

1. INTRODUCTION

1.1 Project Profile

Project Title	SplitMate - Expense Tracking App
Front-End	<ul style="list-style-type: none">• XML• Java (for Dynamically UI interaction)
Back-End	<ul style="list-style-type: none">• Java (for app logic)• Firebase (for user Authentication & Database Interaction)
Development Environment	Android Studio
Project Duration	2 months
Project Guide	Mr. Tapan Solanki

1.2 Overview of the Project

SplitMate is a finance management application designed to help users track both personal and group expenses effortlessly. The app's standout feature is its ability to manage and split group expenses, making it a must-have tool for anyone who regularly shares costs with others.

Key Features of SplitMate:

1. User Registration and Authentication:

- **User Account Creation:** Users can create accounts using their email or phone number, with secure password authentication.
- **Firebase Integration:** Secure authentication is handled via Firebase Authentication, ensuring user data protection.
- **Password Recovery:** Users can recover forgotten passwords through email.

2. Personal Expense Tracking:

- **Expense Management:** Users can add, update, and delete personal expenses with ease.
- **Categorization :** Expenses can be categorized, and users can assign payment methods such as cash, online, or other.
- **Real-Time Expense Calculation:** The app dynamically calculates the total amount of expenses as they are entered.

3. Group Expense Management:

- **Group Creation and Management:** Users can create and manage groups to track shared expenses, making it easier to split bills among friends or family
- **Bill Splitting:** SplitMate automatically splits expenses among group members, helping users keep track of who owes what.

- **Group Balance Tracking:** Users can view balances for each group to see how much they owe or are owed.

4. Secure Data Handling:

- **Firebase Realtime Database and Firestore:** All data is securely stored and managed using Firebase's robust database services.
- **Security Protocols:** Firebase security rules are implemented to ensure user privacy and data protection, providing a safe environment for managing financial information.

5. Platform Compatibility:

- **Android Support:** SplitMate is currently available exclusively on Android devices, offering a consistent and reliable user experience on this platform.

❖ Back-End

The back-end of **SplitMate** is built on Firebase, providing a robust and scalable foundation for managing user authentication, data storage, real-time updates, and security. This architecture allows for seamless and secure expense tracking, both personal and group-based.

1. Firebase Authentication:

- **Purpose:** Handles user authentication.
- **Summary:** Firebase Authentication is responsible for managing user sign-ups, logins, and authentication states. SplitMate supports authentication via email/password and phone number, ensuring that users have secure and flexible login options.

2. Firebase Cloud Firestore:

- **Purpose:** Real-time NoSQL database for storing expense data.
- **Summary:** Firestore is used to store and manage personal and group expense data in a flexible, scalable, and real-time manner. The data is organized into collections and documents, which is ideal for applications like SplitMate where real-time synchronization is essential for accurate expense tracking.

3. Firestore Security Rules:

- **Purpose:** Define access control rules to secure data.

- **Summary:** Firestore security rules ensure that only authenticated users can access and modify their own data. These rules are critical in protecting user privacy and securing financial information stored within the app.

4. Real-Time Updates:

- **Purpose:** Enable real-time updates for expense tracking.
- **Summary:** Firebase Firestore's real-time capabilities allow users to see updates to their expenses immediately, whether they are tracking personal spending or managing group expenses. This feature is crucial for a responsive and accurate expense management

Overall, the back-end architecture of SplitMate leverages Firebase's powerful tools to manage authentication, real-time data updates, and secure storage. This infrastructure allows developers to focus on delivering an intuitive user experience, while Firebase efficiently handles the complexities of the back-end operations.

❖ Page To application

- **Splash screen page** (for starting Splash with app logo)
- **Welcome page**(it will display only while user start app without login)
- **Login page**(it contains Login,Signup and login with phone number)
- **Home page** (display the personal expenses on homepage)
- **addExpense page**(for append your personal expenses)
- **groups Data page** (it contain all the group that user created)
- **Creategroup page** (it is for create new group)
- **Groupdetails page** (it will shows all expense for individual group details)
- **Addgroupexpense page** (for add expense to the perticuller group)
- **Update page** (it will shows dialogue for the update and delete the group and personal expenses)

2. PROPOSED SYSTEM

2.1 Aim and Objectives

1) Aim

The aim of the SplitMate project is to develop a mobile expense-tracking application that is convenient, reliable, and secure. The app should be user-friendly and specifically designed to manage personal and group expenses efficiently, ensuring that users can easily track and split costs with others.

a) Convenience

- The app should offer an intuitive interface, making it easy for users to track and manage their expenses, both personal and group-based.
- Users should be able to quickly add expenses, create groups, and track shared costs, regardless of their location or time zone.
- The app should be simple to install and use, ensuring a smooth user experience.

b) Reliability

- The app should be stable and dependable, with a focus on accurate data storage and retrieval.
- Users should be able to rely on the app for precise expense tracking without any data loss or performance issues.

c) Security

- The app must ensure that user data, including financial details, is securely stored and managed.
- Strong security protocols should be implemented to protect against unauthorized access, with all sensitive data safeguarded.

2) Objectives

The following objectives outline the key features and functionalities that SplitMate aims to achieve:

a) Personal and Group Expense Tracking

- Implement features that allow users to easily track personal expenses and manage shared expenses within groups.
- Provide clear insights into who owes what, making it simple to settle group expenses.

b) User-Friendly Interface

- Design an intuitive interface that allows users to effortlessly navigate through the app, add expenses, and manage groups.
- Ensure that users can find and use the app's features quickly and easily.

c) Group Expense Splitting

- Develop a robust system for splitting group expenses, allowing users to automatically calculate and track how much each person owes within a group.

d) Data Persistence

- Utilize Firebase Cloud Firestore to store and retrieve expense data, ensuring that all financial information is securely stored and accessible across multiple devices.

e) Authentication and Authorization

- Implement Firebase Authentication to manage user logins, ensuring that only authenticated users can access their data.
- Use Firebase security rules to protect user data and ensure that only authorized users can access or modify specific information.

f) Real-Time Updates

- Ensure that any changes made to expenses are instantly reflected across the app, providing users with up-to-date information on their financial status.

g) Scalability and Performance

- Design the backend architecture to be scalable, ensuring that the app can handle a growing number of users and expenses without compromising performance.

h) Deployment and Distribution

- Prepare the app for deployment on the Google Play Store, making it accessible to Android users.

i) Group Management

- Allow users to create, update, and delete groups, providing a flexible and user-friendly experience for managing shared expenses.

j) Mobile App Compatibility

- Ensure compatibility with a wide range of Android devices to reach a broad audience of users.

By achieving these objectives, the SplitMate project aims to deliver a comprehensive expense-tracking solution that is user-friendly, reliable, and secure. This project will also demonstrate proficiency in mobile development and cloud-based data management using Firebase, with a focus on creating a high-quality, user-centric application.

2.2 Hardware and Software Requirements

❖ Hardware Requirements:

- **Mobile Device:**
 - A mobile device running **Android 8.0 (Oreo)** or higher.
 - ARM or ARM64 processor.
 - At least **2GB of RAM**.
 - At least **100MB of free storage** for app installation and data.
- **Development Machine** (for developing and testing the app):
 - A computer with a 64-bit operating system (Windows, macOS, or Linux).
 - At least **4GB of RAM** (8GB recommended for smoother performance).
 - At least **40GB of free disk space**.
 - A fast processor (Intel i5 or equivalent recommended).
- **Testing Devices:**
 - Physical Android devices running different Android versions, or emulators with similar configurations.

❖ Software Requirements:

- **Android Development:**
 - **Android Studio** (latest version recommended).

- **Java Development Kit (JDK) 8.**
- **Gradle** (automatically managed by Android Studio).
- **Firebase:**
 - **Firebase CLI** for managing Firebase services.
 - **Google Services JSON** file for configuring Firebase.
- **Testing Tools:**
 - **Android Emulator** for initial testing.
 - **Physical Android Device** for final testing (highly recommended).
- **Version Control:**
 - **Git** for version control.

1) Note:

- **Development Machine:** A machine with a fast processor and ample RAM will enhance development speed and efficiency.
- **Testing:** While an emulator is useful for initial testing and debugging, testing on a physical device is essential for accurate results.

❖ Tools and Technology Used

- **Java Programming Language:** The primary language used for writing application logic.
- **Firebase:**
 - **Firebase Authentication:** For user authentication.
 - **Cloud Firestore (NoSQL Database):** For storing and syncing data.
 - **Firebase Storage:** For storing user files such as images.
 - **Firebase Analytics and Performance Monitoring:** To monitor app performance and user behavior.
 - **Firebase Crashlytics:** For real-time crash reporting.
- **Android Studio:** The official IDE for Android development, providing the tools needed to build apps.
- **Build Features:**
 - **View Binding:** Simplifies code interaction with UI elements.

- **Libraries and Dependencies:**
 - **AndroidX Libraries:** For backward compatibility and modern Android components.
 - **Material Design Components:** For a consistent and modern UI.
 - **JUnit:** For unit testing.
 - **Espresso:** For UI testing.

2.3 Scope

The scope of the SplitMate project is to develop a comprehensive mobile application for tracking and managing both personal and group expenses. The following features and functionalities outline the scope of the project:

- **Personal and Group Expense Tracking:** Users should be able to track their personal expenses as well as create and manage group expenses, ensuring easy splitting and settlement of shared costs.
- **Real-Time Expense Updates:** Changes to expenses should be instantly reflected within the app, ensuring that all users have access to the most up-to-date information.
- **Group Expense Splitting:** The app should automatically calculate and display how much each person in a group owes, simplifying the process of managing shared expenses.
- **File and Receipt Sharing:** Users should be able to attach and share receipts, bills, and other relevant files related to their expenses, providing a clear record of transactions.
- **User Presence:** The app should display the presence status (online/offline) of users within a group, enhancing the social aspect of the application.
- **Data Persistence:** Utilize Firebase Cloud Firestore to store and retrieve all financial data, ensuring data integrity and availability across multiple devices.
- **User Authentication and Security:** Implement Firebase Authentication to manage user logins securely, and use Firebase security rules to ensure that only authorized users can access and manage data.
- **User-Friendly Interface:** The app should have an intuitive and easy-to-navigate interface, allowing users to manage their expenses with minimal

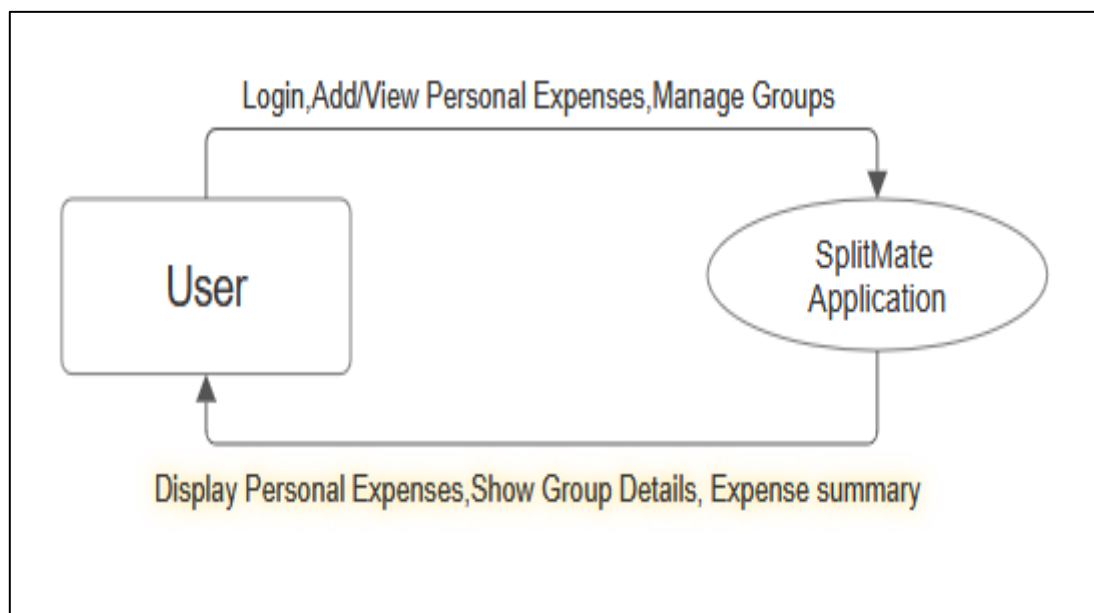
effort.

- **Mobile App Compatibility:** The app should be compatible with a wide range of Android devices, ensuring accessibility for a broad user base.
- **Cloud Deployment:** Deploy the app to the cloud to ensure scalability and reliability, allowing it to handle a growing number of users and transactions without compromising performance.

3. SYSTEM DESIGN

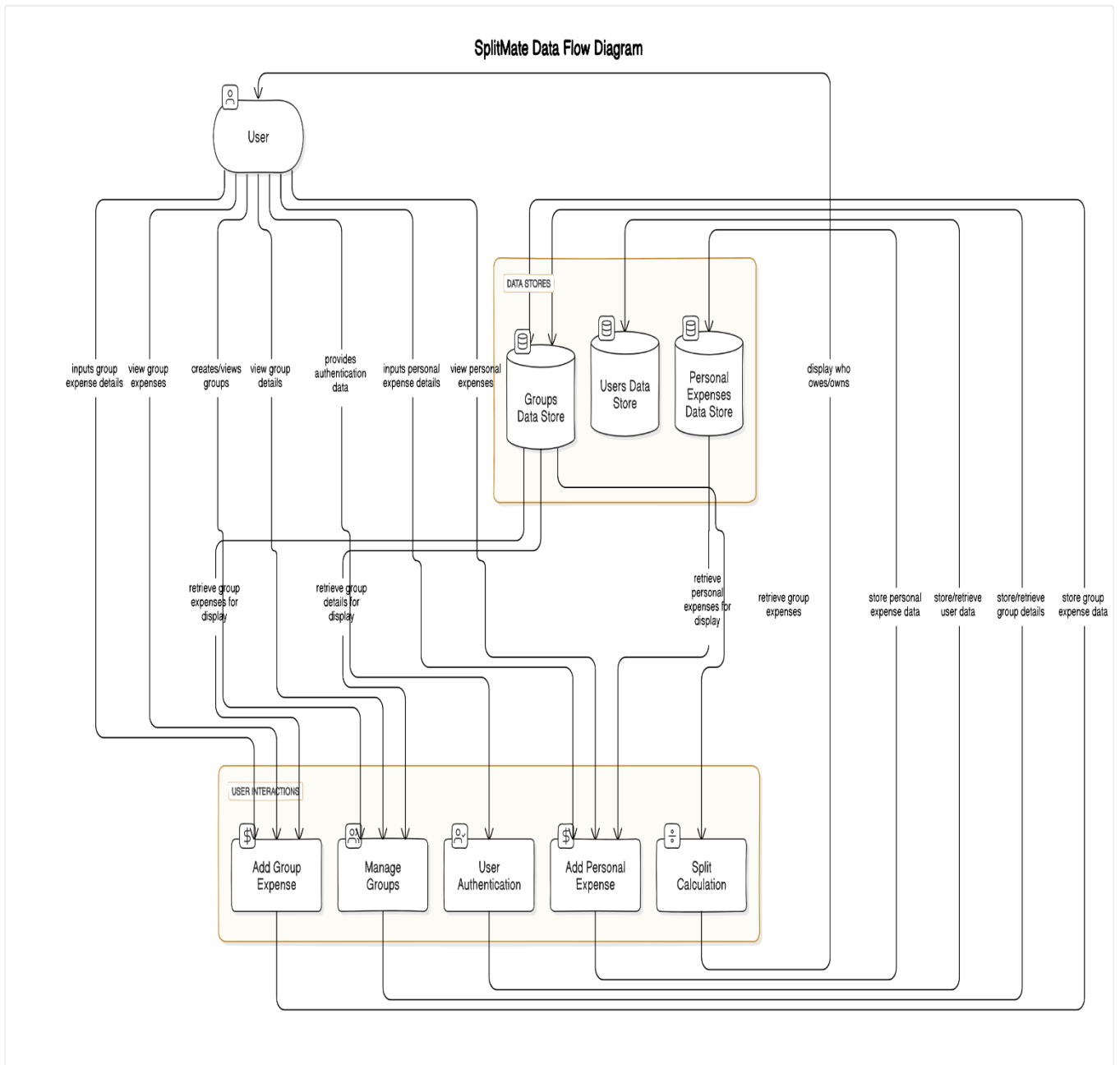
3.1 Data Flow Diagram

❖ Level 0 Data Flow Diagram / context diagram



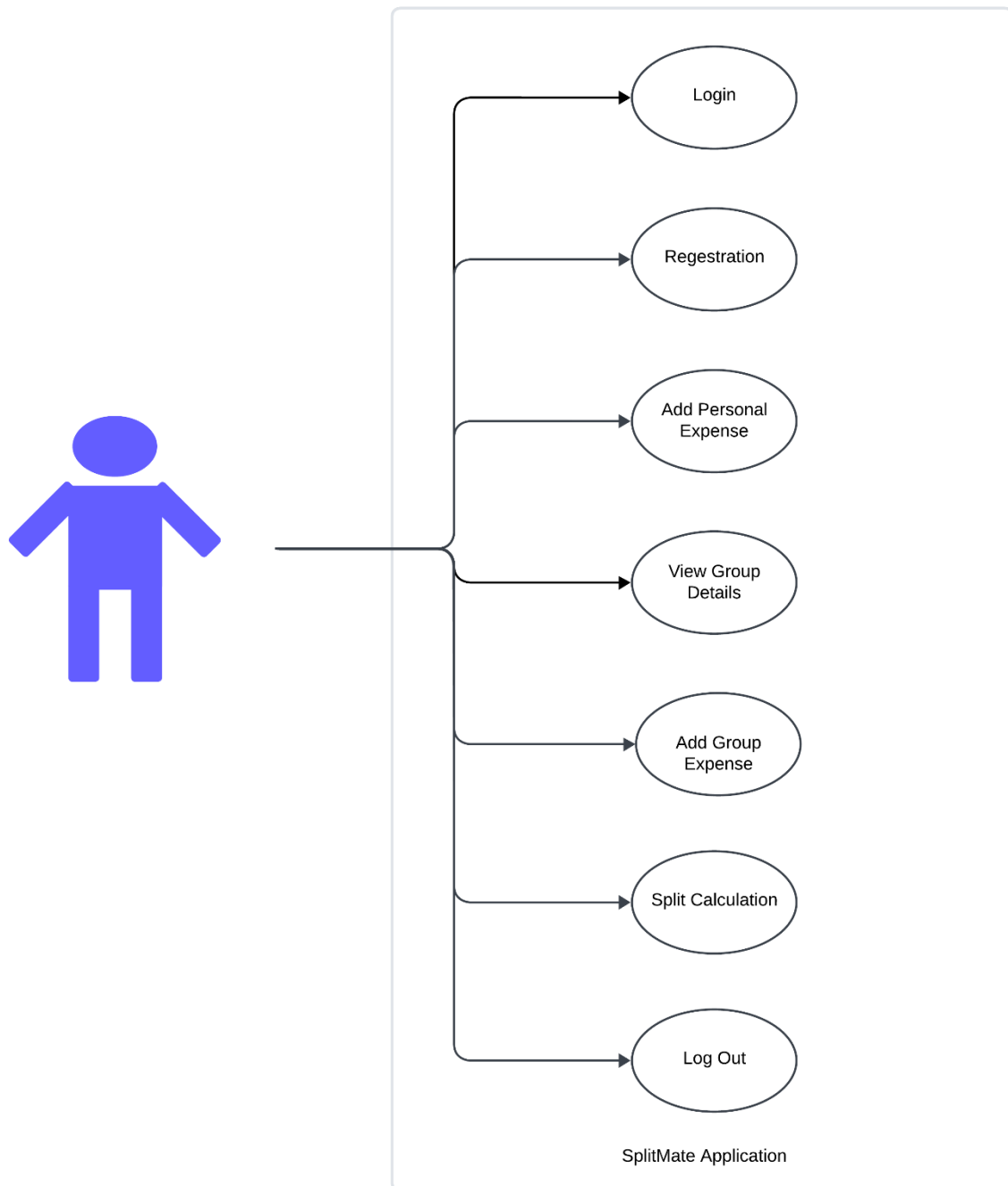
❖ Level 1 data flow diagram

Figure 1



3.2 UML Diagram

❖ Use Case Diagram



- Login Activity Diagram



3.3 Data Dictionary

Users Collection

Field Name	Data Type	Constraint	Description
email	String	Not Null	User Email
PhoneNumber	String	Not Null	User Phone
UID	String	Unique Id	Unique Username

PersonalExpenses Collection

Field Name	Data Type	Constraint	Description
amount	String	Not Null	Expense Amount
category	String	Not Null	Expense Description
date	String	Not Null	Expense Date
paymentMethod	String	Not Null	Payment Method

groups Collection (for Storing Multiple Group)

Field Name	Data Type	Constraint	Description
createdAt	String	Not Null	Date of group creation
groupname	String	Not Null	Name of Group
members	String[] (Array)	Not Null	Members Name
uid	String	Unique id	Unique id for user

expenses Collection (for group Expenses)

Field Name	Data Type	Constraint	Description
amount	String	Not Null	Expense Amount
category	String	Not Null	Expense Description
date	String	Not Null	Expense Date
paidBy	String	Not Null	Amount payer name

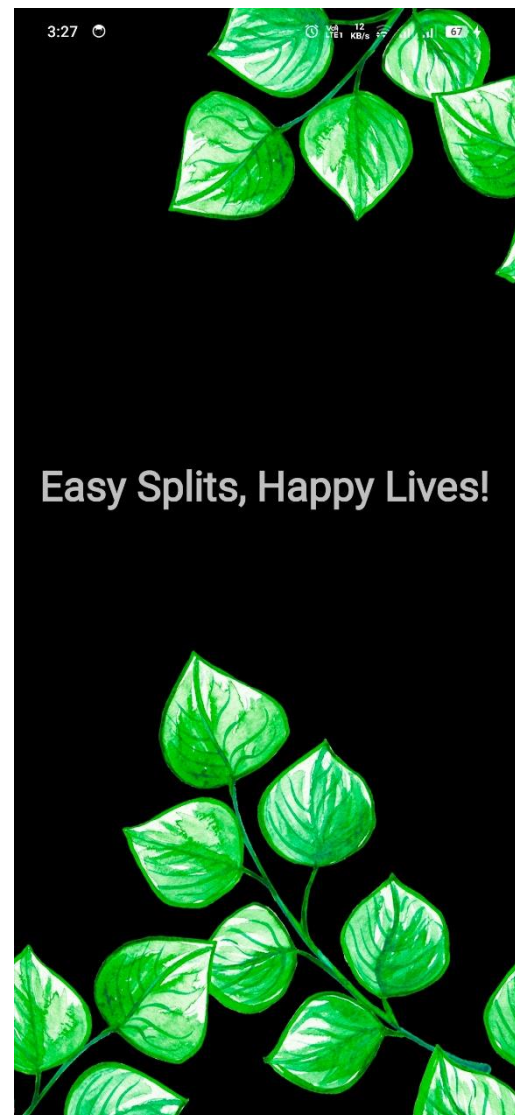
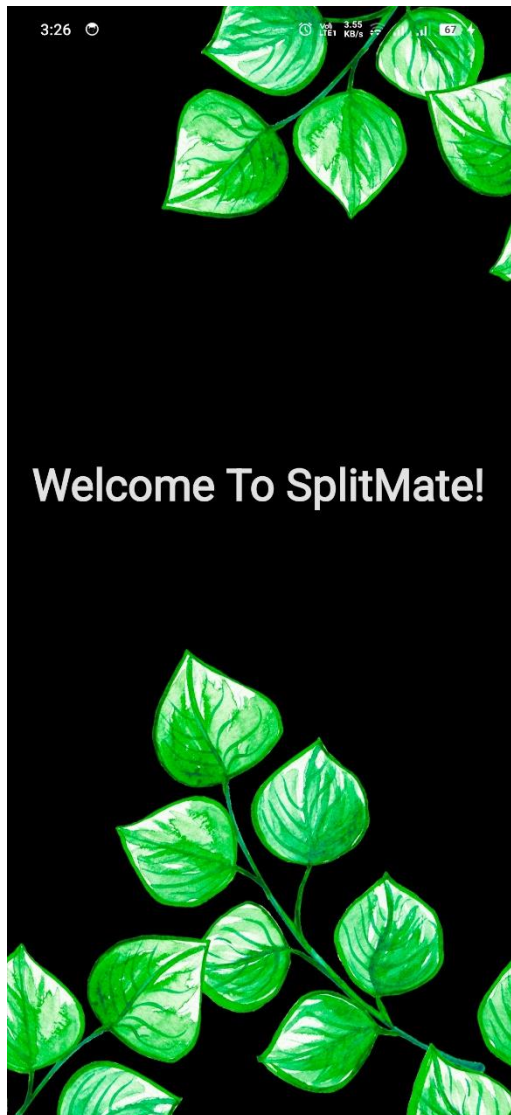
3.4 Interface Designs

[1] Animated Splash Screen



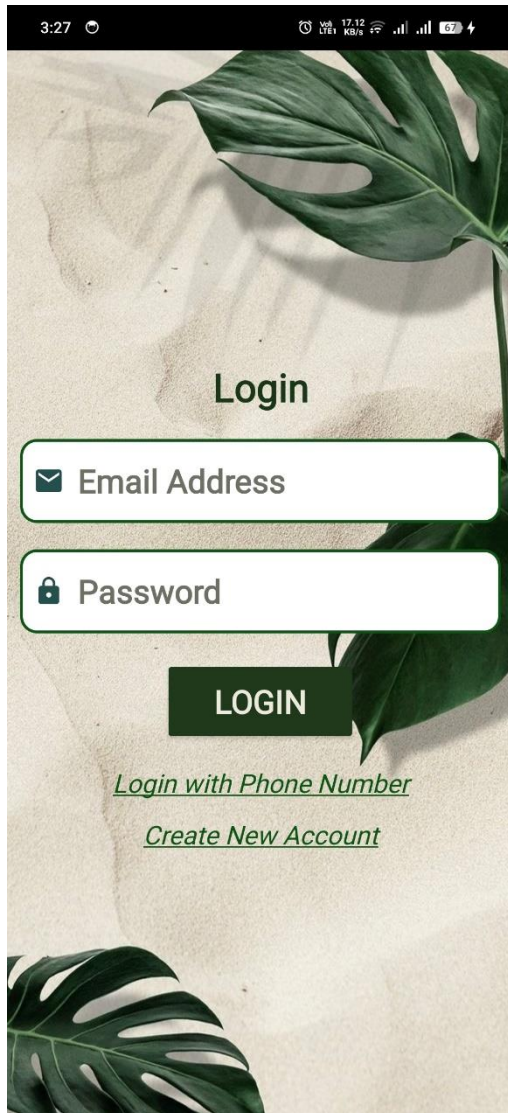
-This animated screen will be appearing each time while user open the app

[2] Welcome Screen

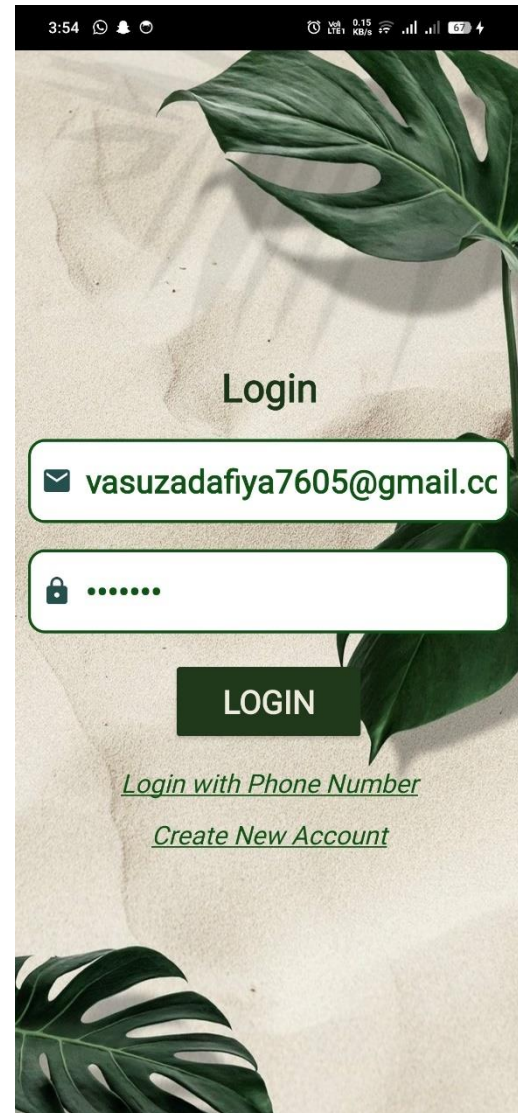


-this welcome screen only appears while user is logged out or opening app first time on their devices. If user is already logged in then it will not show to the user.

[3] Loginpage Activity



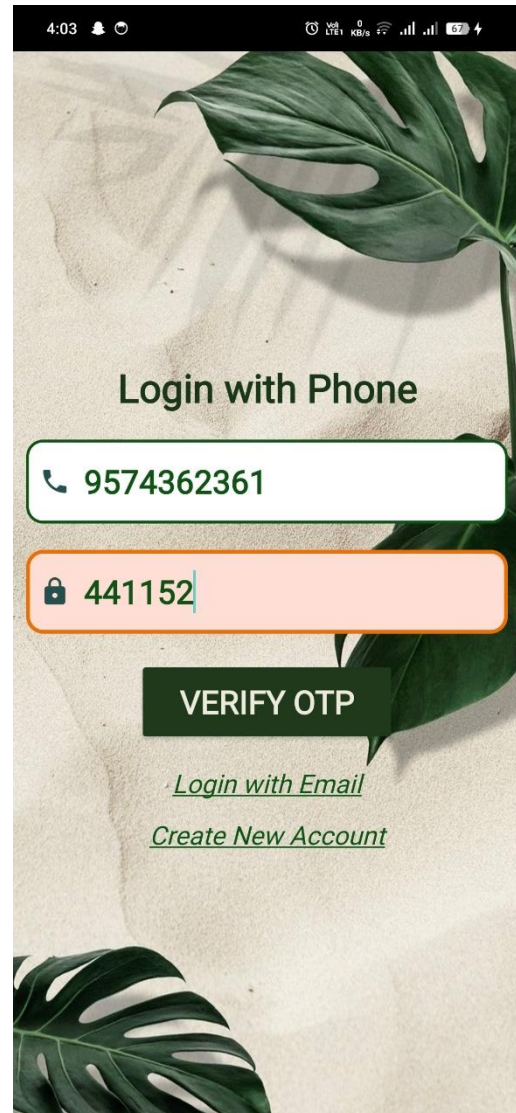
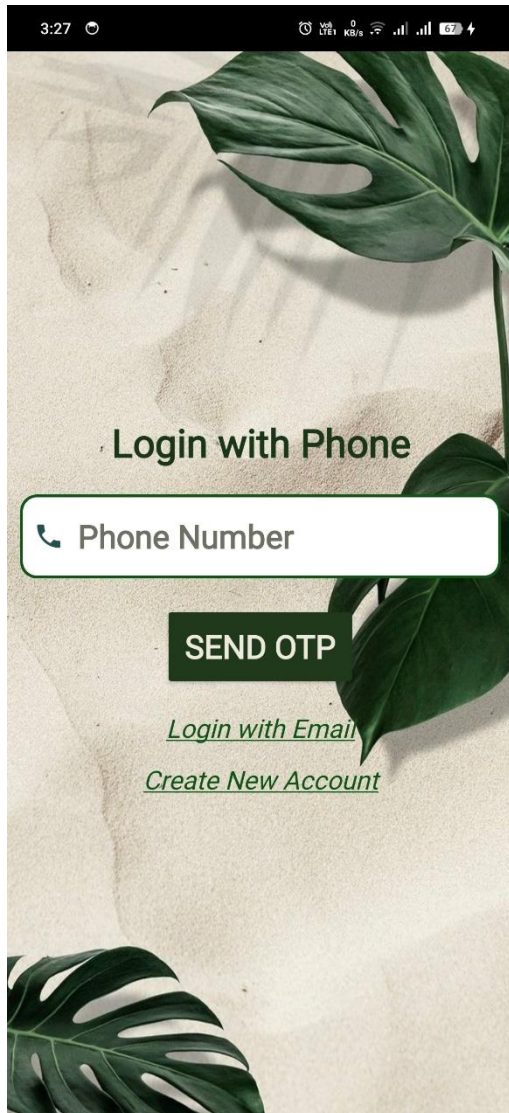
-without Data inserted



-with Data inserted

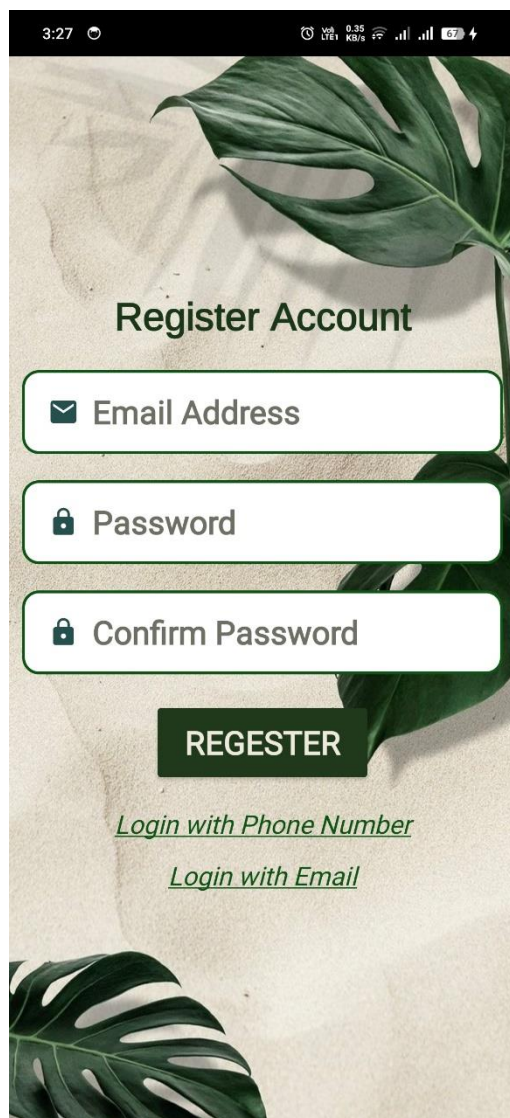
-Login Screen is fully Dynamic, it includes Register new user and Login with Phone number functionality in single Activity...

Login with Phone Number



-When user enter mobile number if number is valid then it will send OTP to the user and appear OTP input box that input 6 digits OTP...

Register New Account with Email



3:27

Register Account

Email Address

Password

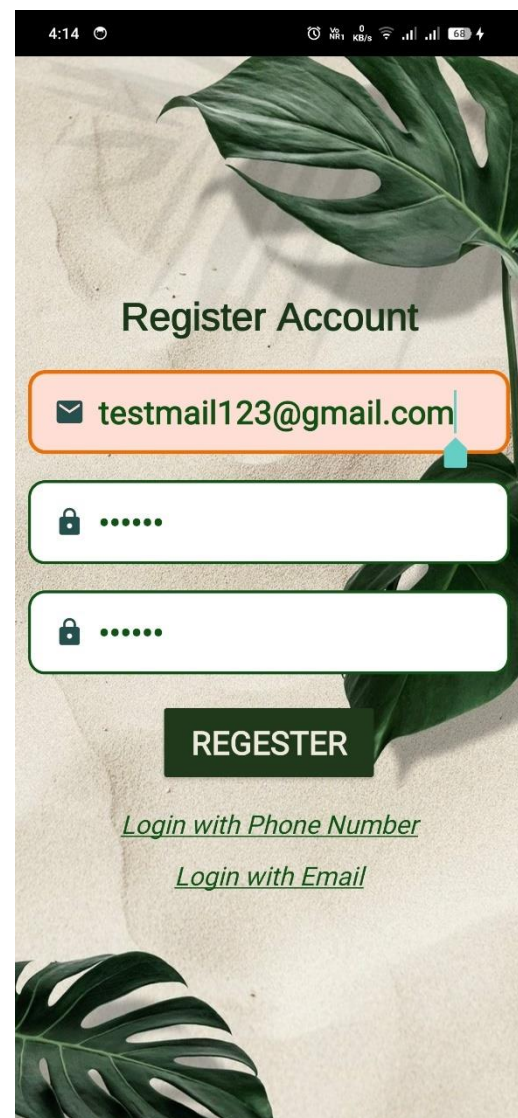
Confirm Password

REGISTER

[Login with Phone Number](#)

[Login with Email](#)

The image shows a mobile app registration screen. The background is a light beige color with a large green monstera leaf in the top right corner. The title 'Register Account' is centered at the top. Below it are three input fields: 'Email Address', 'Password', and 'Confirm Password'. Each field has a small icon (envelope for email, lock for password) on the left. Below the fields is a dark green button with the text 'REGISTER' in white. At the bottom, there are two links: 'Login with Phone Number' and 'Login with Email', both in green text.



4:14

Register Account

testmail123@gmail.com

.....

.....

REGISTER

[Login with Phone Number](#)

[Login with Email](#)

The image shows the same mobile app registration screen as the previous one, but with the 'Email Address' field filled with 'testmail123@gmail.com'. The 'Password' and 'Confirm Password' fields are still empty. The 'REGISTER' button and links are the same.

-New Signup are Applicable if...

- If email is already not Exist
- If email format is valid
- Email and password are not empty.
- Password are same in both inputBox.
- Password must be greater than 6.
- Password must be contained [a-z, A-Z,0-9].

[4] HomePage Activity

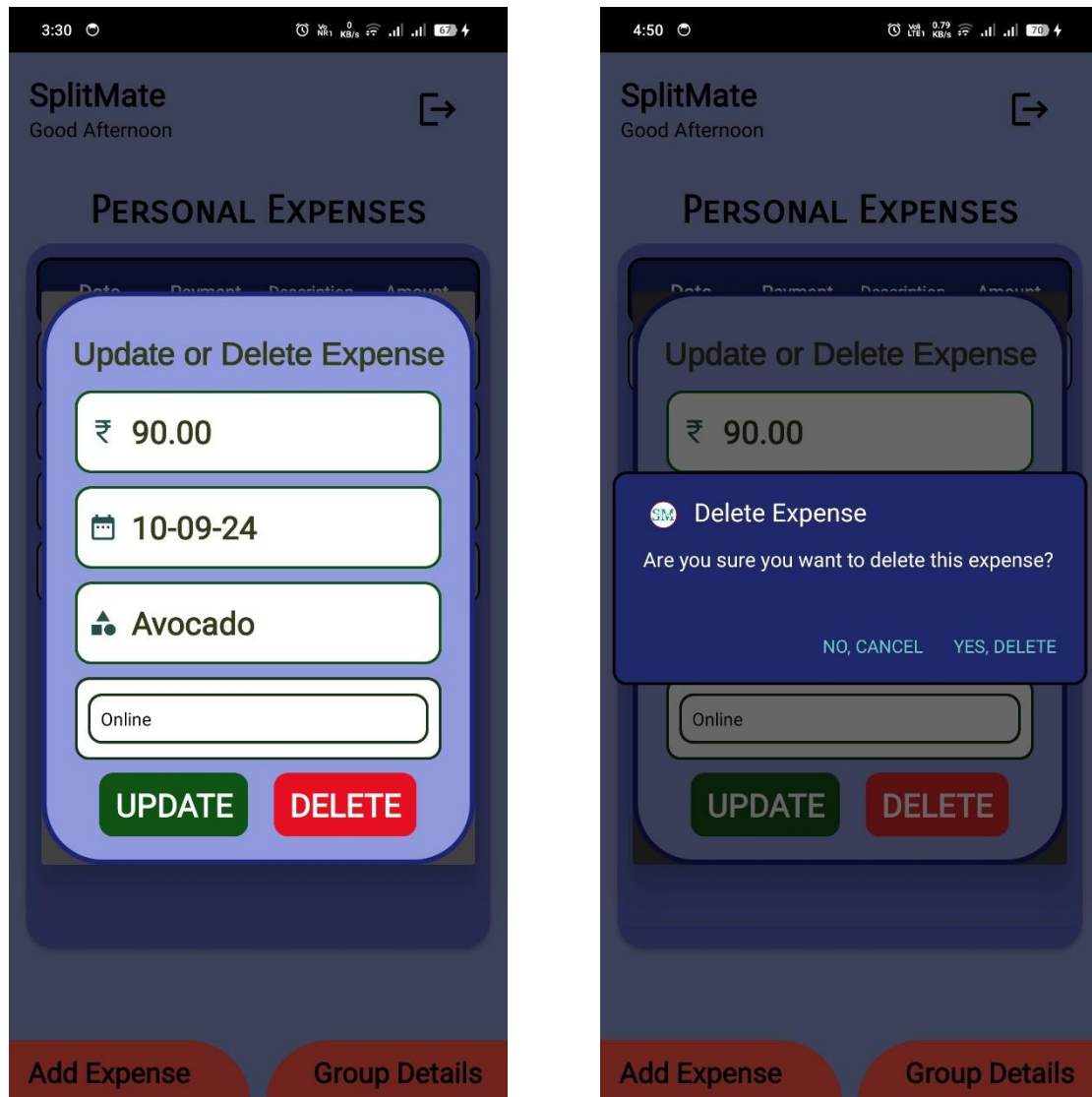


-without Personal Expense



-with Personal Expenses

[5] Update Delete Dialogue for Personal Expenses



- this dialogue will appear when user click on any of one expense.
- This dialogue automatically fetches the data from clicked expense.
- When user click on the DELETE button then another alert dialogue will appear for confirmation.

[6] Add Personal Expense Activity

The image displays two screenshots of a mobile application interface for adding a personal expense. Both screenshots show a form titled "Add your Personal Expense" with a background image of a deer in a forest.

Left Screenshot (Form Fields):

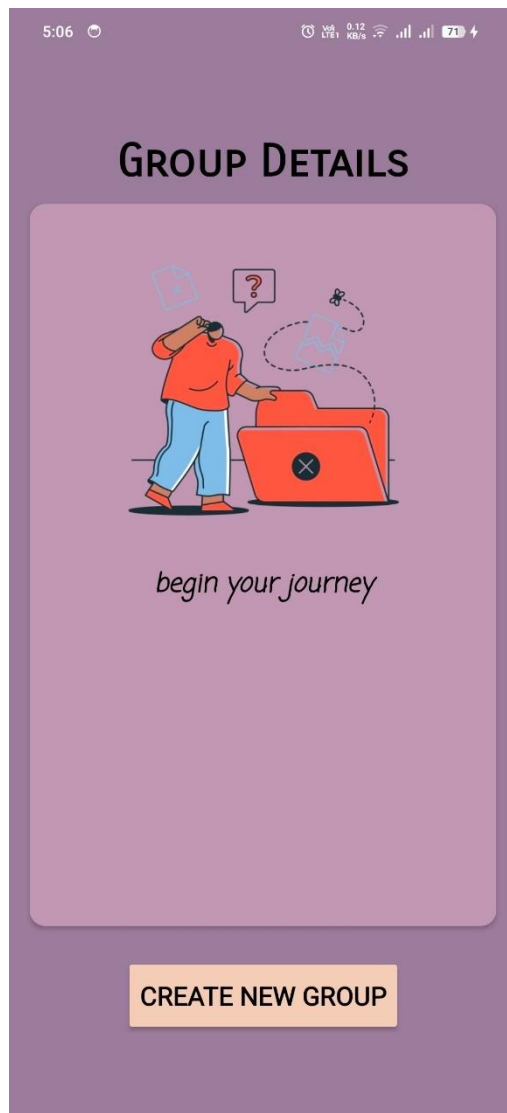
- Amount:** ₹ Amount
- Date:** 12-09-24
- Category (Optional):** Category (Optional)
- Payment Method:** Select Payment Method (dropdown menu with options: Cash, Online, Other)

Right Screenshot (Filled Form):

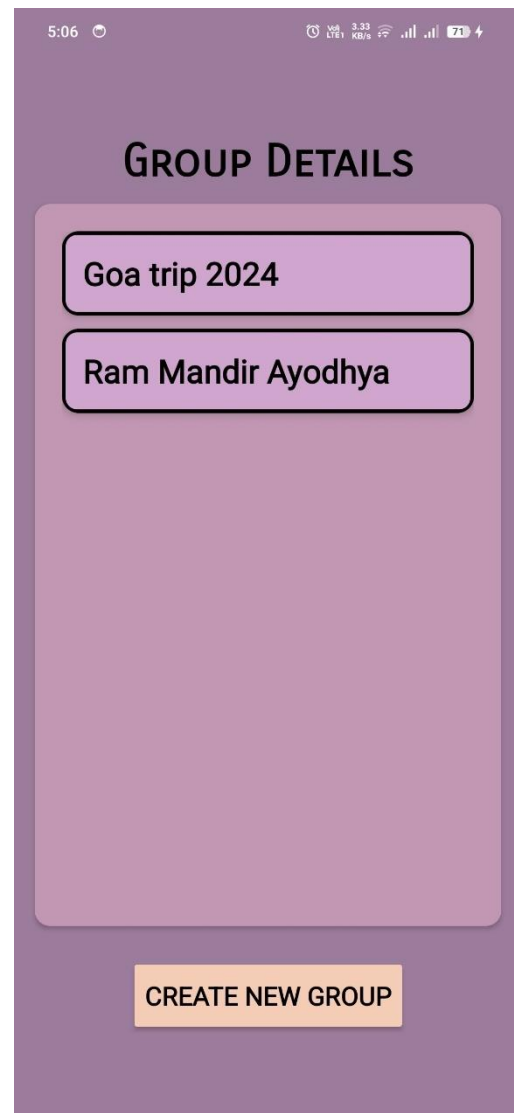
- Amount:** ₹ 170.43
- Date:** 12-09-24
- Category:** Breakfast
- Payment Method:** Online
- Action:** ADD EXPENSE button

- this activity help to add your personal expense into the database
- amount, date and payment method field are mandatory.
- When you click on the Add Expense Button then it will add expense to the Database and that data will show you on a homepage.

[7] Group Data Activity



- without groups



-with groups

- All Groups that created by current user will showing here
- When user click on the groupname then all groups expense will be appear dynamically.
- When user click on CREATE NEW GROUP button then it redirect to creategroup activity...

[8] Create Group Activity

The image displays two sequential screenshots of a mobile application interface for creating a group. Both screens have a background image of a Monstera plant and a status bar at the top.

Left Screenshot (5:19): The title is "Create Your Squad". It features two input fields: "Group Name" and "Number of members". At the bottom, there is a red button labeled "ADD FIELDS".

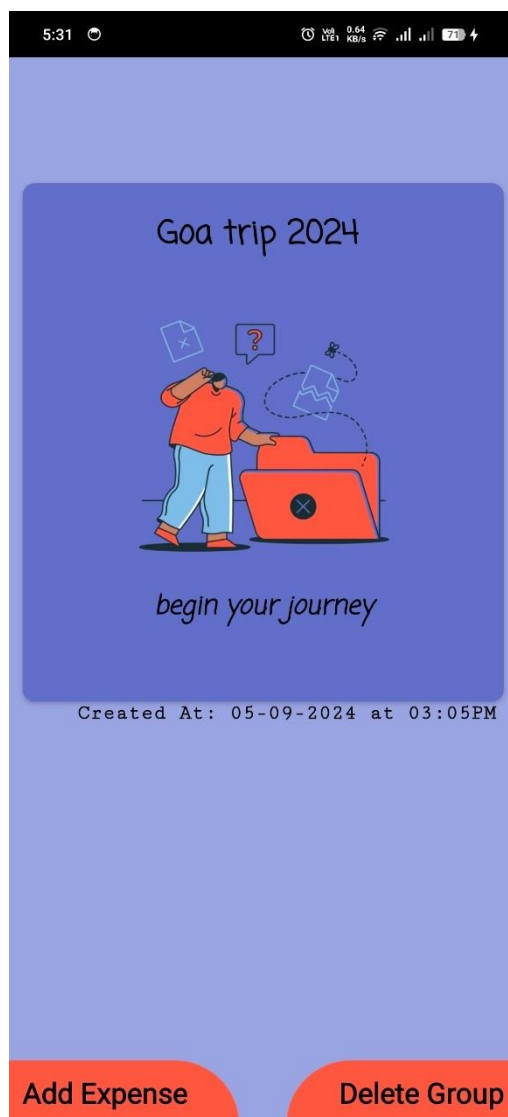
Right Screenshot (5:20): After clicking "ADD FIELDS", the interface updates. The "Group Name" field now contains the text "test". Below it, a section titled "Enter Member Names" contains three input fields with the names "vasu", "sahil", and "tushar". The "tushar" field is highlighted with an orange border. At the bottom, there is a red button labeled "CREATE GROUP".

- When user enter integer on number of members field, after click on the ADD FIELDS Button then it will dynamically give fields for group members name.
- After entering the name of group members user can create the group.

Validations

- Group name must be unique then existing Group name.
- Group Member name must be unique

[9] Group Details Activity



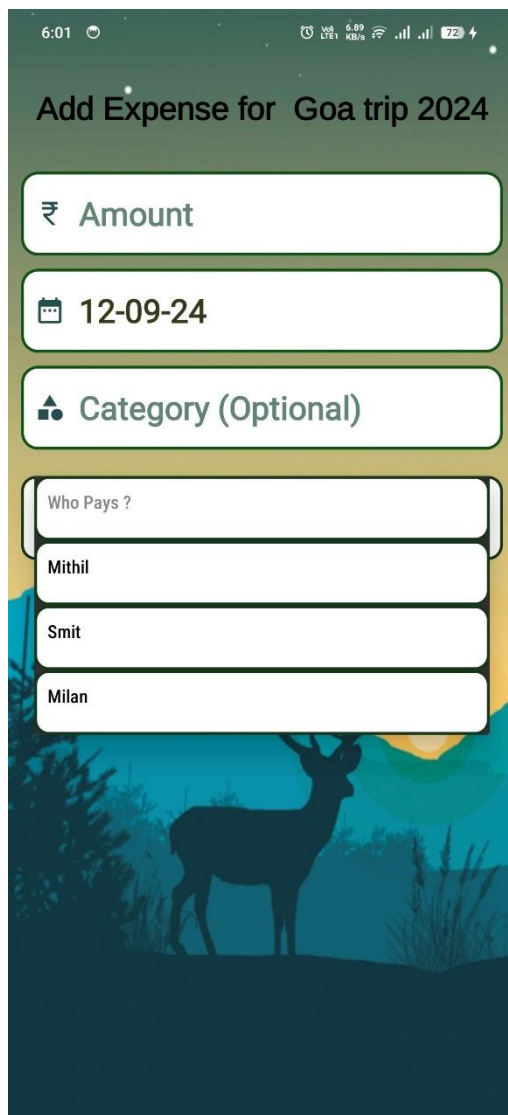
-without group expense



-with group expense

- This activity will shows individuals group expense dynamically
- this will fetch automatically group details according to groupId
- automatically make split and calculate who owes or own amount.

[10]. Add Group Expense Activity.



6:01

Add Expense for Goa trip 2024

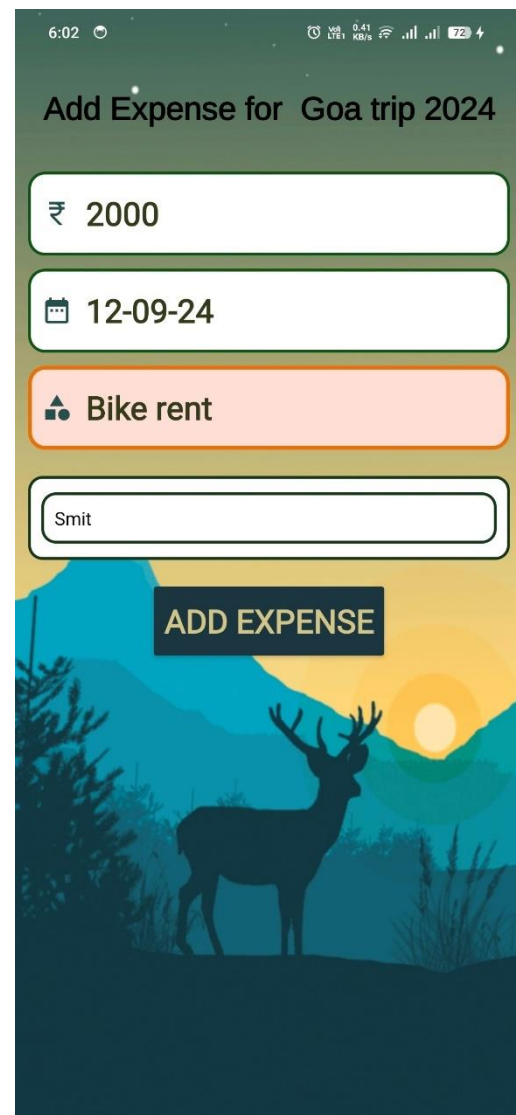
₹ Amount

12-09-24

Category (Optional)

Who Pays ?

- Mithil
- Smit
- Milan



6:02

Add Expense for Goa trip 2024

₹ 2000

12-09-24

Bike rent

Smit

ADD EXPENSE

- here member names are dynamically fetched from the database...

4. TESTING

1. User Registration and Authentication

Test Case 1: User Registration

- **Objective:** Verify that a new user can register successfully.
- **Preconditions:** The app is installed, and the user is on the registration screen.
- **Steps:**
 1. Enter a valid email address.
 2. Enter a strong password.
 3. Click on the "Register" button.
- **Expected Result:** The user should receive a confirmation message, and the app should redirect to the login screen.

Test Case 2: User Login

- **Objective:** Verify that a registered user can log in successfully.
- **Preconditions:** The user has a registered account.
- **Steps:**
 1. Enter the registered email address.
 2. Enter the correct password.
 3. Click on the "Login" button.
- **Expected Result:** The user should be logged in and redirected to the home screen displaying personal expenses.

Test Case 3: Invalid Login

- **Objective:** Verify that the app handles invalid login attempts.
- **Preconditions:** The user is on the login screen.
- **Steps:**
 1. Enter an unregistered email address or incorrect password.
 2. Click on the "Login" button.
- **Expected Result:** The app should display an error message indicating invalid credentials.

2. Personal Expense Logging

Test Case 4: Add Personal Expense

- **Objective:** Verify that a user can add a new personal expense.
- **Preconditions:** The user is logged in and on the home screen.
- **Steps:**
 1. Click on the "Add Personal Expense" button.
 2. Fill in the amount, date, category, and payment method fields.
 3. Click on the "add expense" button.
- **Expected Result:** The new personal expense should be saved and displayed on the home screen under personal expenses.

Test Case 5: Edit Personal Expense

- **Objective:** Verify that a user can edit an existing personal expense.
- **Preconditions:** The user has at least one personal expense recorded.
- **Steps:**
 1. Select an existing expense from the home screen.
 2. Click on the "Edit" button.
 3. Modify the expense details (e.g., amount, category).
 4. Click on the "Save" button.
- **Expected Result:** The changes should be saved, and the updated expense should be displayed.

Test Case 6: Delete Personal Expense

- **Objective:** Verify that a user can delete a personal expense.
- **Preconditions:** The user has at least one personal expense recorded.
- **Steps:**
 1. Select an existing personal expense from the home screen.
 2. Click on the "Delete" button.
 3. Confirm the deletion.
- **Expected Result:** The personal expense should be removed from the home screen.

3. Group Expense Management

Test Case 7: Create Group

- **Objective:** Verify that a user can create a new group.
- **Preconditions:** The user is logged in and on the "Group Details" screen.
- **Steps:**
 1. Click on the "Create New Group" button.
 2. Enter the group name, add members, and click "Save".
- **Expected Result:** The group should be created and displayed in the group list.

Test Case 8: Add Group Expense

- **Objective:** Verify that a user can add an expense to a group.
- **Preconditions:** The user is a member of a group.
- **Steps:**
 1. Select an existing group from the group list.
 2. Click on the "Add Group Expense" button.
 3. Enter the amount, date, category, and paid by details.
 4. Click on the "Save" button.
- **Expected Result:** The expense should be added to the group and should reflect in the group expense summary.

Test Case 9: Split Expense Calculation

- **Objective:** Verify that group expenses are split correctly.
- **Preconditions:** There are multiple group members, and expenses have been added.
- **Steps:**
 1. Navigate to the group details screen.
 2. Check the calculated owed/own amounts for each member.
 3. **Expected Result:** The app should correctly calculate and display who owes and who owns money in the group.

Test Case 10: Edit Group Expense

- **Objective:** Verify that a user can edit an existing group expense.
- **Preconditions:** The group has at least one recorded expense.
- **Steps:**
 1. Select an existing expense from the group details.
 2. Click on the "Edit" button.
 3. Modify the expense details (amount, paid by, etc.).
 4. Click on the "Save" button.
- **Expected Result:** The changes should be saved, and the updated expense should be displayed in the group.

Test Case 11: Delete Group Expense

- **Objective:** Verify that a user can delete a group expense.
- **Preconditions:** The group has at least one recorded expense.
- **Steps:**
 1. Select an existing group expense from the group details.
 2. Click on the "Delete" button.
 3. Confirm the deletion.

- **Expected Result:** The group expense should be removed from the group details.

4. General App Behaviour

Test Case 12: Log Out

- **Objective:** Verify that the user can log out of the app.
- **Preconditions:** The user is logged in.
- **Steps:**
 1. Click on the "Log Out" button in homepage.
- **Expected Result:** The user should be logged out and redirected to the login screen.

5. FUTURE ENHANCEMENT

1. Integration with Payment Gateways:

- **Description:** Allow users to directly settle group balances through integrated payment gateways (e.g., UPI, PayPal, etc.).
- **Benefit:** Users can instantly pay each other, reducing the hassle of tracking manual payments outside the app.
- **Implementation:** Integrate payment APIs with group expenses so users can make payments and track them directly within the app.

2. Multi-Currency Support:

- **Description:** Provide support for multiple currencies, allowing groups with members from different countries to track expenses in their local currency and automatically convert between them.
- **Benefit:** Increases the app's usability for international users.
- **Implementation:** Use currency conversion APIs to track and display expenses in various currencies with real-time exchange rates.

3. Recurring Expenses:

- **Description:** Add the option for users to set recurring expenses (e.g., rent, subscriptions) that are automatically added at specified intervals.
- **Benefit:** Simplifies expense tracking for regular, predictable expenses.
- **Implementation:** Provide scheduling functionality where users can set up recurring expenses with reminders.

4. Analytics Dashboard:

- **Description:** Implement a visual analytics dashboard that displays spending patterns, monthly reports, and group balance summaries in charts and graphs.
- **Benefit:** Gives users better insights into their spending habits, helping them budget more effectively.
- **Implementation:** Use charting libraries to visualize data and allow users to filter and analyze their personal and group expenses.

5. OCR (Optical Character Recognition) for Bills:

- **Description:** Allow users to upload pictures of receipts or bills, and automatically extract key data (amount, date, etc.) to populate expense details.
- **Benefit:** Reduces manual entry of expense details and improves the app's user experience.
- **Implementation:** Integrate OCR tools like Google Vision to scan images and extract text for automatic input.

6. Expense Notifications and Reminders:

- **Description:** Send notifications for overdue payments, when new group expenses are added, or when expenses are settled.
- **Benefit:** Keeps users informed about their financial obligations and upcoming expenses.
- **Implementation:** Use Firebase Cloud Messaging (FCM) to trigger real-time notifications based on user activity or pre-set reminders.

7. Advanced Group Splitting Options:

- **Description:** Provide more flexibility for splitting group expenses, including custom percentage splits, shares based on different roles, or per-item cost splitting.
- **Benefit:** Supports more complex group payment scenarios, such as trips or events with unequal contributions.
- **Implementation:** Modify the group expense logic to allow different split methods (percentage, itemized, or custom rules) when adding a group expense.

8. Expense Export and Import:

- **Description:** Allow users to export their expenses to CSV, Excel, or PDF format and import them back if needed.
- **Benefit:** Provides flexibility for users who want to back up their data or use it for external reporting or tracking.
- **Implementation:** Implement functionality to export and import data from the app's database to common file formats like CSV or Excel.

6. BIBLIOGRAPHY

1. Firebase Documentation

- Firebase, "Firebase Documentation." Available: <https://firebase.google.com/docs> . Accessed: September 2024.

2. Android Development Best Practices

- Google, "Android Developer Guide." [Online]. Available: <https://developer.android.com/guide>. Accessed: September 2024.

3. Firestore Database Guide

- Google, "Firestore: NoSQL Database for the Mobile Apps." [Online]. Available: <https://firebase.google.com/docs/firestore> . Accessed: September 2024.

4. User Interface Design Principles

- Alan Cooper, Robert Reimann, David Cronin, and Christopher Noessel, *About Face: The Essentials of Interaction Design*, 4th ed., Indianapolis: Wiley Publishing, 2014.

5. Software Testing Techniques

- Cem Kaner, *Testing Computer Software*, 2nd ed., New York: John Wiley & Sons, 1999.

6. Object-Oriented Design and UML

- Martin Fowler, *UML Distilled: A Brief Guide to the Standard Object Modelling Language*, 3rd ed., Addison-Wesley, 2003.