# INDEX

# 1

# Introduction

## 1.1 Project Description

The goal of the "Split Expense Management" application is to facilitate seamless management of expenses within groups, providing users with a comprehensive platform to organize and track shared expenses efficiently. By offering various features such as group creation, expense splitting, settlement, and payment integration, the app aims to simplify the process of managing group finances and promoting transparency among members.

The app allows users to create groups effortlessly, enabling them to add members from their contacts and customize group details such as name, profile picture, and estimated budget. This functionality streamlines the setup process, ensuring that users can quickly establish a dedicated space for managing shared expenses with their chosen group of individuals.

Within each group, members can engage in real-time communication through a dedicated chat interface. This feature fosters collaboration and enables users to discuss expenses, coordinate plans, and share important updates seamlessly. By centralizing communication within the app, users can easily stay informed and coordinate effectively with their group members.

Expense management is a key component of the app, empowering users to add expenses directly within the group chat. When adding an expense, users can provide detailed information such as the expense amount, description, payer, and date. Additionally, the app offers multiple options for splitting expenses, including equally, by amount, by percentage, and share-wise, allowing users to distribute costs fairly based on their preferences.

The app provides users with comprehensive visibility into their group's expenses, allowing them to track spending, view detailed breakdowns, and monitor individual contributions. By offering transparent expense tracking features, the app promotes accountability and helps users stay informed about their financial obligations within the group.

To streamline the expense settlement process, the app calculates each user's outstanding balance and facilitates the settlement of debts within the group. Users can easily settle their dues, ensuring that all expenses are accounted for and resolved promptly. This functionality eliminates the need for manual calculations and promotes hassle-free expense management.

Integration with payment gateways enables users to make secure payments to settle their debts conveniently. By supporting popular payment methods and ensuring secure transaction processing, the app provides users with a reliable platform for managing financial transactions within their groups. This functionality enhances user convenience and promotes efficient expense settlement.

## 1.2 Project Profile

| | |
|---|---|
| Project Title : | Split Expense Management |
| Project Type : | Mobile Application |
| Front End : | Flutter Framework |
| Back End : | Firebase Database |
| Database : | Cloud Firestore, Firebase Storage |
| Development Tools : | Android Studio, VS Code |
| Documentation Tool : | Microsoft Word |
| Developed By : | Savaliya Kevin B. |

## 1.3 Project Modules

### a ] Authentication Module:

- Responsible for user registration, login, and authentication using email or social media accounts.

### b ] Group Management Module:

- Allows users to create groups, add members, and manage group details such as name, profile picture, and estimated budget.

### c ] Group Chat Module:

- Provides a platform for real-time communication among group members, facilitating discussions, sharing updates, and coordinating plans.

### d ] Expense Management Module:

- Enables users to add expenses within the group chat, specifying details such as amount, description, payer, and date.
- Offers various methods for splitting expenses among group members, including equally, by amount, by percentage, and share-wise.

### e ] Split Expense Tracking Module:

- Allows users to track group expenses, view detailed breakdowns, and monitor individual contributions, promoting transparency and accountability.

### f ] Expense Settlement Module:

- Facilitates the settlement of expenses within the group, calculating each user's balance and providing options for settling dues.

### g ] Payment Integration Module:

- Integrates with payment gateways to enable users to make secure payments for settling their expenses conveniently.

# 2

# Environment Description

## 2.1 Hardware And Software Requirements

- **Operating System :**
  - Windows : 7 or higher
  - MAC : OS X v10.7 or higher
  - Linux : ubuntu

- **Hardware Specification :**
  - Processor : Minimum 1GHz, recommended 2GHz or more
  - Ethernet connection (LAN) or wireless adapter (Wi-Fi)
  - Hard drive : Minimum 32 GB, recommended 64GB or more
  - Memory (RAM) : Minimum 1 GB, recommended 4GB or more
  - Keyboard & mouse

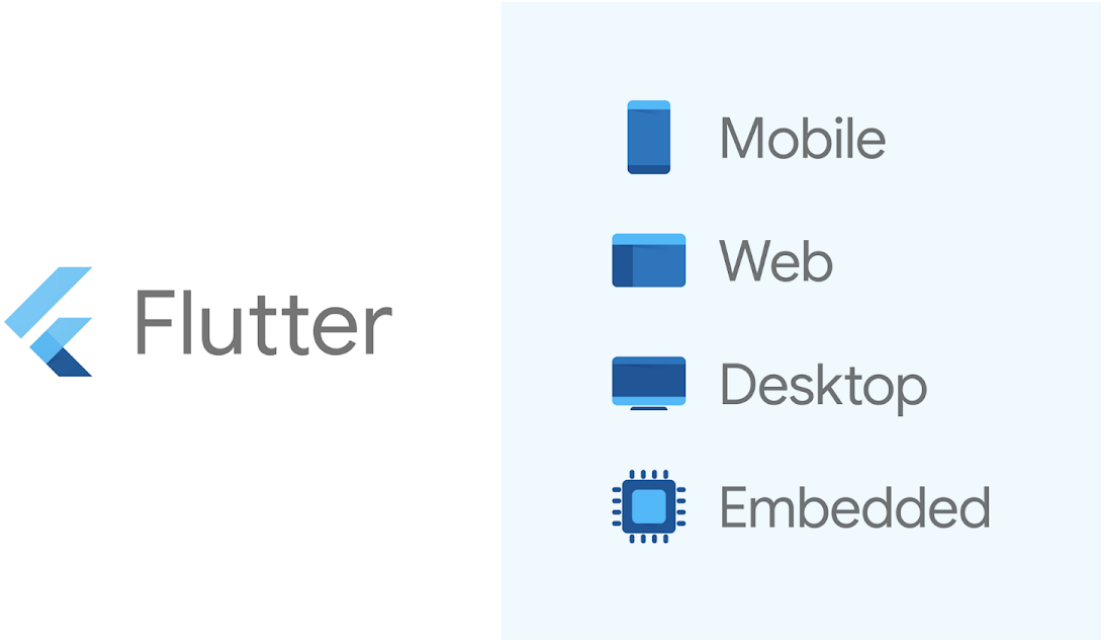
- **Software Specification :**
  - Android Studio IDE
  - Visual Studio Code IDE
  - Recommended Browsers :
    - Google Chrome
    - Microsoft Edge
    - Safari
    - Internet Explorer
    - Firefox

## 2.2 Technologies Used

### ❖ Frontend Technology :

#### 1 ) Flutter Framework :

Flutter is Google's portable UI toolkit for crafting beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source.




- **Advantages of Flutter :**

  - **Dart Programming Language :**

    Flutter uses Dart as its primary programming language. Dart is an object oriented, class-based language with C-style syntax.

    Dart offers features such as strong typing, asynchronous programming, and a reactive programming style, making it well-suited for building modern, reactive user interfaces.

- **Widget-based Framework:**

  Flutter uses a widget-based approach to building user interfaces. Everything in Flutter is a widget, including structural elements like buttons and containers, layout elements like rows and columns, and even entire screens.

  Widgets are immutable and declarative, meaning that instead of changing the properties of existing widgets, you create new widgets to reflect changes in the UI.

- **Hot Reload:**

  One of the most powerful features of Flutter is its hot reload capability. Hot reload allows developers to make changes to their code and see the results instantly reflected in the running app without losing the app state.

  This rapid iteration cycle significantly speeds up the development process and enables developers to experiment and iterate on UI designs quickly.

- **Cross-platform Development:**

  Flutter enables cross-platform development, allowing developers to write code once and deploy it to multiple platforms, including iOS, Android, web, and desktop (Windows, macOS, Linux).

  By sharing a single codebase, developers can save time and effort in maintaining separate codebases for different platforms.

- **Performance:**

  Flutter provides high-performance rendering using its own graphics engine called Skia. Skia is a mature, cross-platform 2D graphics library used by various projects, including Google Chrome and Android.

  Flutter's architecture allows it to achieve consistent 60 frames per second (fps) performance, resulting in smooth and fluid animations and interactions.

- **Customization and Flexibility:**

  Flutter offers extensive customization options, allowing developers to create highly polished and visually stunning user interfaces.

Developers can customize every aspect of their app's UI, including animations, transitions, and gestures, using Flutter's rich set of built-in widgets and APIs.

- **Features of Flutter :**
    - Easy connection of back-end and asynchronization.
    - Fast and responsive layout.
    - Superior execution application
    - Runs same UI for numerous stages.
    - Colossal gadget list.
    - Delightful and liquid UIs.
    - Quick turn of events.
    - Present day and receptive structure.

Since the launch of Flutter in May 2017, it has resolved many of the existing problems in the app development industry. Flutter is a powerful technology, or we can say a tool backed by Dart language packed with a powerful mobile framework that can be used in both iOS and Android applications. Flutter is often used with DART, which is an object-oriented programming language by Google. The flutter development tools come with a graphics library and material design, and the Cupertino design allows faster operations of the app and also gives the app a stunning look, irrespective of its operating platform! The biggest advantage of flutter is that it can be used to create cross-platform apps. Using flutter, one can create iOS apps, Android apps, Websites, and much cross-platform software in just one go, there is no need to write code for different platforms.

1. iOS Apps:

These apps are made for Apple devices and wear. iOS apps are made using the Swift language. The iOS apps have an extension of .ipa.

2. Android Apps:

These apps are made for android devices and wear. Android apps are made using Java and Kotlin, with an extension of .apk. Many app developers who had to work in a cross-platform work environment, and are responsible for the development of both android and iOS apps, found it a difficult and lengthy process to develop apps for both platforms. The major problems encountered by companies and developers were:

No Cross-Platform Dependency: iOS and Android apps work very differently internally, so the developers had to redesign and reconfigure the same content for individual platforms.

Time Constraints: Making a professional app, from coding to designing, requires a lot of time. Companies usually set a time limit by which the app should be ready to be launched into the market. Those developers who had to work on both these platforms often found time limit issues, and the efficiency and quality of work degraded.

More Employees: This problem was encountered by companies. Since they have to develop an app for both platforms, a greater number of app developers knowing about the individual platform had to be hired.

## ❖ Backend Technology :

### 1 ) Firebase :

Firebase is a comprehensive mobile and web development platform provided by Google, offering a wide range of tools and services to help developers build high-quality apps, grow their user base, and earn revenue. Firebase provides a unified platform for backend services, analytics, authentication, cloud messaging, remote configuration, and more, allowing developers to focus on building great user experiences without worrying about managing infrastructure or scaling their backend.

Firebase offers a suite of backend services, including Realtime Database, Cloud Firestore, Cloud Functions, Authentication, Cloud Storage, and Hosting. These services provide developers with scalable and reliable infrastructure for storing data, managing user authentication, and executing server-side logic.


- **Features of Firebase :**
  - **Backend Services:**

Firebase offers a suite of backend services, including Realtime Database, Cloud Firestore, Cloud Functions, Authentication, Cloud Storage, and Hosting. These services provide developers with scalable and reliable infrastructure for storing data, managing user authentication, and executing server-side logic.

- **Analytics:**

  Firebase Analytics provides insights into user behavior and app performance, allowing developers to understand how users interact with their app, track user engagement, and measure key metrics such as retention, conversion, and revenue.

- **Authentication:**

  Firebase Authentication offers easy-to-use authentication solutions, including email/password authentication, social authentication (using providers like Google, Facebook, Twitter), phone number authentication, and anonymous authentication. It allows developers to securely authenticate users and manage user identities across platforms.

- **Cloud Messaging:**

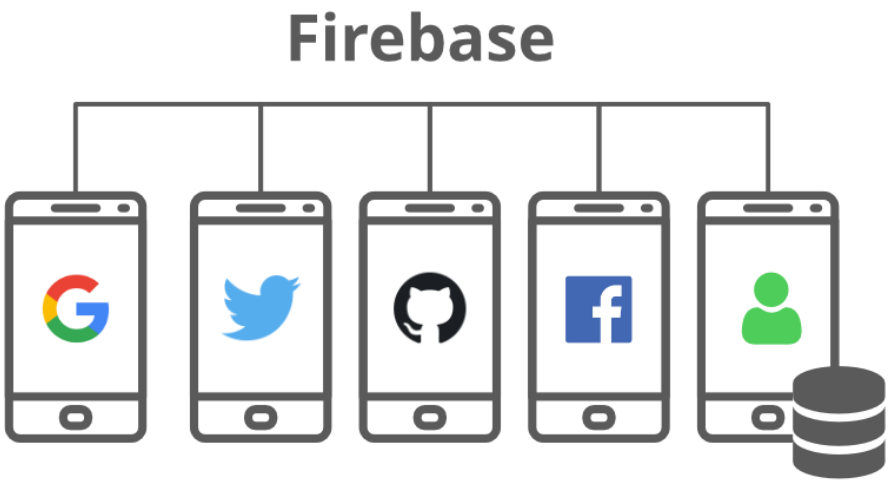
  Firebase Cloud Messaging (FCM) enables developers to send targeted messages and notifications to users across platforms, including Android, iOS, and web. It supports various message types, including notifications, data messages, and topic-based messaging, helping developers engage and re-engage users effectively.

- **Remote Configuration:**

  Firebase Remote Config allows developers to remotely configure app settings, feature flags, and parameters without requiring app updates. It enables dynamic content personalization, A/B testing, and staged rollouts, empowering developers to optimize app experiences based on user feedback and behavior.

- **Cloud Storage:**

  Firebase Cloud Storage provides secure and scalable cloud storage for storing user-generated content such as images, videos, and files. It offers powerful storage capabilities, including multi-region replication, access control, and integration with Firebase Authentication.




## ❖ Cloud Firestore Database :

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.

Cloud Firestore is newer, but it is not replacing the Firebase Real-time Database. Cloud Firestore is a flexible as well as scalable NoSQL cloud database. It is used to store and sync data for client and server-side development. It is used for mobile, web, and server development from Google Cloud Platform and Firebase. Like the Firebase Real-time Database, it keeps syncing our data via real-time listeners to the client app. It provides offline support for mobile and web so we can create responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also provides seamless integration with Google Cloud Platform products and other Firebase, including cloud functions.

After Cloud Firestore's NoSQL data model, we can store data in documents that have field mappings for values. The documents are stored in a container called collections. These containers are used to organize our data and create queries. There are different data types, from simple string and numbers to complex nested objects, supported by documents. We can also create sub-collection within a document and create a hierarchical data structure that scales to the growth of our database. The Firestore data model supports whatever data structure works best for our app.

## ❖ Payment Gateway Integration :

### 1 ) Razorpay Integration in Flutter :

Razorpay is a leading payment gateway provider in India, offering a secure, reliable, and feature-rich platform for processing online payments. It enables businesses to accept payments seamlessly across multiple channels, including websites, mobile apps, and more.

- **Payment Methods :**

  - **Credit and Debit Cards :**

    Customers can make payments using their credit or debit cards issued by Visa, Mastercard, Maestro, American Express, and other major card networks. Razorpay supports both domestic and international card payments.

  - **Net Banking :**

    Razorpay supports payments via net banking, allowing customers to use their bank accounts to make online payments directly from their bank's website or mobile app. It offers integration with numerous banks across India for seamless net banking payments.

  - **Unified Payments Interface (UPI):**

    UPI is a popular payment method in India that allows users to transfer money between bank accounts instantly using their mobile phones. Razorpay supports UPI payments, enabling customers to make payments using UPI apps such as Google Pay, PhonePe, Paytm, BHIM, and others.

- **Wallets :**

  Razorpay supports payments through various digital wallets, providing customers with the convenience of making quick and secure payments using their wallet balances. Supported wallets include Paytm, PhonePe, Google Pay, FreeCharge, Mobikwik, and others.

- **EMI Options:**

  Razorpay offers EMI (Equated Monthly Installments) options for customers who wish to pay for their purchases in installments over time. Customers can choose from a range of EMI plans offered by their banks and financial institutions at the time of checkout.

- **International Payments:**

  Razorpay supports international payments, allowing businesses to accept payments from customers located outside India. Customers can make payments using their international credit or debit cards, and transactions are processed in multiple currencies.

- **QR Code Payments:**

  Razorpay offers QR code-based payment solutions, allowing businesses to generate QR codes that customers can scan using their mobile banking apps or UPI apps to make payments quickly and securely.

# 3

# System Analysis And planning

## 3.1 Existing System And Its Drawbacks

❖ **Existing Systems :**

**Splitwise :**

Splitwise is a popular expense-sharing application that simplifies the process of splitting bills and expenses among groups of friends, roommates, or colleagues. It offers a user-friendly platform with features for creating groups, adding expenses, splitting bills, tracking balances, and settling debts. Splitwise is available on web and mobile platforms, with apps for both iOS and Android devices.

**Key Features:**

- **Group Creation:**

  Users can create groups for various purposes, such as trips, rent sharing, or household expenses.

- **Expense Management:**

  Users can add expenses to the group, specifying details such as amount, description, and payer.

- **Expense Splitting:**

  Splitwise offers various methods for splitting expenses, including equally, by shares, by percentage, and custom amounts.

- **Real-time Balances:**

  The app provides real-time updates on balances and settlements, allowing users to track their contributions and debts accurately.

- **Settlements:**

  Splitwise facilitates settlements between group members, enabling users to record payments and mark debts as settled.

- **Notifications:**

  Users receive notifications for new expenses, reminders for pending settlements, and updates on group activity.

## ❖ Drawbacks of Splitwise :

- **Limited Payment Options :**

  Splitwise primarily focuses on expense tracking and splitting, but it does not offer built-in payment functionality. Users need to settle debts manually outside the app, which can be inconvenient, especially for large groups or frequent expenses.

- **Dependency on Manual Settlements :**

  Splitwise relies on manual settlements between users, requiring them to transfer funds independently using external payment methods. This can lead to delays, discrepancies, or misunderstandings during the settlement process, particularly if users forget to mark debts as settled or fail to communicate effectively.

- **Lack of Integration with Payment Gateways :**

  Splitwise lacks integration with payment gateways or financial platforms, making it challenging for users to make direct payments or transfers within the app. Users may need to rely on third-party services or traditional payment methods to settle debts, adding complexity to the process.

- **Limited Platform Support:**

  While Splitwise offers mobile apps for iOS and Android devices, it does not provide native desktop or web browser support. Users may prefer accessing Splitwise from a web browser for certain tasks or managing expenses on larger screens, which is not currently supported by the platform.

- **Limited Export and Reporting Options:**

  Splitwise provides basic export and reporting options for generating expense reports or sharing data with group members. However, users may find these options limited in terms of customization, formatting, or data analysis capabilities, especially for advanced reporting needs or business use cases.

- **Dependency on Internet Connectivity:**

  Like many cloud-based applications, Splitwise relies on internet connectivity for data synchronization, updates, and notifications. Lack of internet access or poor network connectivity can disrupt users' ability to access or update expense information, particularly in offline or remote environments.

## 3.2 Feasibility Study

- **Technical Feasibility :**

  - Assessment of Technology Stack:
    Evaluate the suitability of Flutter framework for cross-platform development, as well as the compatibility of chosen technologies such as Firebase for backend services and Razorpay for payment integration.

  - Availability of Resources:
    Determine if the necessary technical expertise, development tools, and infrastructure are readily available or can be acquired within the project timeline and budget.

  - Scalability and Performance :
    Consider whether the chosen technologies can scale to accommodate increasing user traffic and data volume without compromising performance.

- **Market Feasibility :**

  - Market Analysis **:**
    Conduct market research to understand the demand for split expense applications, identify target demographics, and analyze existing competitors in the market.

  - User Needs and Preferences:
    Determine the key features and functionalities that users expect from a split expense application, such as expense tracking, group management, payment integration, and collaboration tools.

  - Market Trends and Opportunities:
    Identify emerging trends, market gaps, and potential opportunities for innovation within the split expense application space.

- **Financial Feasibility :**

  - Cost Estimation:

Estimate the initial investment required to develop the split expense application, including development costs, licensing fees, infrastructure expenses, and marketing expenses.

- Revenue Model:

  Define the revenue model for the application, such as subscription fees, transaction fees, freemium models, or monetization through ads or premium features.

- Profitability Analysis:

  Evaluate the projected revenues and expenses to determine the profitability of the split expense application over time and assess the return on investment (ROI).

- **Operational Feasibility :**

  - Resource Allocation:

    Assess the availability of human resources, time, and budget for developing, launching, and maintaining the split expense application.

  - Operational Processes:

    Define the operational processes and workflows involved in managing the application, such as user onboarding, customer support, data management, and security measures.

  - Legal and Compliance Considerations:

    Ensure compliance with relevant laws and regulations governing data privacy, payment processing, and financial transactions, as well as adherence to industry standards and best practices.

- **Risk Analysis :**

  - Identification of Risks:

    Identify potential risks and challenges that could impact the success of the split expense application, such as technical issues, market competition, regulatory changes, or financial constraints.

  - Risk Mitigation Strategies:

    Develop strategies to mitigate identified risks, such as contingency plans, risk transfer mechanisms, or alternative approaches to address potential obstacles.

## 3.3 Requirement Gathering And Analysis

- **Interviews :**

We conducted interviews with potential users, including individuals involved in group expenses such as roommates, friends, colleagues, and travelers. Through these interviews, we gathered insights into their pain points, preferences, and desired features for managing shared expenses.

- **Surveys/Questionnaires :**

We distributed surveys and questionnaires to a broader audience to collect quantitative data on common expense-sharing practices, preferred payment methods, and user expectations regarding expense management applications. The survey responses provided valuable insights into user behavior and requirements.

- **Observation :**

We observed real-world scenarios of group expenses and shared financial activities to understand common challenges and behaviors among users. By observing group interactions and expense-splitting practices firsthand, we gained valuable insights into the complexities of managing shared expenses.

- **Document Analysis :**

We analyzed existing split expense applications, user reviews, and industry reports to identify common features, trends, and best practices in expense management. This helped us understand the competitive landscape and benchmark our application against existing solutions.

- **Brainstorming Sessions :**

We conducted brainstorming sessions with the development team and stakeholders to generate ideas, identify potential features, and prioritize requirements based on user needs and business objectives. These sessions

fostered creativity and collaboration, leading to innovative solutions and feature prioritization.

Based on the insights gathered through these techniques, we identified the following key requirements for the split expense :

- User Registration and Authentication
- Group Creation and Management
- Adding and Editing Expenses
- Expense Splitting Methods (Equal, Unequal, Percentage, Shares)
- Real-time Updates on Balances and Settlements
- Integration with Payment Gateways (e.g., Razorpay)
- Notification and Reminder System
- Expense Categorization and Reporting
- User-friendly Interface and Navigation
- Cross-platform Compatibility (iOS, Android, Web)

By meticulously gathering and analyzing requirements through these methods, we ensured that the split expense addresses the diverse needs of users and provides a seamless and intuitive experience for managing shared expenses effectively.

# 4

# Proposed System

## 4.1 Scope

- Users can register and login to the application.
- Users can create and manage expense groups.
- Users can add expenses within groups and categorize them.
- The application supports various expense splitting methods (equal split, percentage split, etc.).
- Real-time synchronization of expense data across group members.
- Users receive notifications for new expenses and settlement requests.
- Integration with payment gateways for secure transactions.
- Users can settle expenses using preferred payment methods.
- The application offers tools for tracking spending patterns and analyzing expenses.
- Users can manage their profiles and notification preferences.
- Administrators can manage users, groups, and expenses.
- Administrators can enforce rules, resolve disputes, and monitor activity.
- The application implements security measures to protect user data and transactions.
- Develops applications for multiple platforms (e.g., iOS, Android, web).
- Ensures consistent user experience across devices.

## 4.2 Project Modules

### ❖ User Module :

- User Authentication and Authorization
- Group Management
- Expense Management
- Expense Splitting
- Real-time Updates and Notifications
- Payment Integration
- Expense Analysis and Reporting
- User Profile Management
- Admin Panel
- Feedback and Support
- Security and Privacy
- Cross-platform Compatibility

## 4.3 Module Wise Objectives

### ❖ User Module :

- **User Registration and Authentication :**
  - ✓ Handles registration and authentication for all users.
  - ✓ Manages user roles and permissions.

- **Expense Group Management :**
  - ✓ Allows users to create and manage expense groups.
  - ✓ Enables group admins to configure group settings and manage members.

- **Expense Tracking and Management :**
  - ✓ Facilitates adding, editing, and deleting expenses within groups.
  - ✓ Supports categorization, descriptions, and attachment of receipts.

- **Expense Splitting Methods :**

- ✓ Implements various expense splitting methods (e.g., equal split, percentage split).

- **Real-time Updates and Notifications :**
  - ✓ Provides real-time synchronization of expense data.
  - ✓ Sends notifications for new expenses, updates, and settlement requests.

- **Payment Integration :**
  - ✓ Integrates with payment gateways for secure transactions.
  - ✓ Allows users to settle expenses using preferred payment methods.

- **Expense Analysis and Reporting :**
  - ✓ Offers tools for tracking spending patterns and analyzing expenses.
  - ✓ Provides visualizations and reports for expense management.

- **User Profile Management :**
  - ✓ Enables users to manage their profiles and settings.
  - ✓ Allows customization of notification preferences.

- **Admin Panel :**
  - ✓ Provides administrators with access to manage users, groups, and expenses.
  - ✓ Allows admins to enforce rules, resolve disputes, and monitor activity.

- **Feedback and Support :**
  - ✓ Includes features for users to provide feedback and support requests.
  - ✓ Offers customer support channels for assistance and guidance.

- **Security and Privacy :**
  - ✓ Implements security measures to protect user data and transactions.
  - ✓ Ensures compliance with data privacy regulations.

- **Cross-platform Compatibility :**
  - ✓ Develops applications for multiple platforms (e.g., iOS, Android, web).
  - ✓ Ensures consistent user experience across devices.

## 4.4 Expected Advantage

Split expense application offers several advantages to its users, enhancing the management of shared expenses within groups. Here are some expected advantages of using a split expense application:

- **Convenience and Efficiency :**
  - ✓ Simplifies the process of tracking and managing shared expenses among group members.
  - ✓ Provides a centralized platform for adding, categorizing, and splitting expenses, reducing the need for manual calculations and coordination.

- **Transparency and Accountability :**
  - ✓ Promotes transparency by providing real-time updates on expenses, balances, and settlements within the group.
  - ✓ Enhances accountability by clearly documenting each expense and its contributors, reducing disputes and misunderstandings.

- **Fairness in Expense Distribution:**
  - ✓ Offers various expense splitting methods, allowing users to divide expenses equitably based on their preferences and agreements.
  - ✓ Ensures fairness in expense distribution by providing flexibility in choosing splitting methods (equal split, percentage split, custom split, etc.).

- **Cost Sharing and Budget Management :**
  - ✓ Facilitates cost-sharing arrangements among group members for shared expenses such as rent, utilities, groceries, and travel expenses.
  - ✓ Helps users effectively manage their budgets by tracking spending, setting limits, and analyzing expense patterns over time.

- **Streamlined Payment Settlements :**
  - ✓ Integrates with payment gateways to facilitate seamless and secure transactions for settling shared expenses.

- ✓ Enables users to settle their dues directly within the application, eliminating the need for manual transfers or cash exchanges.

- **Enhanced Communication and Collaboration :**
  - ✓ Improves communication and collaboration among group members by providing features for sharing expense details, discussing transactions, and resolving discrepancies.
  - ✓ Enables users to stay informed about group expenses, contributions, and settlements through real-time notifications and updates.

- **Time and Effort Savings :**
  - ✓ Saves time and effort by automating repetitive tasks such as expense tracking, splitting, and settlement.
  - ✓ Reduces the administrative burden associated with managing shared expenses, allowing users to focus on other priorities and activities.

- **Data Analysis and Insights:**
  - ✓ Offers analytical tools and reports to analyze spending patterns, identify trends, and make informed financial decisions.
  - ✓ Provides valuable insights into group expenses, helping users optimize their spending habits and budget allocations.

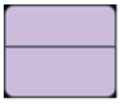
# 5

# Detail Planning

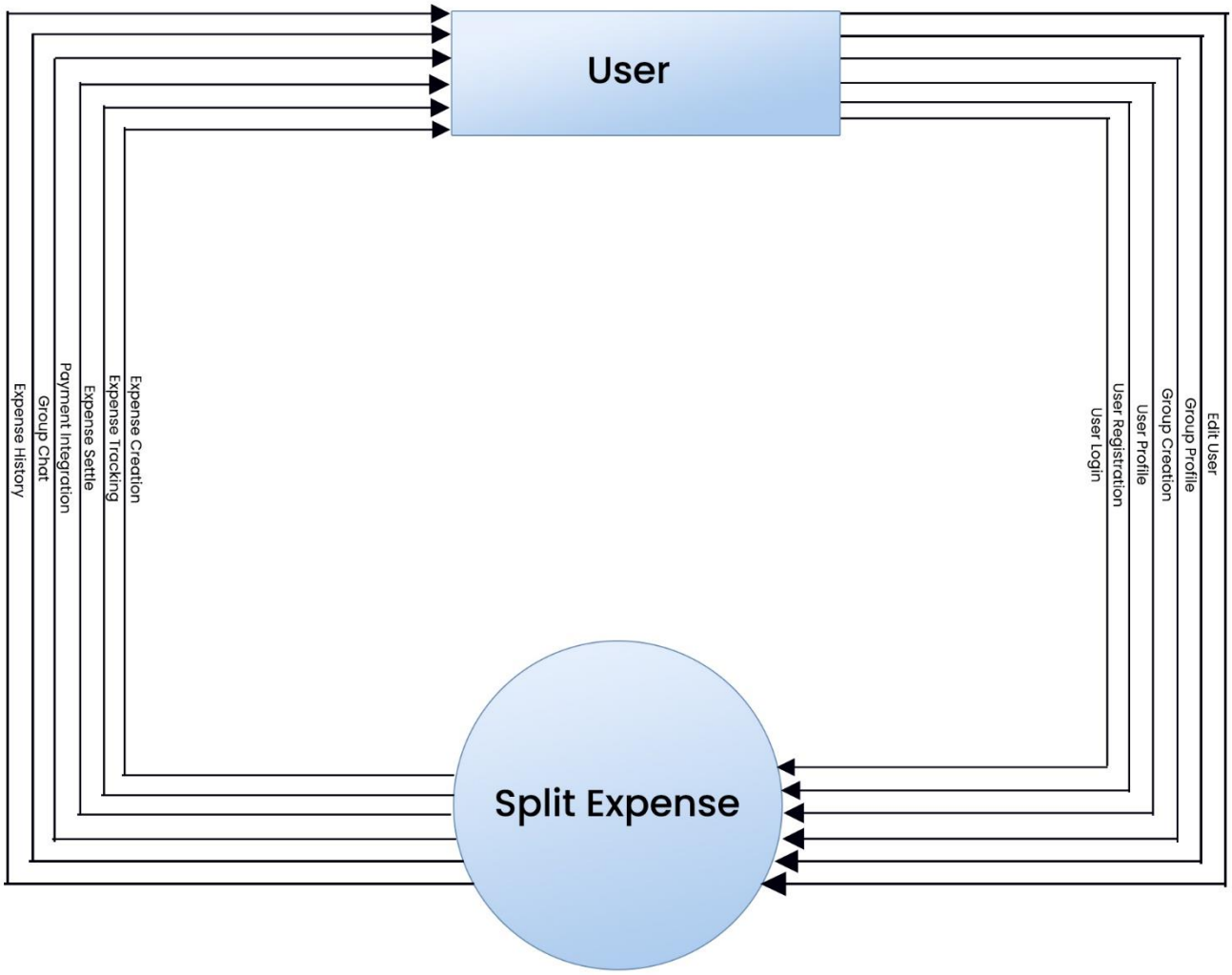
## 5.1 Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data thorough an information system. A data flow diagram can also be used for the visualization of data processing (structure design). It is common practice for a designer to draw a context level DFD first which shows the interaction between the system and outside entities. This context-level DFD id then "exploded" to show more detail of the system being modelled.

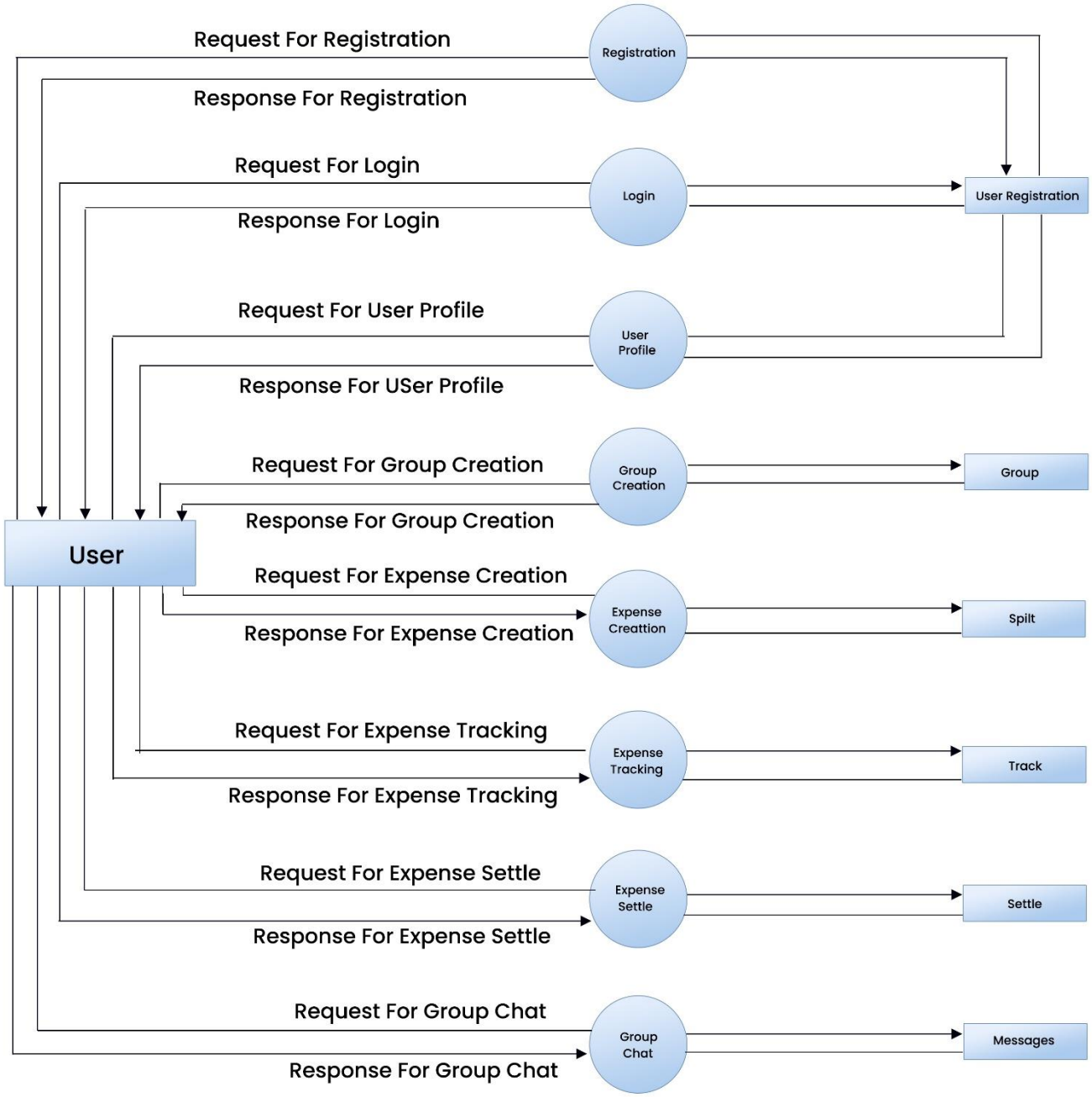
- Modules Used in Data Flow Diagram :-

| Notation | Symbol |
|---|---|
| External Entity |  |
| Process |  |
| Data Store |  |
| Data Flow |  |

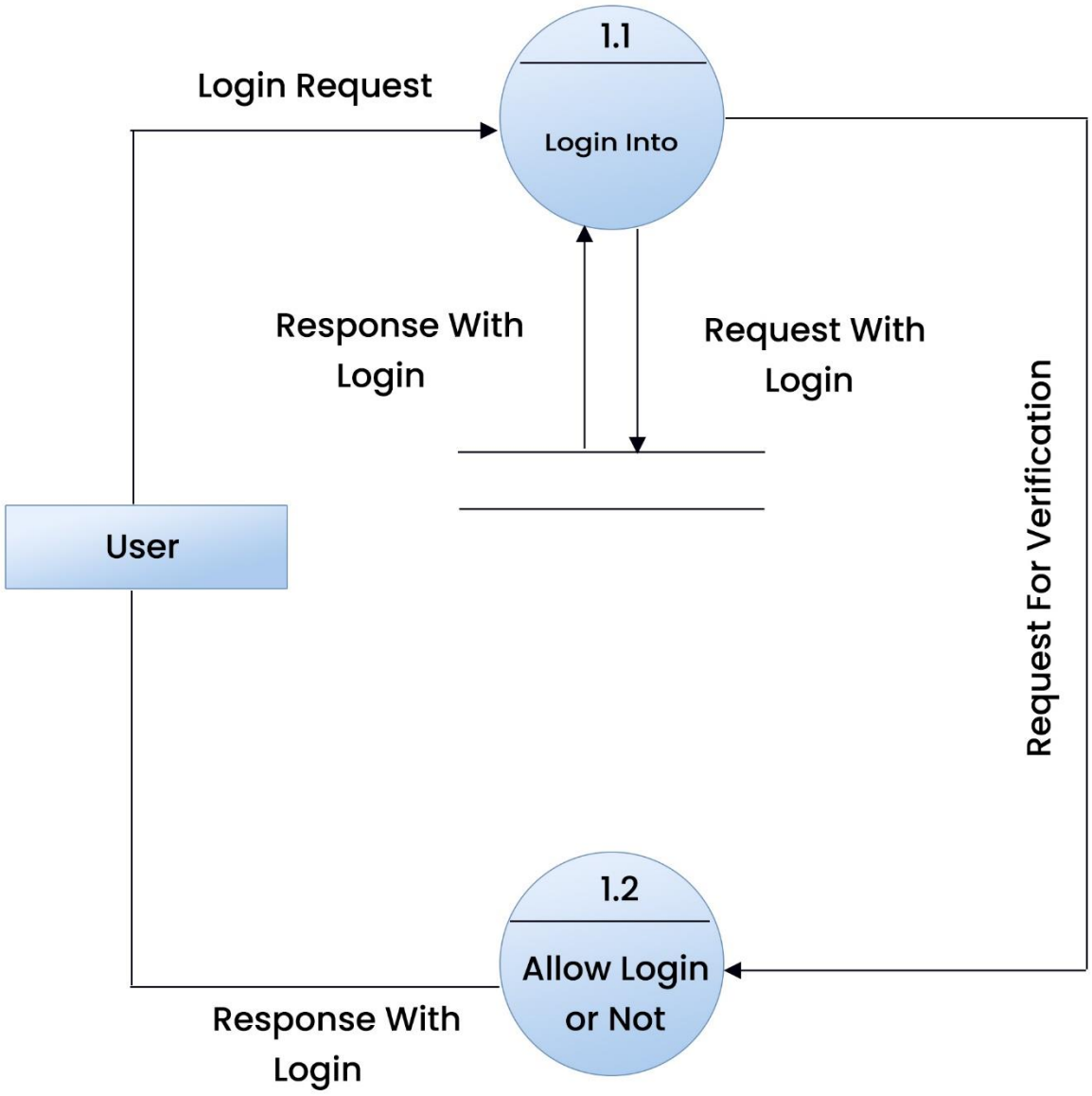
❖ **Context Level of Split Expense Management :**



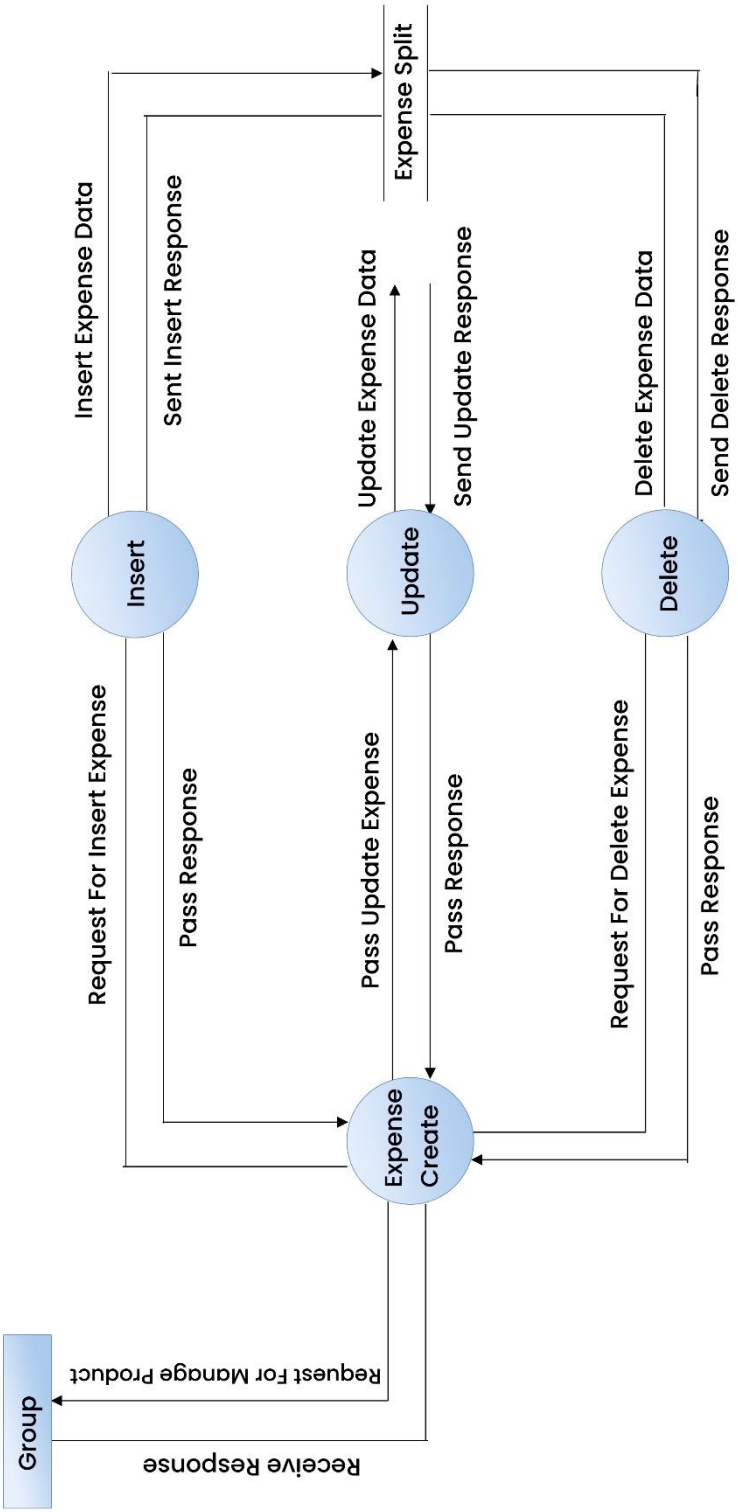
## ❖ 1st Level of Split Expense Management :



❖ **2st Level of Split Expense Management :**

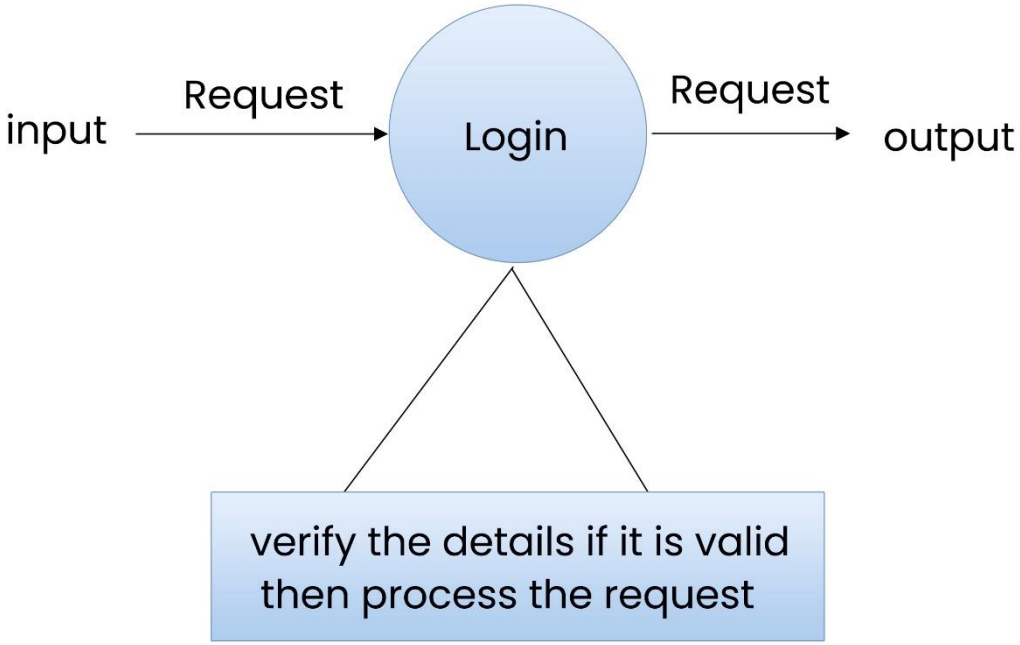


❖ **2st Level of Group Management :**

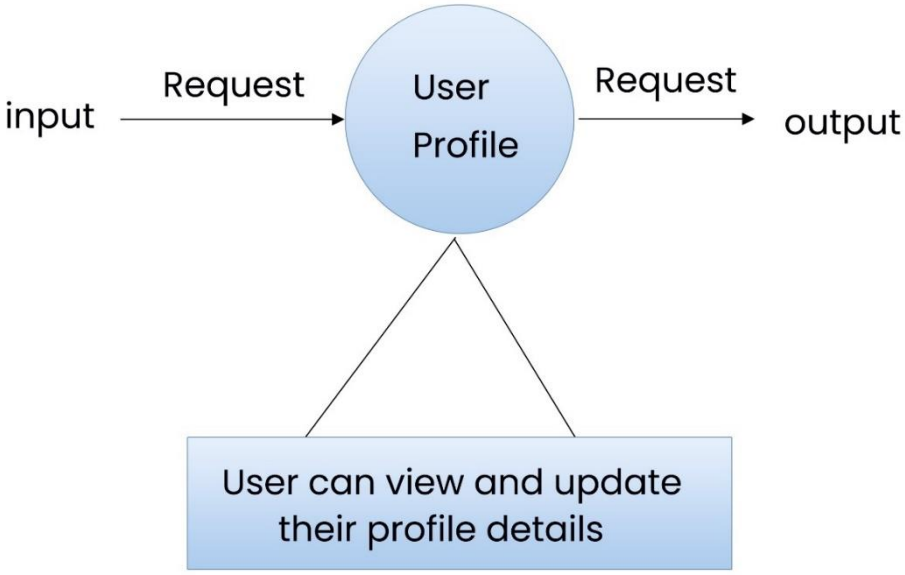


## 5.2 Process Specification / Activity Flow Diagram

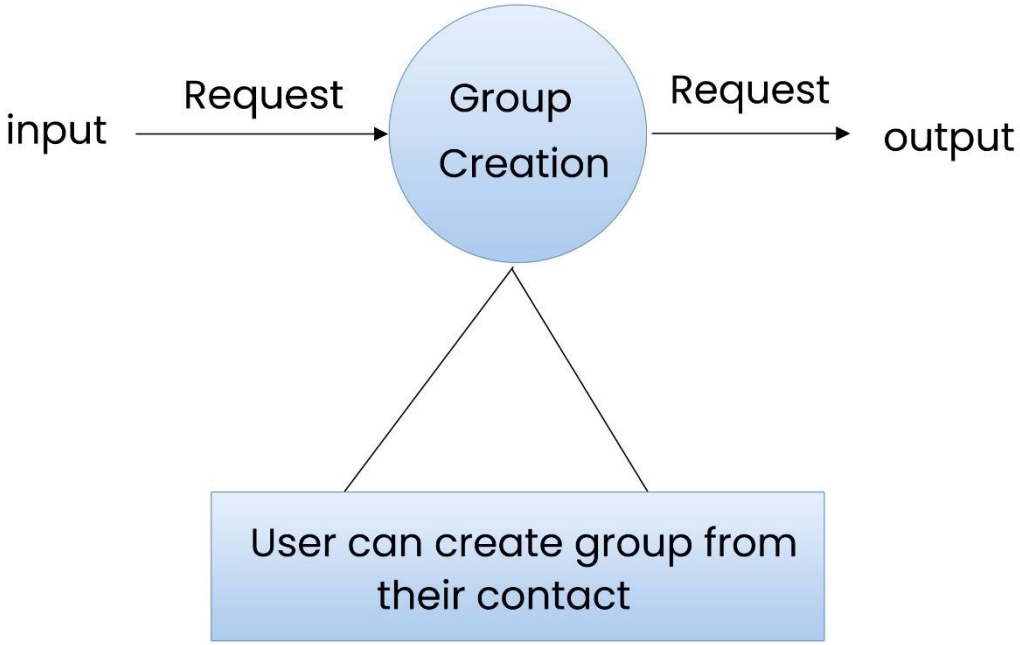
❖ **Login :**



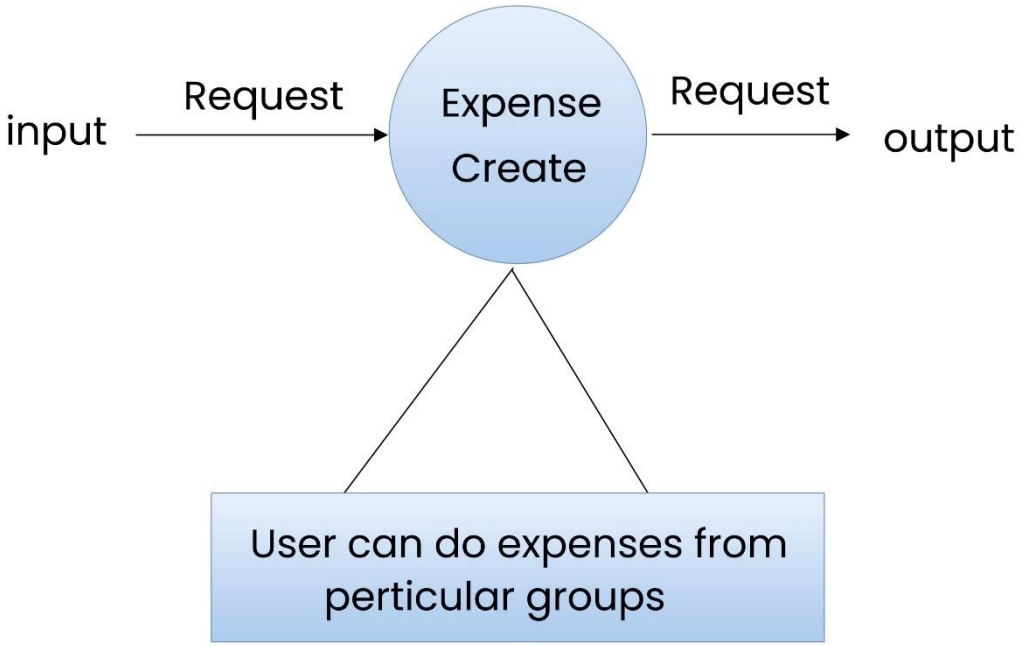

❖ **User Profile :**



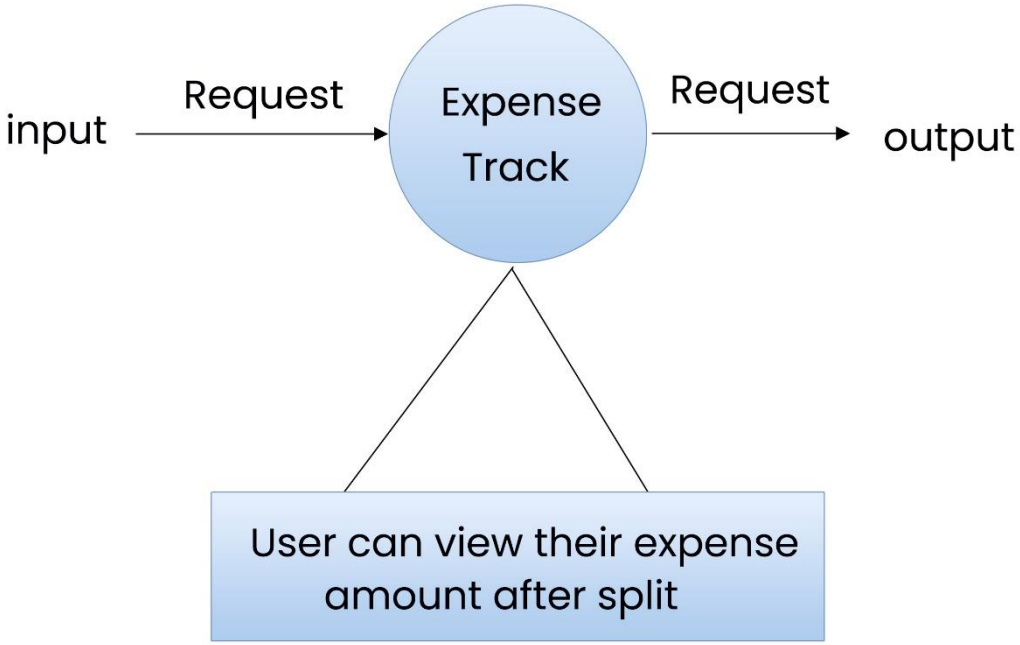
❖ **Group Creation:**



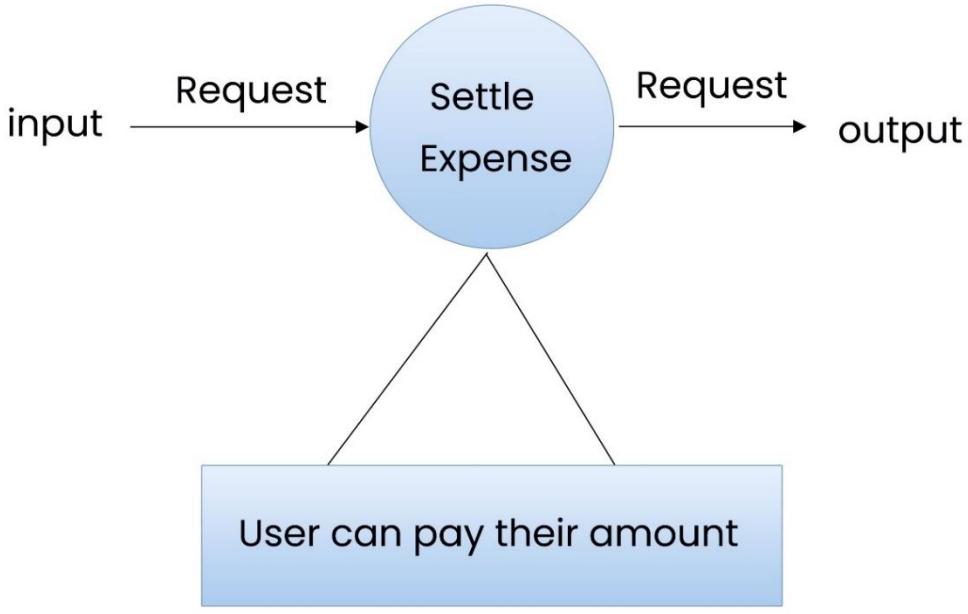

❖ **Expense Creation:**



❖ **Expense Tracking:**




❖ **Settle Expense :**



## 5.3 Data Dictionary

- A data dictionary is a catalogue, a repository of the element system. As the name suggest it is used to know requirement and needs of organization, Thus data dictionary gives the detailed description of the database store.
- Data dictionary used to store data like alias, how used/ where used, and values.
- Data dictionary store data into table form.
- All data stores in table form make it easy to understand our database.

| **Users** | |
|---|---|
| Alias : | User |
| **Where used? /** How used? | To store and retrieve user details |
| Description : | id,name,email,mobileNo,gender,countryCode,currencyCode, profilePicture,avatarId,fcmToken,notification |
| Supplementary information | Id must be unique and login process verify data in this table. |

| **Group** | |
|---|---|
| Alias : | Group |
| **Where used? /** How used? | To store and retrieve group details |
| Description : | id,name,description,budget,groupProfile,memberIds, adminIds,createdAt,lastUpdated |
| Supplementary information | Id must be unique and user can send message and expense. |

| Expense | |
|---|---|
| Alias : | Expense |
| **Where used? /** How used? | To store and retrieve expense details |
| Description : | id,title,amount,payerID,behalfAddUser,createdAt,groupId, tripId,splitMode,percentage,equality,expenseType |
| Supplementary information | Id must be unique and user can add expense. |

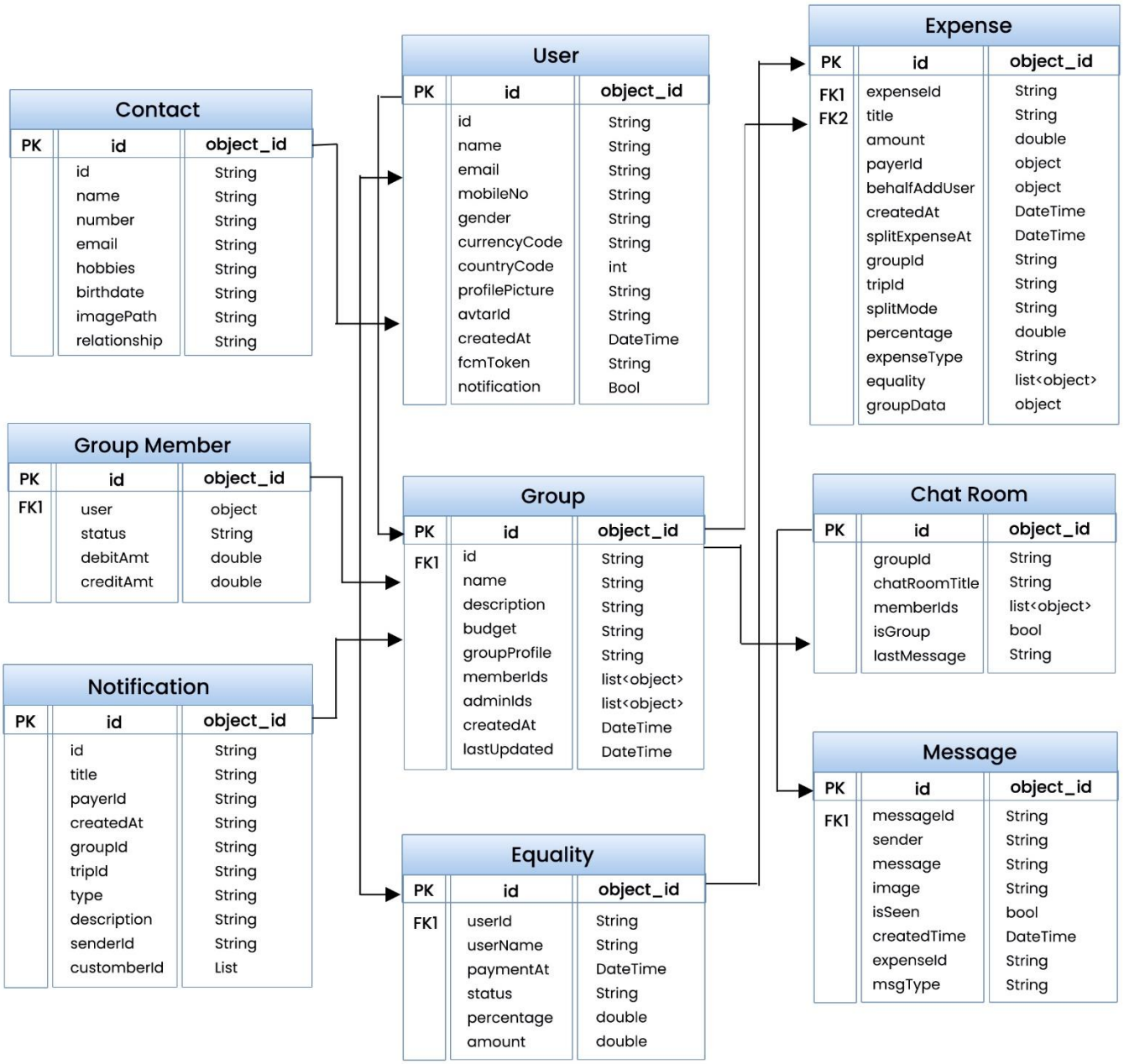
| Message | |
|---|---|
| Alias : | Message |
| **Where used? /** How used? | To send and retrieve message details |
| Description : | messageId,sender,message,image,createdTime,expenseId, msgType, |
| Supplementary information | Send message in groups and retrive it. |

## 5.4 Entity-Relationship Diagram / Class Diagram

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system. An ERD uses data modeling techniques that can help define business processes and serve as the foundation for a relational database.

Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a reference point, should any debugging or business process re-engineering be needed later.

However, while an ERD can be useful for organizing data that can be represented by a relational structure, it can't sufficiently represent semi-structured or unstructured data. It's also unlikely to be helpful on its own in integrating data into a pre-existing information system.


Contact
PK id object_id
id String
name String
number String
email String
hobbies String
birthdate String
imagePath String
relationship String
User
PK id object_id
id String
name String
email String
mobileNo String
gender String
currencyCode String
countryCode int
profilePicture String
avtarId String
createdAt DateTime
fcmToken String
notification Bool
Expense
PK id object_id
FK1 expenseId String
FK2 title String
amount double
payerId object
behalfAddUser object
createdAt DateTime
splitExpenseAt DateTime
groupId String
tripId String
splitMode String
percentage double
expenseType String
equality list<object>
groupData object
Group Member
PK id object_id
FK1 user object
status String
debitAmt double
creditAmt double
Group
PK id object_id
FK1 id String
name String
description String
budget String
groupProfile String
memberIds list<object>
adminIds list<object>
createdAt DateTime
lastUpdated DateTime
Chat Room
PK id object_id
groupId String
chatRoomTitle String
memberIds list<object>
isGroup bool
lastMessage String
Notification
PK id object_id
id String
title String
payerId String
createdAt String
groupId String
tripId String
type String
description String
senderId String
custombeId List
Equality
PK id object_id
FK1 userId String
userName String
paymentAt DateTime
status String
percentage double
amount double
Message
PK id object_id
FK1 messageId String
sender String
message String
image String
isSeen bool
createdTime DateTime
expenseId String
msgType String

# 6

# System Design

## 6.1 Database Design

❖ **User :**

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| id | String | True | Id of User |
| name | String | True | Name of User |
| email | String | True | Email of User |
| mobileNo | String | True | Mobile No. of User |
| gender | String | True | Gender of User |
| currencyCode | String | True | User Currency Code |
| countryCode | Int | True | User Country Code |
| profilePicture | String | True | User Profile |
| avatarId | String | True | User Avatar |
| createdAt | DateTime | True | Created Time |
| fcmToken | String | True | FCM Token |
| notification | Bool | True | Notification Enable |

❖ **Group :**

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| id | String | True | Id of Group |
| name | String | True | Name of Group |
| description | String | True | Description of Group |
| budget | String | True | Budget of Group |
| groupProfile | String | True | Profile of Group |
| memberIds | List<Object> | True | Group members |
| adminIds | List<Object> | True | Group Admin |
| createdAt | DateTime | True | Created Time |
| lastUpdated | DateTime | True | Last Update Time |

❖ **Expense :**

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| expenseId | String | True | Id of Expense |
| title | String | True | Title of Expense |
| amount | Double | True | Expense amount |
| payerId | Object | True | Expense Payer User |
| BehalfAddUser | Object | True | Expense Behalf User |
| createdAt | DateTime | True | Created Time |
| expenseAt | DateTime | True | Expense Time |
| groupId | String | True | Group Id |
| tripId | String | True | Trip Id |
| splitMode | String | True | Split Mode |
| percentage | Double | True | Percentage Data |
| expenseType | String | True | Expense Type |
| equality | List<Object> | True | Equality Data |

### ❖ ChatRoom :

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| groupId | String | True | Id of Group |
| chatroomTitle | String | True | Chat title |
| memberIds | List<Object> | True | Member of Group |
| isGroup | Bool | True | Group or Not |
| lastMessage | String | True | Last Message |

### ❖ Message :

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| messageId | String | True | Id of Message |
| sender | String | True | Sender User |
| message | String | True | Message |
| image | String | True | Image |
| isSeen | Bool | True | Message seen or not |
| createdTime | DateTime | True | Created Time |
| expenseId | String | True | Expense Id |
| msgType | String | True | Message Type |

### ❖ Group Member :

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| user | object | True | User of Group |
| status | String | True | User Status |
| debitAmount | Double | True | Debit Amount |
| creditAmount | Double | True | Credit Amount |

### ❖ Contact Data:

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| id | String | True | Id of Contact |
| name | String | True | Contact Name |
| number | String | True | Contact Number |
| email | String | True | Contact Email |
| hobbies | String | True | Hobbbies |
| birthDate | String | True | User Bdate |
| imagePath | String | True | User Image |
| relationShip | String | True | Relationship |

### ❖ Notification Model :

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| id | String | True | Id of Notification |
| title | String | True | Title of Notification |
| payerId | String | True | Payer |
| createdAt | String | True | Created Time |
| groupId | String | True | Group Id |
| tripId | String | True | Trip Id |
| type | String | True | Type |
| description | String | True | Description |
| senderId | String | True | Sender Id |
| customerId | List | True | Customer Id |

### ❖ Equality Data :

| Field Name | Data Type | Required | Description |
|---|---|---|---|
| userId | String | True | Id of User |
| userName | String | True | User Name |
| paymentAt | DateTime | True | Payment At |
| status | String | True | Status |
| percentage | Double | True | Percentage |
| amount | Double | True | Amount |

## 6.2 Directory Structure

❖ **Controllers :**

That perform all the funtionality related operations.

Controller

|

|-- auth_controller.dart

|

|-- user_controller.dart

|

|-- home_controller.dart

|

|-- intro_controller.dart

|

|-- splash_controller.dart

|

|-- group_controller.dart

|

|-- expense_controller.dart

|

|-- expense_history_controller.dart

|

|-- network_controller.dart

|

|-- otp_verify_controller.dart

|

|-- user_repository_controller.dart

|

|-- pick_image_controller.dart

|

|-- notification_history_controller.dart

❖ **Model :**

That contain all the object of Data.

Model
|
|-- user_model.dart
|
|-- intro_model.dart
|
|-- user_contact_model.dart
|
|-- notification_data.dart
|
|-- message_model.dart
|
|-- expense_data.dart
|
|-- chat_room.dart
|
|-- group_members.dart
|
|-- group_data.dart

❖ **Screen :**

That contain all the view of all the screen.

Screen
|
|-- add_contact_screen.dart
|
|-- add_expense_form.dart
|
|-- add_expense_screen.dart
|
|-- add_member_screen.dart
|
|-- contact_us.dart
|
|-- country_pick_screen.dart
|
|-- create_new_group.dart
|
|-- dashboard.dart
|
|-- expense_details.dart
|
|-- group_details.dart
|
|-- group_expense_history.dart
|
|-- group_screen.dart
|
|-- group_setting.dart
|

|-- history_screen.dart
|
|-- home_screen.dart
|
|-- image_preview_screen.dart
|
|-- intro_screen.dart
|
|-- notification_screen.dart
|
|-- otp_verify.dart
|
|-- pdf_viewer_screen.dart
|
|-- phone_login_screen.dart
|
|-- privacy_policy.dart
|
|-- profile_screen.dart
|
|-- record_payment_screen.dart
|
|-- settle_up_screen.dart
|
|-- splash_screen.dart
|
|-- split_screen.dart
|
|-- total_screen.dart
|
|-- view_all_groups.dart
|
|-- view_split_screen.dart

❖ **Theme :**

Theme
|
|-- colors.dart
|
|-- colors_theme.dart

❖ **Utils :**

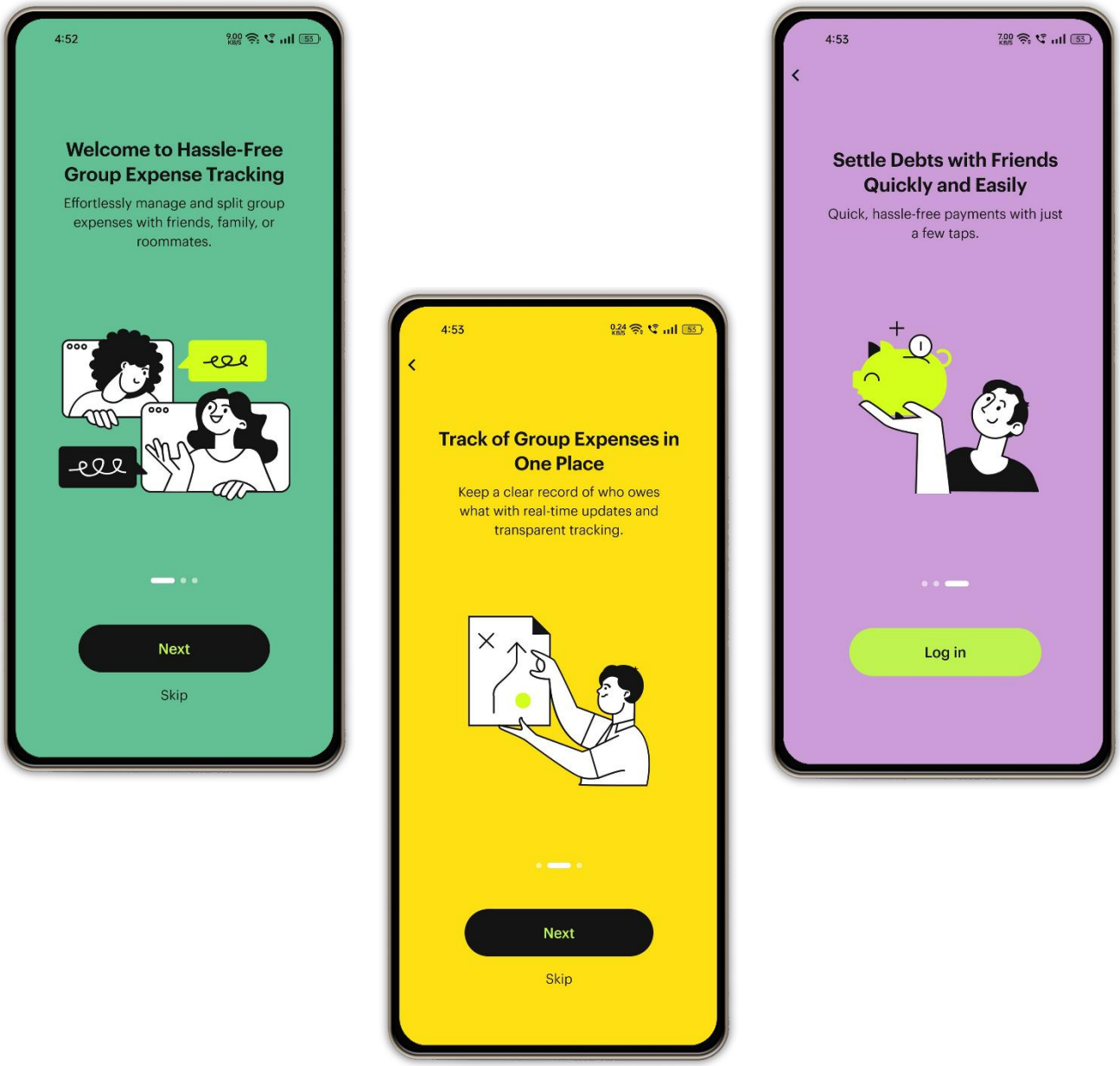
Utils
|
|-- app_font.dart
|
|-- app_storage.dart
|
|-- assets.dart
|
|-- controller_ids.dart
|
|-- extensions.dart
|
|-- firestore_timestamp_extenstions.dart
|
|-- network_dependency.dart
|
|-- strings.dart
|
|-- utils.dart

❖ **Utils :**

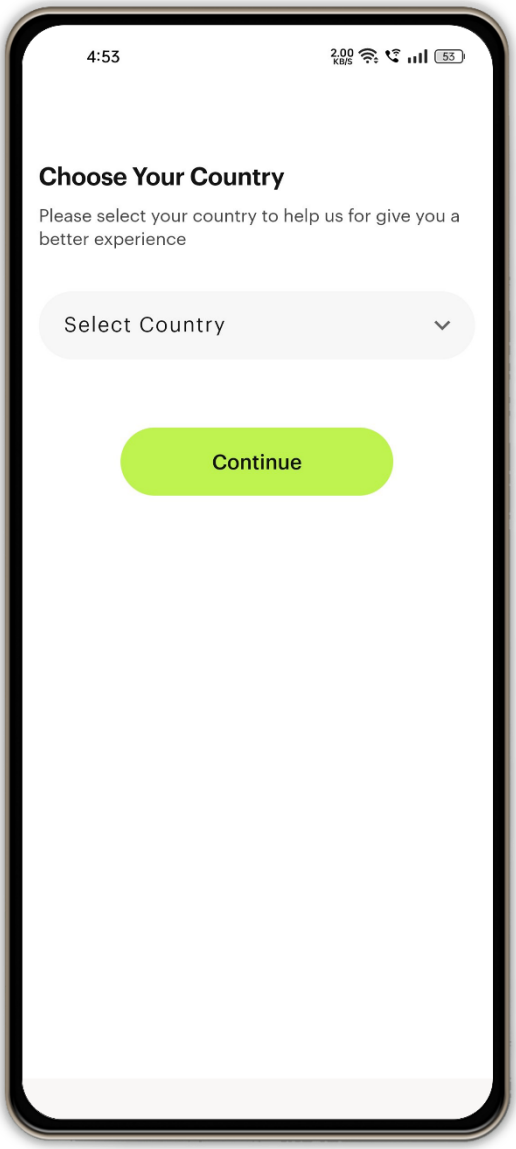
Utils
|
|-- app_dialogue.dart
|
|-- chat_message_content.dart
|
|-- connectivity_widget.dart
|
|-- custom_loading_widget.dart
|
|-- shimmer_widget.dart
|
|-- user_profile_widget.dart
|
|-- custom_group_avatar_widget.dart
|
|-- horizontal_avatar_widget.dart
|
|-- User
|
|-- my_name_text_widget.dart
|
|-- my_number_text_widget.dart
|
|-- my_profile_widget.dart
|
|-- other_profile_widget.dart

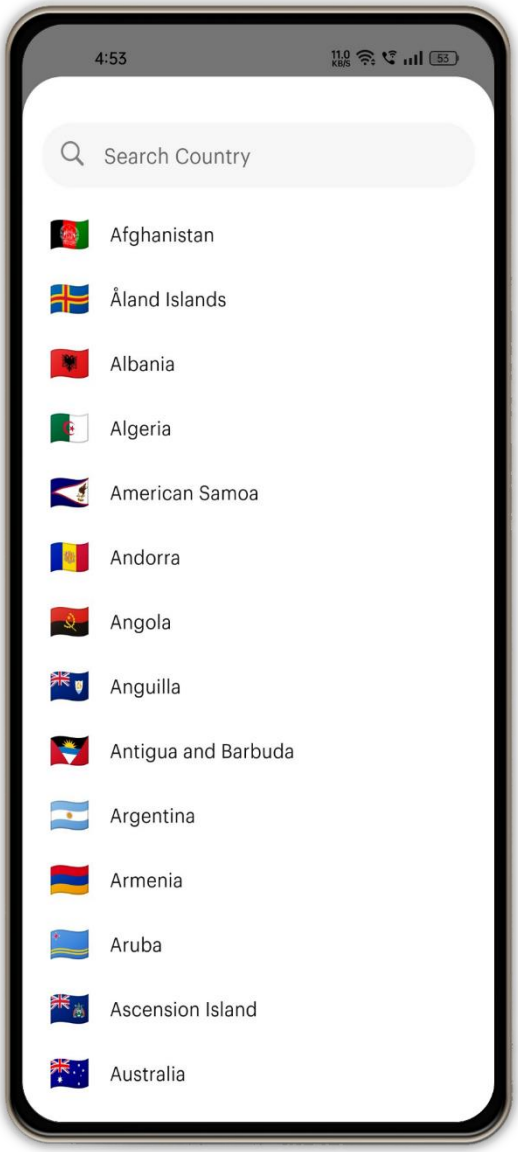
## 6.3 Input Layouts

- **Intro Screen :**

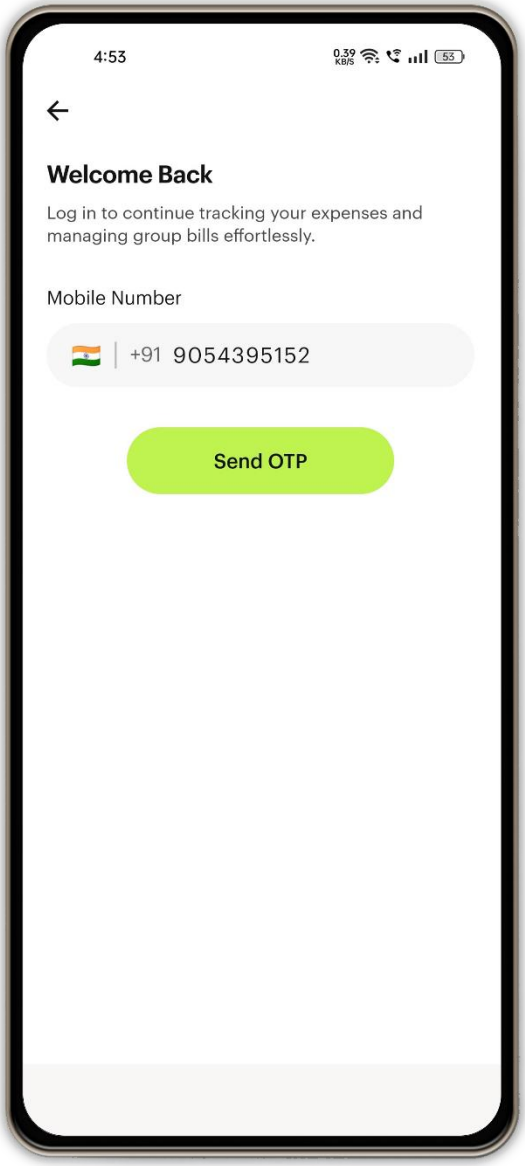


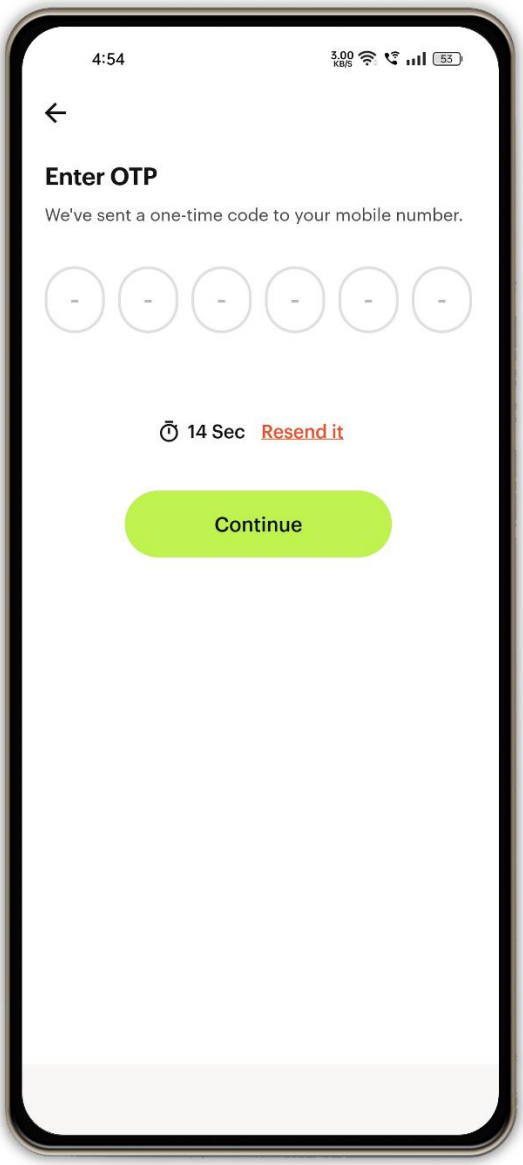
## ❖ Country Select Screen :



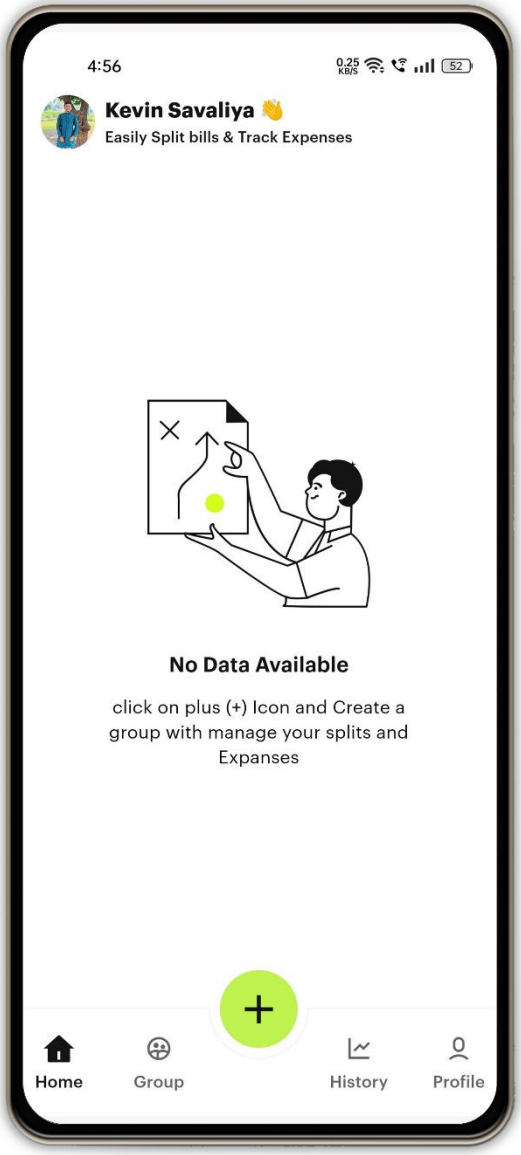



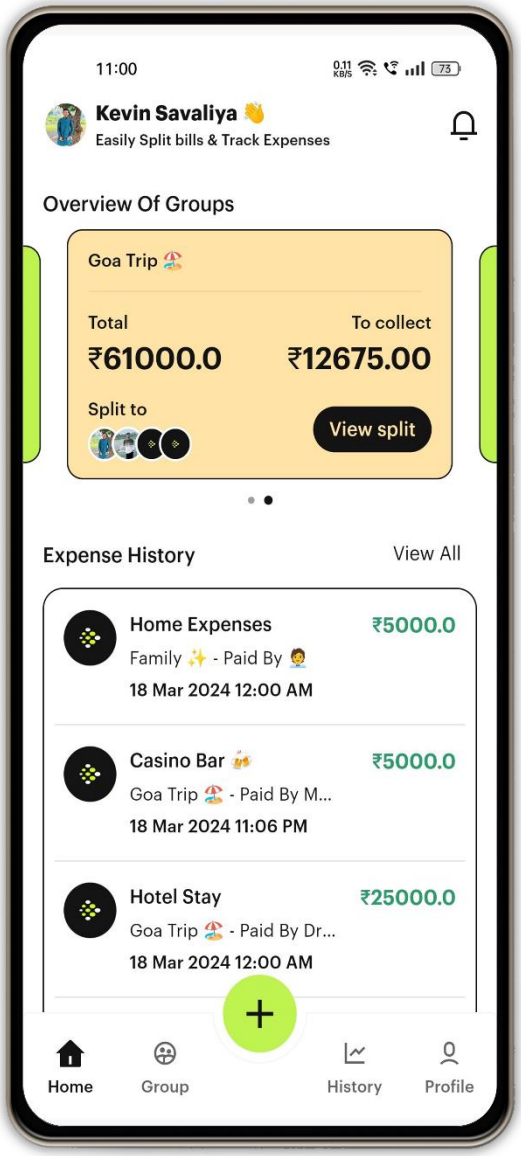
## ❖ Phone Login & OTP Screen :



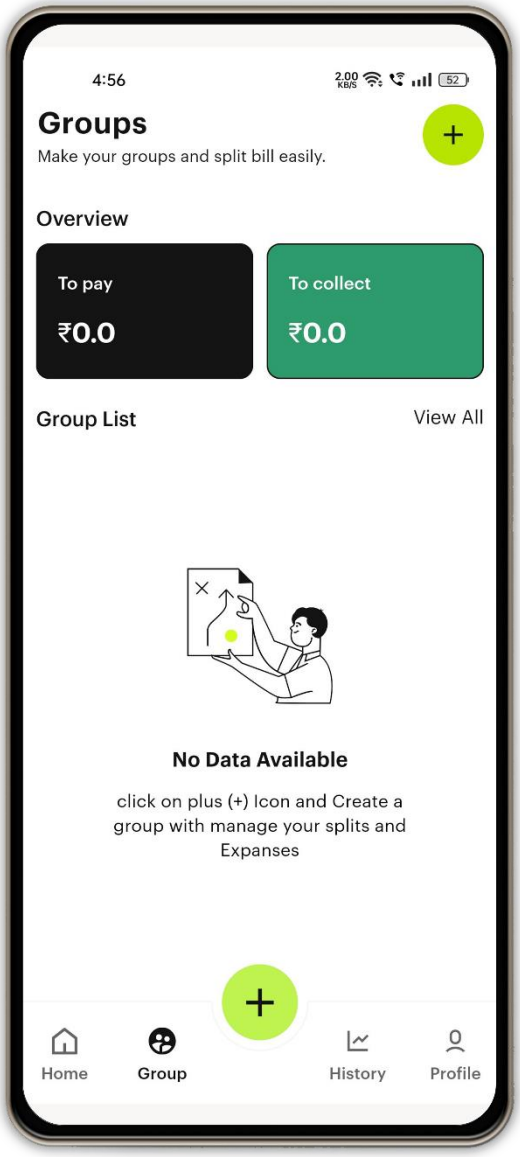



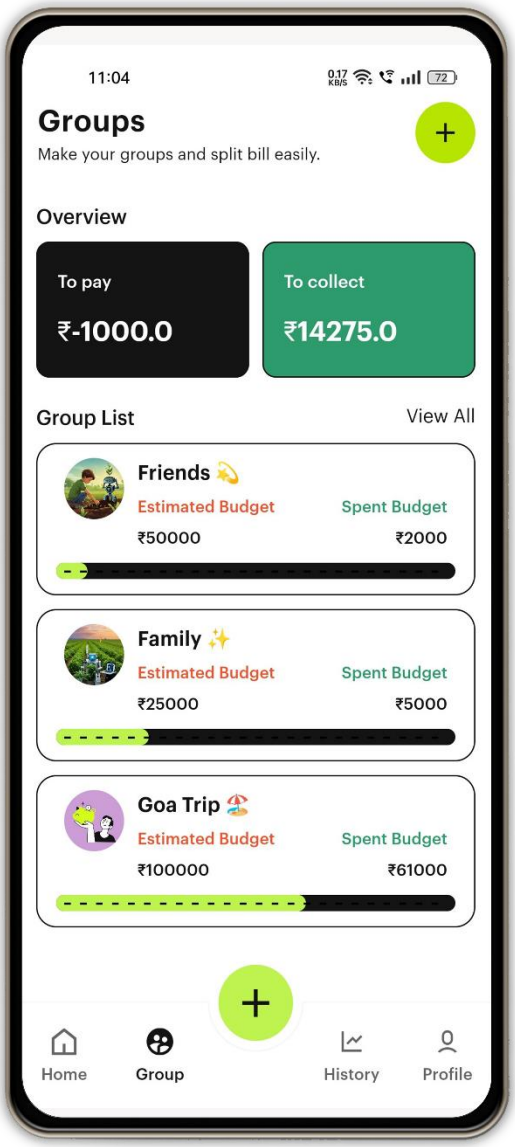
## ❖ Home Screen :


4:56
Kevin Savaliya 👋
Easily Split bills & Track Expenses
No Data Available
click on plus (+) Icon and Create a group with manage your splits and Expanses
Home
Group
History
Profile

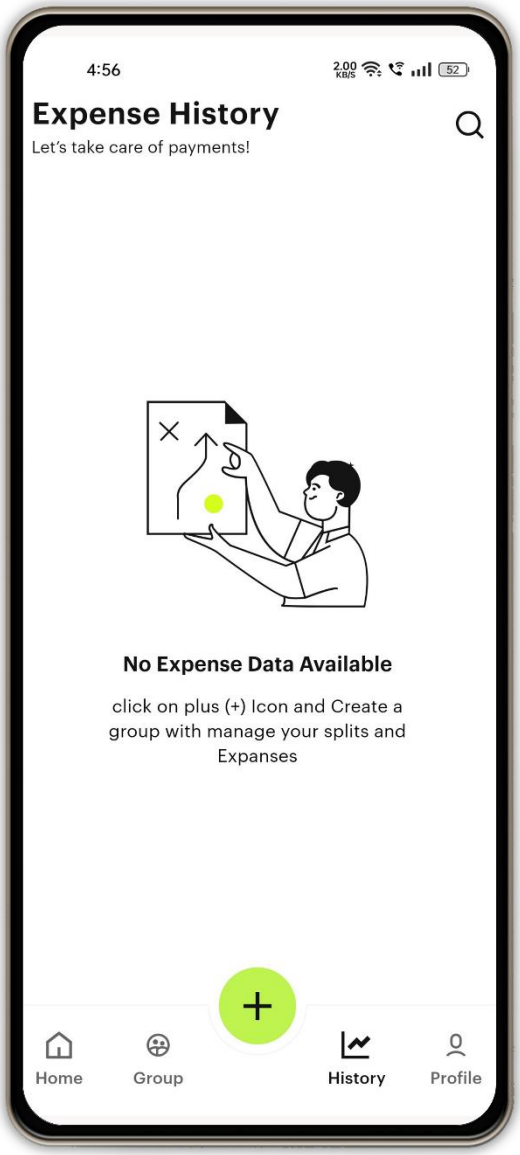

11:00
Kevin Savaliya 👋
Easily Split bills & Track Expenses
Overview Of Groups
Goa Trip 🏖️
Total
₹61000.0
To collect
₹12675.00
Split to
View split
Expense History
View All
Home Expenses
₹5000.0
Family ✨ - Paid By 🧑‍💼
18 Mar 2024 12:00 AM
Casino Bar 🍻
₹5000.0
Goa Trip 🏖️ - Paid By M...
18 Mar 2024 11:06 PM
Hotel Stay
₹25000.0
Goa Trip 🏖️ - Paid By Dr...
18 Mar 2024 12:00 AM
Home
Group
History
Profile

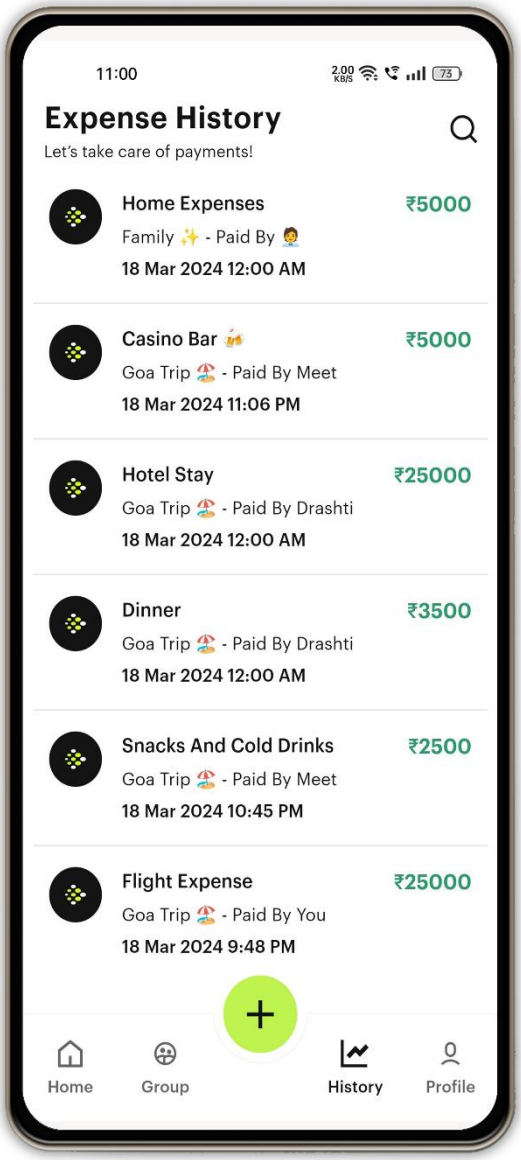
## ❖ Group Screen :



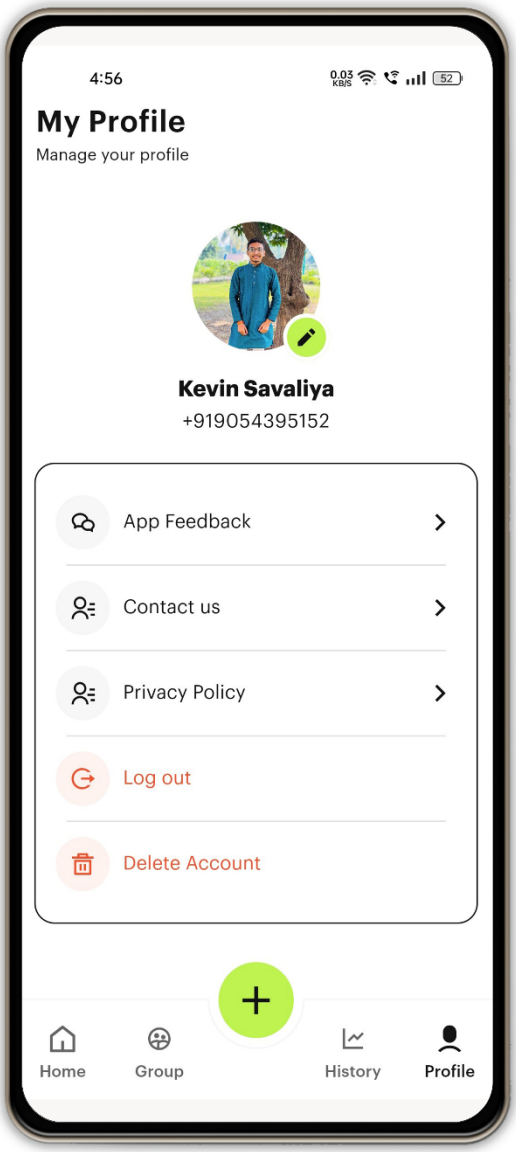



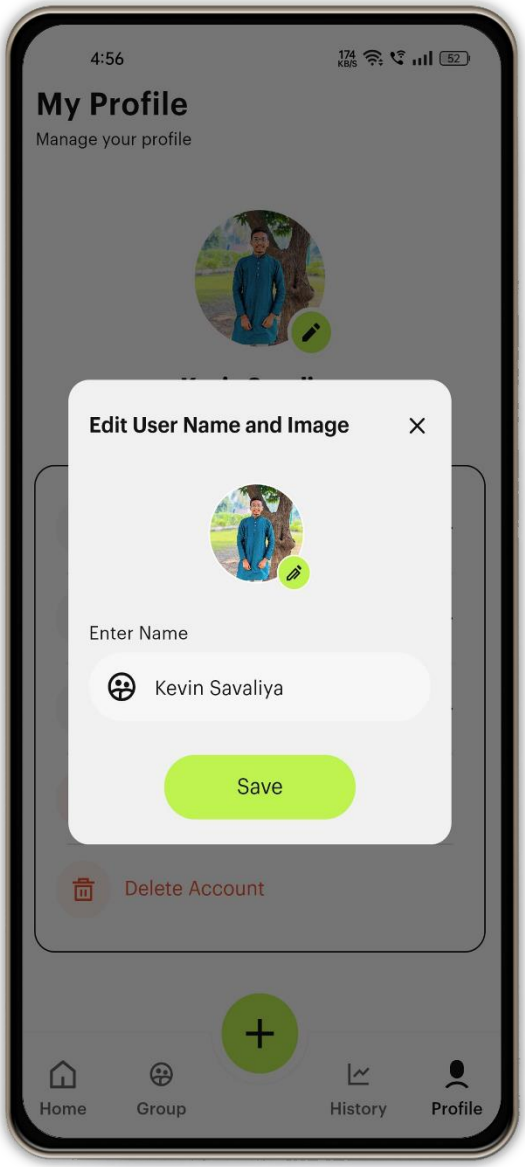
## ❖ History Screen :



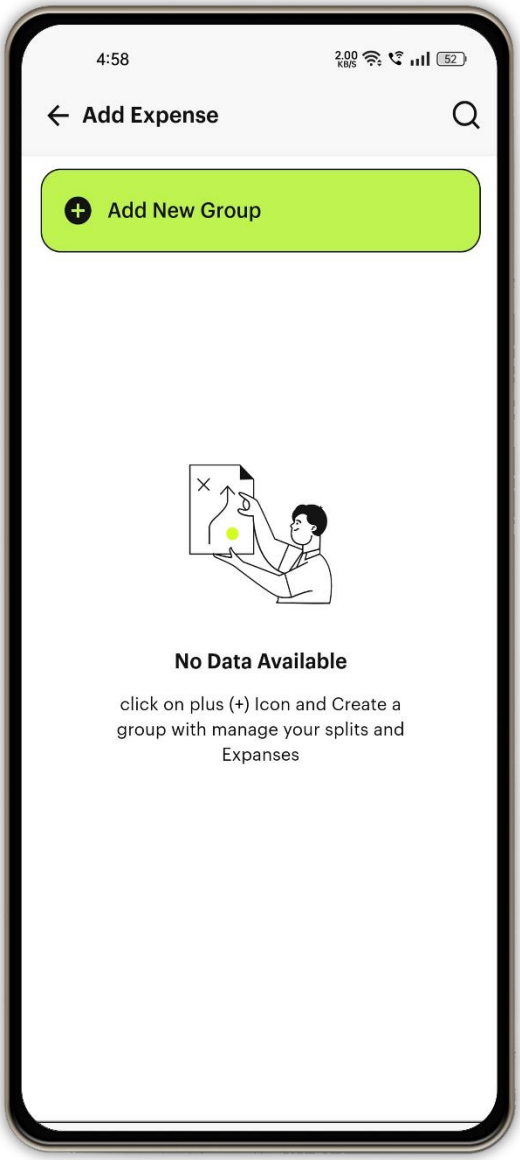



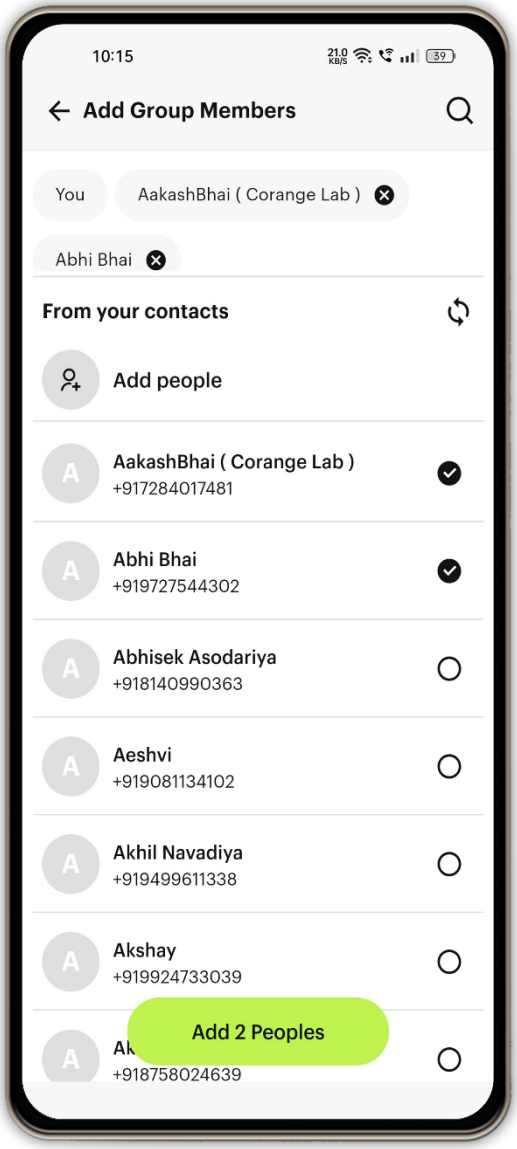
## ❖ Profile Screen :

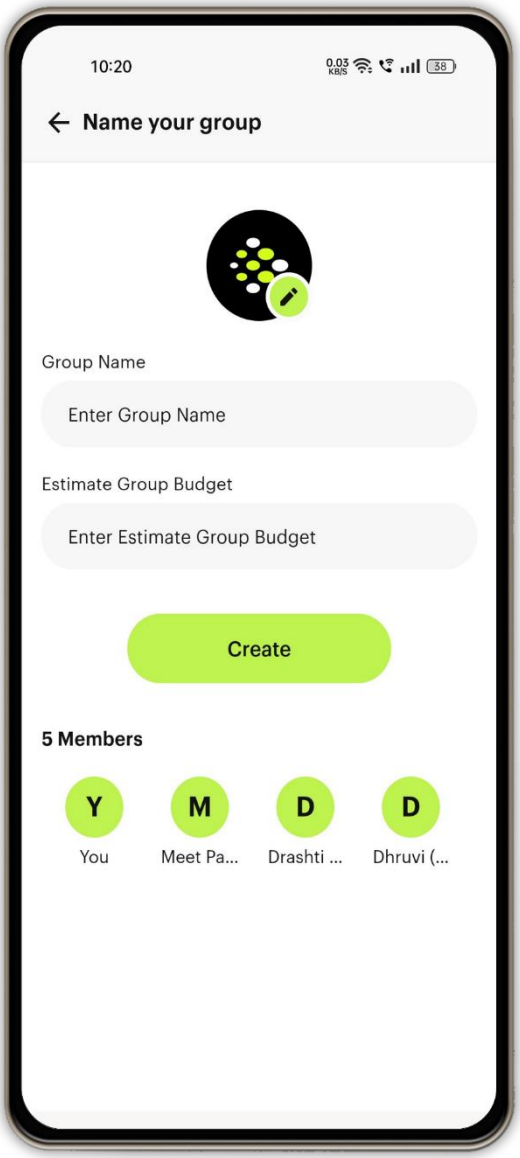


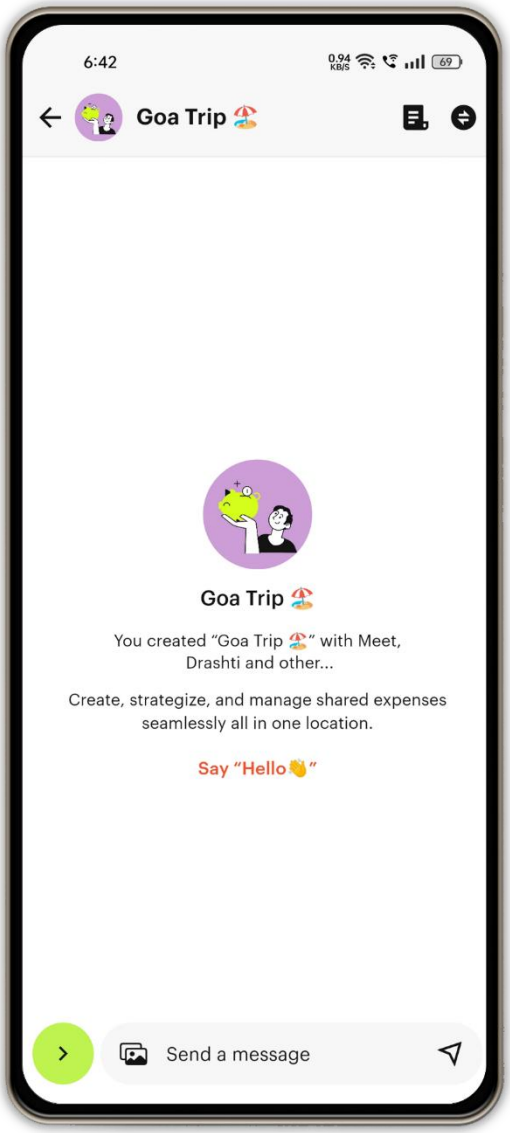



## ❖ Create Group :

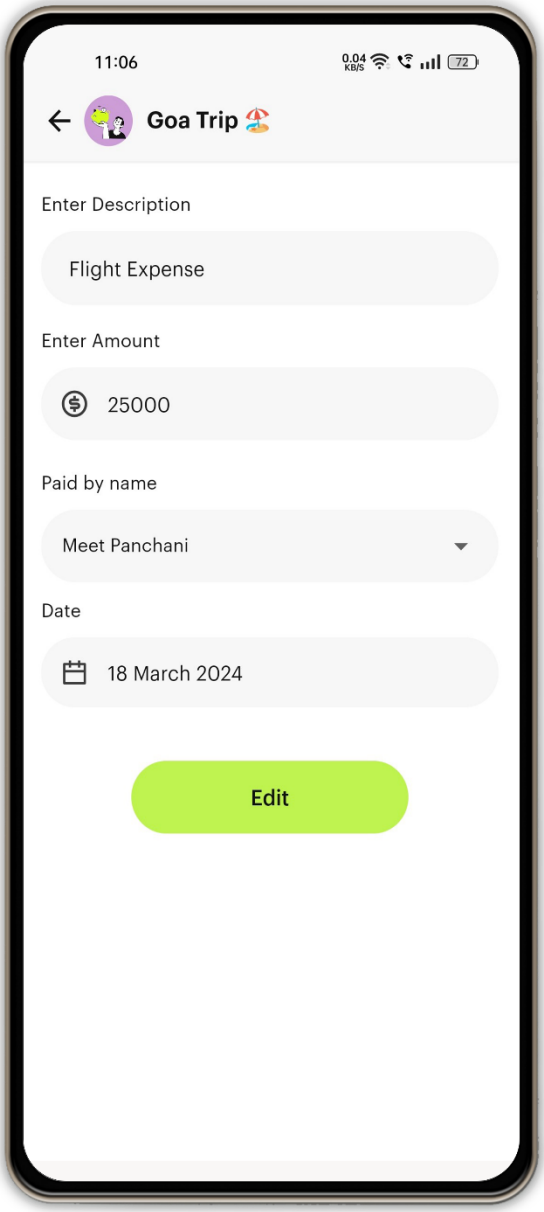


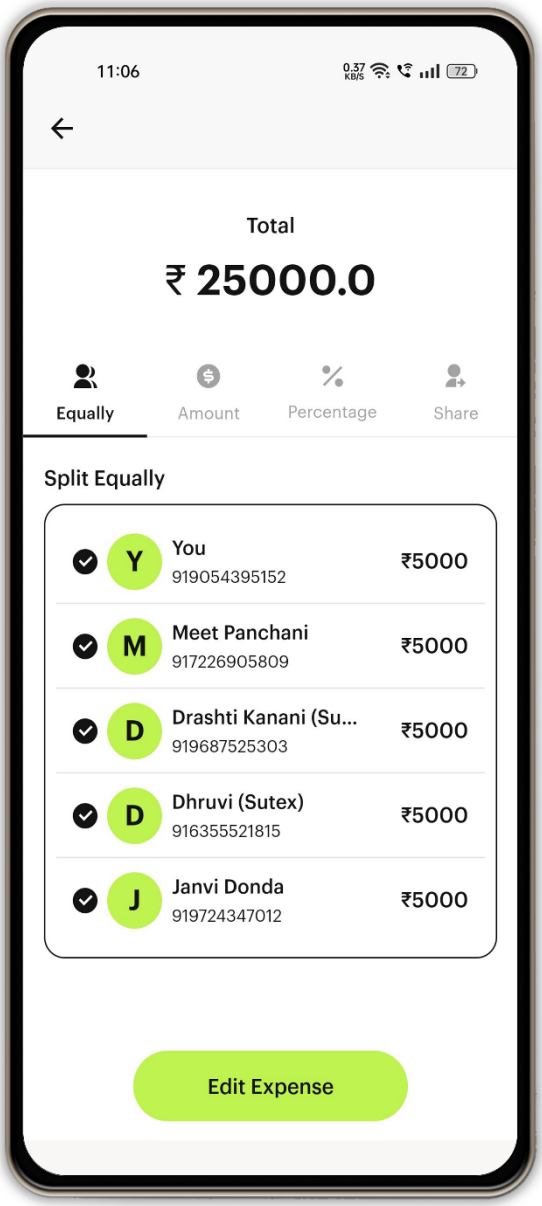

10:20
Name your group
Group Name
Enter Group Name
Estimate Group Budget
Enter Estimate Group Budget
Create
5 Members
Y
You
M
Meet Pa...
D
Drashti ...
D
Dhruvi (...

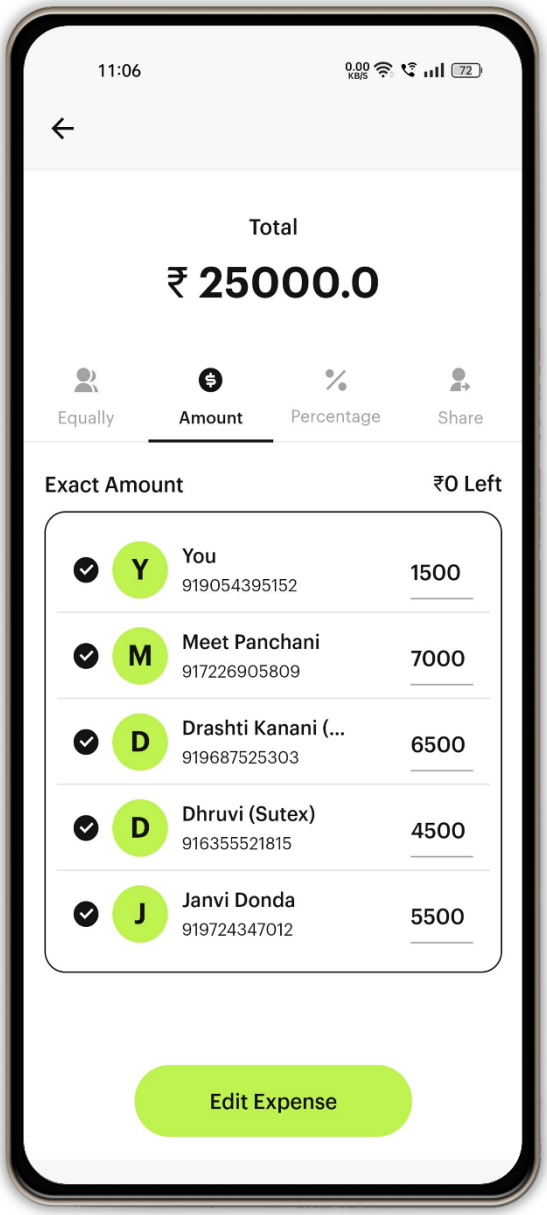

6:42
Goa Trip
Goa Trip
You created "Goa Trip" with Meet, Drashti and other...
Create, strategize, and manage shared expenses seamlessly all in one location.
Say "Hello"
Send a message

## ❖ Add Expense Screen :

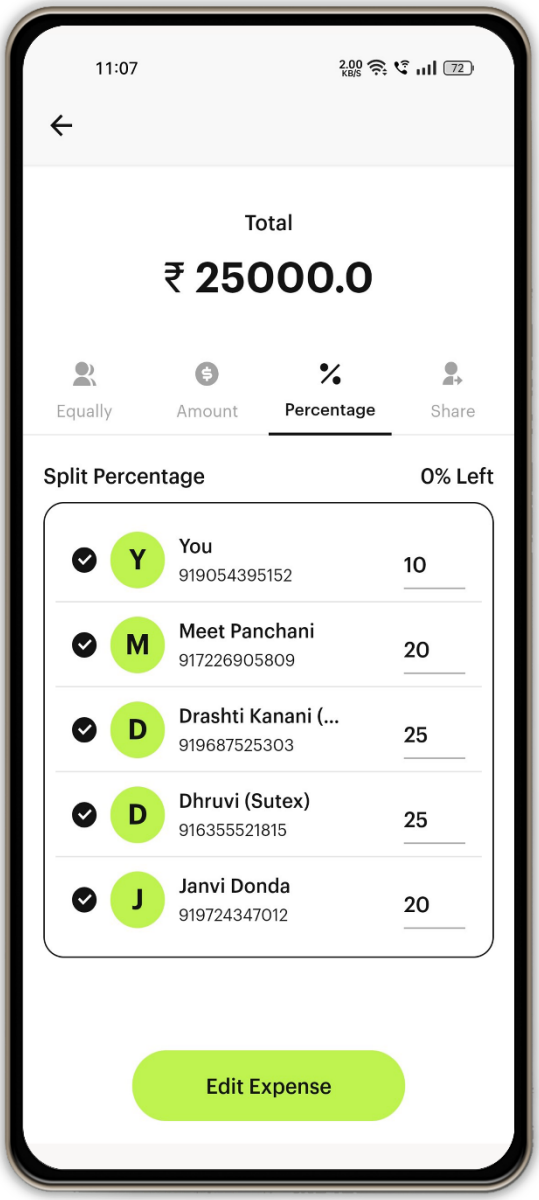


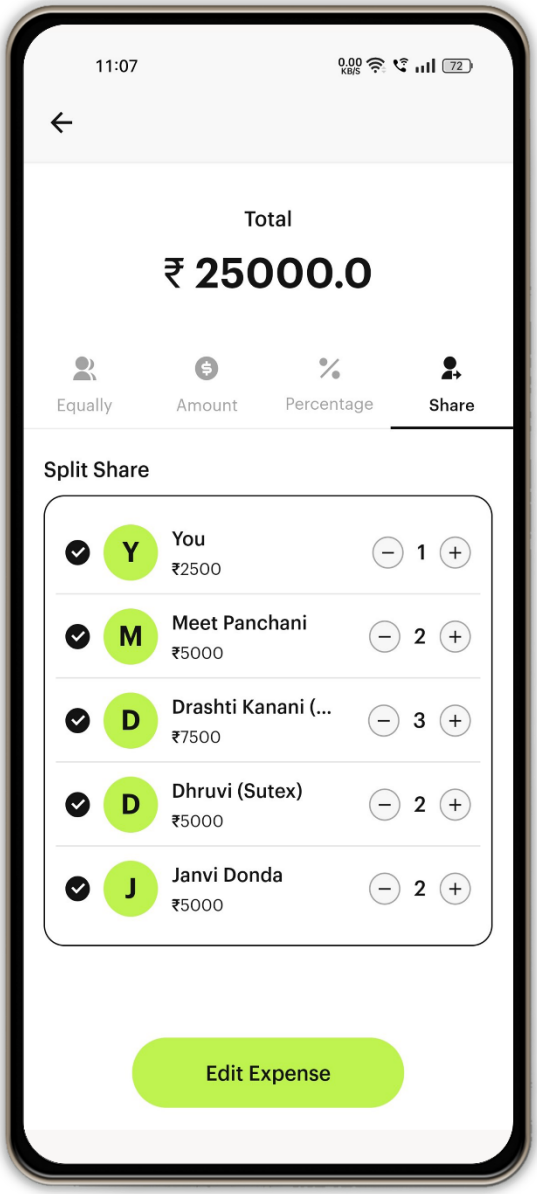
❖ **Split Expense Eqaully and Amount Wise :**



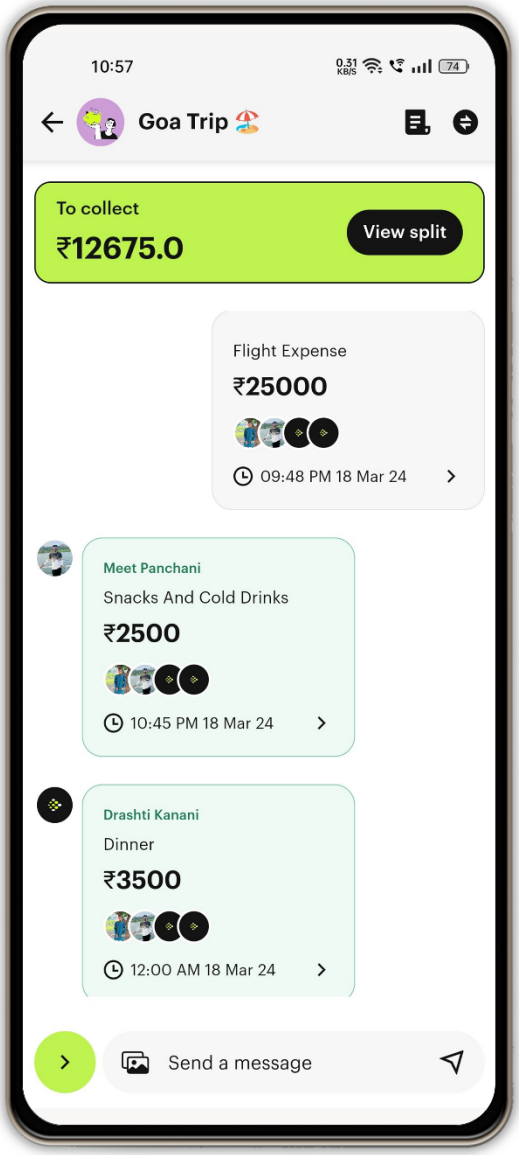



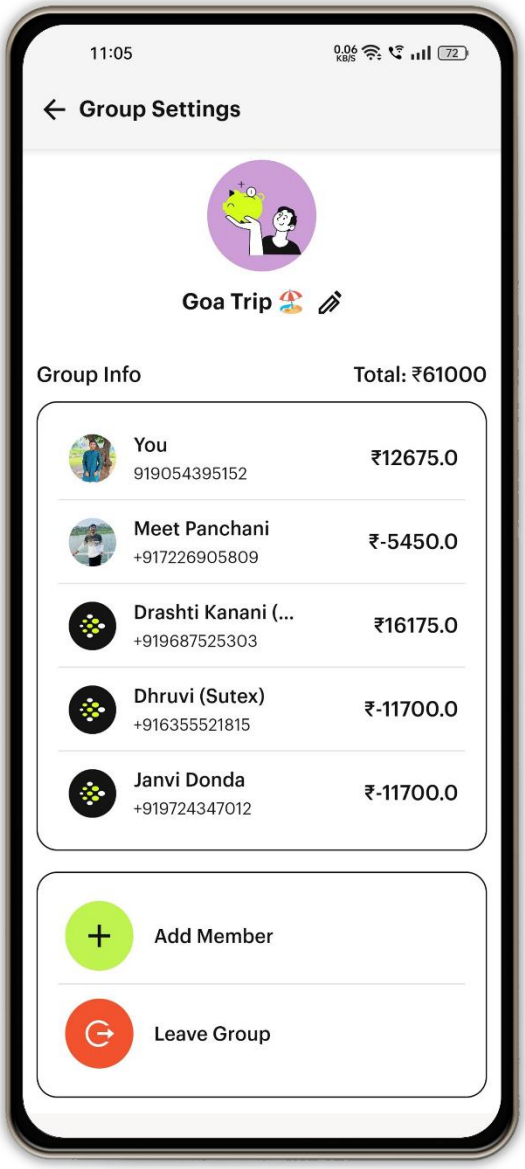
## ❖ Split Expense Percentage and Share Wise :



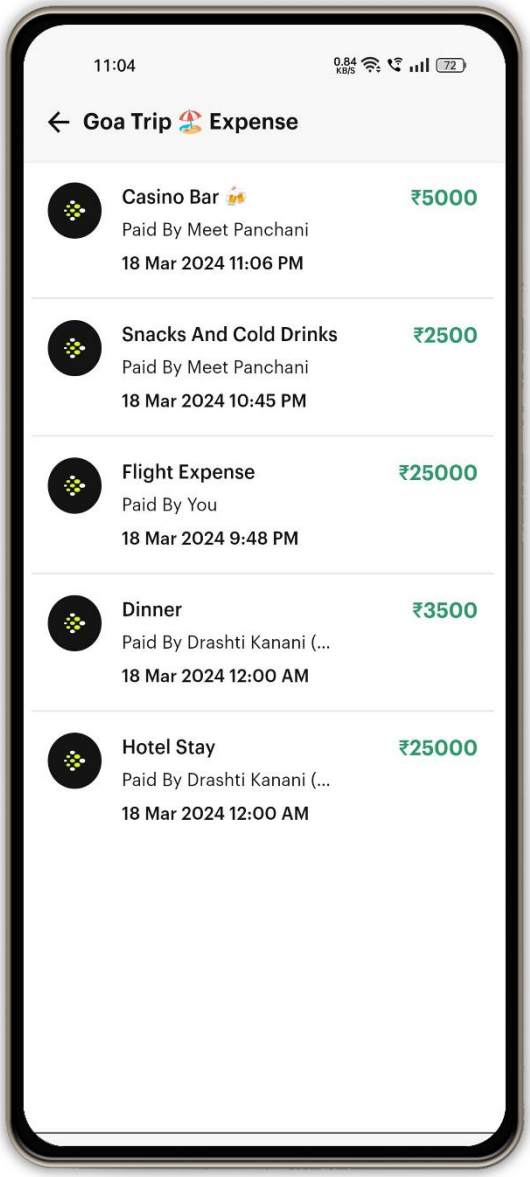



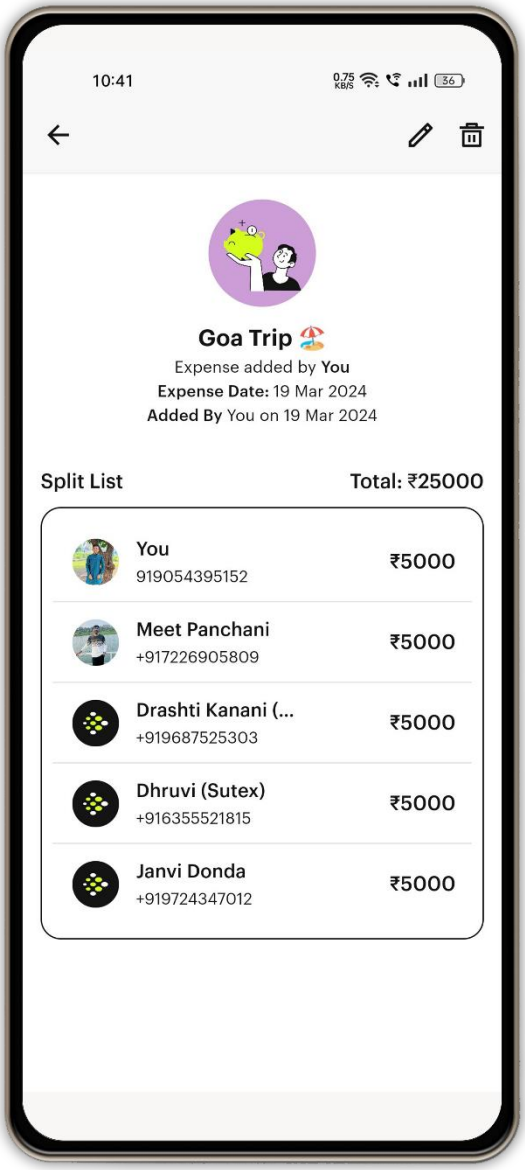
## ❖ Chat Screen :


10:57
Goa Trip
To collect
₹12675.0
View split
Flight Expense
₹25000
09:48 PM 18 Mar 24
Meet Panchani
Snacks And Cold Drinks
₹2500
10:45 PM 18 Mar 24
Drashti Kanani
Dinner
₹3500
12:00 AM 18 Mar 24
Send a message

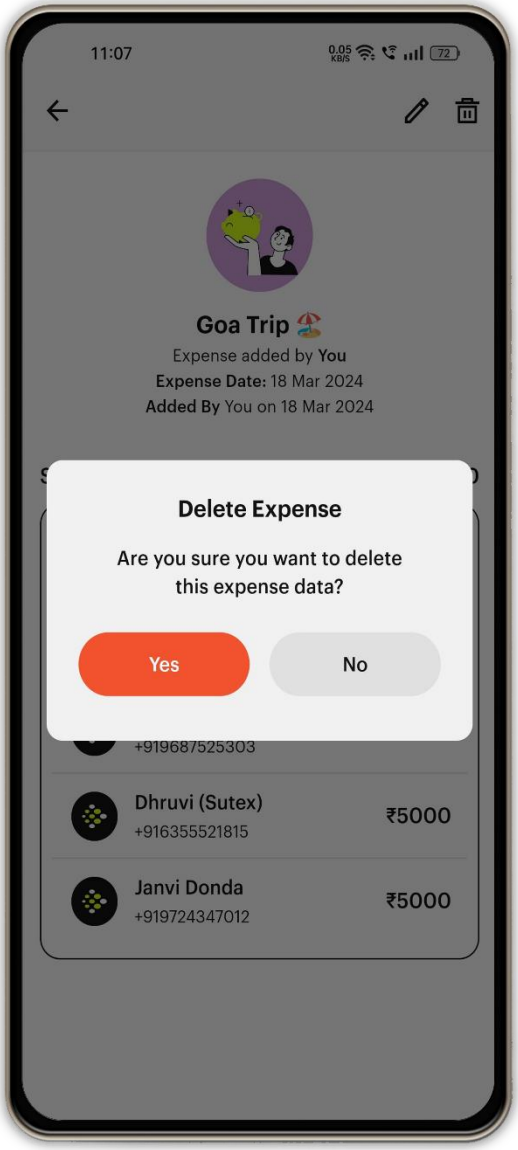

11:05
Group Settings
Goa Trip
Group Info
Total: ₹61000
You
919054395152
₹12675.0
Meet Panchani
+917226905809
₹-5450.0
Drashti Kanani (...
+919687525303
₹16175.0
Dhruvi (Sutex)
+916355521815
₹-11700.0
Janvi Donda
+919724347012
₹-11700.0
Add Member
Leave Group

## ❖ Group Expense :

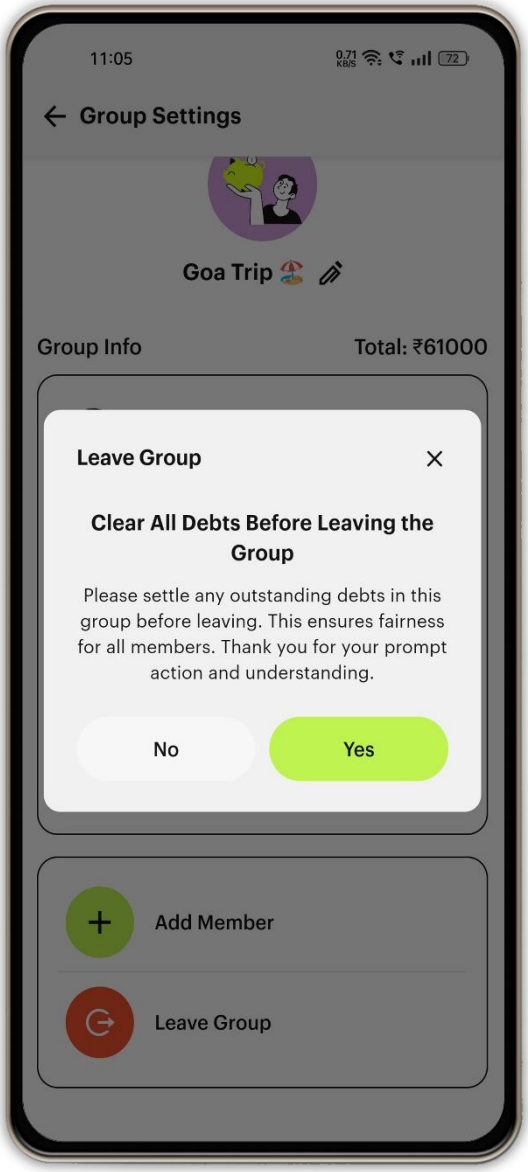


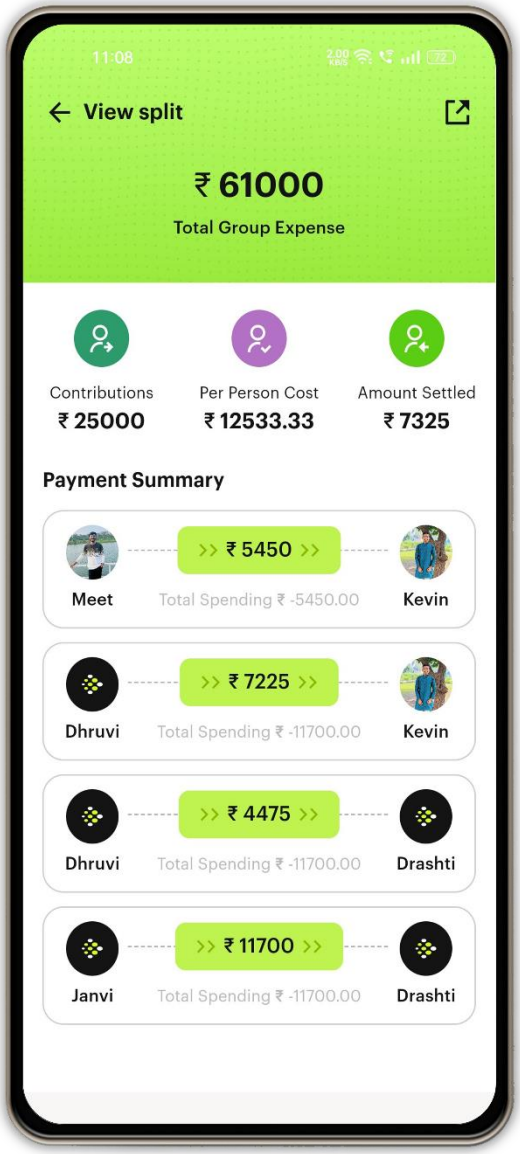
❖ **Delete Expense :**



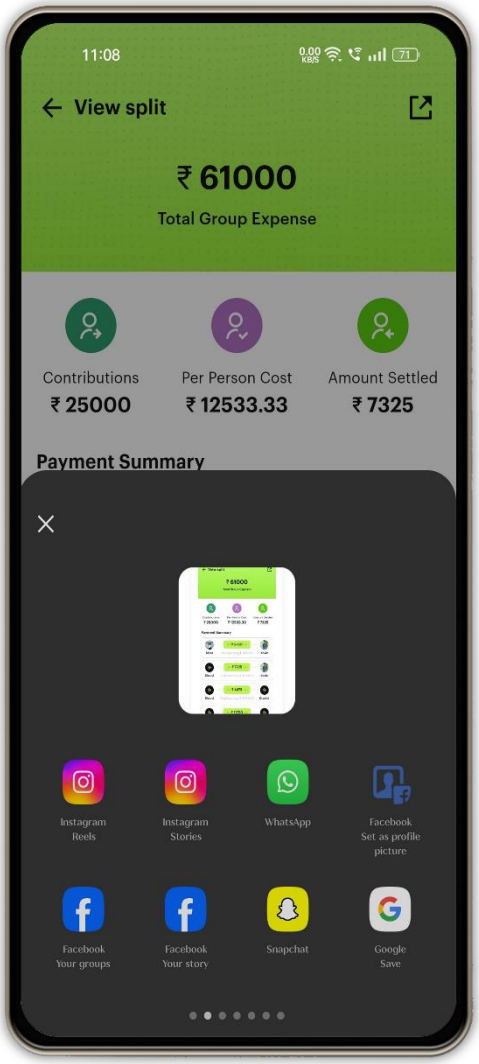

**Leave Group :**



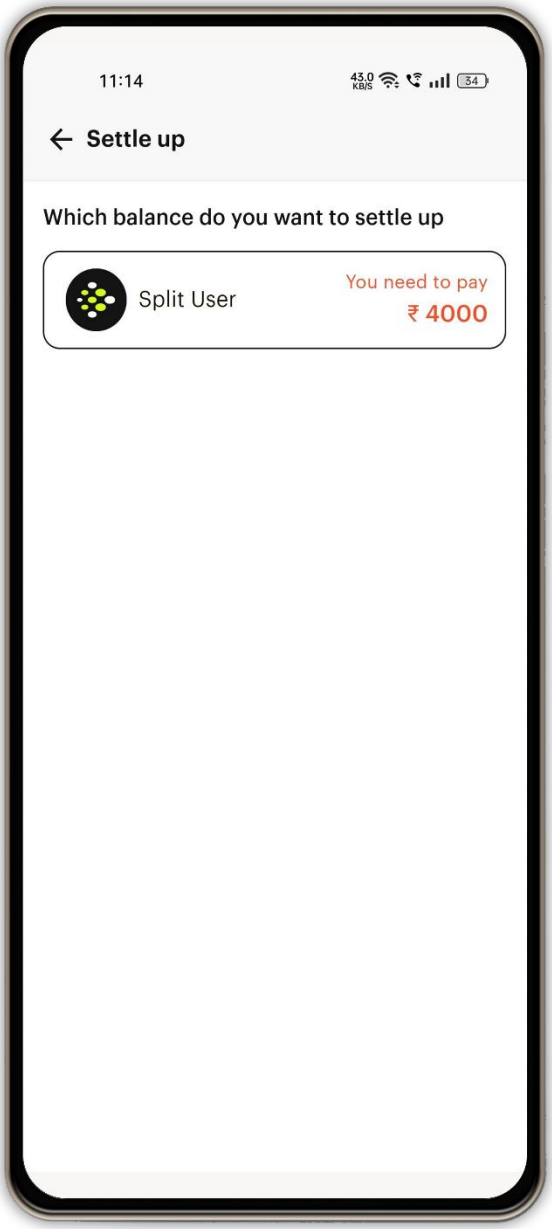
❖ **View Split Screen :**



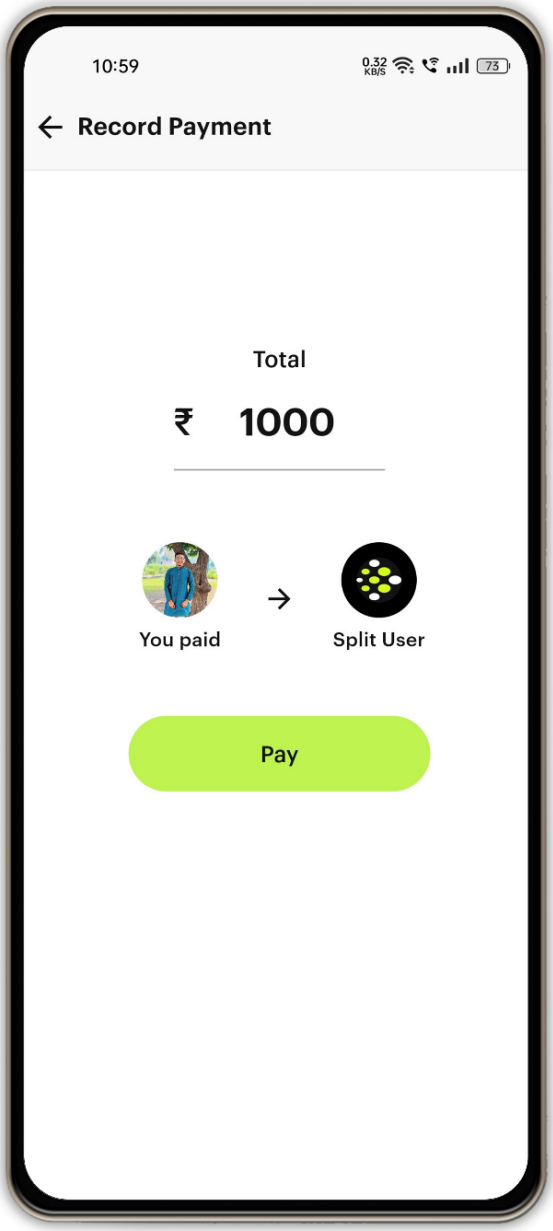

**Share Split Image :**



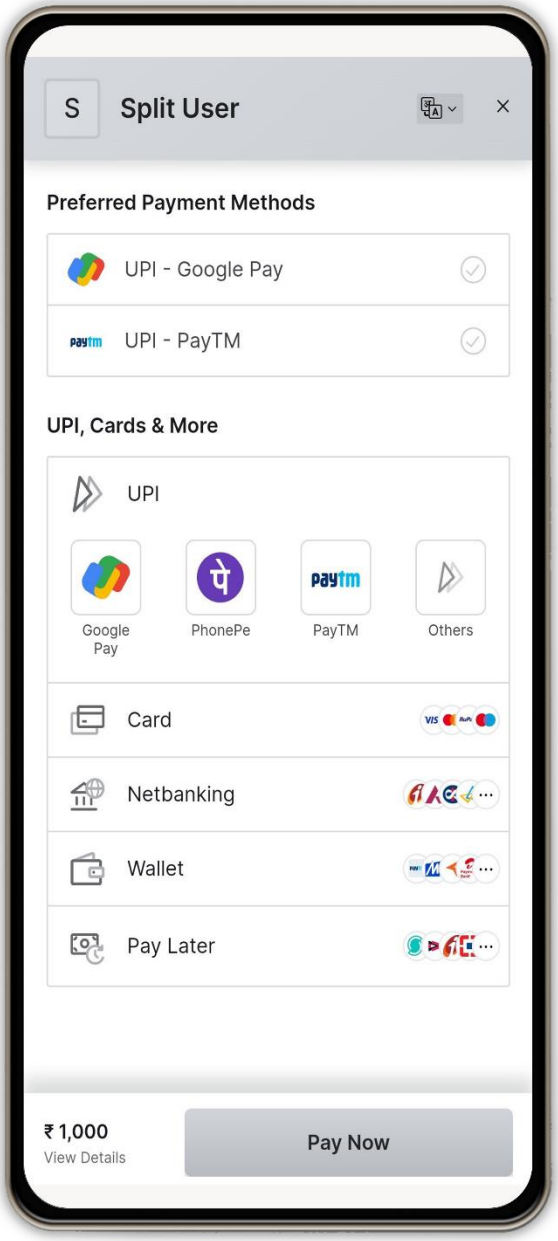
❖ **Settle Up Screen :**



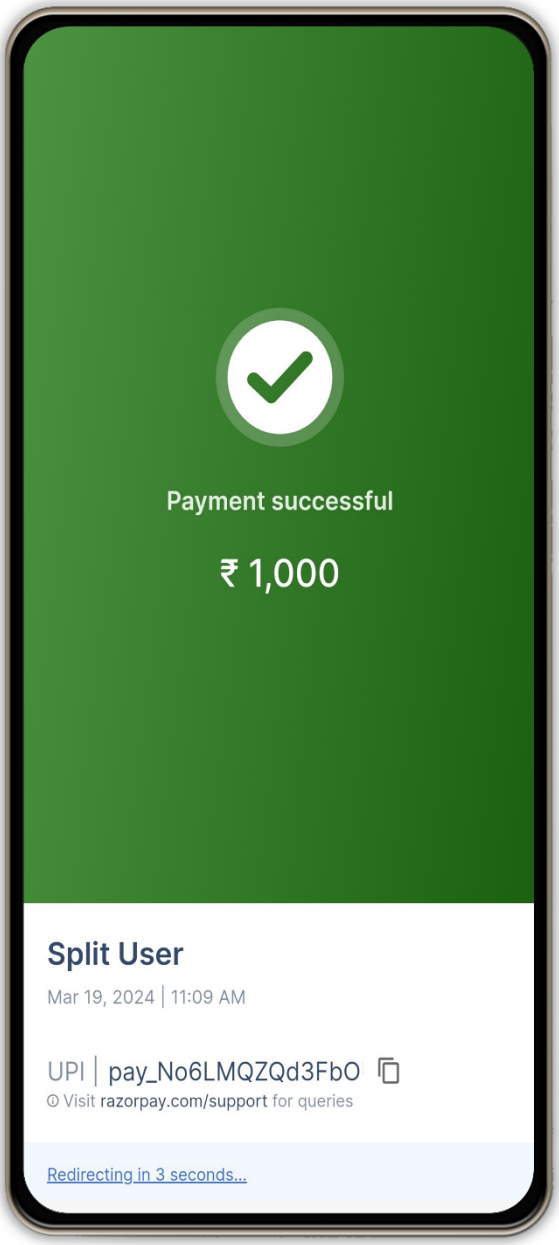

**Payment Screen :**



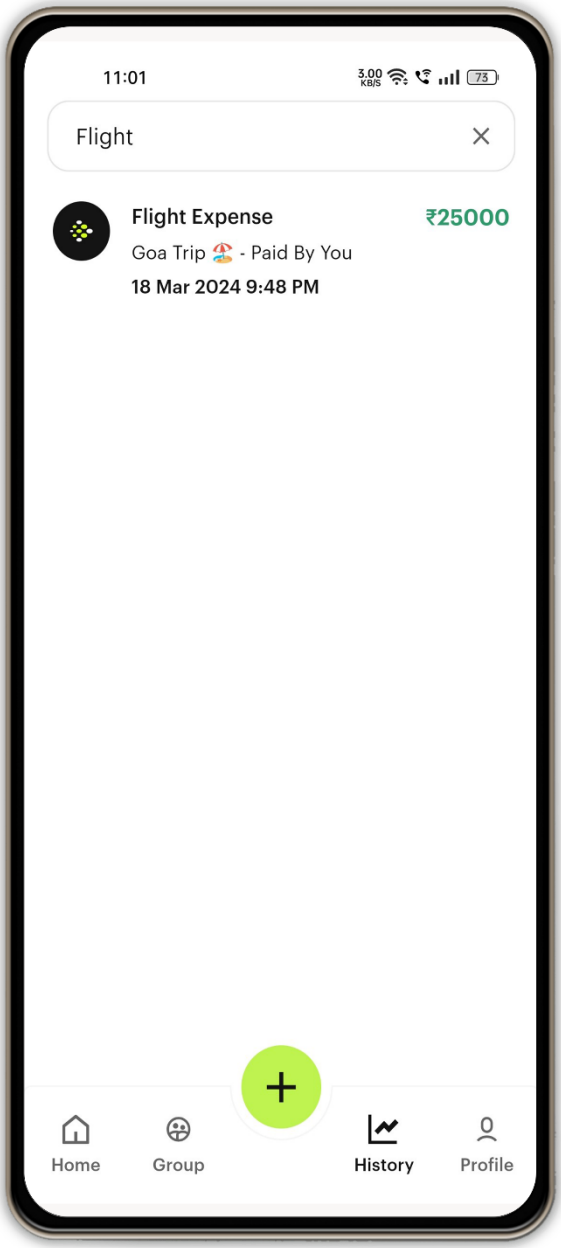
❖ **Payment By :**




**Payment Success :**

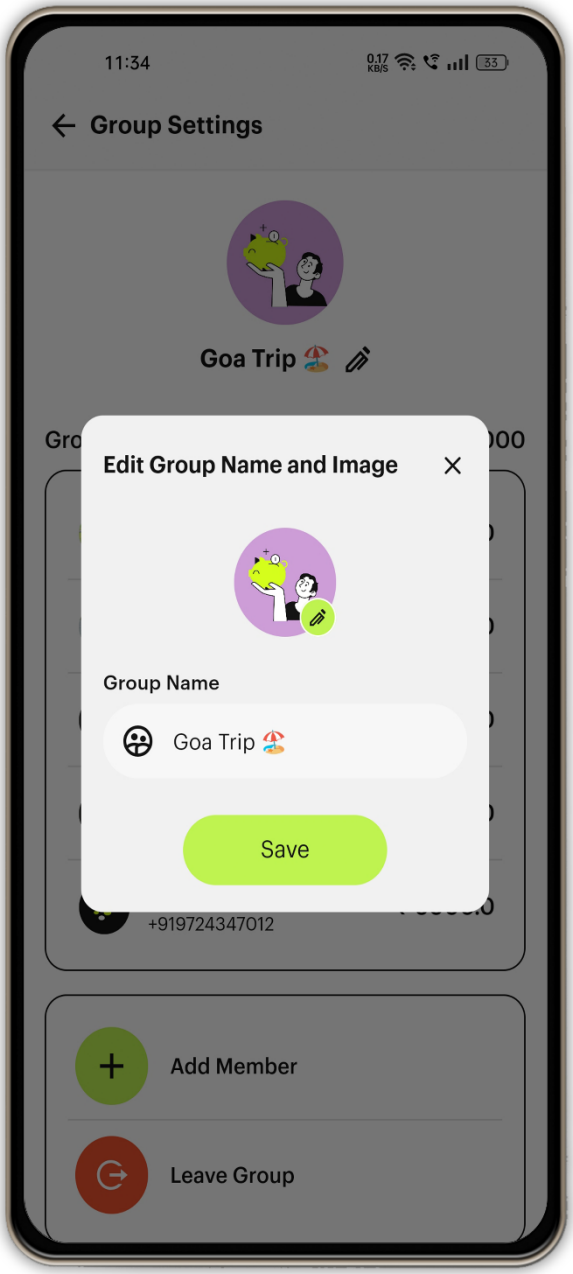


❖ **Search Expense :**

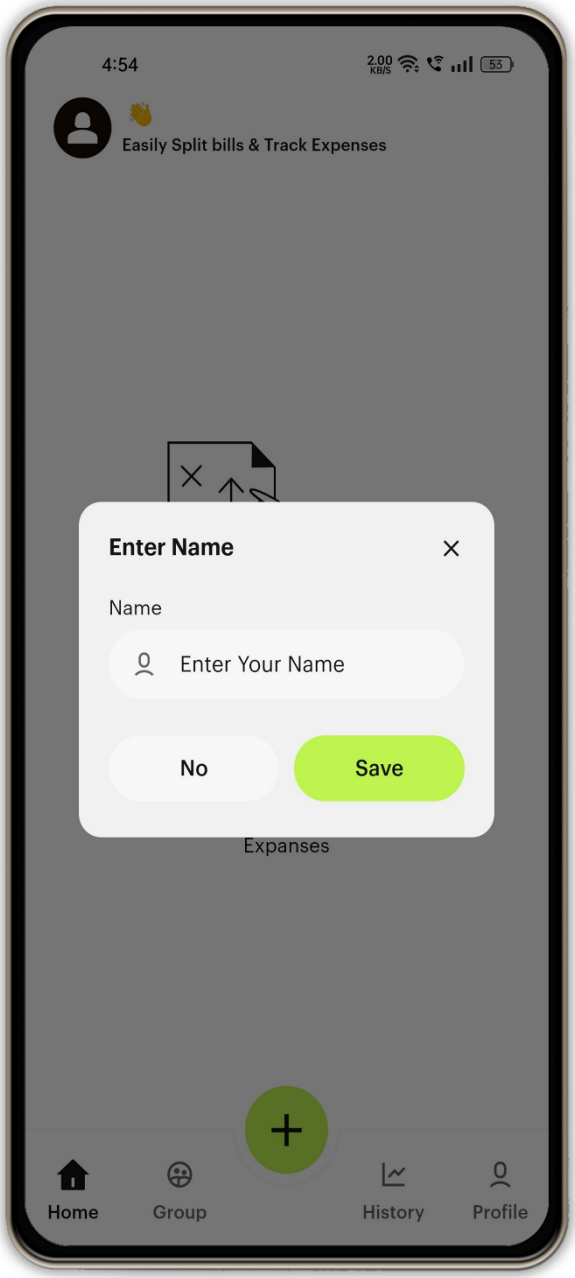



**Search Group :**

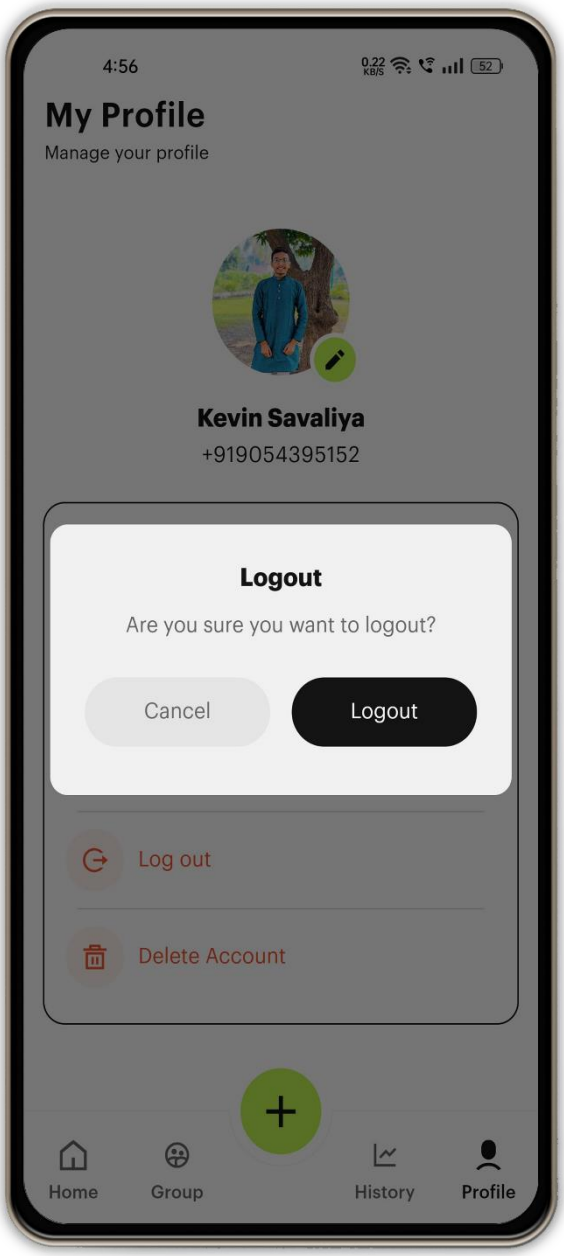
❖ **Edit Group :**



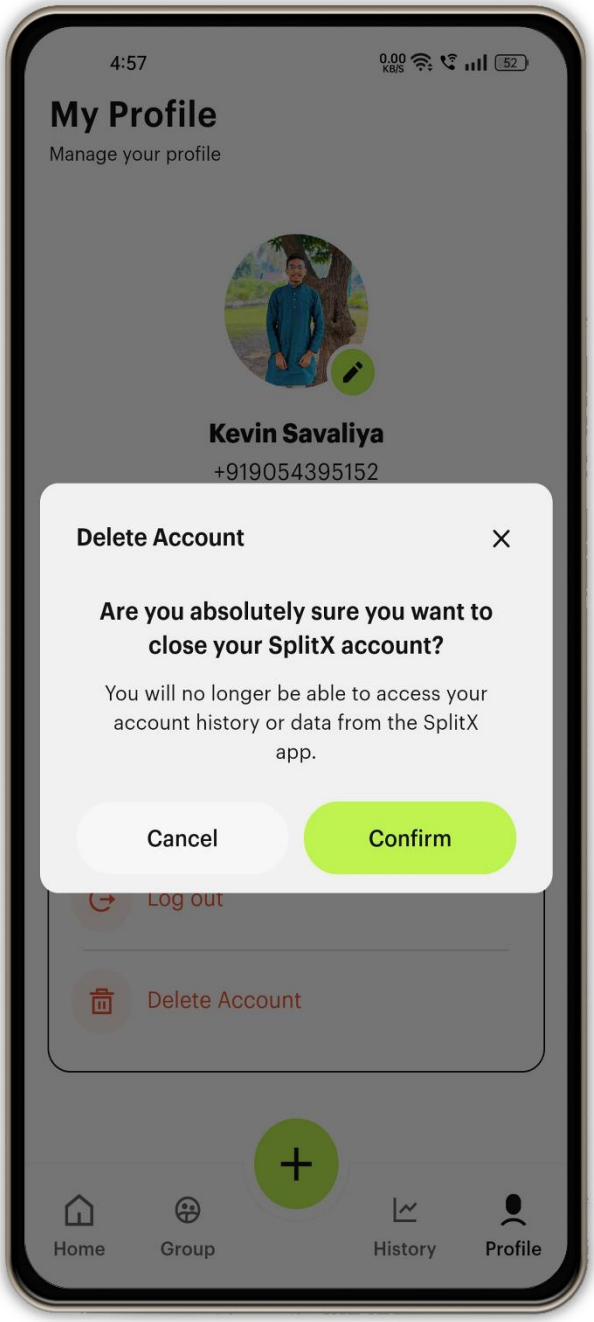

**Save Name Dialogue :**



❖ **Log Out Dialogue :**




**Delete Account Dialogue :**



# 7

# Software Testing

Testing is the process of detecting errors. Testing performs a very critical role for the quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

Software testing is a critical element of software quality assurances and represents the ultimate review of specification, design and coding. Testing is an exposure of a system to trial input to see whether it produces correct output. Testing cannot be determined whether software meets user's needs, only whether it appears to confirm to requirements. Testing can show that a system is free of errors, only that it contains error. Testing finds errors, it does not correct errors. Software success is a quality product, on time and within cost. Though testing can reveal critical (costly) mistakes. Testing should therefore,

- Validate Performance
- Detect Errors
- Identify Inconsistencies

## ❖ What is Testing ?

- Testing is the process of examining an application to ensure it fulfils the requirements for which it was designed and meets quality expectations.
- More importantly, testing ensure the application meet customer expectations.
- Testing accomplished a verity of things but most importantly it measures the quality of websites you are developing.
- This view presupposes there are defect in your website waiting to be discovered and this view is rarely disproved or even disputed.

## ❖ Testing Principles

- All the test should meet the customer requirements
- To make our software testing should be performed by a third party.
- Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
- All the test to be conducted should be planned before implementing it.
- It follows the Pareto rule(80/20 rule) which states that 80% of errors come from 20% of program components.
- Start testing with small parts and extend it to large parts.

## ❖ Strategy of Testing

- **Unit Testing :**

  It focuses on smallest unit of software design. In this we test an individual unit or group of inter related units. It is often done by programmer by using sample input and observing its corresponding outputs.

- **Integration Testing :**

  The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components are combined to produce output. Integration testing is of four types:

  (i) Top down
  (ii) Bottom up
  (iii) Sandwich
  (iv) Big-Bang.

- **Regression Testing :**

  Every time new module is added leads to changes in program. This type of testing makes sure that whole component works properly even after adding components to the complete program.

- **Alpha Testing :**

  This is a type of validation testing. It is a type of acceptance testing which is done before the product is released to customers. It is typically done by QA people.

- **Beta Testing :**

  The beta test is conducted at one or more customer sites by the enduser of the software. This version is released for the limited number of users for testing in real time environment.

- **System Testing :**

  In this software is tested such that it works fine for different operating system. It is covered under the black box testing technique. In this we just focus on required input and output without focusing on internal working. In this we have security testing, recovery testing, stress testing and performance testing.

- **Performance Testing :**

  It is designed to test the run-time performance of software within the context of an integrated system. It is used to test speed and effectiveness of program.

## ❖ Purpose of Testing

- Identify defects or bugs in the software.
- Ensure the software meets functional and non-functional requirements.
- Validate that the software performs as expected under various conditions.
- Enhance the quality, reliability, and usability of the software.
- Verify compliance with regulatory standards and industry best practices.

## ❖ Software Testing Process

- Test Planning:
  Define test objectives, scope, resources, and timelines. Develop test plans and test cases.

- Test Execution:
  Execute test cases, record results, and identify defects. Conduct exploratory testing to uncover additional issues.

- Defect Management:
  Document and prioritize defects. Assign defects for resolution, re-test fixes, and verify closure.

- Test Reporting:
  Generate test reports to communicate test results, defect metrics, and overall software quality to stakeholders.

- Test Closure:
  Evaluate test completion criteria, finalize test documentation, and archive test artifacts.

## ❖ Software Testing Technique

- Equivalence Partitioning:
  Divides input data into equivalent classes to reduce test cases.

- Boundary Value Analysis:
  Tests input values at the boundaries of equivalence partitions.

- Decision Table Testing:
  Maps combinations of inputs to corresponding actions or outcomes.

- State Transition Testing:
  Tests transitions between different states of a system.

- Pairwise Testing:
  Tests all possible combinations of input parameters with a reduced number of test cases.

## ❖ Software Testing Technique

- Test Management Tools:
  Manage test plans, test cases, and test execution.

- Automation Testing Tools:
  Execute automated test scripts and generate test reports.

- Load Testing Tools:
  Simulate user load and measure system performance under load conditions.

- Defect Tracking Tools:

Capture, track, and manage defects throughout the testing process.

## ❖ Best Practices in Software Testing

- Early Testing:
  Begin testing activities as early as possible in the SDLC to detect defects sooner.

- Test Automation:
  Automate repetitive and time-consuming test cases to improve efficiency and reliability.

- Continuous Integration:
  Integrate testing into the development process to identify defects early and ensure code quality.

- Traceability:
  Establish traceability between requirements, test cases, and defects to ensure comprehensive test coverage.

- Test Environment Management:
  Maintain stable and consistent test environments to minimize environmental issues during testing.

# 8

# Limitations and Future Scope of Enhancements References

## ❖ Limitation :

### ➢ Dependency on User Input :

- Users may forget to enter certain expenses or provide inaccurate information, leading to discrepancies in expense calculations.
- Inaccurate expense data can result in incorrect splitting and may cause disputes among group members.

### ➢ Complex Expense Scenario :

- Handling complex scenarios, such as shared expenses involving multiple currencies, uneven contributions, or reimbursements, can be challenging.
- Current implementations may struggle to accurately split expenses in such scenarios, leading to manual intervention and potential errors.

### ➢ Limited Payment Integration :

- Integrating with a limited number of payment gateways may restrict users' ability to settle expenses conveniently.
- Users may face challenges in settling expenses if their preferred payment method is not supported by the application.

- **Security Concerns :**
    - Storing sensitive financial data within the application requires robust security measures to prevent unauthorized access and data breaches.
    - Inadequate security measures can compromise user privacy and expose them to risks such as identity theft or fraud.

- **Platform Dependency :**
    - Developing separate applications for different platforms (e.g., iOS, Android, web) may result in inconsistencies in user experience and functionality.
    - Users may experience differences in features, performance, or usability when accessing the application across different devices or platforms.

## ❖ Future Enhancement :

- **Enhanced Expense Splitting Algorithms :**
    - Develop advanced algorithms that can handle complex expense scenarios more effectively, such as machine learning algorithms that analyze spending patterns to suggest optimal expense distributions.

- **Integration with Financial Management Tools :**
    - Integrate the split expense application with popular financial management platforms such as Mint or Personal Capital to provide users with comprehensive financial insights and budgeting tools.

- **Expanded Payment Options :**
    - Partner with additional payment gateways to offer users a wider range of payment options, including digital wallets, peer-to-peer payment platforms, and international payment methods.

- **Enhanced User Experience :**

  - Conduct user research and usability testing to identify pain points and areas for improvement in the user interface and user experience. Implement user-centric design principles to enhance usability and accessibility.

- **Automated Expense Tracking :**

  - Develop features for automatic expense tracking, such as OCR-based receipt scanning or integration with banking APIs to import transaction data, reducing the manual effort required from users.

- **Real-time Expense Notifications :**

  - Implement push notifications or in-app alerts to notify users of new expenses, pending settlement requests, or significant changes in group balances, ensuring timely communication and action.

- **AI-driven Insights :**

  - Leverage AI and ML algorithms to analyze expense data and provide personalized insights, such as spending trends, budgeting recommendations, or predictive analytics to help users make informed financial decisions.

- **Multi-platform Support :**

  - Develop a responsive web application or progressive web app (PWA) that offers consistent functionality and user experience across different devices and platforms, eliminating the need for separate native applications.

❖ **References :**


- **Splitwise - Expense Splitter**
- **Settle Up**
- **Tricount**
- **SplitBill**
- **Expense Splitter**


---

# THANK YOU