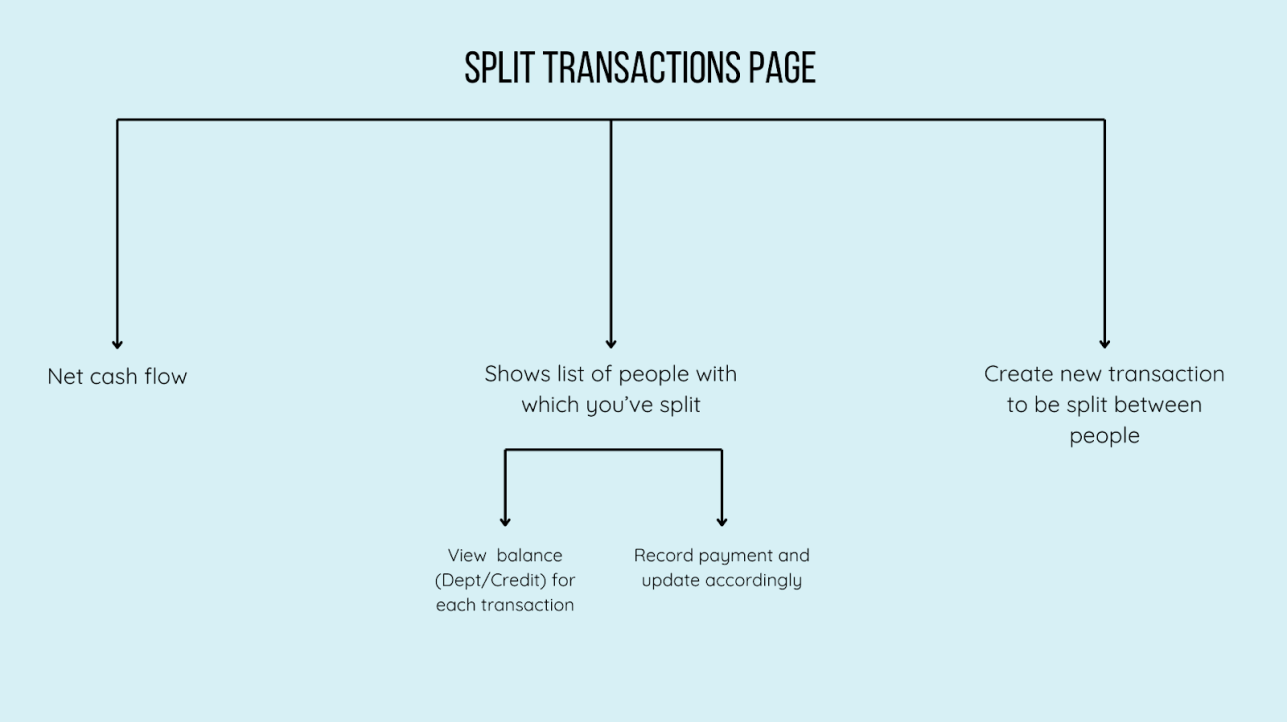


- **Transactions subpage:** This subpage will display a list of all transactions carried out. Users will have the option to modify transactions, classify them into different spending categories, and split transactions with other contacts.
- **Spending categories subpage:** This subpage will provide a list of spending categories currently configured. Users can create new categories, view the monthly spending for each category, and set budgets for individual categories as well as the overall budget.
- **List of merchants subpage:** This subpage will allow users to classify merchants into different spending categories. It will also provide the option to view transactions with a specific merchant and see the total amount spent on that merchant.

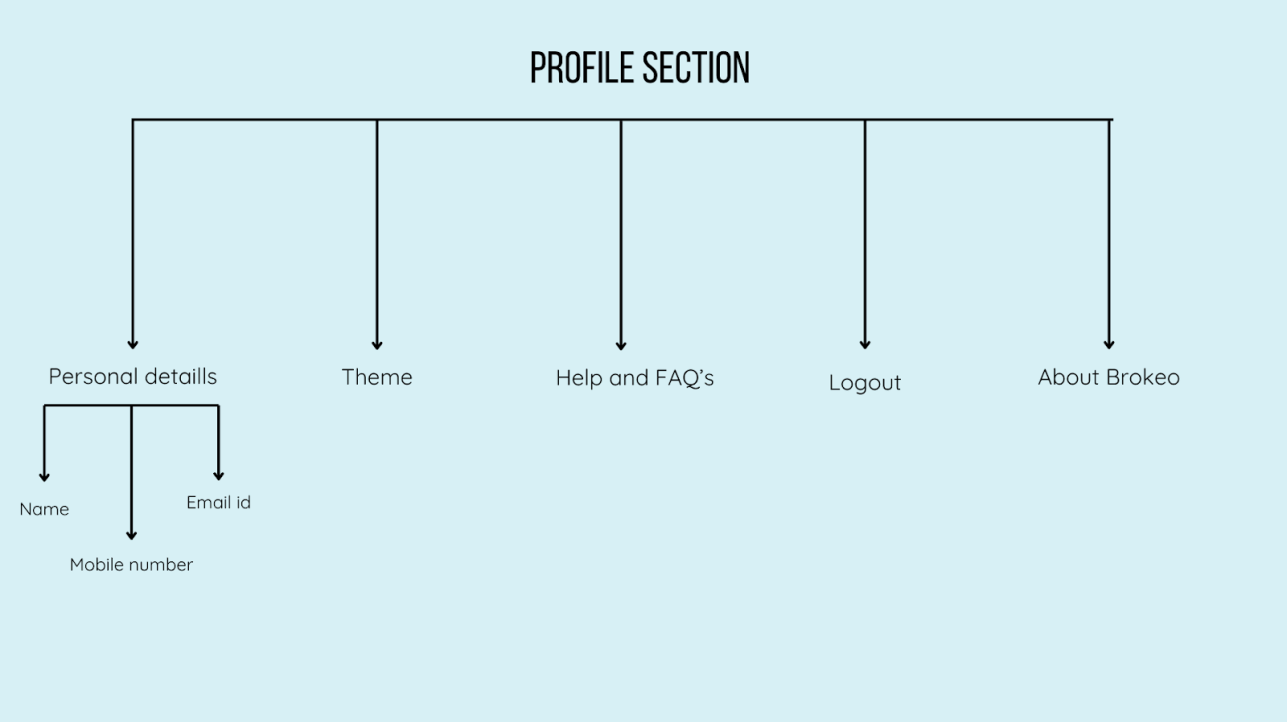
- The **third page** will be the analytics page. It will include an option to select a time frame for analysis, such as year, month, quarter, or week. Users can view the total and average amount of money spent during the selected time frame and compare it to the previous time frame.



- The **fourth page** will be the split transactions page, which will provide details about the transactions split with other people. At the top, it will display the net cash flow expected after clearing all dues (indicating whether you are in net debt or net surplus). This will be followed by a list of people you have split transactions with, sorted in decreasing order of the amount owed to them or owed by them. Clicking on a card for any person will show all transactions associated with that person and provide an option to record a payment to adjust the dues. Additionally, there will be an option to create a new transaction and split it between multiple people.



- The **profile page** will allow users to edit personal information, such as name, mobile number, and email ID. It will also include options to set the app theme, access a help section, view an about section, and log out of the app



3.1.2 Hardware Interfaces

For an expense-tracking app running on smartphones, the following hardware interfaces are considered:

- **Touchscreen Interface**
  - The app will rely on touch gestures for navigation, data entry, and interaction. It will support gestures like taps, swipes, and pinch-to-zoom for a user-friendly experience.
  - Compatible with touchscreens found on modern smartphones.

- **Storage**
  - Requires access to internal or external storage to save application data, such as user preferences, transaction records, etc.
  - Supports built-in smartphone storage and external SD cards if available.

- **Biometric Authentication**
  - Allows users to secure the app with fingerprint..
  - Utilizes the device's biometric sensors (e.g., fingerprint scanner).

- **Network Interface**
  - Requires access to Wi-Fi or cellular data for syncing information with a global server.
  - Compatible with the smartphone's Wi-Fi and cellular hardware.

3.1.3 Software Interfaces

- The phone application must be compatible with modern Android and iOS operating systems, ensuring smooth functionality across a wide range of devices. Specifically:
    - **Android:** The application must support devices running Android 8.0 (API Level 26) and above
    - **iOS:** The application must support devices running iOS 13 and above
    - The application must function seamlessly across various screen sizes and resolutions
- The application should be capable of parsing stored SMS messages on the device to automatically detect transactions.
- The application will use external APIs for OTP generation and email verification to confirm the user's phone number and email address.
- The application will use different databases to store and manage information related to users, transactions, spending categories, merchants, and other relevant data.

3.2 Functional Requirements

The final application should have the following functionalities:

3.2.1 Automatic Detection of transactions through SMS

- The system will identify online transactions by monitoring incoming messages on the user's phone.
- After detecting a transaction, the system will determine the merchant associated with the payment and categorize the purchase by both merchant and spending category (e.g., Food and Drinks, Groceries, Medical Supplies, etc.).

3.2.2 Transaction management

- The system will allow users to add, modify, and delete recorded transactions
- When adding a transaction, users will have the option to:
    - Enter the amount paid
    - Select the merchant from a dropdown menu containing existing merchants or create a new one.
    - Choose a spending category corresponding to the transaction
    - Split the transaction by selecting a list of contacts
- Users will be able to modify any recorded transaction, with access to the same fields as those available during the addition process: amount, merchant, spending category, and split details.
- The system will allow users to delete any previously recorded transaction.

3.2.3 Spending Category Management

- The system will provide users with the option to view their spending across various spending categories such as food and drinks, medical supplies, groceries, etc.
-

- Users will have the ability to manage spending categories, including adding, editing, or removing categories from the list.
- The system will allow users to assign a category to previously unknown merchants when making a payment to a merchant for the first time. Users will also be able to change the category of a specific transaction or merchant as needed.
- Users will have the option to specify a monthly budget for each spending category and view details about budget usage, including the amount spent and the remaining budget.

3.2.4 Merchant management

- Users will be able to view transactions organized by merchants, providing a detailed breakdown of all transactions associated with each merchant.
- The system will display metrics for each merchant such as the total money paid and the number of transactions made within the current month.
- Users will have the option to assign or modify the spending category associated with a merchant.

3.2.5 Spending Analytics

- Users will have the option to set a time interval, such as one week, one month, a quarter, a year, or a custom range, to view detailed analytics.
- The analytics will include**:**
    - Total money spent during the selected interval
    - Average spending for the period
    - A comparison with spending in previous intervals of the same duration
    - Category-wise breakdowns of spending
- Users will have the ability to export transaction data for a specific time interval (e.g., a week, month, quarter, year, or custom range) into a CSV file. The exported data will include a list of transactions during that time period, category-wise totals and the overall total amount spent.

3.2.6 Split transactions

- The system will use phone numbers and contact information as identifiers to manage the splitting of transactions among different users.s
- Users will have the option to split any existing or new transaction with other users.
- Users will be able to view their transaction history with any contact, including**:**
    - Transactions that have been split.
    - Outstanding amounts owed by or to the user.
    - The ability to record cash payments made to or received from a contact to settle outstanding debts. These cash payments will also appear as separate entries in the transaction history with that contact.
- The user should be able to notify any other contact to request them to settle their debt.
- The user should also be able to view the net amount of money that they owe/ that is owed to them.

3.2.7 Scheduled dues and upcoming payments

- The system will display a dashboard listing all pending dues and upcoming payments, along with their respective due dates
- Users will receive notifications for upcoming due dates through push notifications.
- The system will allow users to manually add or edit dues and upcoming payments.
- Users will have the option to mark dues as either paid or pending.
- The system must allow users to set up recurring payments (e.g., mess fees, subscriptions).

3.2.8 User Profile page

- Users will be able to view and edit their details, including their name, phone number, and email address. The system will verify these details using a one-time password for the phone number and a verification email for the email address.
- Users will have the option to customize the app's theme by choosing between a light theme, dark theme, or syncing it with the default theme currently set on their mobile device.
- The application will include a help section containing a user manual to guide users in understanding and utilizing its features.
- The user should have the option to log out of the app.
- An "About Brokeo" section will be available, providing information about the application and the development team behind it.

3.3 Use Case Model

3.3.1 Use Case #1: Track Transactions Automatically (U1)

Authors: This use case was written by Anjali Patra and Marisha Thorat.

Purpose: To capture all user transactions automatically via SMS parsing for financial tracking.

Requirements Traceability: 3.2.1

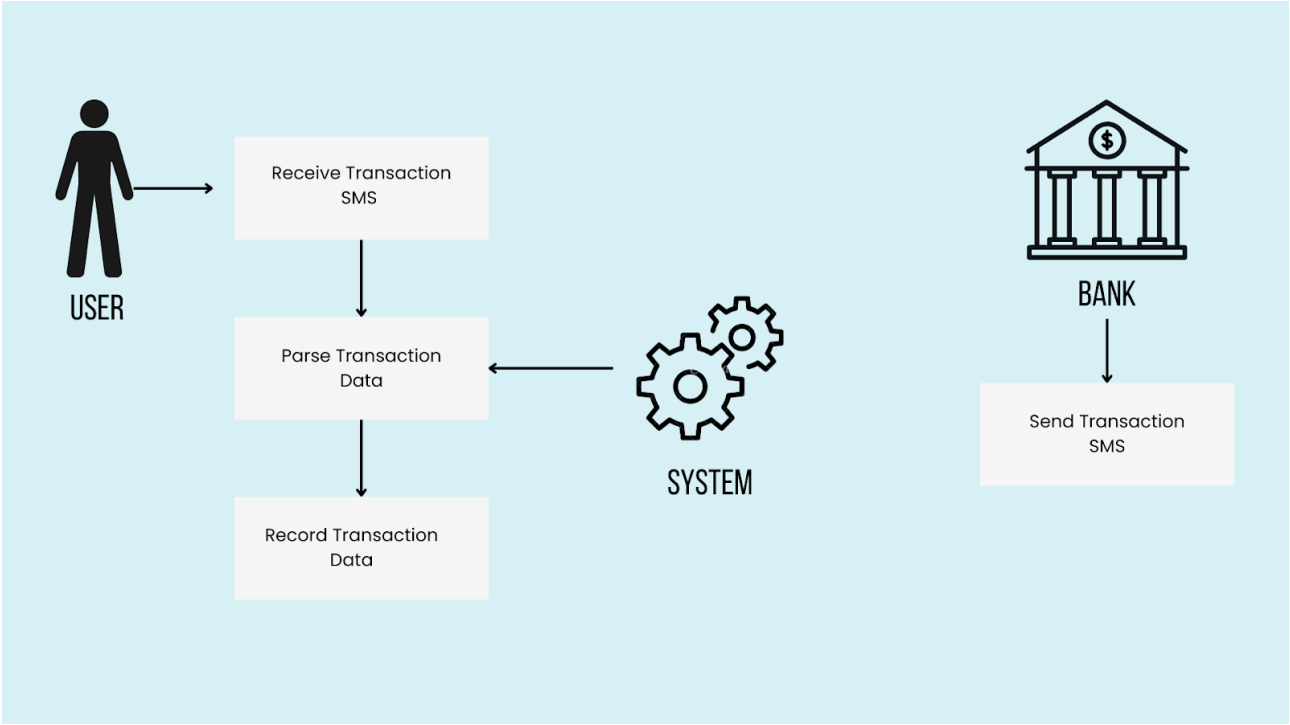
Priority: High - Critical for automatic transaction capture and accurate financial tracking.

Preconditions: The user must grant SMS permissions to the application and have SMS notifications enabled for financial transactions from their bank or payment provider. Additionally, the system must be integrated with an SMS parsing engine capable of identifying financial transaction patterns, and a secure database must exist to store transaction data.

Post-conditions: Transaction data is automatically captured, parsed, and stored. Transactions are securely stored in the database and categorized.

Actors: User, SMS Parsing Engine, Transaction Database

Exceptions: Invalid or unreadable SMS format – the system cannot parse transaction details. SMS permissions not granted – transaction data cannot be captured.



3.3.2 Use Case #2: View and Manage Transactions (U2)

Authors: This use case was written by Rudransh Verma and Bhavnoor Singh.

Purpose: To allow users to view detailed information for each transaction, edit existing transactions, or add new ones.

Requirements Traceability: 3.2.2

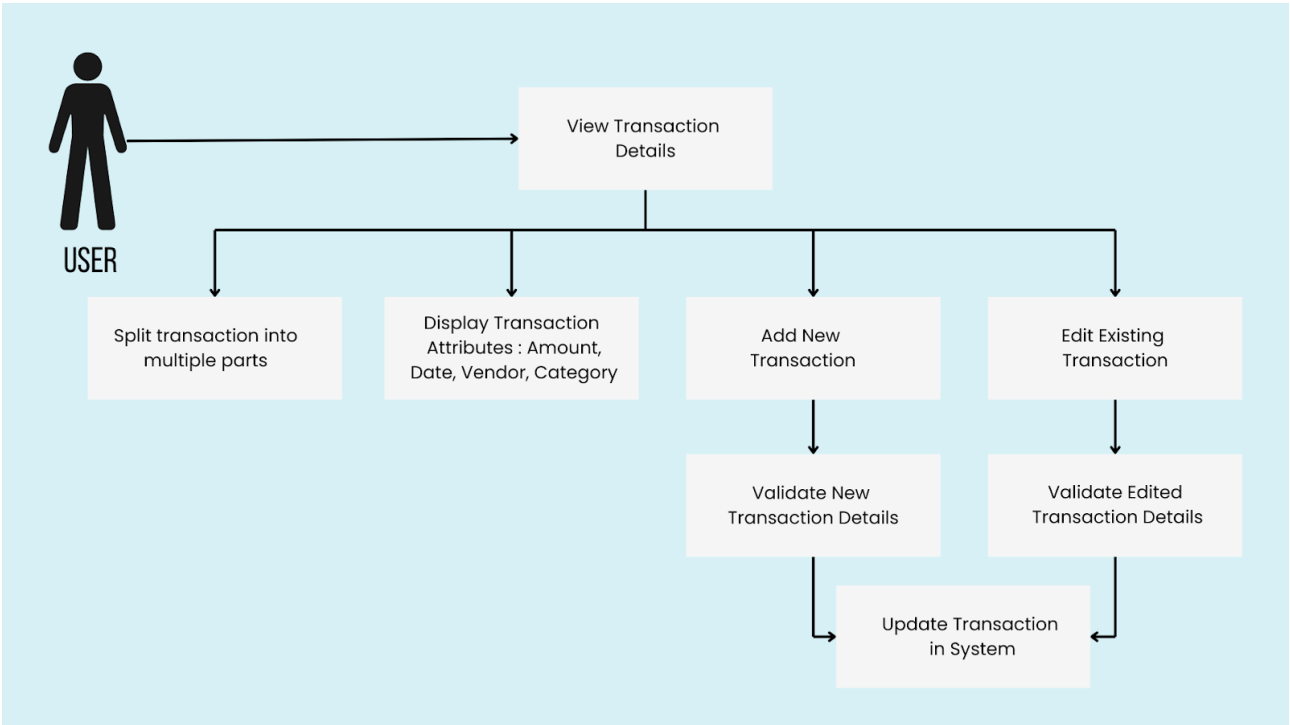
Priority: High - Essential for users to manage and edit their transactions effectively.

Preconditions: The system must have existing transaction data stored in the database and provide an interface that allows users to view, edit, and add transactions. Each transaction must be associated with defined categories and attributes. Additionally, the system must enforce validation rules for user-input transaction details to ensure the accuracy and integrity of the data.

Post-conditions: Transaction details are updated or new transactions are added. Validated transaction data is stored securely.

Actors: User, Transaction Database, System

Exceptions: Invalid transaction data input – system flags or rejects the entry.



3.3.3 Use Case #3: Set and Manage Budgets (U3)

Authors: This use case was written by Aujasvit Datta and Suryansh Verma.

Purpose: To set and track overall and category-specific (e.g., Food, Travel) budgets.

Requirements Traceability: 3.2.3

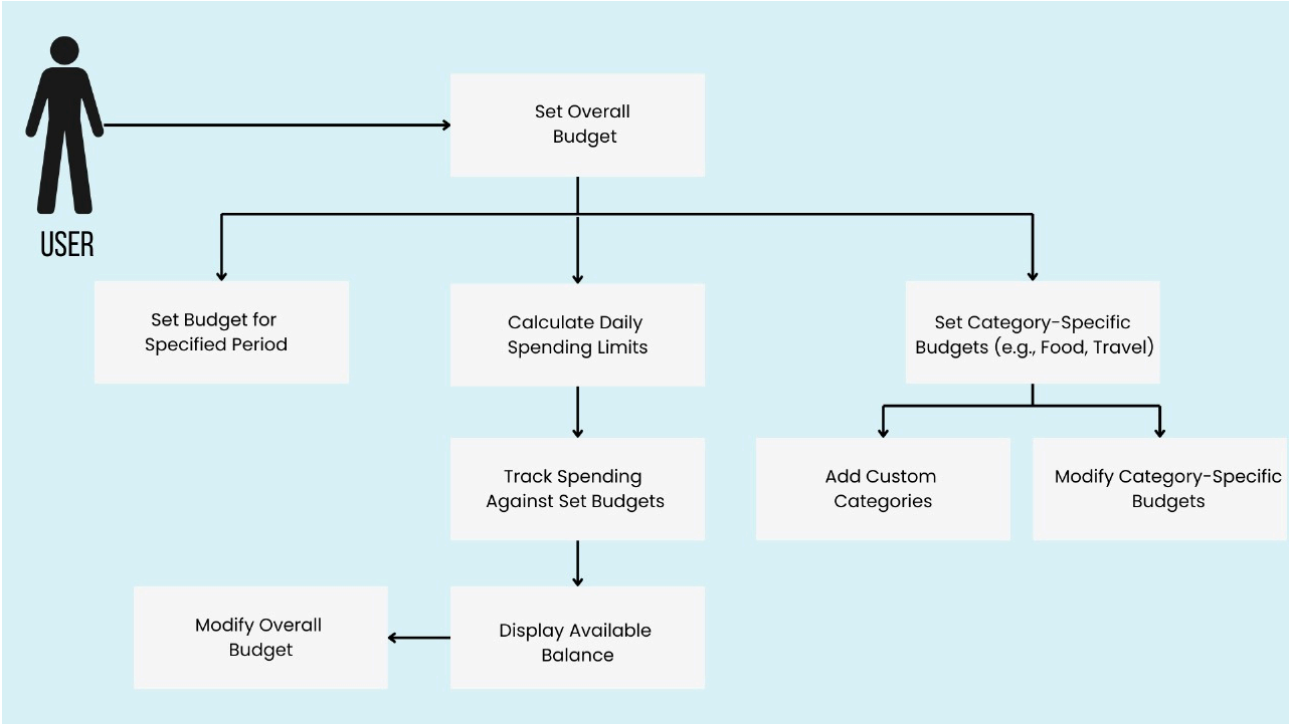
Priority: High - Core feature for managing overall and category-specific budgets to track expenses.

Preconditions: The system must have predefined categories or allow users to create custom ones. The system must have access to historical spending data for calculating daily limits. The system must have a budget setup interface available to users.

Post-conditions: Budgets are successfully set, modified, and tracked. Spending limits are updated, and users are notified of overspending.

Actors: User, System

Exceptions: Budget limit exceeded– user is alerted.



3.3.4 Use Case #4: Vendor Categorization and Expense Tracking (U4)

Authors: This use case was written by Darshan Sethia and Sanjna S.

Purpose: To categorize vendors into predefined or custom categories (e.g., Food, Travel) and track spending, including total amount and visit frequency.

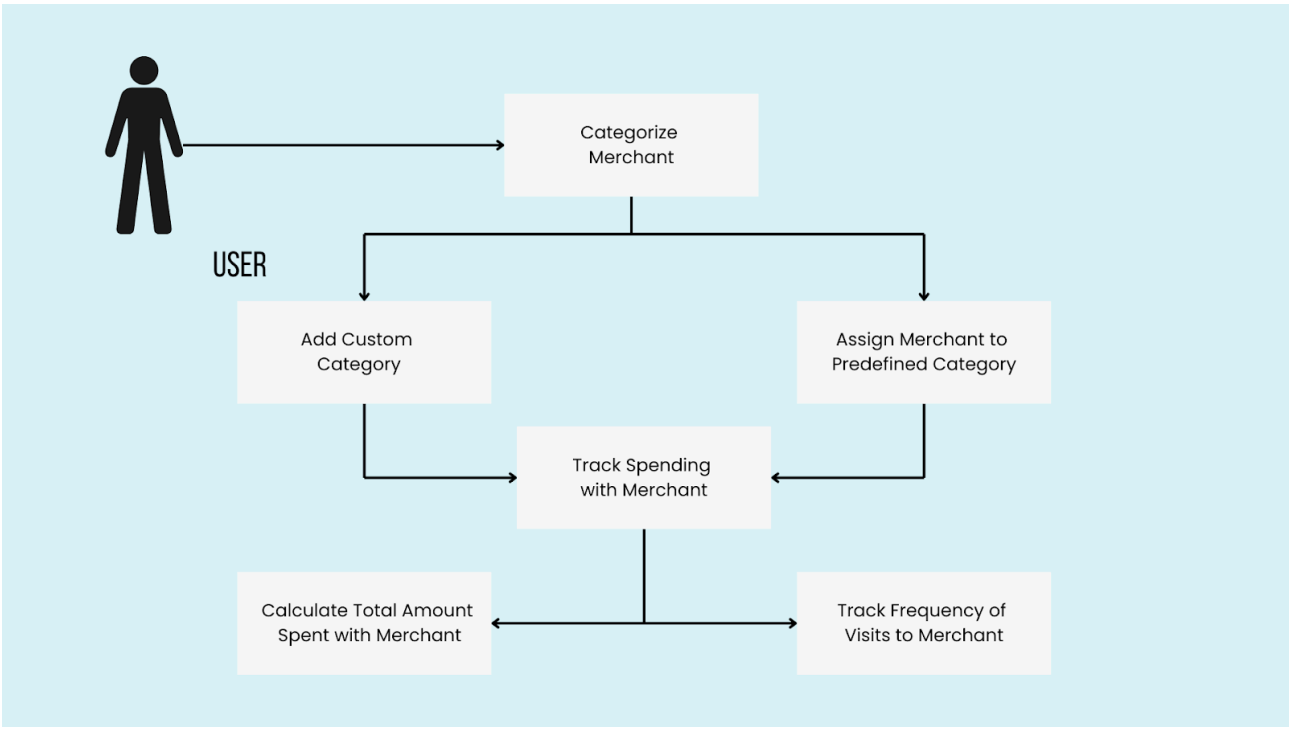
Requirements Traceability: 3.2.3, 3.2.4

Priority: Medium - Important for categorizing vendors and tracking spending but secondary to core functions.

Preconditions: The system must have stored transaction data with associated vendor details. Predefined and user-defined categories must be configured in the system. The system must have mechanisms to track spending amounts and frequency for each vendor.

Post-conditions: Vendors are categorized, and expenditure is tracked. Total amounts and visit frequencies are updated and displayed.

Actors: User, Transaction database, System



3.3.5 Use Case #5: Analyze Trends of Transactions (U5)

Authors: This use case was written by Solanki Shrey Jigneshbhai and Dhriti Barnwal.

Purpose: To view spending trends by month, week, or day.

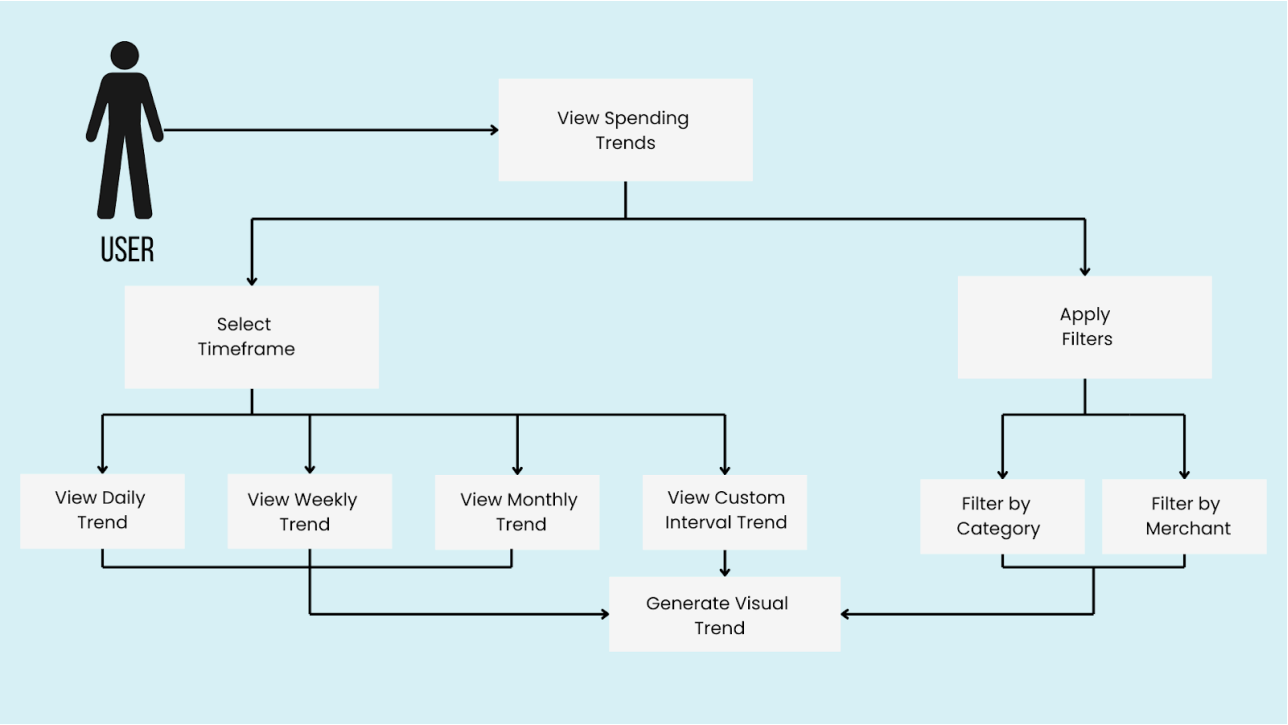
Requirements Traceability: 3.2.5

Priority: Medium - Useful for users to analyze spending trends but not a critical feature.

Preconditions: The system must have stored transaction data categorized by timeframes and categories. The system must have a visualization interface for displaying trends. Users must be able to apply filters to trends based on categories or merchants.

Post-conditions: Spending trends are visualized and can be filtered by categories or merchants.

Actors: User, System, Transaction Database



3.3.6 Use Case #6: Export Expense Report (U6)

Authors: This use case was written by Anjali Patra and Marisha Thorat.

Purpose: To allow users to export expense reports for specified timeframes (monthly, weekly, or custom).

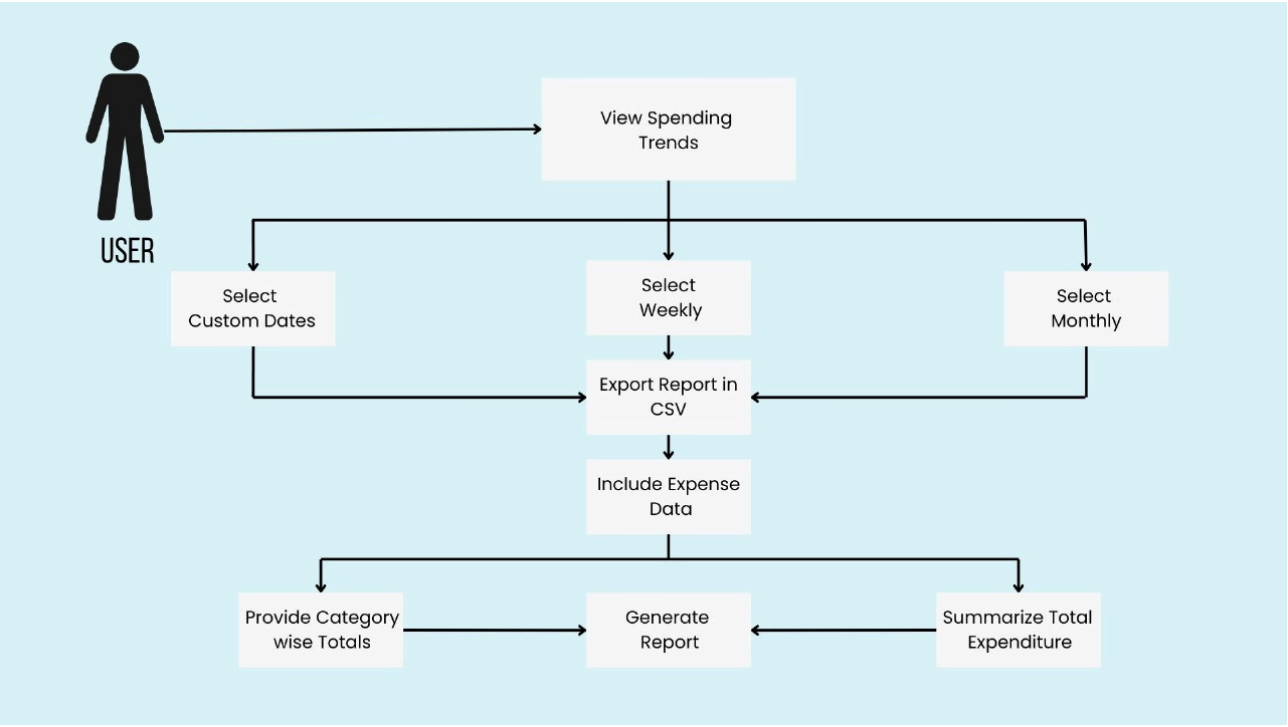
Requirements Traceability: 3.2.2, 3.2.5

Priority: Medium - Valuable for exporting detailed reports, but not as essential as real-time tracking.

Preconditions: The system must have stored transaction data categorized by timeframes and attributes. The system must have an export interface available for users to select timeframes and data options.

Post-conditions: The expense report is successfully exported in the chosen format. Reports can be shared via email or other platforms (if configured).

Actors: User, System, Transaction Database



3.3.7 Use Case #7: Visualize Expenditure with Charts (U7)

Authors: This use case was written by Rudransh Verma and Suryansh Verma.

Purpose: To provide a visual representation of overall expenditure and category-wise (e.g., Food, Travel) spending using charts.

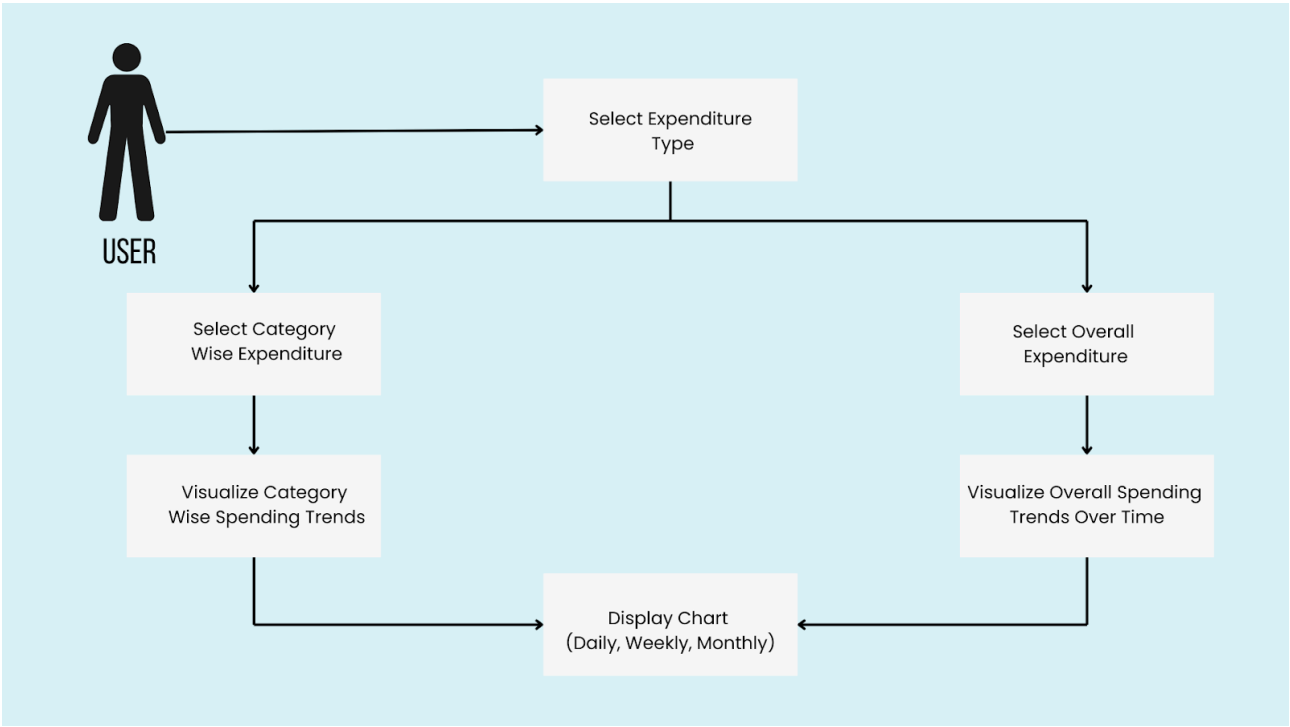
Requirements Traceability: 3.2.2, 3.2.5

Priority: Medium - A helpful visualization tool, but not central to financial management.

Preconditions: The system must have stored and categorized transaction data. The system must have a charting or visualization tool integrated into the application. Users must be able to apply filters for categories or timeframes for more accurate visualizations.

Post-conditions: The user can visualize expenditure trends via charts. Filters for categories or time frames are applied for better visual insights.

Actors: User, System, Transaction Database



3.3.8 Use Case #8: View and Manage Dues and Upcoming Bills/Payments (U8)

Authors: This use case was written by Aujasvit Datta and Bhavnoor Singh.

Purpose: To view and manage pending dues and upcoming bills.

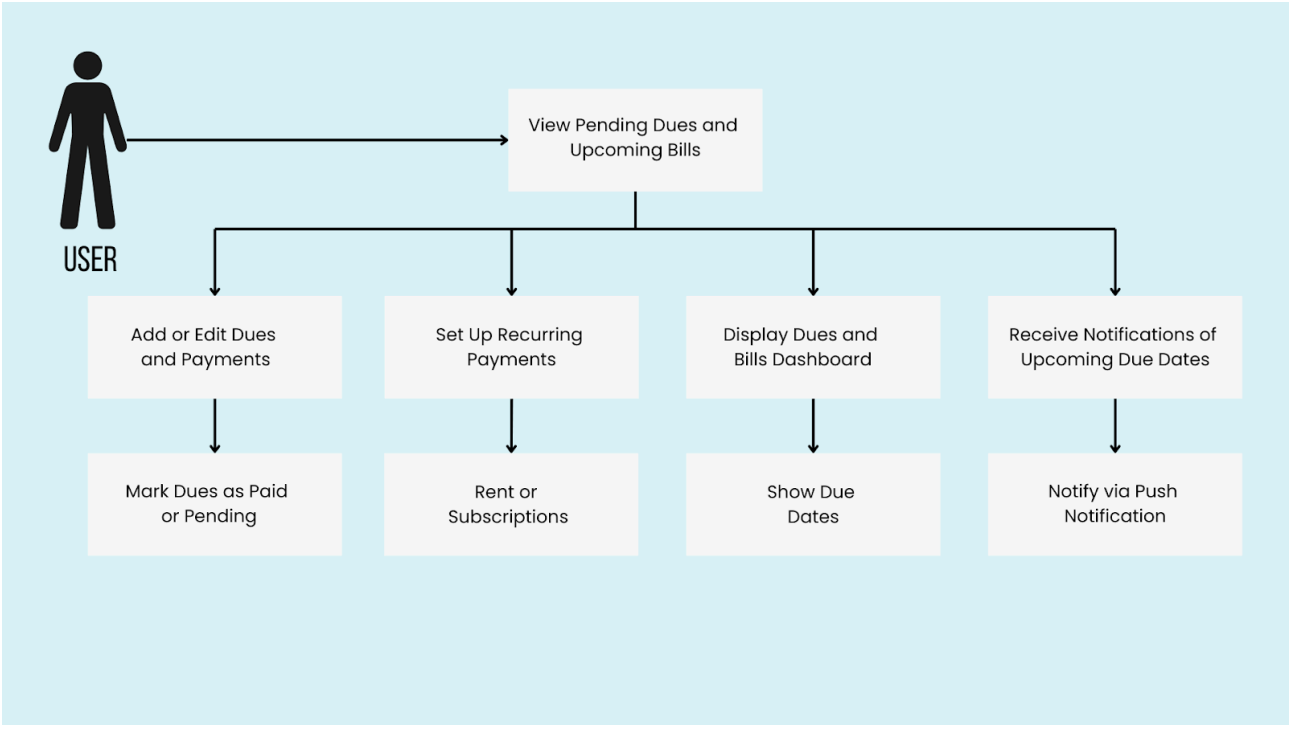
Requirements Traceability: 3.2.7

Priority: High - Crucial for managing bills and avoiding missed payments, impacting user financial health.

Preconditions: The system must have existing dues and bill data (either manually entered or imported). Recurring payments must be set up in the system if users wish to track them.

Post-conditions: Dues and bills are displayed on the dashboard. Users are notified of upcoming due dates. Recurring payments are tracked and notified appropriately.

Actors: User, System



3.3.9 Use Case #9: Split Bills and Track Debts (U9)

Authors: This use case was written by Darshan Sethia and Dhriti Barnwal.

Purpose: To split bills and track owed or borrowed amounts.

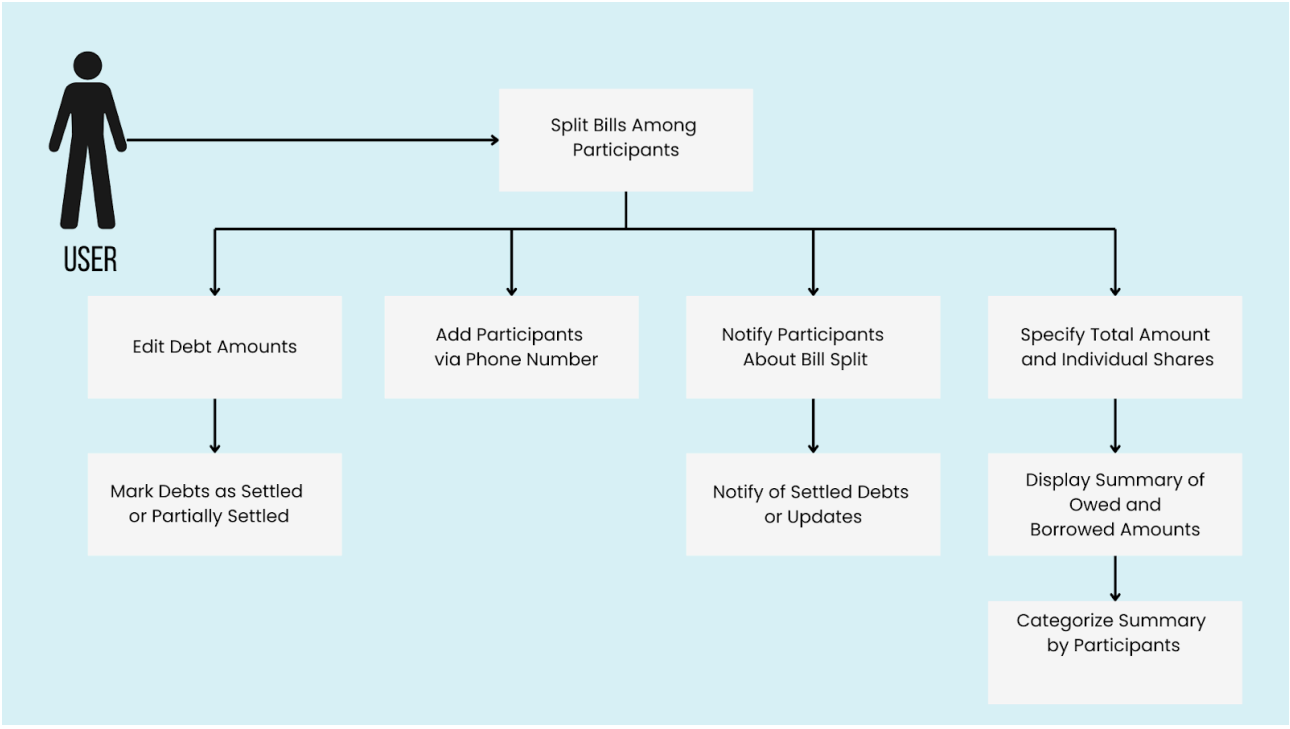
Requirements Traceability: 3.2.6

Priority: Medium - Helpful for managing shared expenses but not critical to core financial tracking.

Preconditions: The system must have an interface for splitting bills and managing debts. The system must have a notification system to inform participants about updates (email, SMS, or app notifications).

Post-conditions: Bills are split and tracked. Debt amounts are updated and participants are notified. Settled debts are marked appropriately.

Actors: Users, System



4 Other Non-functional Requirements

4.1 Performance Requirements

- **Concurrency** - The application can seamlessly support 10,000 concurrent users.
- **Responsiveness** - Average response time below 3 seconds at launch for a seamless user experience. Average response time under 500 milliseconds during normal operation.
- **Resource Efficiency** - Adhere to industry best practices for storage, memory, and processing to minimize resource consumption.

4.2 Safety and Security Requirements

- **Login Mechanism**: Proper login mechanisms will be implemented to prevent unauthorized access or hacking attempts.
- **Database Backup**: Periodic and secure backups of the entire database will be taken to ensure no loss of data.
- **Secure Data Transmission**: All information will be transmitted securely and reliably to prevent data breaches during communication.
- **Password Storage**: Passwords will be securely stored using salting and hashing techniques. This ensures that the system admin can only verify if the password is correct but will not know the actual password submitted by the user, protecting against eavesdropping.
- **Minimum Platform Standards**: The application will support a minimum set of mobile operating system versions that meet the required safety standards, including:
    - Android 8+
    - iOS 13+
- **Transaction Notifications**: A notification will be sent to the concerned users whenever a transaction is added to ensure transparency and awareness.
- **Password Strength Enforcement**: The system will enforce strong passwords by preventing users from setting passwords that meet any of the following criteria:
    - Length is less than 8 characters.
    - Contains only lowercase or uppercase alphabets.
    - Uses commonly known phrases, examples of which will be stored in a database for reference.
- **Forgot Password Feature**: The "Forgot Password" feature will include authentication via security questions to minimize the risk of misuse.

4.3 Software Quality Attributes

4.3.1 Maintainability

- The architecture, design, implementation, and documentation of the software must be such that they minimize maintenance overhead as much as possible.
- Fixing a security defect, including updating the documentation and testing, must not take more than two working days. This will be achieved by ensuring the software is modular and well-structured.
- The average work time required to add a minor feature, including documentation and unit testing, should not exceed one person-week.

4.3.2 Availability

- In case of a server crash, the system state must be restored within two hours to avoid disruption to users.
- The app must handle a surge in user activity at the end of each month, as many users analyze their budgets and expenses during this period.
- During peak times, such as tax season or holidays when shared expenses are higher, the system should provide full functionality without any performance degradation.

4.3.3 Reliability

- The **Mean Time to Failure (MTTF)** shall be more than one week to ensure consistent and reliable service.
- The system must undergo extensive feature testing, load testing, and regression testing before release or deployment to identify and resolve any issues.
- The system must provide accurate financial calculations and consistently deliver correct results to users.

4.3.4 Portability

- The front end will be designed using **Flutter**, ensuring compatibility across various devices, including, tablets, and mobile phones both Android and i
- Brokeo will support Android and iOS platforms to ensure portability across diverse user devices.

Appendix A – Data Dictionary

1 User Class

Description	Represents all users of the application. This master class contains the fundamental details of each user.
Class Variables	- name : string - mail_id : string - phone_number : string - user_id : int - transaction_list : list of transaction objects - category_list : list of category objects
Methods	- register_user - login - Logout - delete_user - edit_user - add_category - delete_category - modify_category - get_transactions - get_categories
Relationships	This is the main user class containing all core information about the user.

2 Spending Category Class

Description	Represents spending categories for tracking and organizing user expenses.
Class Variables	- category_name : string - category_id : int - budget_limit : float
Methods	- create_category - set_budget

Relationships	This class enables the User Class to manage categories and their respective budgets.

3 Transaction Class

Description	Represents a financial transaction within the system, containing relevant information for effective tracking and organization.
Class Variables	- transaction_id : int - amount : float - date : date - description : string - category_id : int - user_id : int - split_with : dict (key: phone number as string, value: amount owed as float)
Methods	- create_transaction - modify_transaction - delete_transaction
Relationships	Linked to both the User Class and the Spending Category Class, as transactions are categorized and associated with specific users.

Appendix B - Group Log

Date	Timings	Duration	Minutes
7th Jan	11:00-12:00	1hr	● Finalised the group name as **Runtimeɜrror** ● Ideas were suggested by everyone. The main ideas proposed were: ○ Campus eBay ○ Carpooling System ○ Study spot tracker ○ Personal Finance Management System ○ Food Delivery System ○ Streamlining Lost and Found
10th Jan	10:00-11:00	1hr	● Preliminary discussion on feasibility and usefulness of the ideas. ● Felt that Financial Management is a problem faced by most students and can be extended to anyone with a budget. ● Decided to implement an application for the same. ● Finalized Personal Finance Management.
12th Jan	18:00-18:30	30mins	● Finalised the description of the project and decided on the name “Brokeo”
17th Jan	10:00-11:00	1hr	● Started work on the SRS document. ● Discussed the UI and application features.
18th Jan	20:00-22:00	2hrs	● Formalized the features of the software. ● Exploration of possible implementation, challenges. ● Discussed the various frameworks to be used in frontend and backend

20th Jan	17:00-20:00	3hrs	● Divided the various sections of the SRS document amongst ourselves. ● Discussed the design of various pages, including the home page, login page etc. on Figma. ● Started working on the written portion of the SRS document
22th Jan	21:00-00:00	3hrs	● Discussed and applied minor design changes. ● Raised and settled contentions regarding various use cases and features, finalising the document. ● Creation of the first draft of the SRS document.
23th Jan	18:00-18:45	45mins	● Discussed the first document draft with our TA Mr. Parag Ghodeshwar.
24th Jan	10:00-11:00	1hr	● Proofread the whole document. ● Raised contentions and corrected them.