JSS Mahavidyapeeta

JSS Science and Technology University
JSS Technical Institution Campus,
Mysore

Department of Information Science & Engineering

Subject: Software Engineering (IS610)

Project: Payment Tracker

Submitted to: Dr. B S Mahanand

Submitted by:

Sindhura L (01JST17IS050)
Spoorthi S (01JST17IS062)

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the “Payment Tracker” app which tracks and notifies the user about the payments due, by the user or to the user. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team.

1.2 Document Conventions

Term	Definition
User	Someone who interacts with the mobile phone application
GUI	Graphical User Interface
PTS	Payment Tracking System
ADT	Android development Tools
IDE	Integrated Development Environment
AVD	Android Virtual Device

1.3 Intended Audience and Reading Suggestions

This document is to be read by the development team, the project managers, marketing staff, testers, and documentation writers. The software

engineer/Developer and project managers need to become intimately familiar with the SRS. Others involved need to review the document.

Testers need an understanding of the system features to develop meaningful test cases and give useful feedback to the developers. The developers need to know the requirements of the software product they need to build.

1.4 Product Scope

The scope of this product is to help people who have to manage a lot of periodic payments from various sources. It also aids to track the payments due by the user. This product avoids the problem of repetitively setting reminders for various tasks. The application takes details about the payment and periodic interval after which the user will be notified. It stores the related images such as bond in encrypted form for the user's reference. It also helps users to calculate their overall income and expense over a time range and the balance remaining after that interval.

1.5 References

- IEEE Software Engineering Standards Committee," IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications"
- https://www.scribd.com/document/204773379/Srs-Example-2010-Group2
- https://developer.android.com/guide/topics/providers/calendar-provider
- https://firebase.google.com/docs/android/setup

2. Overall Description

2.1 Product Perspective

In the existing android applications, we generally have to set different reminders in the mobile phone for our schedule. These reminders only notify the user once on the intended date but for payment related activities periodic reminders are necessary. So, we have to set the remainder again and again. This is a tedious task. Our product avoids all these repetitive tasks and helps the users to keep track of their payments in an organized way. It also provides a way to track the user's investments and commitments in various platforms.

2.2 Product Functions

- The app requests login credentials of the user to access the contents of the app providing a layer of security.
- After logging in the user can set periodic reminders regarding various payments/events or view details of the previously set reminders.
- It gives notifications to the user along with the details entered by them on the day of the event.
- The app also allows the user to save photos related to the payment/event like bonds along with the reminder in encrypted format. Thus, keeping sensitive information like account number, policy number, etc safe.
- The details regarding the payment along with the image is stored in a server thus the data will be safe from any system malfunctions.
- The app allows user to keep track of their investments and commitments in various financial platforms by helping the user to know their total income and expenses over a period of time.

2.3 User Classes and Characteristics

- Insurance policy holders can use the app in order to set periodic reminders, which notify them about the amount and date of premium payment.
- Fixed deposit holders can use the app to set the reminder about the maturity amount and date.
- Users who need to pay or collect rent can also use the app to set reminders. This helps users who need to collect rent/payments periodically from large number of parties.
- Users can set even other general-purpose reminders.
- App also provides a facility for the users to store the images such as bonds etc related to reminders in encrypted format safely.
- It helps users who have to keep track of a lot of investments and commitments in different financial institutions. It shows the remaining non-invested balance after a given time range thus allowing the user decide to whether to cut down their expenses or to invest in other schemes.

2.4 Operating Environment

Hardware Requirements:

- The absolute minimum for Android were originally a 200 MHz processor, 32 MB of RAM, and 32 MB of storage.

Software Requirement:

All versions of Android which are compatible,

- Android 10
- Android 9.0 "Pie"
- Android 8.0 and Android 8.1 "Oreo"
- Android 7.0 and Android 7.1 "Nougat"
- Android 6.0 "Marshmallow"
- Android 5.0 “Lollipop”
- Android 4.4 “KitKat”
- Android 4.3, Android 4.2, Android 4.1 “Jelly Bean”

2.5 Design and Implementation Constraints

- The internet connection is a constraint for the application. Since the application fetches data from the database over the internet, it is crucial that there is an internet connection for the application to function.
- Mobile application is constrained by the capacity of the database.

2.6 User Documentation:

Detailed document describing the usage of the product shall be delivered along with the product after the completion of the product.

2.7 Assumptions and Dependencies

- One assumption about the product is that it will always be used on mobile phones that have enough performance. If the phone does not have enough hardware resources available for the application, for example the users might

have allocated them with other applications, there may be scenarios where the application does not work as intended or does not work at all.

- If the app is not well supported, for example: not providing required permissions then it may not work as intended.
- Another assumption is that the product is installed in an android operating system. Other operating system example: iOS doesn't support the app
- It is assumed that the user will have reasonable internet connection facility, since app uses firebase, internet connection is necessary.

3. External Interface Requirements

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

3.1 User Interfaces

A first-time user of the mobile application should see the sign-in page when he/she opens the application. If the user has not registered, he/she should be able to do that on the sign-in page. If the user is not a first-time user, he/she should be able to login using his registered password and username.

After logging in the user's dashboard page appears where all the previous reminders are displayed. This page has a new reminder button using which user can set new reminders and also a calculator button. The add button navigates to a new page where the new reminders details are taken such as date, type of reminder, photo (optional) etc. The submit button in this page creates new reminders as per the details provided. The calculator button navigates to a new page where balance and time range can be provided to know the total income and expenditure over that interval and the remaining balance.

3.2 Hardware Interfaces

Since Mobile application does not have any designated hardware, it does not have any direct hardware interfaces. The System clock in the mobile phones and hardware connection to the database server is managed by the underlying operating system on the mobile phones and the server.

3.3 Software Interfaces

- Operating System ---- Android operating system for its best support and user-friendliness.
- Database ---- Firebase, used to save user's information
- IDE ---- To implement the project we use android studio3.5 framework (ADT), which provides integrated android developer tools for development and debugging.
- Language ---- Java
- Adobe XD is used for user interface design.
- Git is used for version control and collaboration of the project.

3.4 Communications Interfaces

When app connected to Firebase, it does not connect through normal HTTP. It connects through a WebSocket. WebSocket's are much, faster than HTTP. We don't have to make individual WebSocket calls, because one socket connection is plenty. All of your data syncs automatically through that single WebSocket as fast as your client's network can carry it.

4. System Features

4.1 Remainder

4.1.1 Description and Priority:

The users should be able to set reminders according to their needs. They should also be able to view, update, and delete the reminders that they have previously set.

4.1.2 Stimulus/Response Sequences:

The system shall request for the following details regarding the payment, from the user to set the reminder.

- Name
- Description
- Image
- Date

- Period

The system should notify the user on the specified date. It should also notify the user repeatedly, separated by a time interval specified by the user till the user deletes the reminder.

4.1.3 Functional Requirements:

- Add reminder
  The users should be able to set reminders according to their needs by providing required details.

- View reminder
  The users should be able to view all the reminders previously set by them after logging in. The system should display the details of the reminders like the name, description, image, etc.

- Update reminder
  The users should be able to update details specified in a particular reminder. The system should allow the user to perform operations like changing the description, image, date etc.

- Delete reminder
  The users should be able to delete reminders when they no longer want to be notified regarding certain payments. Only the intended reminder should be deleted completely from the firebase.

4.2 Store details in the server

4.2.1 Description and Priority:

The details entered by the users should be stored in the server so that they can access their data from any device.

4.2.2 Stimulus and Response:

The details entered by the user should be stored in the server and the user should be able to access them whenever required.

4.2.3 Functional Requirement:

- Access the data from any device.
- Even if the user uninstalls the app data should not be lost.

4.3 Calculate gross income and expenditure:

4.3.1 Description and Priority:

This feature should help the user to calculate gross investment amount or interest amount for a particular period of time which will be specified by the user. This information will help the user to plan for further investments.

4.3.2 Stimulus and Response:

The user should give the balance amount which is the non-invested money he is left with, start date and end date as input. The System will consider investments (or any other commitments like EMI's) as debit and interest amount as credit and should be able to calculate the gross expenditure or profits.

4.3.3 Functional Requirement:

- Calculate total gross income or expenditure

  The user should be able to calculate total gross income or expenditure amount which will help them for taking further investment decisions.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

System

The application will run on all Android devices running 4.1 (JellyBean) or Later. It will be around 11mb in size. The application will respond to the size of the screen and/or window the application is running in.

Response Time

The application should take less than 4 seconds when running on an Android phone and less than 8 second when on an emulator or tablet. The application will run fine until the user begins to multi-task between 3 or more processes.

Workload

The application must support approximately 10,000 users and allow 100 simultaneous connections to the database at the time of launch as we have used firebase spark plan.

Scalability

The application can be scaled up to support more customers and higher simultaneous access count by upgrading to a newer plan on firebase.

5.2 Safety Requirements

Since we have used firebase to store the reminders and encrypted images the user information is retained if the app is deleted from the mobile phone. Firebase provides a sense of safety to the user's data. It acts like a backup for the user's data.

5.3 Security Requirements

Since the app tracks payments security is an important aspect. The user's account is authenticated with username and password such that unauthenticated users cannot view the reminder information.

The user's data will be stored in the database in encrypted format as it may contain sensitive financial information.

5.4 Software Quality Attributes

- Reliability:
  The application should have capability to maintain minimum level of performance.
- Availability:
  The application should run 24x7 if internet connection is available.
- Portability:
  User should be able to install applications easily using apk.

6. System organization:

Client server architectural model is used as basic system organizational model for the project. The project uses Firebase as the database works on the basis of client server model. The details entered by the users will be stored in the firebase server. Clients can access their data from any device at any point of time. Good network connection is a necessary requirement for this application. Even if users uninstall the app the user's data will be safe in the server. And also, when app gets connected to Firebase, it does not connect through normal HTTP. It connects through a WebSocket. WebSocket's are much, faster than HTTP and response time is less.

7. Control style:

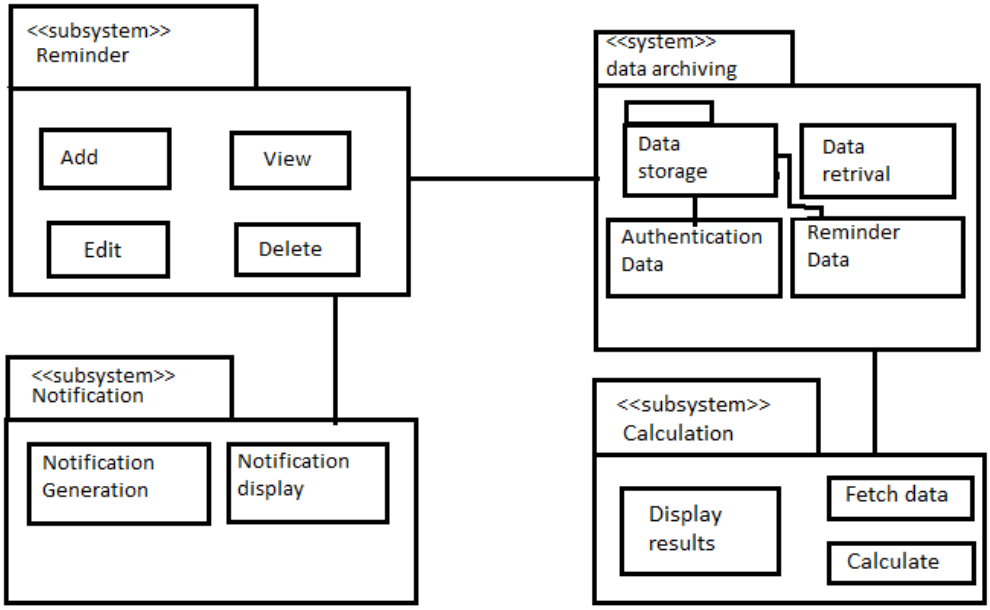
Event driven model is used as the control style for the project. The app is driven by externally generated event. When event occur, control is transferred to the subsystem that can handle the event. When the users want to create new reminder, the user should click the new reminder button and create reminder activity does the job. Here the button click is the acts as event and new reminder is created as a response.

8. Process model:

Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance. Each iteration passes through the requirements, design, coding and testing phases**.** And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.

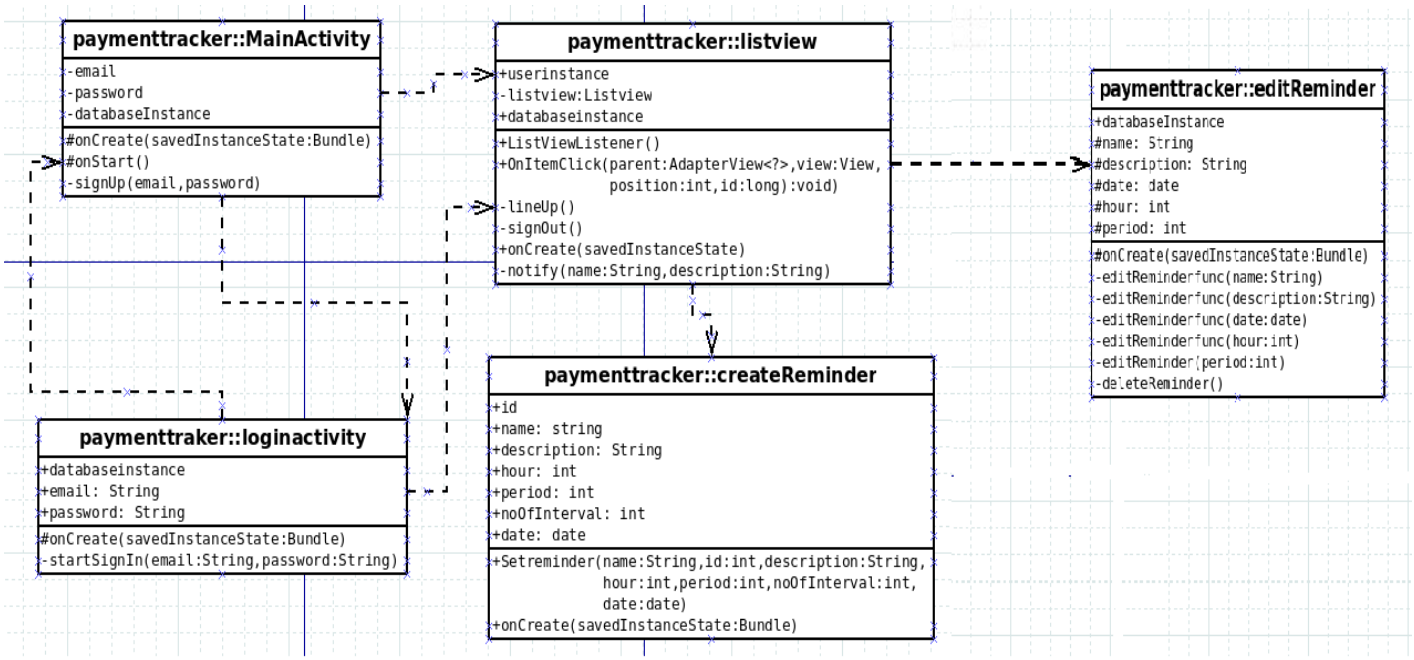
We have used incremental model to develop the app. Functional requirement are broken into 3 modules. First module was to develop feature to add periodic reminders which is the main feature of the app. In the next stage of development we have added an image feature, using which users can store images related to the reminder. In the last module, we have added a small feature which helps the user to calculate gross expenditure and income over a certain time interval and also the balance remaining after that interval.

9. Subsystem model:



The payment tracker application mainly consists of four subsystems, reminder subsystem, notification subsystem data archiving subsystem and calculation subsystem. The reminder subsystem contains the components/modules that help in performing activities such as adding new reminder, viewing existing reminders, edit reminders and delete reminders. This subsystem interacts with the user to take input regarding adding or updating reminders and displays the created reminders through the graphical user interface. The notification subsystem consists of notification generation and display components. The notification generation component generates notifications on the date specified by user. The display component displays the notification to the user in the notification bar. The data archiving subsystem has data storage subsystem and data retrieval module. The data storage subsystem contains two components that store user authentication data and reminder data. The data retrieval module retrieves data stored in the database and provides it to the reminder subsystem to display the retrieved information. The calculation subsystem has three modules that fetch data from database, calculate results and display them to the user.

10. Class diagram:



Payment tracker:: Main Activity

This activity takes email and password and creates an account for the user. The method onStart() checks if the user is logged in, if yes he is automatically directed to the list view activity.

Payment tracker:: login Activity

This activity takes email and password from the user in and logs the user in.

Payment tracker:: listview Activity

This activity lists all the previously set reminder by the user. notify() method sends notification at the time specified by the user.

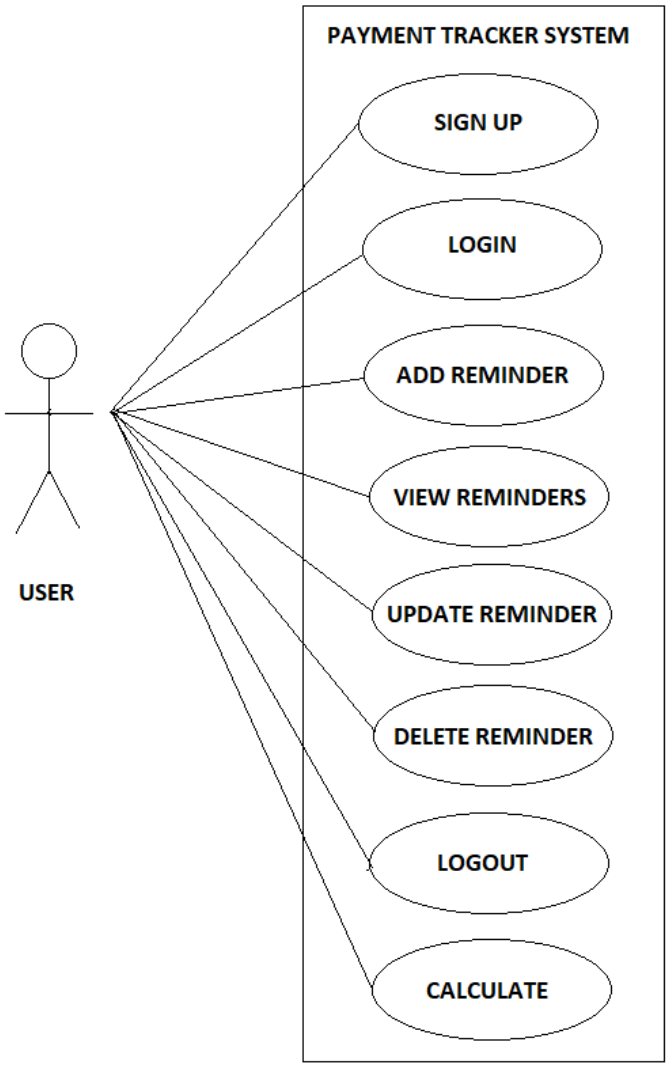
Payment tracker:: createReminder

This activity takes all the necessary information from the users and creates a new reminder.

Payment treacker :: editReminder Activity

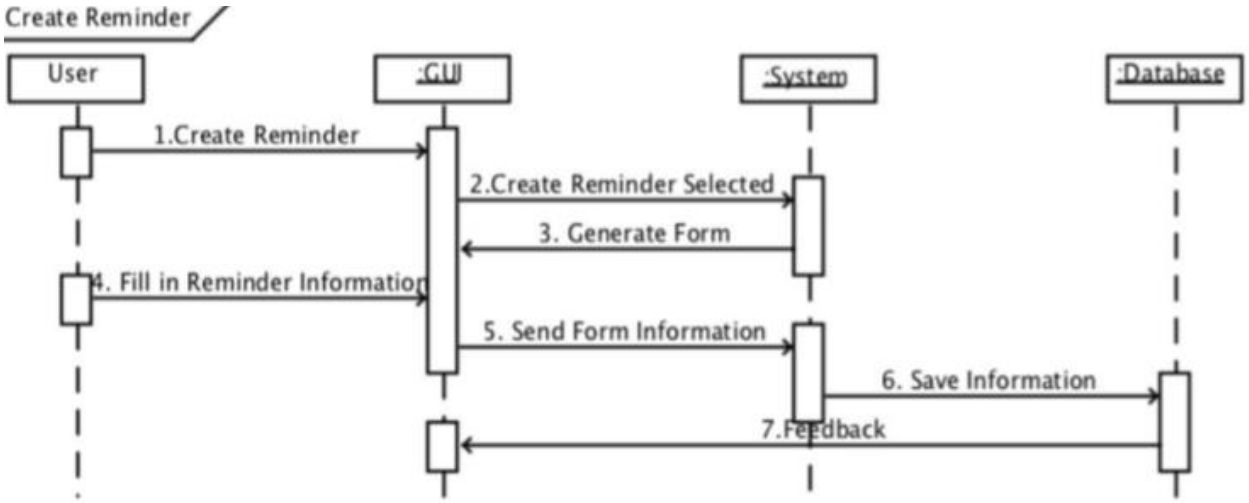
User can edit or delete the reminder information using this activity.

11. Usecase diagram:



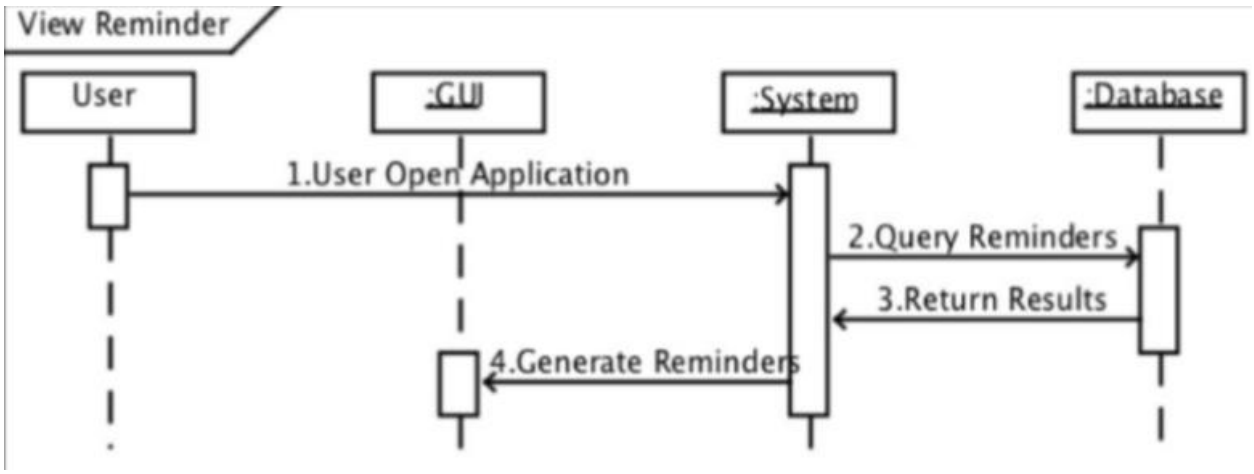
The different use cases are described with the help of sequence diagrams given below.

12. Sequence diagram:



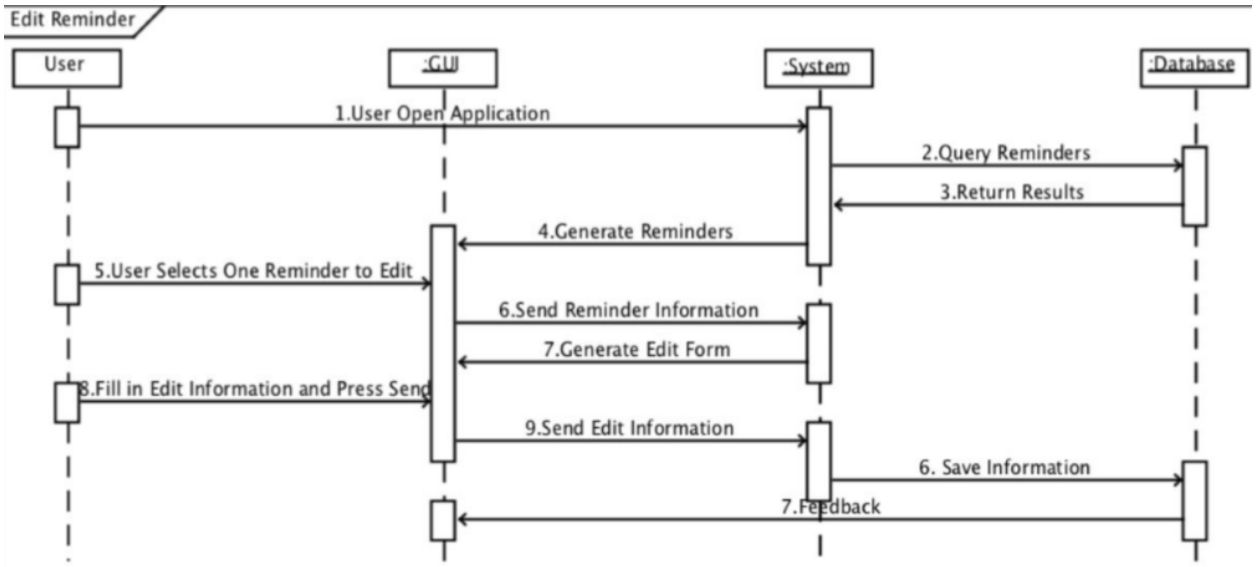
SD1. Sequence Diagram Create Reminder based on Use Case Create Reminder.

The user can create the reminder by selecting the create reminder option. The system generates a form. After entering the details in the form, the system stores the information in the database and sends feedback.



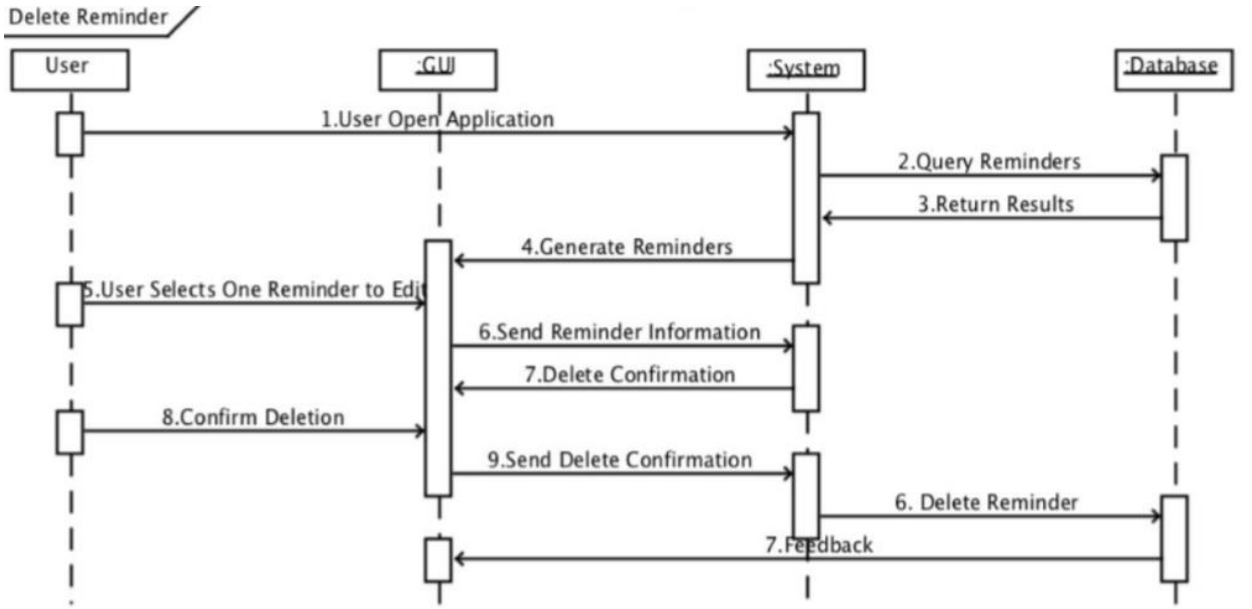
SD2. Sequence Diagram View Reminder based on Use Case View Reminder.

As the user login to the system, the system queries the database for the reminders specific to the user. Database sends the list of already set reminders which is presented to the user



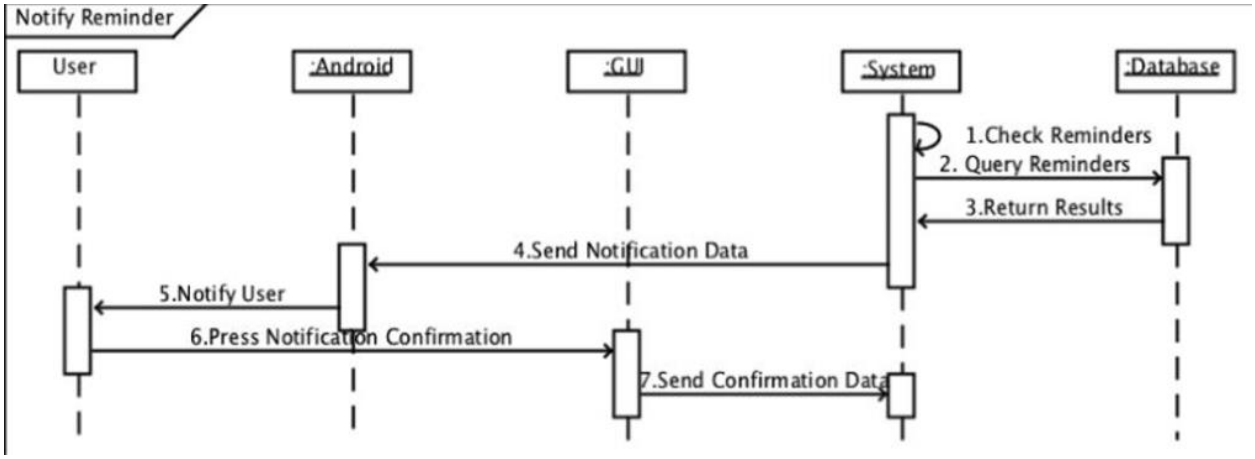
SD3. Sequence Diagram Edit Reminder based on extendable Use Case Edit Remin der.

After viewing the list of reminders, the user can select one among them to edit. User is asked enter new reminder information through a form. Edit information is sent to the database for storage and the database returns feedback to the user.



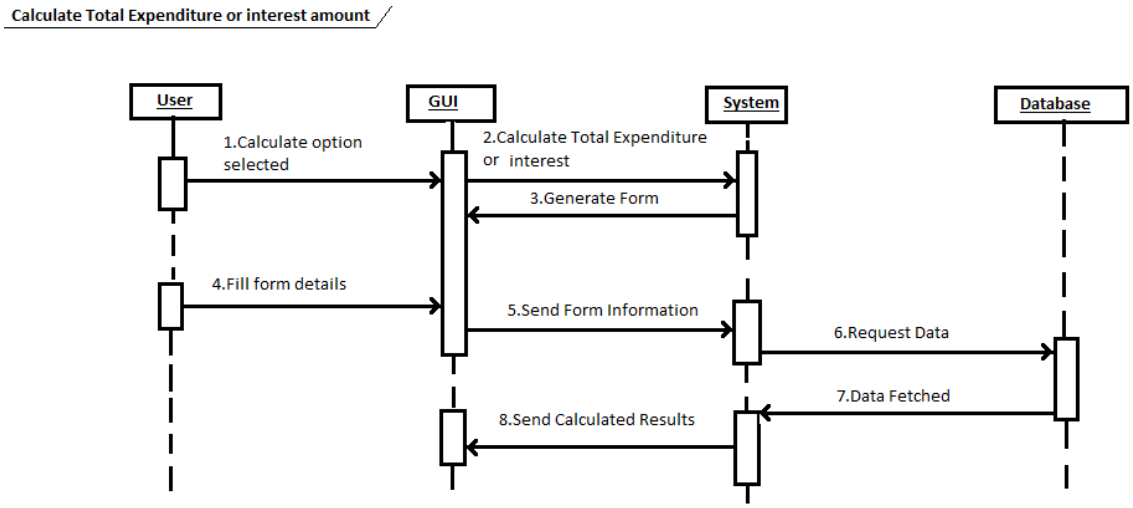
SD4. Sequence Diagram Delete Reminder based on extendable Use Case Delete Reminder.

User can select one among the listed reminders and request for deletion of the reminder. The system asks for the confirmation once again. If user confirms deletion the reminder is deleted from the database.



SD5. Sequence Diagram Notify Reminder based on Use Case Notify Reminder.

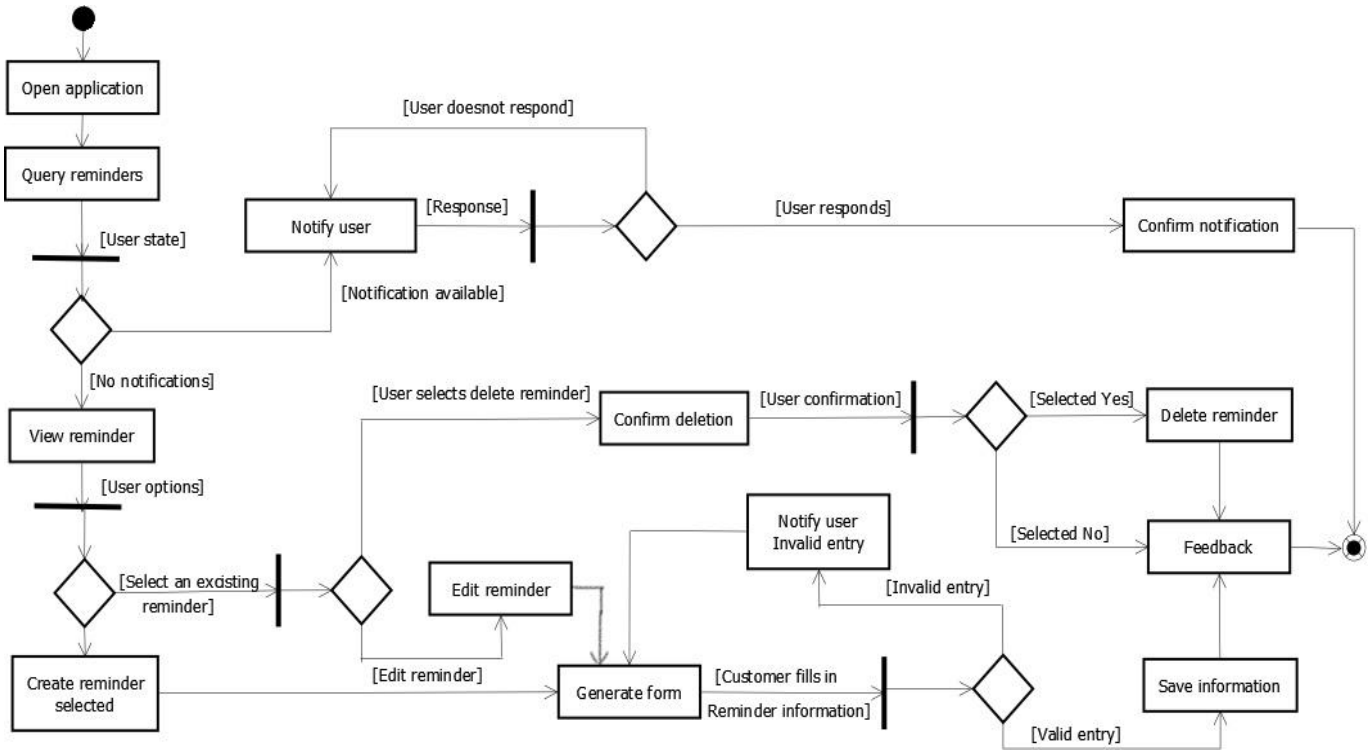
The system checks for the reminders by querying the database. At particular date and time specified by the user the system sends reminder information to the android which is presented as a notification for the user. The users can confirm the notification for acknowledgement.



SD6. Sequence Diagram Calculate total expenditure or interest amount based on Use Case Calculate.

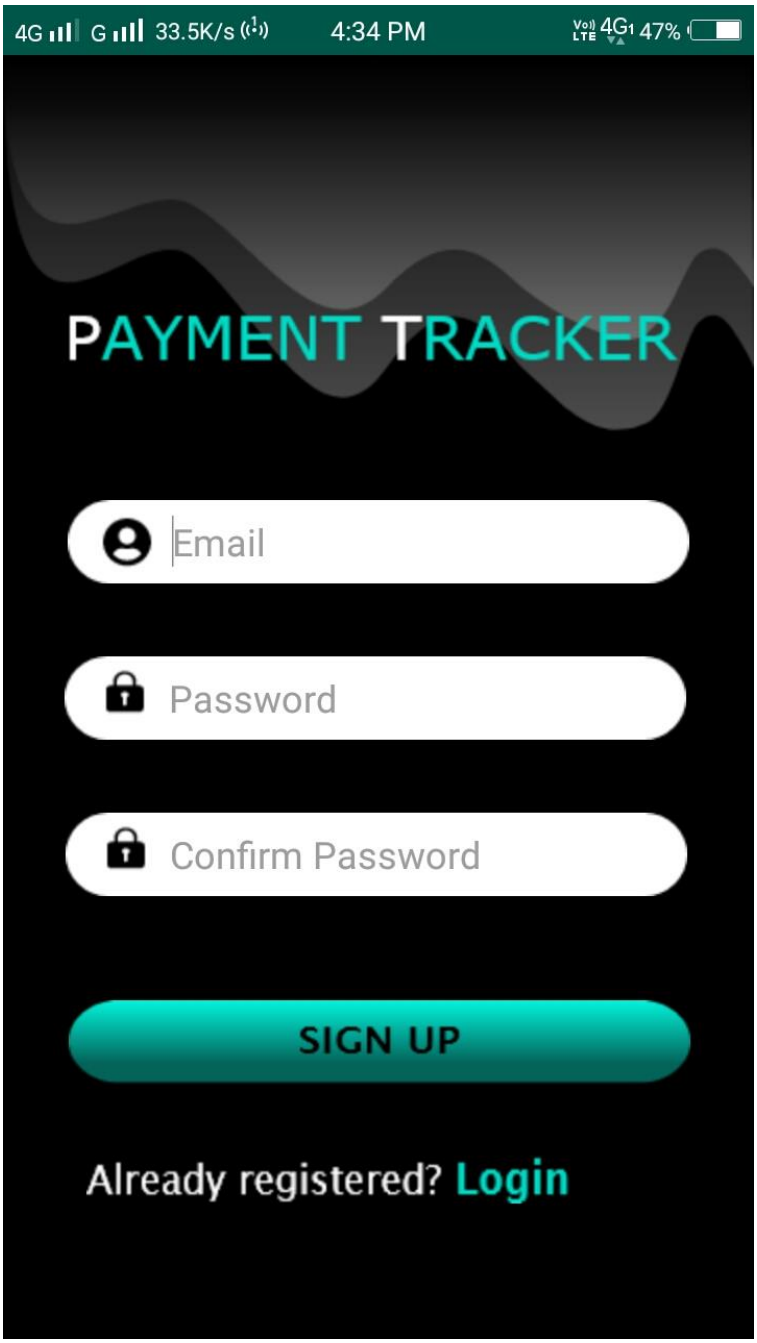
User can select the calculator option to find out their total income, expenditure and remaining balance by providing relevant details in the form. The system fetches relevant data from the database, does calculations and sends the results to be displayed to the user.

13. State diagram:



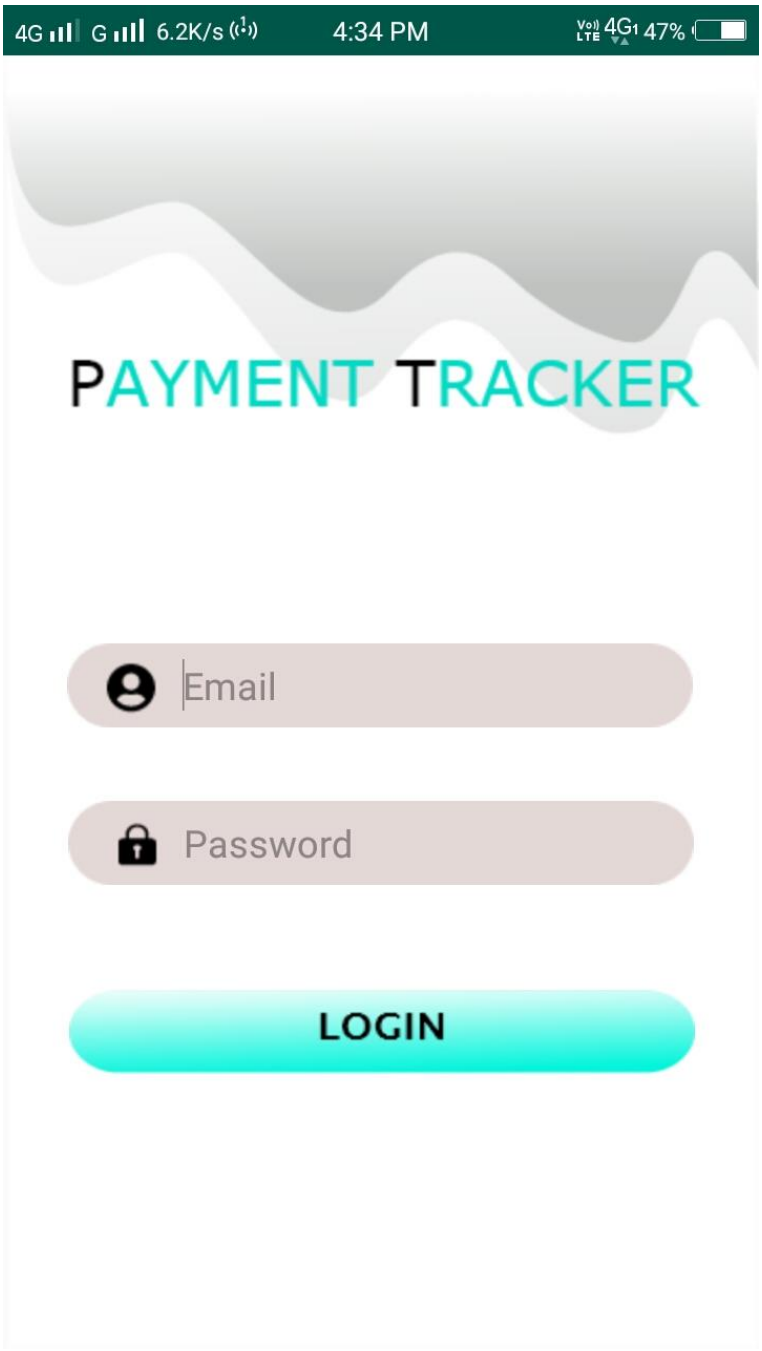
The state diagram above shows how the application goes to various states upon different responses or interactions from the user. Firstly the user opens the application to view the reminders. If there are any notifications he will be notified until he responds to the notification. If there are no notifications he can view the reminders. Then user can either create a new reminder or select a previous reminder and either edit or delete it. If he selects to create a new reminder a form is generated and user should input the details. If the entries are valid it is saved else user is notified about the invalid entry and is again prompted with the form. If the user desires to edit a reminder then again a form will be generated and data will be saved if entries are valid. If the user wants to delete a selected reminder he will be asked for confirmation and it will be deleted if he responds yes. The user will be provided with a feedback confirming the proper execution of the actions he performed (like adding, deleting or updating).

14. Graphical user interface:



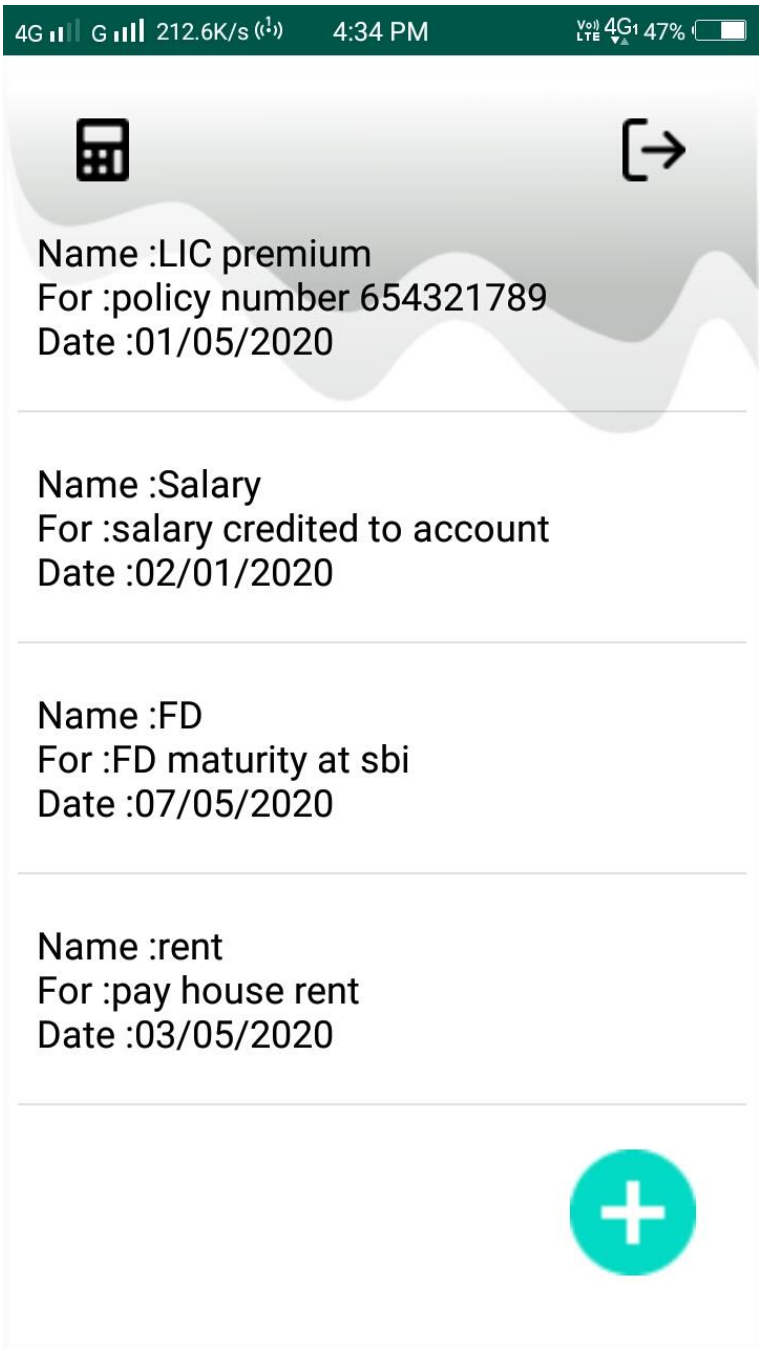
Signup Activity:

It is the opening page of the app. When an unregistered user wants to use the app, it is necessary for him to register by giving his email and password. If the user has an account, he can go to the login Activity.



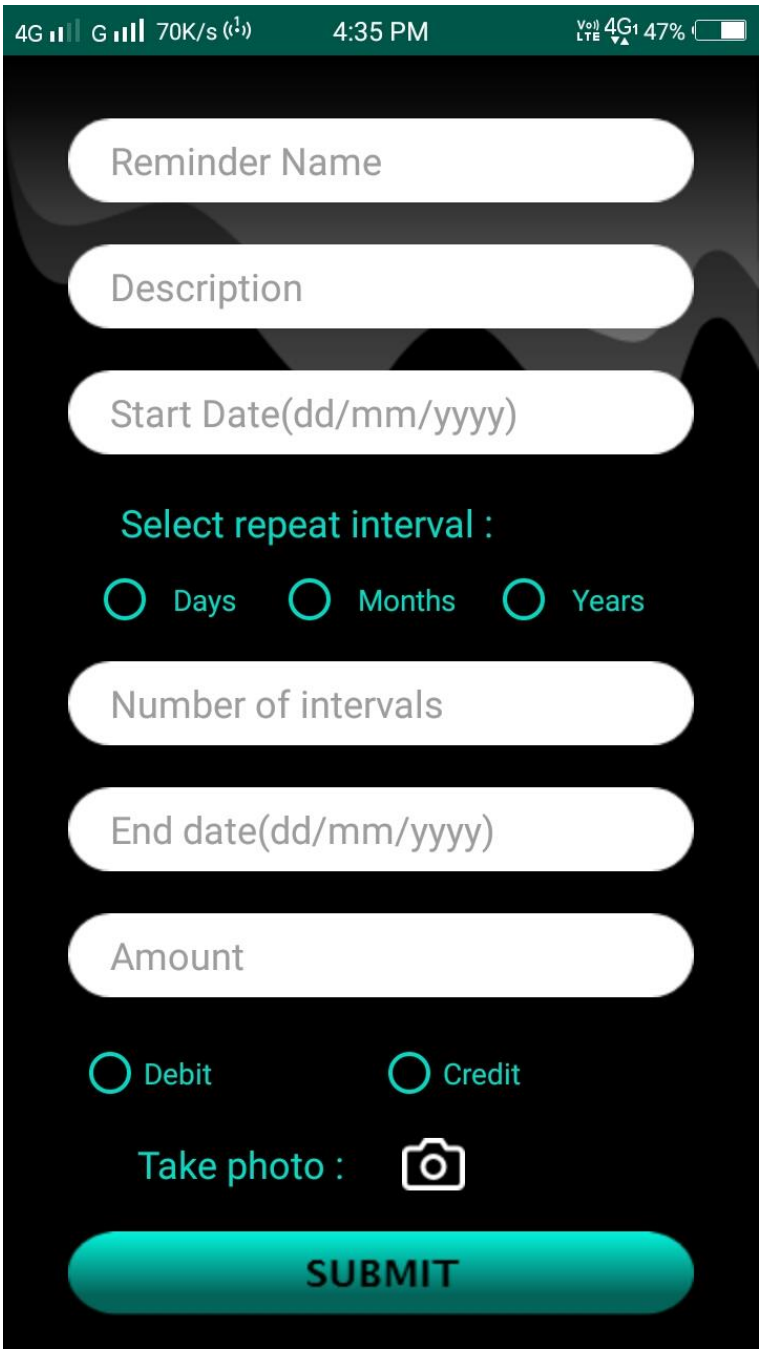
Login Activity:

An already registered user can enter email and password for the login, if the user hasn't logged out then he will be automatically directed to the view reminder Activity.



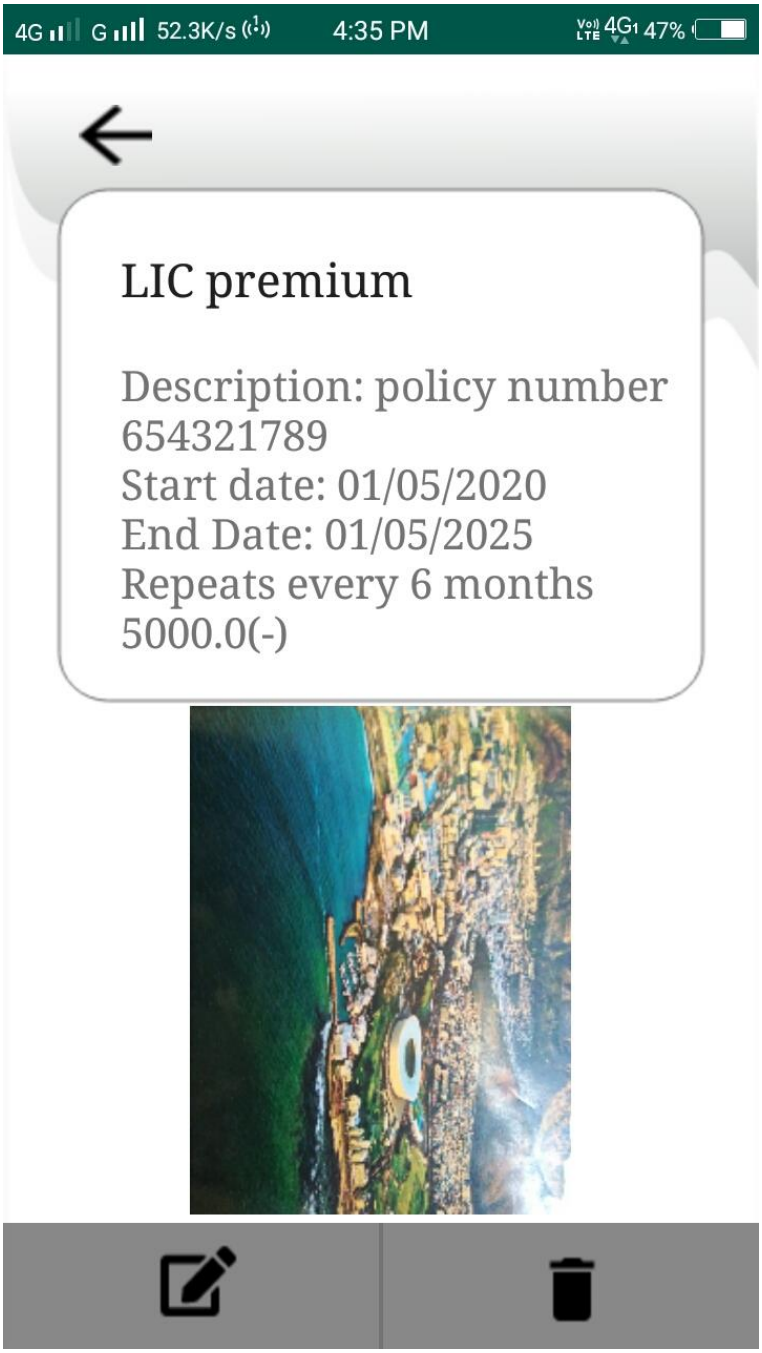
View reminders activity:

This page will have the list of all reminders which are already set by the user. Add new reminder, balance calculator and sign out options are also provided here.



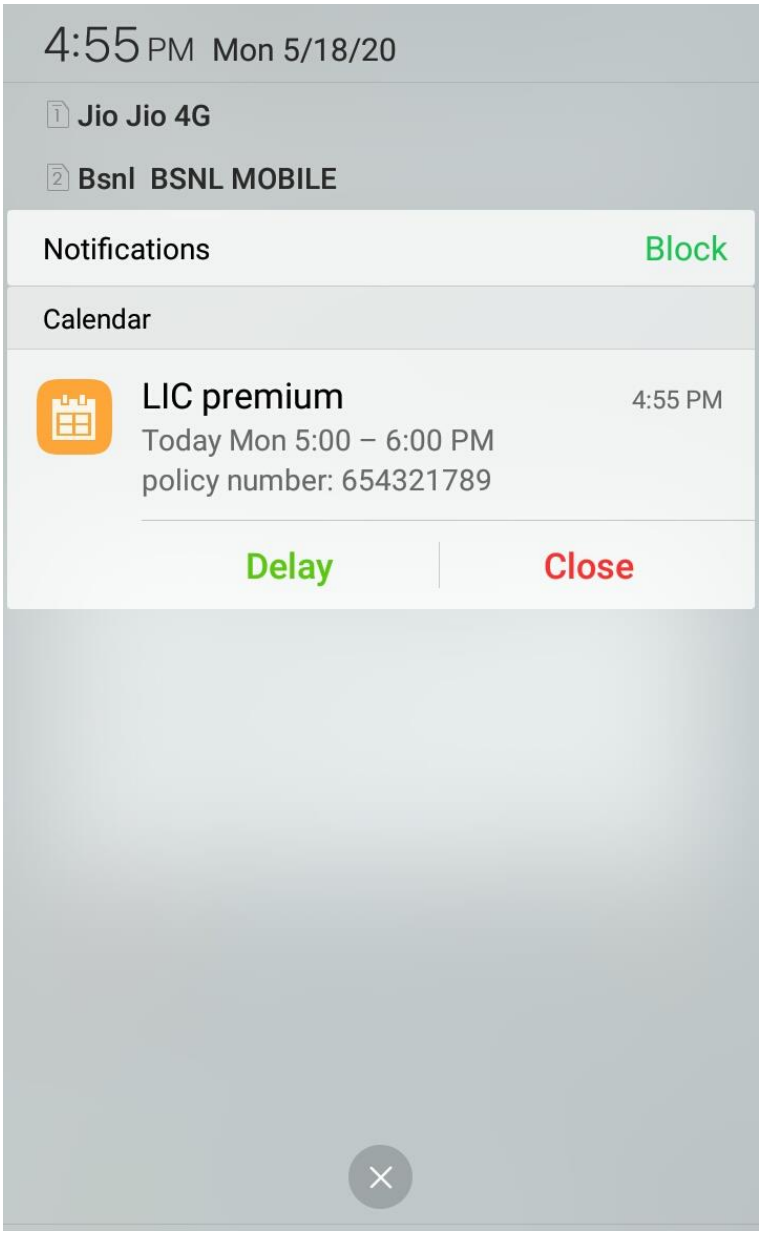
Add reminder activity:

It is an activity which takes name, description, date, interval at which the reminder has to repeat, number of repetitions and amount of money to be paid or received from the user in order to create new reminder. It also provides option to take an image to be stored along with the reminder.



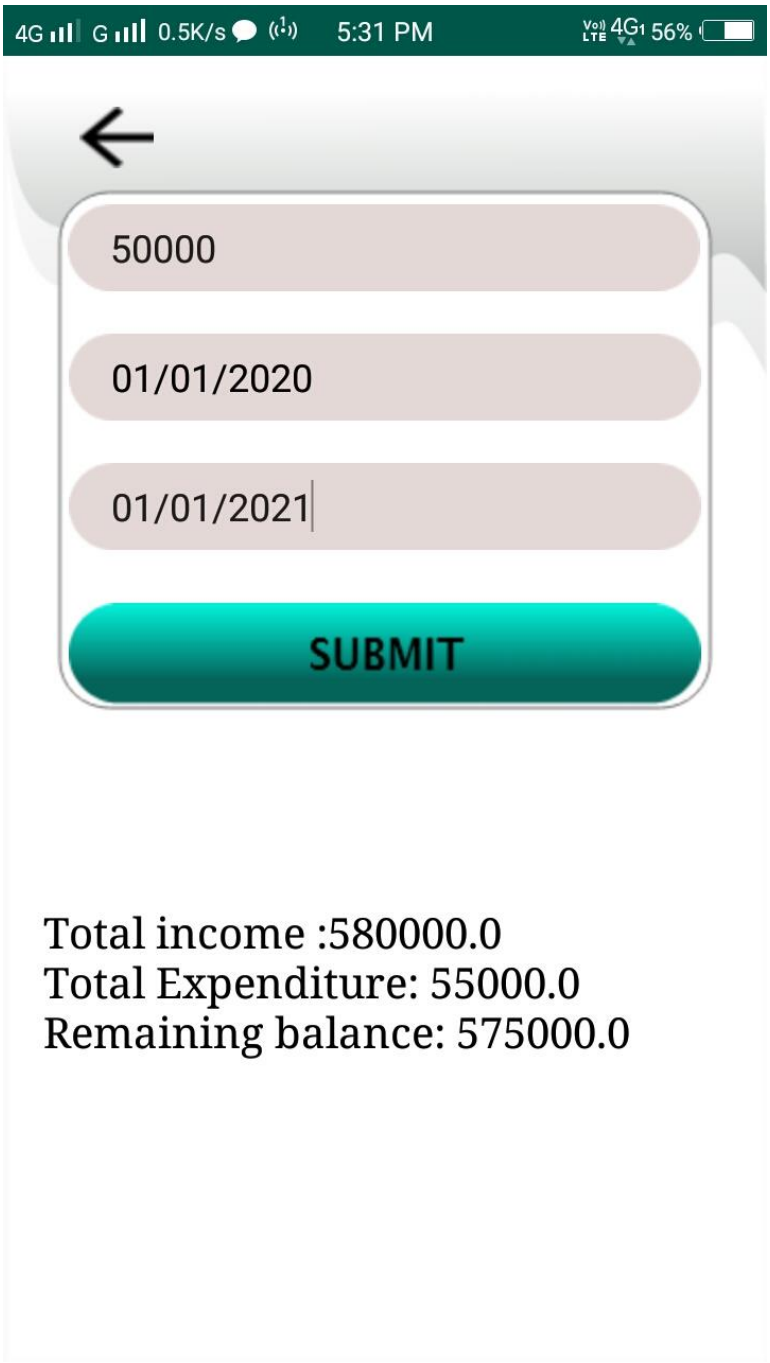
Edit/delete reminder activity:

In this activity, details of the reminder are displayed along with the image when the user clicks on a reminder. The user can either edit or delete the reminder by pressing the edit or delete buttons.



Notification:

The image shows the notification at time specified by the user. It contains name and description of the reminder given by the user at the time of reminder creation.



Calculator:

In this activity the user can provide a current balance and a start and end date for an interval. The total income and expenditure over that interval and the remaining balance of the user is displayed.

15. Test Cases:

Function 1: Registration

Test case 1: If we provide a valid email id and a password and also confirm the password correctly, a new user will be created.

Test case 2: If we do not provide a valid email id without '@' or '.' toast will be shown to enter a valid email id.

Test case 3: If we do not confirm the password correctly toast will be shown to enter same password.

Function 2: Login

Test case 1: If we provide a valid email id of an existing user and the corresponding password correctly, we will be logged in.

Test case 2: If we do not provide a valid email id without '@' or '.' or an email id of a nonexistent user, toast will be shown to enter a valid email id.

Test case 3: If we enter a wrong password, a toast will be shown to enter valid credentials.

Function 3: Create reminder

Test case 1: If we provide all the details in the form while creating the reminder, a repetitive reminder will be created which repeats based on the interval provided by us till the end date.

Test case 2: If we do not provide an end date reminder will repeat forever.

Test case 3: If we do not provide any details regarding the interval and give an end date same as the start date a non-repetitive (one time) reminder will be created on the specified date.

Test case 4: If we give an end date which is smaller than the start date a toast will be shown to provide valid dates.

Test case 5: If we do not take an image while creating the reminder no image will be displayed when we open the reminder to view its details.

Test case 6: If we provide an amount an amount and not specify whether its credit or debit toast will be shown to check any of the two options.

Function 4: Check total income and expense

Test case 1: If we provide all the current balance along with proper start and end date, the total income and expenditure over the interval and the remaining balance will be displayed. If there are no reminders in the given interval total income and expense will be zero and balance will remain unchanged.

Test case 2: If we give an end date which is smaller than the start date a toast will be shown to provide valid dates.

Function 5: Edit reminder

This has test cases same as creating a reminder.

Function 6: Delete reminder

Reminder will be deleted if we press the delete button shown in the reminder details page.

16. Conclusion:

As we all know money is an essential commodity to run our life. Investments and deposits are keys to save money for future. Keeping track of each and every financial activity is very important because the money people invest is the result of their efforts throughout their life. The app helps the people who have to manage a lot of periodic payments from various different sources. Using the app users can set periodic reminders, can store images related to reminders. They can calculate the gross expenditure or income of their monetary activities. Thus users who have to keep track of a lot of investments and commitments in different financial institutions can use the app to ease the process. The users can see the remaining non-invested balance after a given time range thus allowing them to decide on whether to cut down their expenses or to invest in other schemes.

17. Future scope:

- The search bar is a very much recommended improvement for the app. Case insensitive and quick search bar based on binary search algorithm would do the job.
- Each reminder can be provided with a tag which will be provided by the user at time of creation of the reminder. Filters based on these tags can be implemented to categorize the reminders.
- Various investment platforms like banks, insurance companies etc can be provided with the facility of advertising their policies or investment plans. This would further help users to decide future monetary plans.
- The app can be made to intelligently suggest the investment plans to users, based on their income, previous financial activities and profits that the investment plans yield. An ML based model can be trained for the purpose.