



**Symbiosis Institute of Technology**

EXPENDITURE

TRACKER SYSTEM

PROJECT

REPORT

Submitted by

**Arundarasi Rajendran (18070122081)**

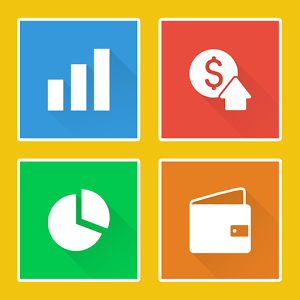
Department of Computer Science

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

Index

1. Introduction
   1. Project summary
   2. Background
   3. Problem statement
   4. Purpose of project
   5. Project scope
   6. Limitations
   7. Motivation
   8. System Purpose
      1. Users
      2. Location
      3. Responsibilities
   9. Overview of document
2. Development of system
   1. Methodology used
   2. Gantt Chart
   3. Software platforms
   4. Hardware platforms
3. Functional Requirements
   1. High priority
   2. Medium priority
   3. Low priority
4. Non-Functional Requirements
5. Data Flow Diagrams
   1. The Context model
      1. Context diagram
      2. System externals
   2. Data Flow Diagram Level 1
   3. Data Flow Diagram Level 2
6. The Use Case Model
   1. System Use Case diagram
   2. Use Case descriptions
7. Entity Relationship Diagram
8. Sequence Diagram
9. The Class Model
10. Code snippets
11. Output and Test Cases
    1. Test cases table
    2. Output
       1. Positive test cases
       2. Negative test cases
12. Validations
13. **Introduction**

1.1 Project Summary



ExpensePal logo

Project Name: Expenditure Tracker System

Mobile App name: ExpensePal

Project Manager: Arundarasi Rajendran

Responsible Users: Admin

User

1.2 Background

Studying or working abroad is an opportunity that many aspire. It gives them a chance to gain experience in the outside world. At the same time, individuals get to immerse themselves into a new culture, and broaden their perspective about living, working and studying abroad. Everything about traveling abroad and pursuing an education or getting a job is tempting to anyone except for one key factor which probably plays the most important role. Traveling, studying and living abroad, irrespective of the country is a costly affair. There are many reasons which cause this hike in expenditure. May it be tuition, current currency exchange or the country you live in, the amount of money that a student or an adult as an employee will have to shell out in order to live abroad will always be high. However, there are many options for students like scholarships when it comes to tuition, all of their personal expenditure must be managed by them.

Individuals moving out of the confines of their homes to the world have a natural tendency to explore all avenues of life, which directly affects the money in their bank accounts. Though some are able to get a hold on themselves and show some restraint, others need help in order to manage their money and finances. In today’s time, as technology has given us so many gifts, one that has been catching a lot of attention is management apps, especially budget management apps. These software applications can be installed on your handheld devices and also synced with a laptop or desktop.

Tracking expenditure on a personal level is always a problematic issue faced on a daily basis by almost each of us. Recording and managing expenses, income/savings are getting very difficult in today’s busy life which ultimately results in the inefficient planning of personal budgets. This proposed application will allow users to track and analyze their expenses, income, and savings to plan their personal budget, organize their expenses and income and record movement of money.

1.3 Problem Statement

This application provides a convenient way to record and track a person’s daily expenditures into a mobile app. Through this application the user can make quick calculations at time of spending money and can verify values in a certain currency into a different currency. It allows them to add reminders for their due payments. In addition, in this application users can record details about their desire to buy certain products or services. Furthermore, this application would help the user to adjust their budget for the expenses they have to make in the future.

1.4 Purpose of Project

The purpose of this project is to create a user interface and portable mobile app that allows a user to record their daily expenditures and adjust their budgets accordingly. It becomes difficult for people to track their expenditures and adjust their budgets for the upcoming month. Therefore, this mobile app will record users daily expenditures in different categories. This application will provide a visual representation of expenditures in different categories. It will allow the user to make quick calculations and compare values in different currencies in order to spend wisely. Also, it provides space to record the products that the user would like to buy in the future. This system aims to manage users daily expenses in a more efficient and manageable way. This application enables the user to not just keep the control on the expenses but also to generate and save reports. It is convenient for college students or foreign employees living in another country to record their expenses, remind them of their payments, convert currency values and make quick calculations.

1.5 Project scope

The scope of this project is to provide a portable mobile app to track and manage their expenses for all users for example, foreign students, and foreign employees. This mobile app will provide the following options. The mobile app will allow users to make reminders for payment dues, make quick calculations using the calculator, convert values in different currencies to compare cost, and create a wishlist. This application can be utilized by any individual who has a difficulty in viewing and recording their daily expenditures. The app can be used by anyone who is living or working abroad from their home country as this application specifically focuses on these customers.

1.6 Limitations

This mobile app does not have a direct connection to your bank account as other expense manager apps do. This app does not consider income and cannot deduct the amount of expenses made from your income. Also, this app does not have the option to take pictures of expenditure bills because there are some apps that have this feature. There are many mobile apps in the app store that record and track your expenditures. They are more efficient and convenient for users as they have more favourable features.

1.7 Motivation

The reason behind building this project is that I realized there should be a mobile app for foreign workers or students who go and study, live or work abroad for managing their expenses in a new place. Along with tracking their expenses, I thought the app should have some additional features such as, a calculator, a payment reminder, a wishlist, and a currency converter. These additional features will help them adjust their budget for the month. Many foreigners struggle in adjusting their budget for each month. Therefore, I decided to build a mobile app that could help them record their expenses meanwhile, make wise choices about their budget for the next month by wisely spending their money. My inspiration to build this project basically came from my sister who is studying abroad and who struggled to adjust her budget at the beginning.

1.8 System Purpose

*1.8.1 Users*

The entities who will benefit from this expenditure tracker system and those who will be affected by the new system include:

1.Users

Any user would have an easier way to record their expenditures and track them. Users will be able to compare costs in different currencies to check how much expenses are made according to their native currency. The users will be able to make payment reminders.

2. Admin

They have the responsibility to update login credentials of users. They also have the authority to delete the account if a user wants to delete their account. The system is not beneficial to admins as they are the ones in control of login credentials.

*1.8.2 Location*

The mobile app will be available to anyone who downloads it from google play. It can be only used in online mode for login but once logged in the mobile app can be used in offline mode. The mobile app can be used at any location.

*1.8.3 Responsibilities*

The primary responsibilities of this system:

* allow user to register into the mobile app
* allow user to login
* allow user to reset their password
* permit user to view home page after login
* allow user to add or edit category
* allow user to add or delete their expense entry
* give list of expenses made
* allow admin to delete an account
* provide user with budget reports through pie chart and bar chart
* allow user to list their desired items to buy

The desired features of this system:

* password protection scheme for all users
* reminder for payment dues
* convert values to different currencies using currency converter
* a calculator for quick calculations

1.9 Overview of document

The rest of this document gives the detailed specifications for the expenditure tracker system.

* Section 2: Functional Objectives

Every one of the objectives give a specific behaviour of the system, and certain operations that meet the requirements. These objectives are organized by priority. In order for the system to be successful to users all the high priority objectives must be met.

* Section 3: Non-Functional Objectives

This section is organized by category. Each objective specifies a technical requirement or constraint on the overall characteristics of the system. In this section, each objective is measurable.

* Section 4: Context Model

This section gives a text description of the goal of the system and a pictorial description of the scope of the system in a context diagram. The entities visible outside the system are external and interact with the system, which is described in this section.

* Section 5: Data Flow Diagram Level 1 and 2

This section gives a detailed diagram of the data flow between the processes of the system, external entities and data storage.

* Section 6: Use Case Model

The specific behavioural requirements of the system are detailed in a series of use cases. Each use case takes the user through the expenditure tracker system along with additional features and shows interaction between the system and some outside actor. Each use case is described with both text and an interaction diagram. An interface prototype is also shown. The system use case diagram depicts the interactions between all use cases and system actors.

* Section 7: Entity Relationship Diagram

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of entity relationship model are: entity set and relationship set.

* Section 8: Sequence diagram

The sequence diagram shows interactions between objects and actors arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the different scenarios.

* Section 9: Class Model

A class is a collection of objects in the system that have the same data and behaviour. All analysis classes and their relationships are shown on the class diagram.

* Section 10: Results

This section shows the pages of the expensePal app that are functioning once logged in.

* Section 11: Code snippets

The coding behind the main functions of the pages specifically, their operations are shown in this section.

* Section 12: Test Cases

This section shows the testing of each operation of each page in the expensePal app. The testing is divided into two sections: positive test cases and negative test cases. The positive test cases are the cases that have passed and are working in the app. Meanwhile, the negative test cases are the cases that have failed and are not properly working in the app.

* Section 13: Validation

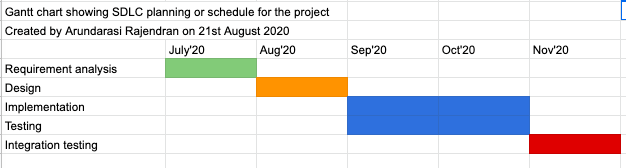
This section validates the functional and non-functional requirements of the project by explaining if it is working or not and is it visible or present or not.

1. **Development of System**

2.1 Methodology used

To make this project, I referred to online video tutorials on android studio coding at the beginning to understand how to make an app. I referred to some of the tutorials on how to make payment reminders, calculator, currency converter, wishlist and more. I used some online source codes to make the components in this project such as adding or deleting expenses, displaying a pie chart and line chart, and more. For the charts, I imported them from online. For the currency converter, I had to use API’s from online. The project is made in the android studio application using Java language and it is linked with Google Firebase. Using google firebase, I was able to make the user register, login, and reset their password.

2.2 Gantt Chart

This is the timeline of my project.

2.3 Software platforms

The software applications used in this project are the following

* Android Studio v4 (Java)
* Android Emulator

2.3 Hardware platforms

The hardware application used in this project are

* MacBook Air laptop

2.4 Online tools used

* LucidChart
* Flowchart maker and online diagram software
* Google Firebase

1. **Functional Requirements**

3.1 High Priority

1. The application should have a register and login page.
2. The application should provide password-based secure access to the user.
3. The user should be allowed to add or delete or update different categories to group their expenses.
4. The user should be allowed to add or delete or update their expenses in different categories such as food, rent, clothing, medical and etc.
5. Design a dashboard or home screen providing an easy and simple view of every expenditure in one place.
6. The application will provide budget reports such as pie charts and distributions
7. The admin should be able to delete a users account.

3.2 Medium Priority

1. The user should have a calculator in handy so they can make quick calculations on mobile.
2. It should provide the facility to set alerts on payment dues e.g. electricity bills will be paid by the end of the month.
3. The application should have a currency converter if the user would like to see the cost of an item to their domestic currency.

3.3 Low Priority

1. The welcome message should appear when the user opens the mobile app should be taken to the login page.
2. The application can have a wishlist where users can enter the cost and name of the item that they would like to buy in future.
3. The user can change the background color.
4. **Non-Functional Requirements**

1.Security

* The system shall provide password protected access to the app.

2.Usability

* Any user should be able to use the mobile app on their phone.
* Users should be able to add, delete, and update expense entries and categories plus, use payment reminder, calculator, currency converter, and wishlist.

3.Portability

* The application should be installed on mobile.

4.Performance and scalability

* Users would receive payment reminders on time within a few seconds.
* Expenditure results will be displayed in second.

5.Reliability

* The system will be completely operable anytime once it is opened.

6.Availability

* The application should be available to all users anytime and to anyone.

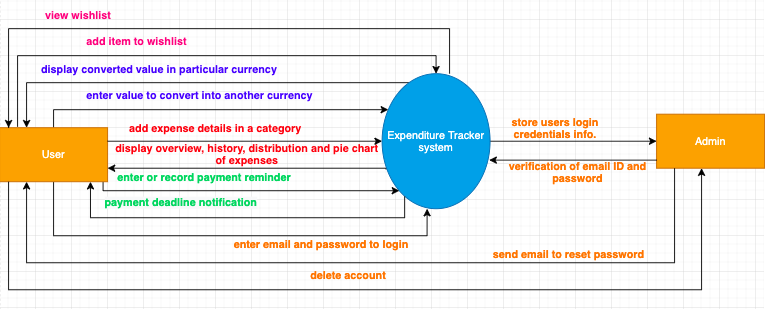
7.Supportability

* The system should be viewable on android.

1. **Data Flow Diagrams**

5.1 The Context Model

*5.1.1 Context Diagram*

**

*5.1.2 System Externals*

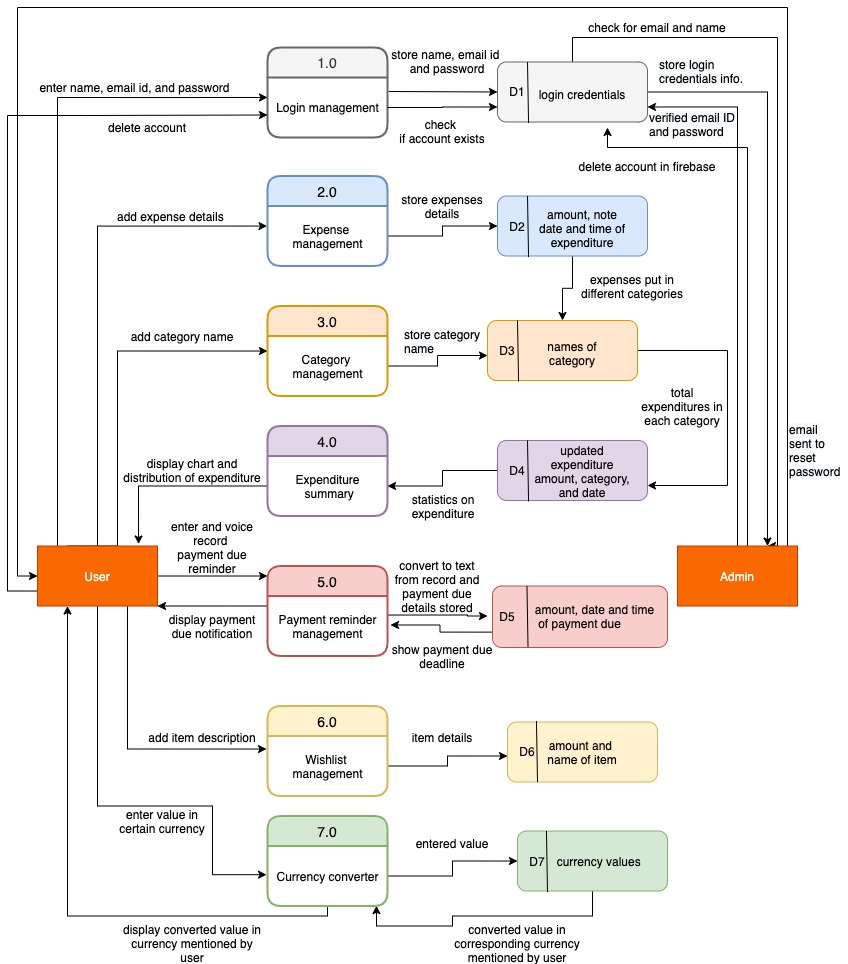
Users

A user will use this system if they are living, studying or working abroad and they require a platform to record their expenses and spend their money wisely. These users can be foreign students studying in another country or foreign employees working for a foreign company and are new to the country and just starting to settle.

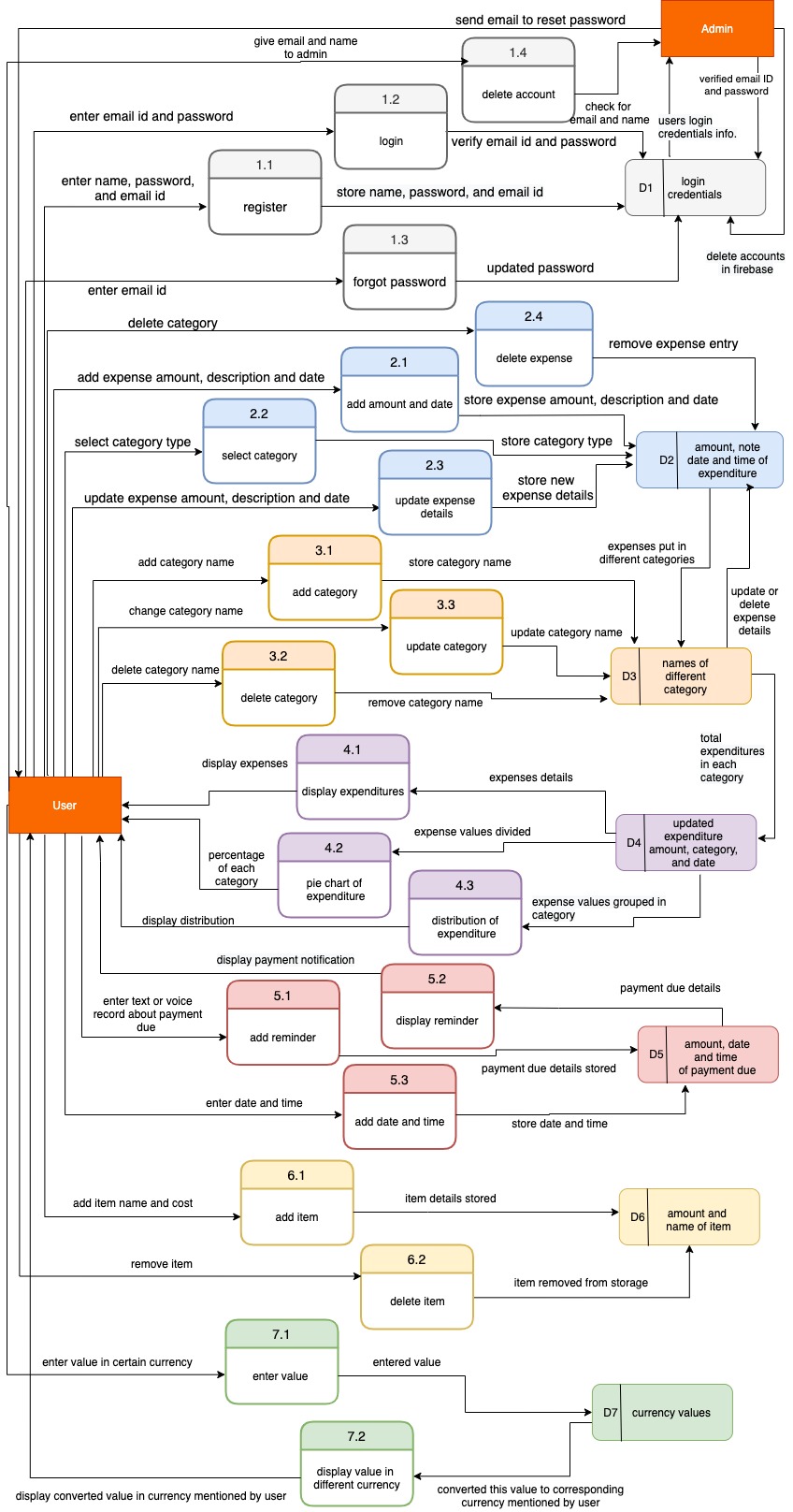
Admin

The admin has the responsibility of controlling the user accounts registered into firebase. They have the authority to delete the accounts and view details of the user.

5.2 Data Flow Diagram Level 1

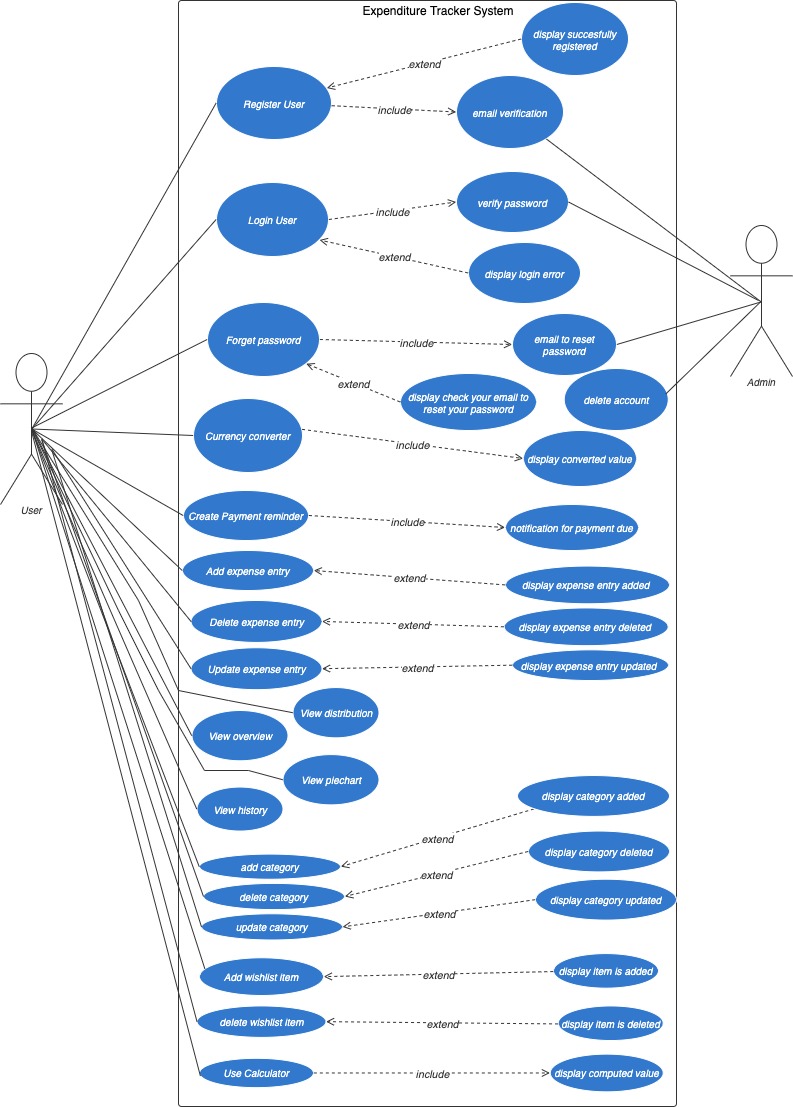


5.3 Data Flow Diagram Level 2



1. **The Use Case Model**

6.1 System Use Case diagram



6.2 Use Case descriptions

**Register User**

| Use Case Name | Register User |
| --- | --- |
| Summary | A user must register themself before logging in to create an account to track their expenditures. |
| Basic Flow | 1.The use case starts when a user indicates that they want to register.  2.The system requests name, email ID and password.  3.The user enters their name, email ID and password.  4.The system checks if the entered email ID is similar to any existing registered email IDs.  5.The system stores all user information at firebase (backend).  6.The user starts a login session. |
| Alternative Flows | Step 3:  If the name is not filled then the system displays “full name is required” then it goes to step 2.  Step 3:  If the email ID is not filled then the system displays “email is required” then it goes to step 2.  Step 3:  If password is not filled then the system displays “password is required” then it goes to step 2.  Step 3:  If the password length exceeds 6 characters, the system requests that the user re-enter the password by displaying the message “minimum password length should be 6 characters”.  Step 4:  If the email ID is similar to another email ID then the system displays failed to register and it goes to step 2.  Step 4:  If the email ID is not duplicate of another email ID then the system displays “you have been registered successfully”. |
| Extension Points | none |
| Preconditions | none |
| Postconditions | The user can now record expenses, add payment due reminders, and perform other functions. |

**Login User**

| Use Case Name | Login User |
| --- | --- |
| Summary | A user must login to get restricted information on their expenditure, wishlist and payment dues. |
| Basic Flow | 1.The use case starts when a user indicates that he wants to login.  2.The system requests email ID and password.  3.The user enters his or her email ID and password.  4.The system verifies the email ID and password against all registered users.  5.The system starts a login session and displays the home page. |
| Alternative Flows | Step 4:  If the email ID is invalid then the system displays a “login error” message and the case goes back to step 2.  Step 4:  If the password is invalid, the system displays a “login error” message. Then the system requests that the user re-enter the password. When the user enters another password the use case continues with step 4 using the original email id and password. |
| Extension Points | Register, Forgot password |
| Preconditions | The user is registered. |
| Postconditions | The user can now record expenses, add payment due reminders, and perform other functions. |

**Forgot password**

| Use Case Name | Forgot password |
| --- | --- |
| Summary | This use case allows the user to reset their password. |
| Basic Flow | 1.The use case starts when a user indicates that he wants to reset their password.  2. The system requests email ID.  3. The user enters their email ID.  4. The system displays a message that please check your email to reset password  5. The firebase system sends an email to the user’s email ID to reset password.  6. The use case opens the email and clicks on the link given in it to reset their password. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is registered. |
| Postconditions | The user can login with a new password and can record expenses, add payment due reminders, and perform other functions. |

**Delete account**

| Use Case Name | Delete account |
| --- | --- |
| Summary | This use case allows the admin to delete an account. |
| Basic Flow | 1.The use case starts when the admin is notified that a user's account needs to be deleted  2. The admin checks the name and email of the user given to them.  4. The admin deletes the users account google firebase console. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user must register. |
| Postconditions | The users account is deleted and can no longer be used. |

**Add category**

| Use Case Name | Add category |
| --- | --- |
| Summary | This use case allows the user to add a category to group their expenses. |
| Basic Flow | 1.The use case starts when a user indicates that he wants to add a category.  2. The system requests the name of the category.  3. The user enters the name of the category.  4. The system stores the category name in the database and displays a “successfully added” message. |
| Alternative Flows | Step 3:  If the user does not add the name of the category then the system displays a “please enter category name” message. |
| Extension Points | none |
| Preconditions | The user is logged in. |
| Postconditions | The user can add expenses made in a category. |

**Update category**

| Use Case Name | Update category |
| --- | --- |
| Summary | This use case allows the user to update a category. |
| Basic Flow | 1.The use case starts when a user indicates that he wants to update a category.  2.The system requests the new name of the category.  3. The user enters the new name of the category.  4. The system stores the new category name in the database and displays a “category name edited to new name” message. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in and category entries have been made. |
| Postconditions | The updated category name is displayed in category list. |

**Delete category**

| Use Case Name | Delete category |
| --- | --- |
| Summary | This use case allows the user to delete a category. |
| Basic Flow | 1.The use case starts when a user indicates that he wants to delete a category.  2.The system displays a message to transfer items present in that category to another category.  3. The system deletes that category name in the database and displays a “category is deleted” message. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in and category entries have been made. |
| Postconditions | The deleted category name is not present in the category list. |

**Add expense entry**

| Use Case Name | Add expense |
| --- | --- |
| Summary | This use case allows the user to add an expense entry. |
| Basic Flow | 1.The use case starts when a user indicates that he wants to add an expense.  2. The system requests the category type, expense date, expense amount, and some description about the expense.  3.The user enters the expense date, expense amount, some description about the expense and chooses the category type.  4.The system stores these details in the database and displays a “new expense added” message.  5.The system resets the page by emptying the input bars. |
| Alternative Flows | none |
| Extension Points | Add category, Piechart , History, Distribution |
| Preconditions | The user has added category name for certain expenses. |
| Postconditions | The user can add more expenses made in a category. |

**Update expense entry**

| Use Case Name | Update expense entry |
| --- | --- |
| Summary | This use case allows the user to update the expenses that were already added. |
| Basic Flow | 1.The use case starts when a user indicates that he wants to update an expense entry.  2.The system requests the new expense date, new expense amount, and some new description about the expense.  3.The user enters the new expense date, new expense amount, and some new description about the expense.  4.The user chooses to change category and picks a different category.  5. The system stores the update date, amount, category, and description in the database and displays a “expense updated” message. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in and expense entries should be added. |
| Postconditions | The user can add more expenses made in an updated category. |

**Delete expense entry**

| Use Case Name | Delete expense entry |
| --- | --- |
| Summary | This use case allows the user to delete the expenses that were already added. |
| Basic Flow | 1.The use case starts when a user indicates that he wants to delete an expense entry.  2.The system asks if to transfer this expense to another category or delete it.  3.The user deletes the entry.  4.The system deletes that expense entry and updates it in the database. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in and expense entries should be added. |
| Postconditions | The user can add more expenses made in an updated category. |

**Create payment reminder**

| Use Case Name | Create payment reminder |
| --- | --- |
| Summary | This use case allows the user to add the payment due reminders. |
| Basic Flow | 1.The use case starts when a user indicates that they want to add payment due reminder.  2. The system requests payment amount, payment details, date, and time.  3. The user voice records the payment details and picks the date and time.  4.The system displays these details in a list. |
| Alternative Flows | Step 2:  If text is not filled or date is not picked then the system displays a “please enter or record the text” message. |
| Extension Points | none |
| Preconditions | The user is logged in. |
| Postconditions | The user can now add more payment reminders and get notification messages of added reminders on time. |

**Add wishlist item**

| Use Case Name | Add wishlist item |
| --- | --- |
| Summary | This use case allows the user to add any item that they want to buy in the future in this wishlist section. |
| Basic Flow | 1.The use case starts when a user indicates that they want to add a wishlist item.  2.The system requests the item cost and item name.  3.The user enters the item name and item cost**.**  4.The system stores these details in a list. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in. |
| Postconditions | The user can now add more items in the list. |

**Delete wish list item**

| Use Case Name | Delete wishlist item |
| --- | --- |
| Summary | This use case allows the user to delete any item. |
| Basic Flow | 1.The use case starts when a user indicates that they want to delete a wish-list item.  2.The user deletes the item.  3.The system deletes the item from the list and displays a “successfully deleted item” message. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in. The user must have added an item. |
| Postconditions | The deleted item is not present in the list. |

**Use Calculator**

| Use Case Name | Use Calculator |
| --- | --- |
| Summary | This use case allows the user to perform arithmetic operations. |
| Basic Flow | 1.The use case starts when a user indicates that they want to use a calculator.  2.The system requests input values and operators.  3.The user enters input values and operators.  4.The system computes the output of the arithmetic calculation. 5.The system displays the output. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in. |
| Postconditions | The user can perform any arithmetical operations. |

**View Overview**

| Use Case Name | View Overview |
| --- | --- |
| Summary | This use case allows the user to view the total amount of expenses and the list of expenses. |
| Basic Flow | 1.The use case starts when a user indicates that they want to see the overview of expenses.  2.The system displays the total amount of expenses and the list of expense entries. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in. The user has added entries of expenses and categories. |
| Postconditions | The user can view the total amount of expenses and the list of expense entries. |

**View History**

| Use Case Name | View History |
| --- | --- |
| Summary | This use case allows the user to view the list of expense entries with date, amount, and name. |
| Basic Flow | 1.The use case starts when a user indicates that they want to see the history of expenses.  2.The system displays the list of expense entries with date, amount, and name. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in. The user has added entries of expenses and categories. |
| Postconditions | The user can view the list of expense entries. |

**View Pie chart**

| Use Case Name | View Pie Chart |
| --- | --- |
| Summary | This use case allows the user to view the spread of expenses made in different categories to get an insight on their expenditures. |
| Basic Flow | 1.The use case starts when a user indicates that they want to see the pie chart of expenses.  2.The system displays a pie chart and a list of categories with their percentage. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in. The user has added entries of expenses and categories. |
| Postconditions | The user can view the pie chart and list of categories with their percentage. |

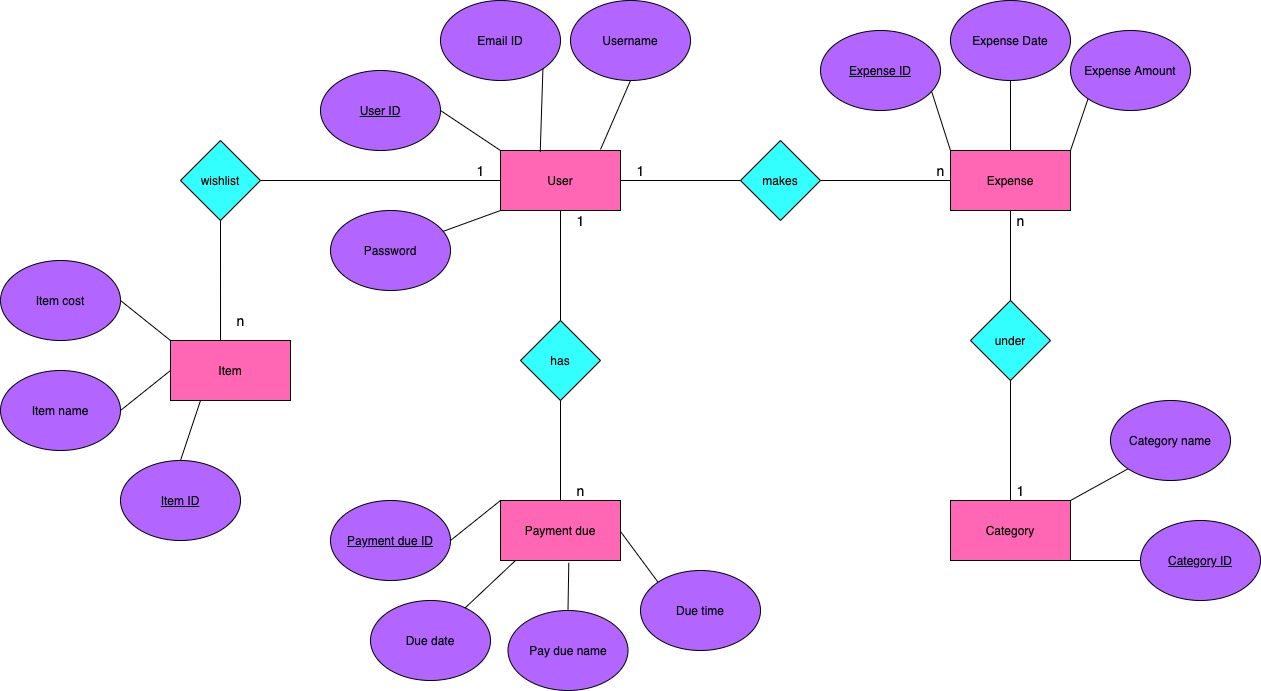
**View Distribution**

| Use Case Name | View Distribution |
| --- | --- |
| Summary | This use case allows the user to view the distribution of expenses in a category to get an insight on their expenditures. |
| Basic Flow | 1.The use case starts when a user indicates that they want to see the bar chart of expenses.  2.The system displays a bar chart |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in. The user has added entries of expenses and categories. |
| Postconditions | The user can view the bar chart. |

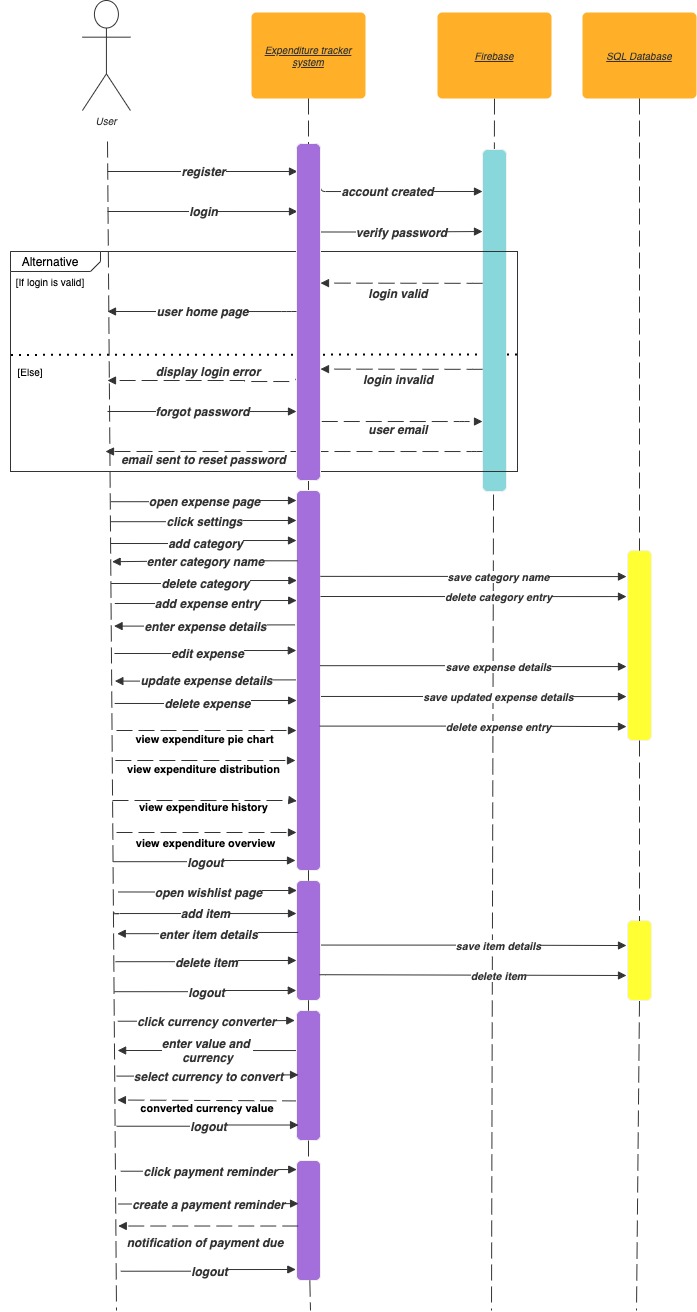
**Currency converter**

| Use Case Name | Currency converter |
| --- | --- |
| Summary | This use case allows the user to enter a value in a certain currency and convert it into another currency. |
| Basic Flow | 1.The use case starts when a user indicates that they want to use the currency converter.  2.The system requests a value to convert.  3.The user enters a value to convert.  4.The user chooses the currency of that value.  5.The system requests in which currency to convert to.  6.Then the user chooses the currency they want to convert the value into.  7.The system computes the converted value and displays it. |
| Alternative Flows | none |
| Extension Points | none |
| Preconditions | The user is logged in. |
| Postconditions | The user can convert values to different currencies. |

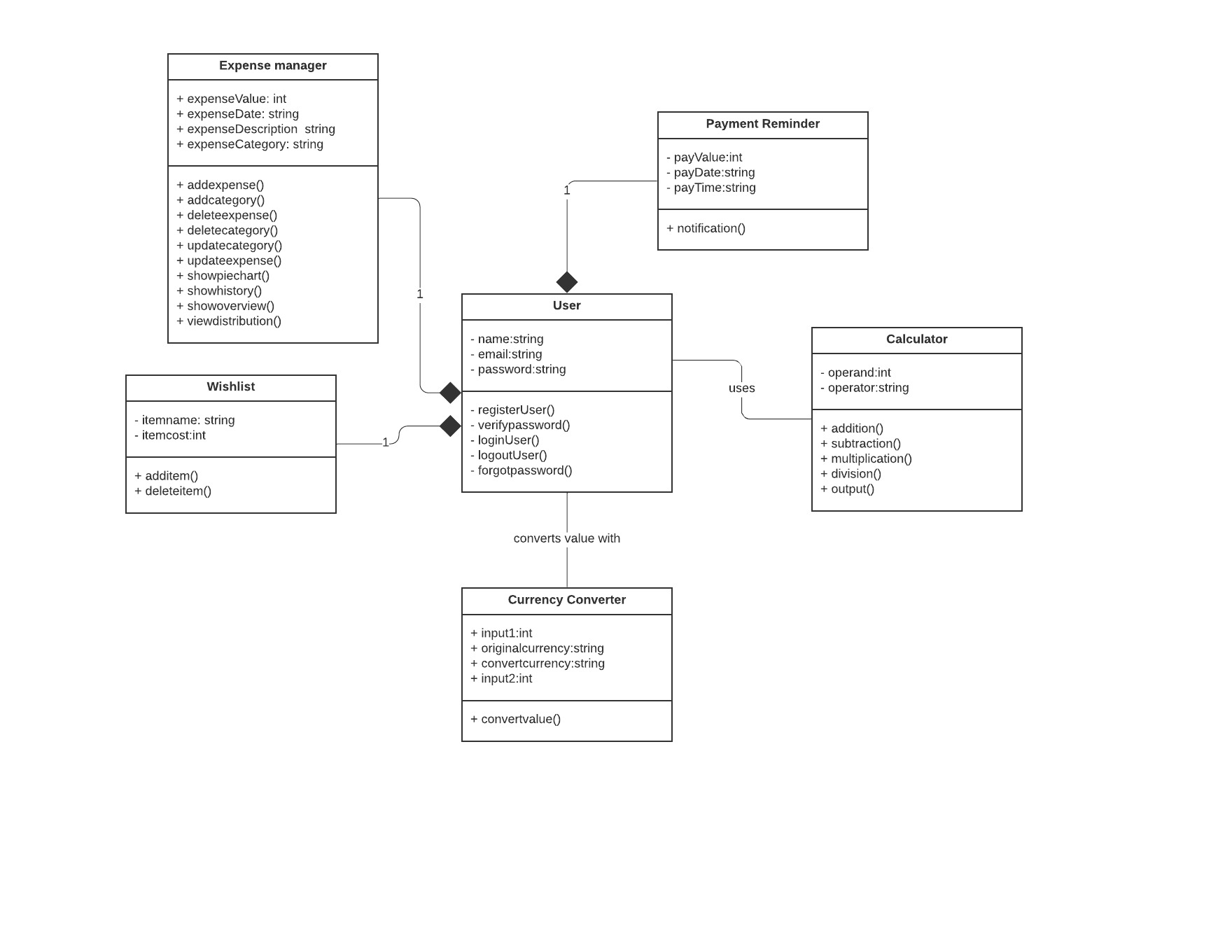
1. **Entity Relationship Diagram**

****

1. **Sequence Diagram**

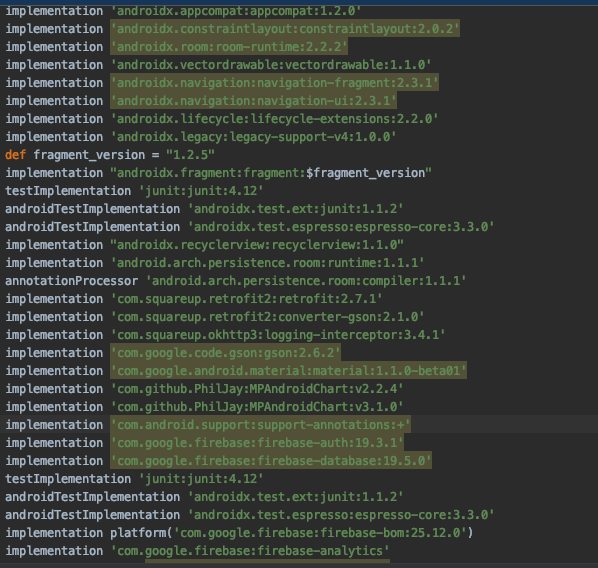
****

1. **The Class Model**

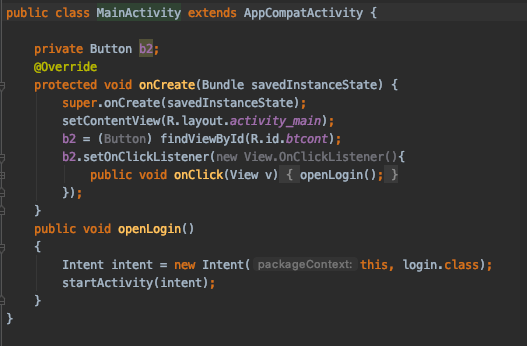
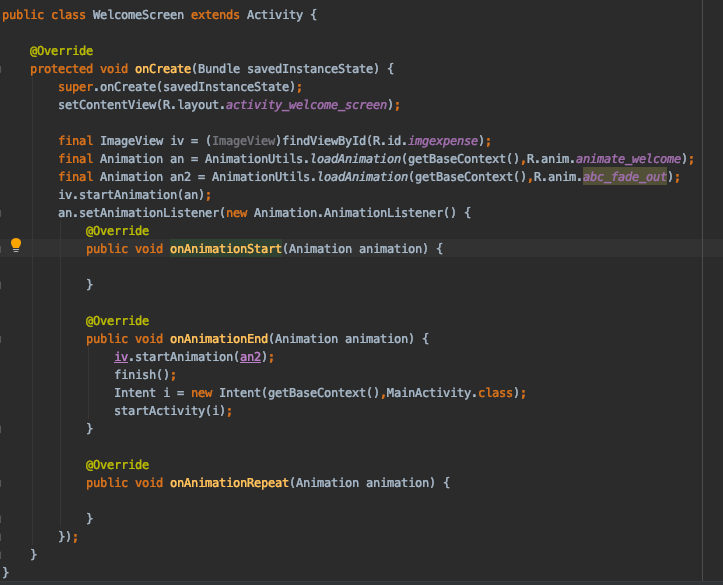
****

1. **Code snippets**

Build Gradle - imports



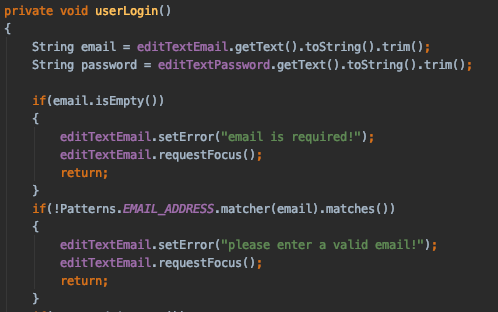
Welcome Screen

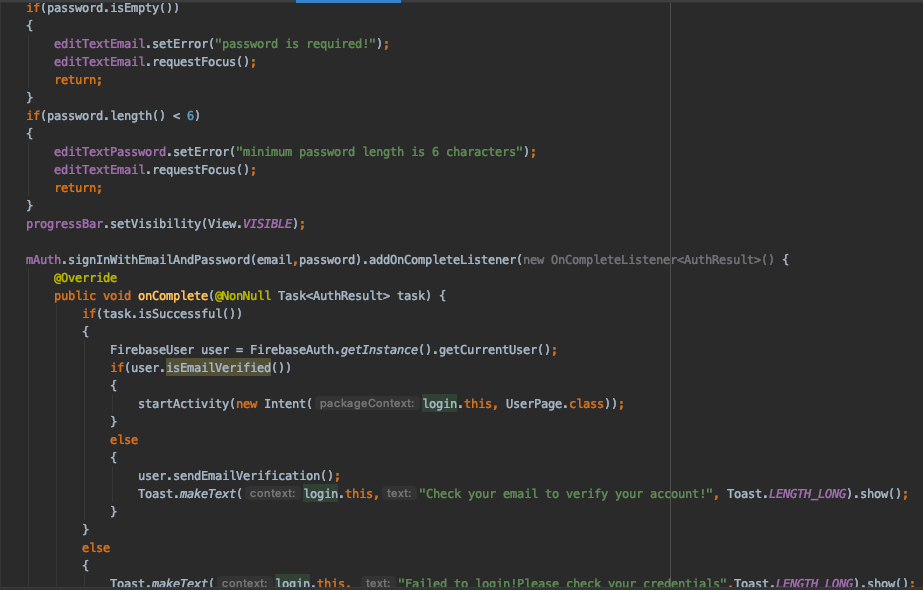


User Page



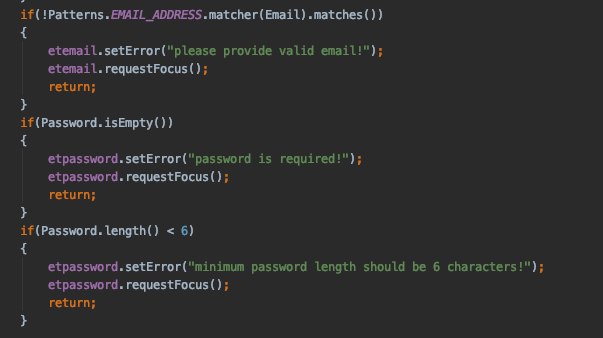
Login



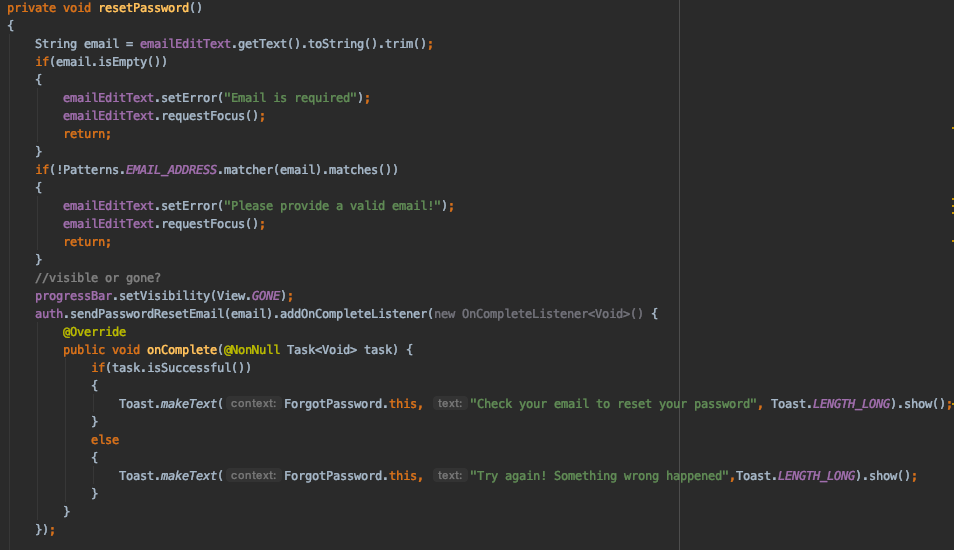


Register

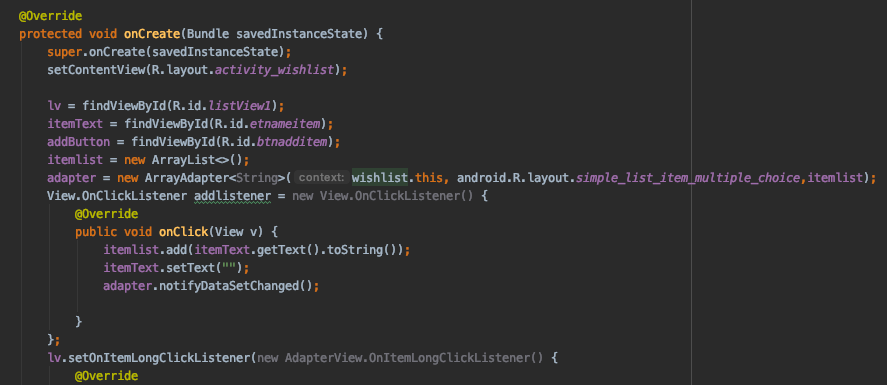




Reset password



Wishlist

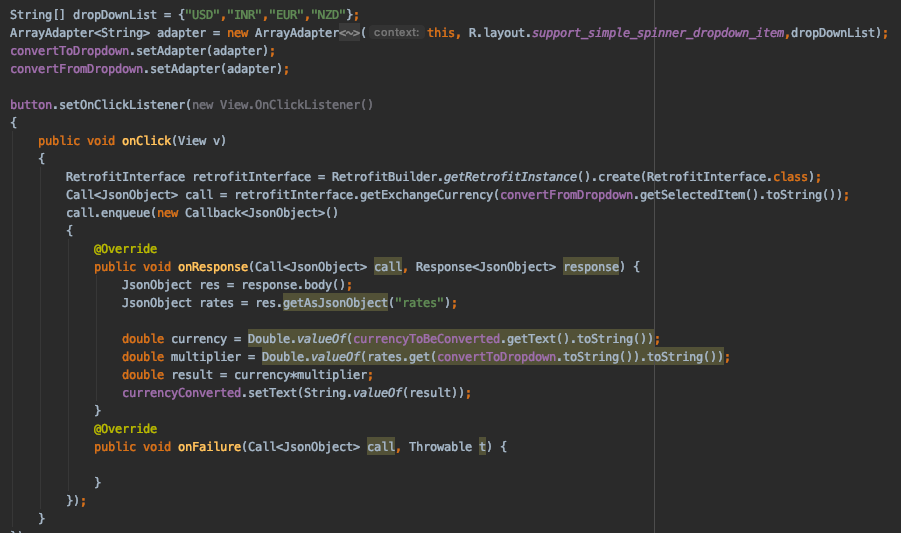




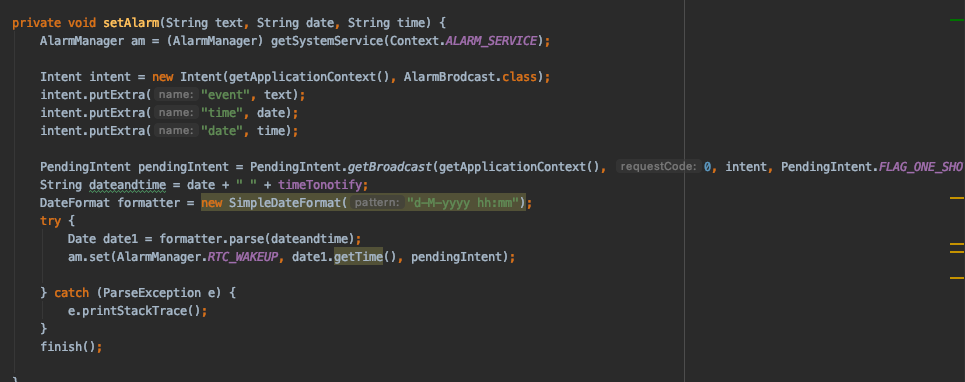
Calculator

****

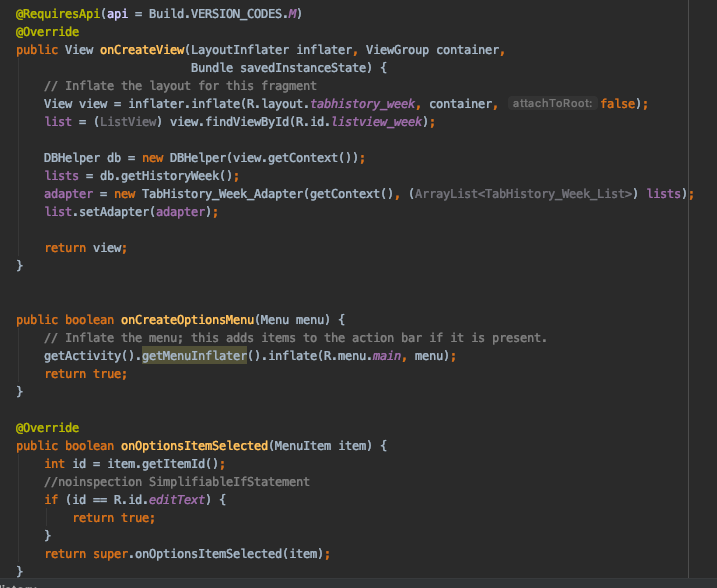
Currency converter

****

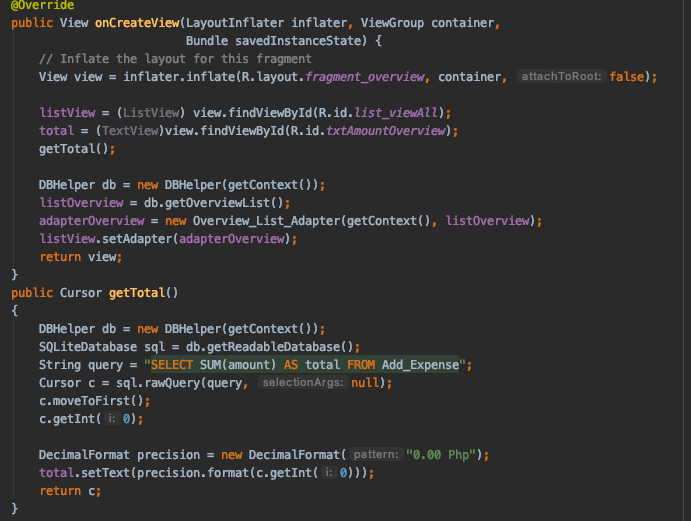
Payment reminder

****

History

****

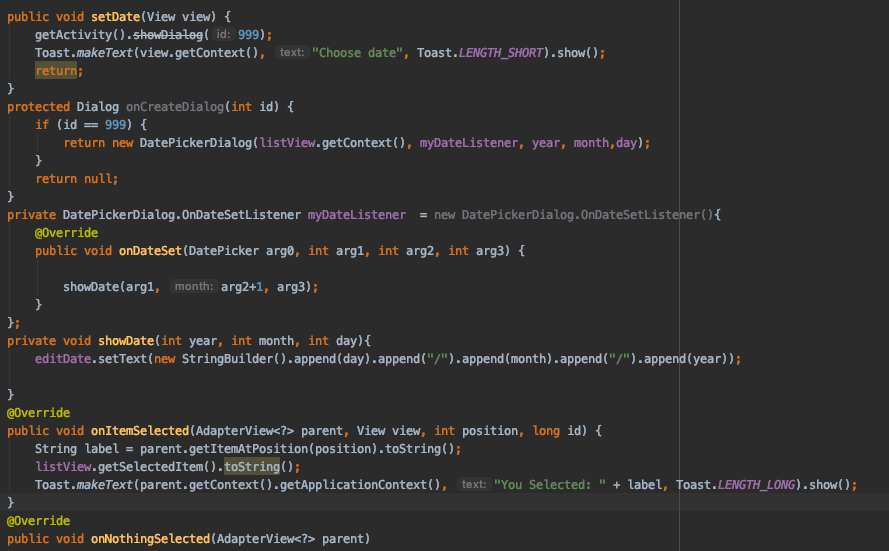
Overview

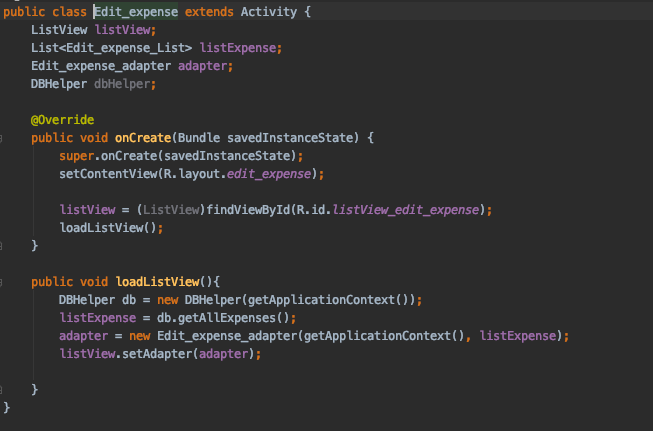
****

Distribution

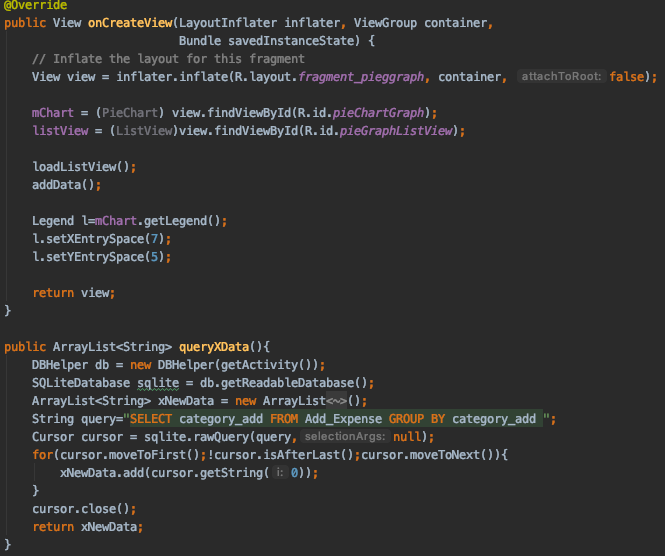


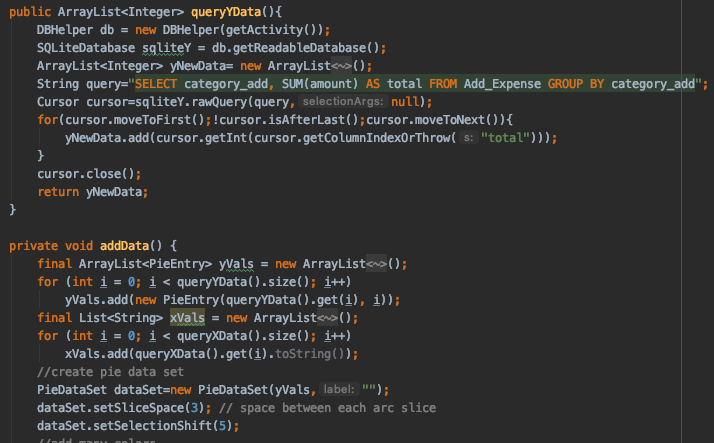
****

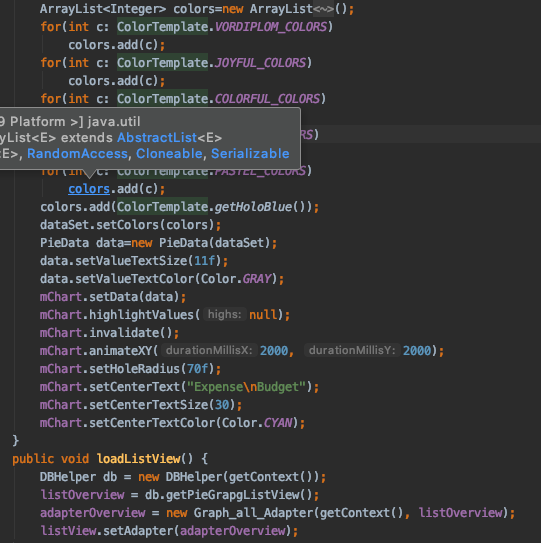
Expense entry

****

Piechart





****

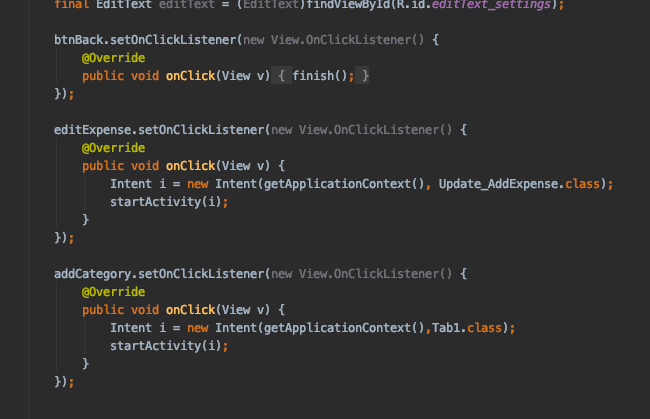
Pie Chart

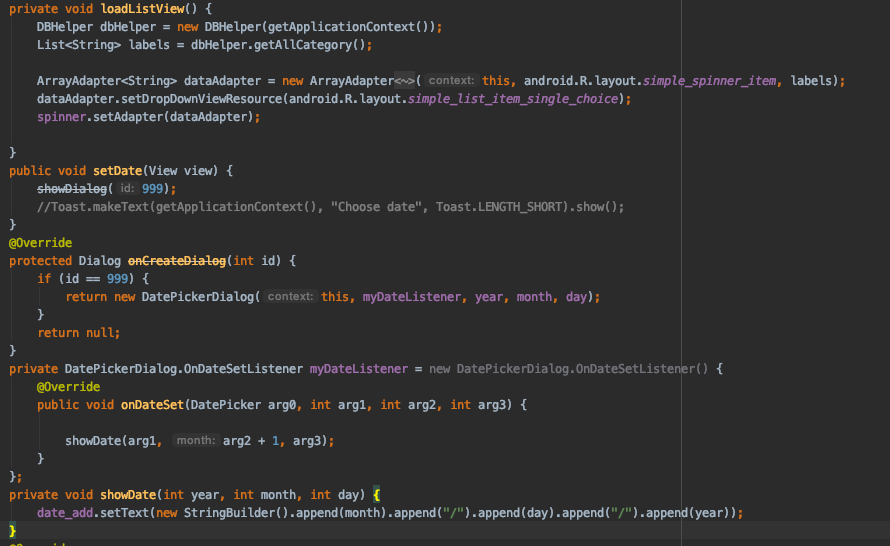
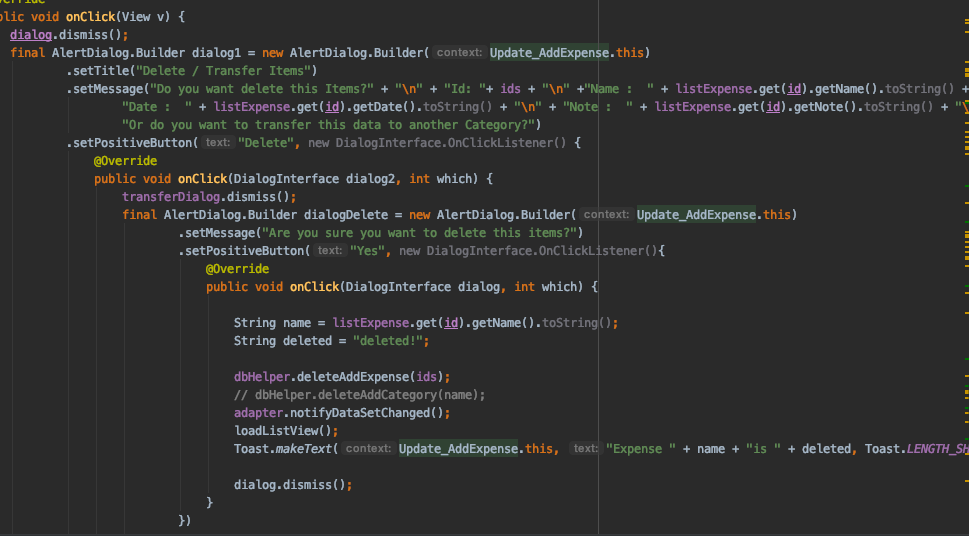
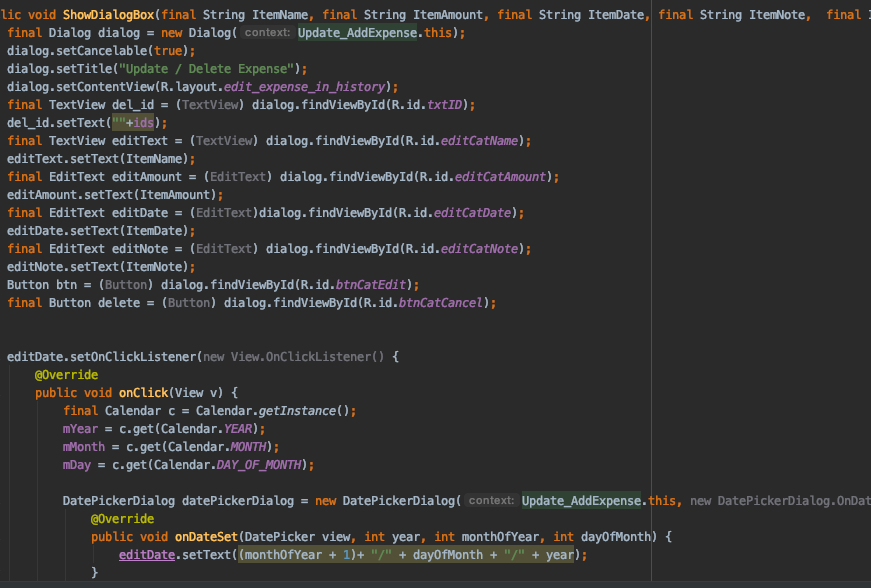
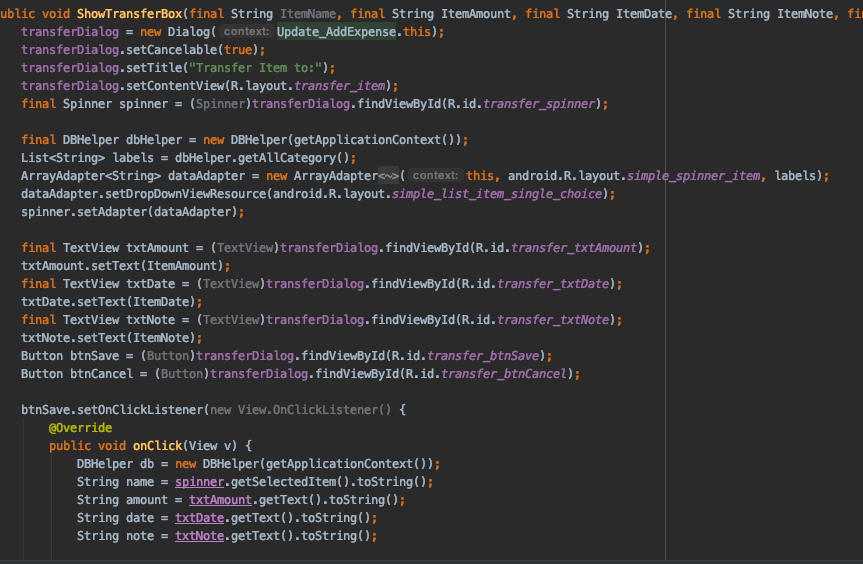
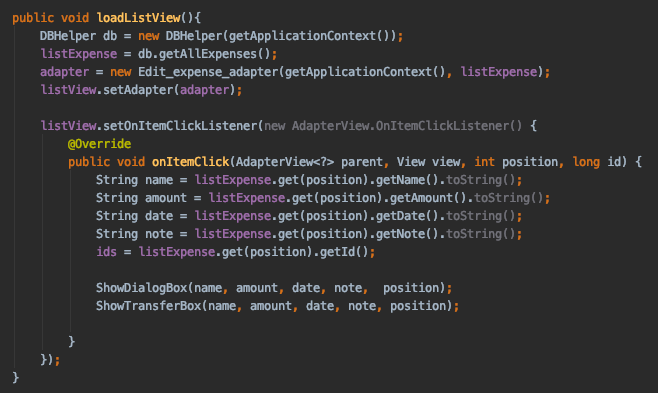


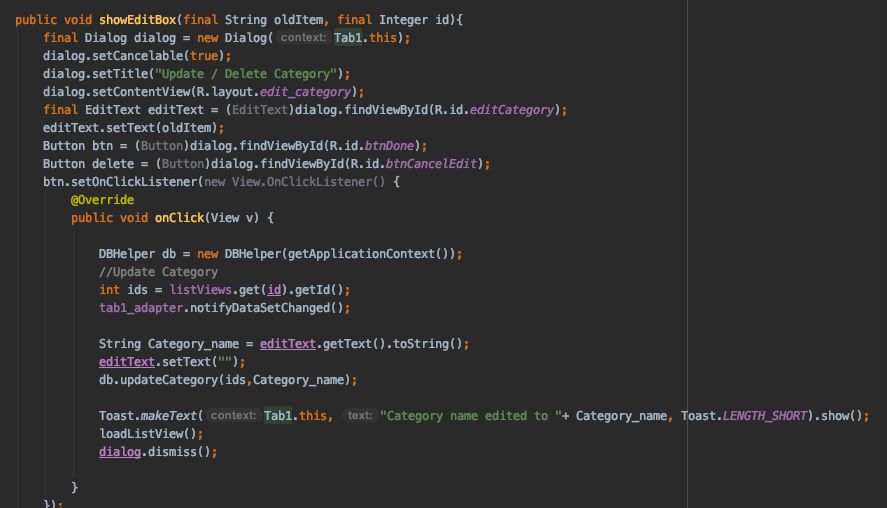




Settings







1. **Output and test cases**

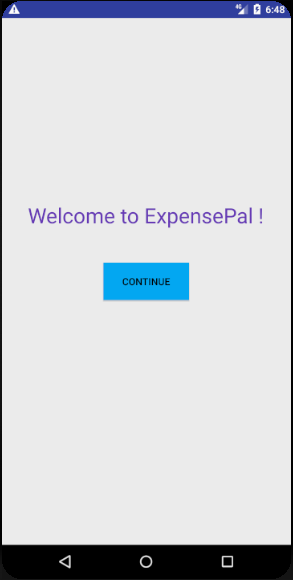
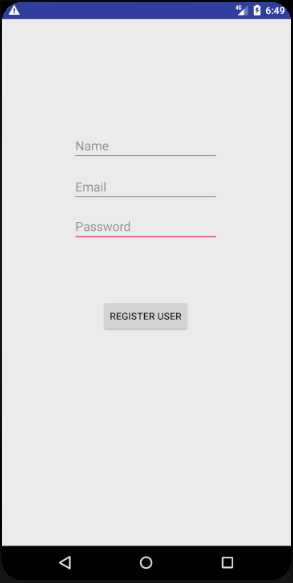
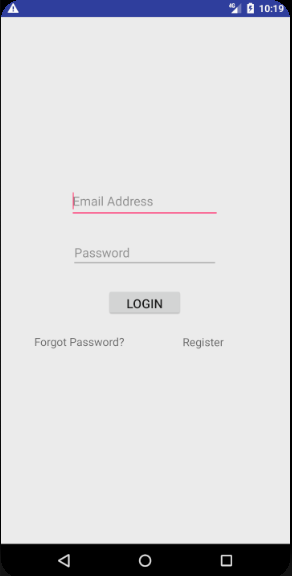
11.1 Test Case table

| No. | Test Case Name | Description | Priority | Result |
| --- | --- | --- | --- | --- |
| 1 | Register user | The app asks the user to enter email id and sends an email to user to verify email only then they are allowed to login | High | Successful |
| 2 | Login | The app asks the user to enter email id and password after email verification | High | Successful |
| 3 | Forgot Password | The app allows the user to reset their password once they enter their email id and an email is sent to user in which they can reset their password | High | Successful |
| 4 | Add category | The app allows the user to add a category | High | Successful |
| 5 | Delete category | The app allows the user to delete a category | High | Successful |
| 6 | Update category | The app allows the user to update a category by changing its name | High | Successful |
| 7 | Add expense entry | The app allows the user to add an expense entry | High | Successful |
| 8 | Delete expense entry | The app allows the user to delete an expense entry | High | Unsuccessful |
| 9 | Update expense entry | The app allows the user to update an expense entry by changing the amount, and date in the entry | High | Successful |
| 10 | Create payment reminder | The app allows the user to enter a payment reminder by selecting the date and time and enter or record a message | Medium | Successful |
| 11 | Add wishlist item | The app allows the user to add a wishlist item | Low | Successful |
| 12 | Delete wishlist item | The app allows the user to delete a wishlist item | Low | Successful |
| 13 | Calculator | The app allows the user to perform addition, subtraction, division, and multiplication operations. | Low | Successful |
| 14 | Currency converter | The app allows the user to enter a value in a certain currency and convert it to another currency. | Medium | Unsuccessful |
| 15 | Overview | Here the app shows the total amount of all expenses entries | High | Successful |
| 16 | History | Here the app shows all the expense entries made | High | Successful |
| 17 | Distribution | Here the app shows the distribution of expenses in each category | High | Successful |
| 18 | Budget | Here the app displays a pie chart | High | Successful |
| 19 | Delete account | Delete a users account | High | Successful |

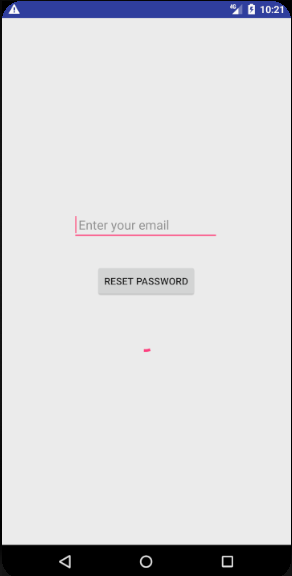
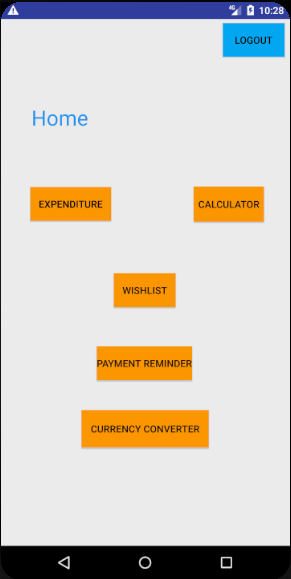
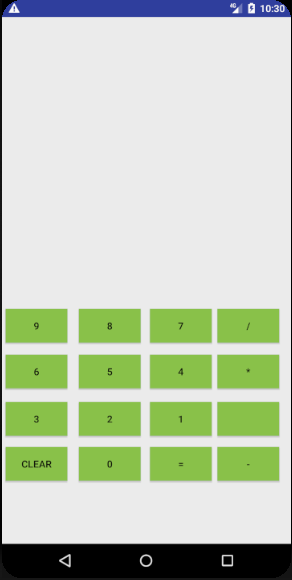
11.2 Output

Pages of ExpensePal App

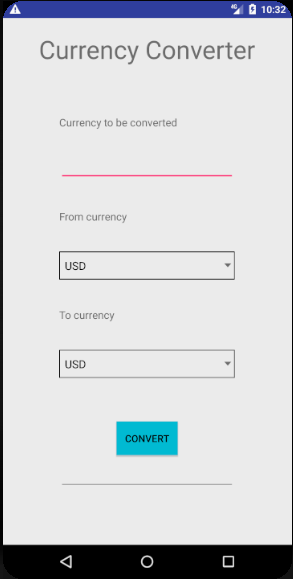
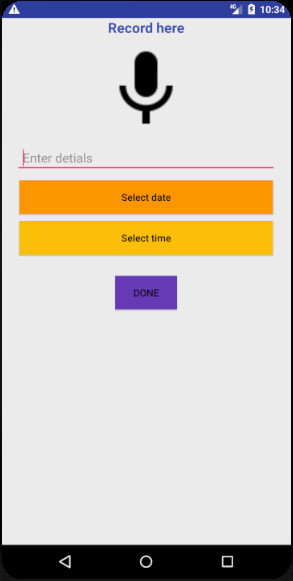
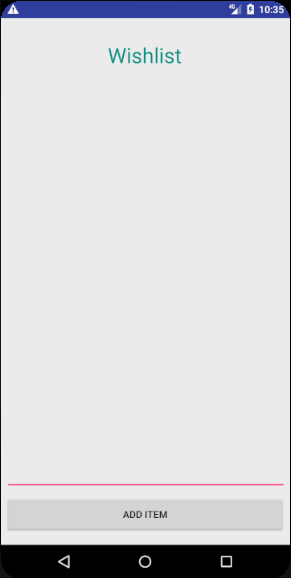
Welcome page Register page Login page

****  ****

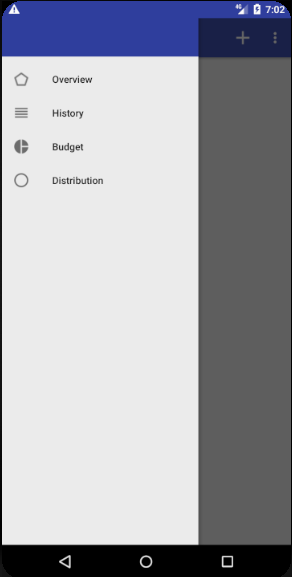
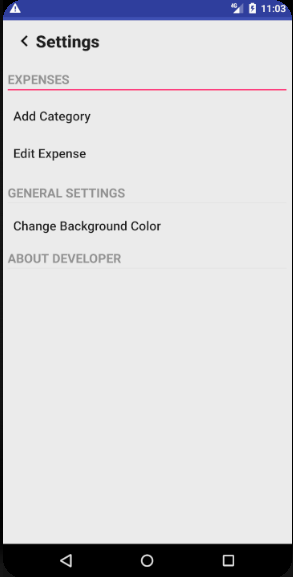
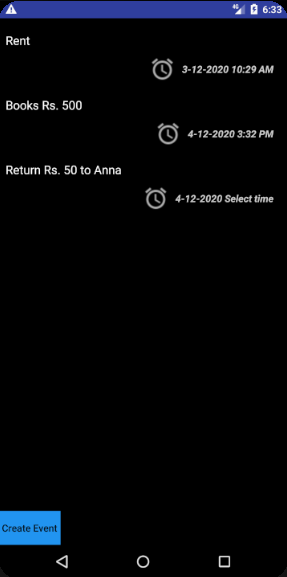
Forgot password page User page Calculator

**  **

Currency converter Create pay reminder Wishlist

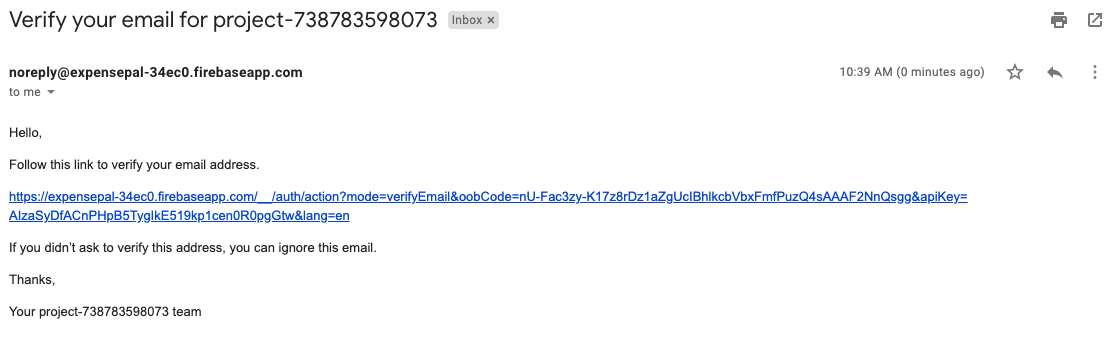
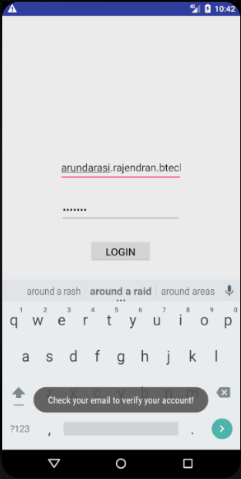
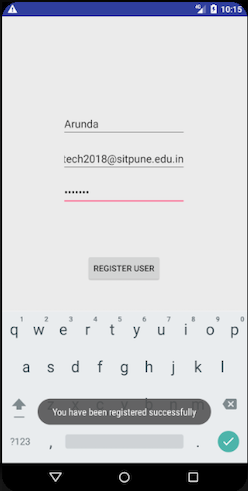
**  **

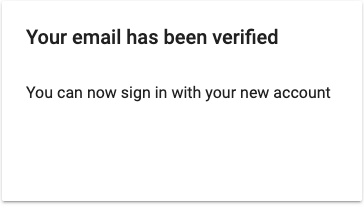
Expenditure Dashboard Settings page Pay reminder page

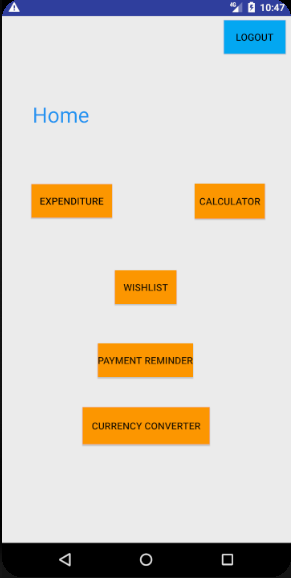
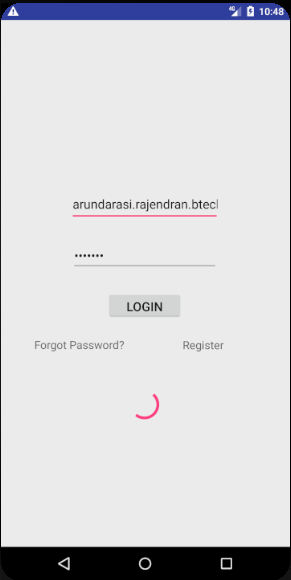
*11.2.1 Positive test cases*

Register user



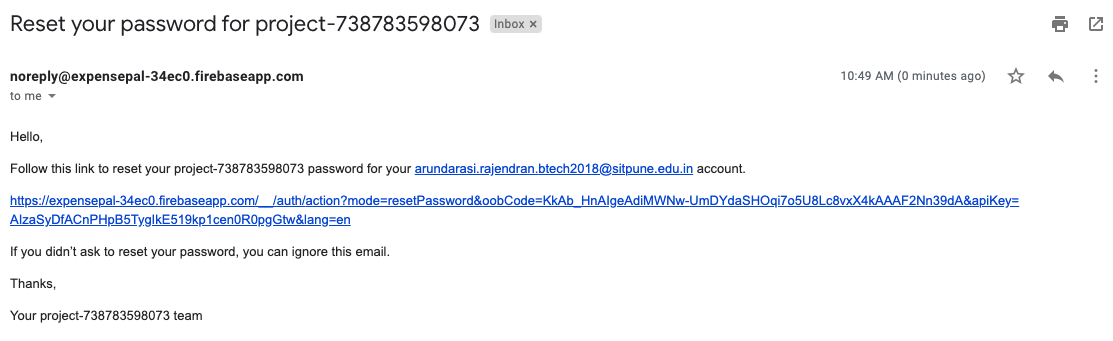


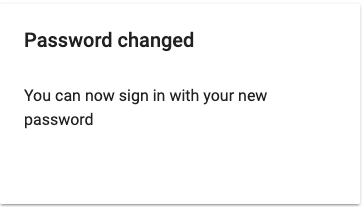
Login



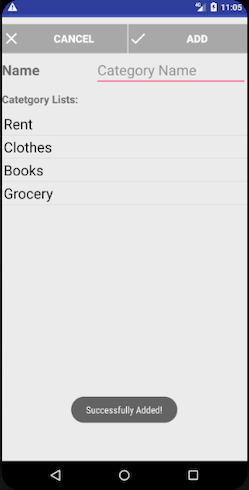
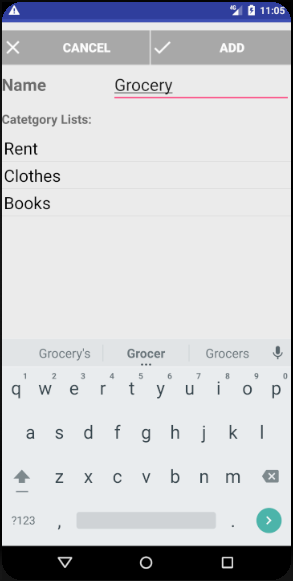
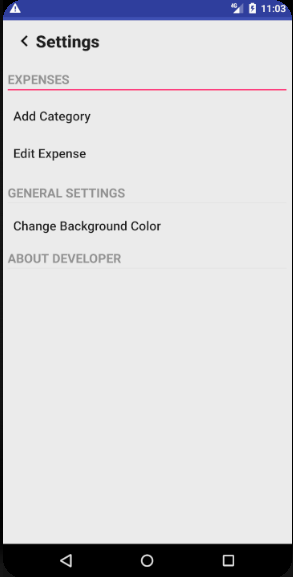
Forgot password



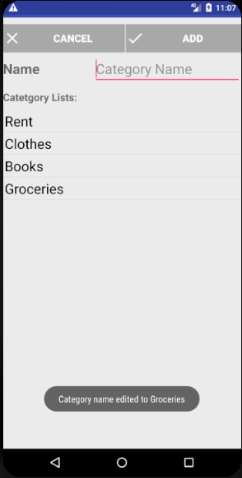
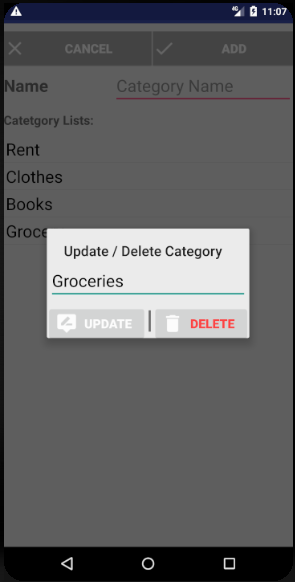
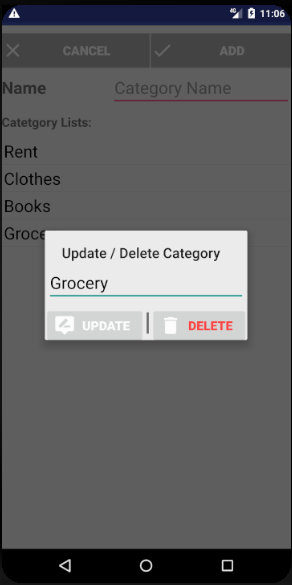




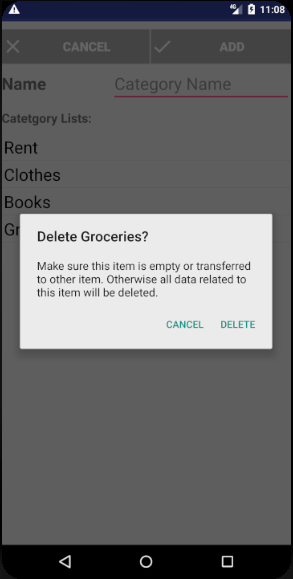
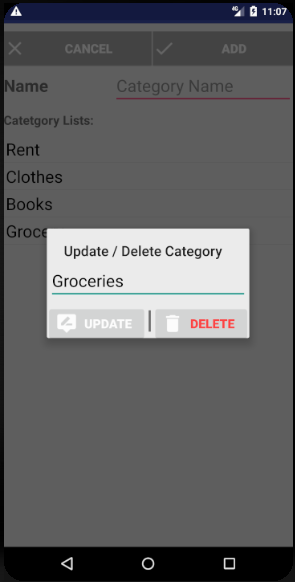
Add category

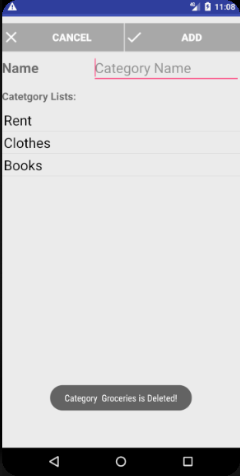
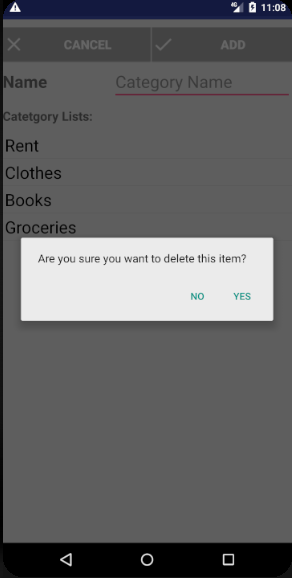


Update category

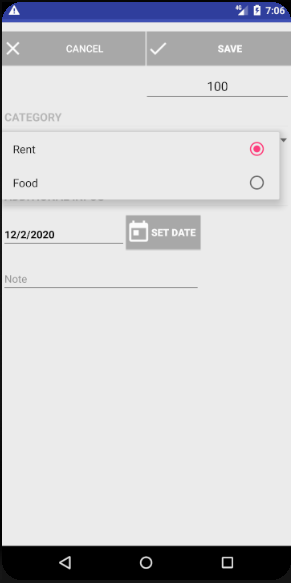
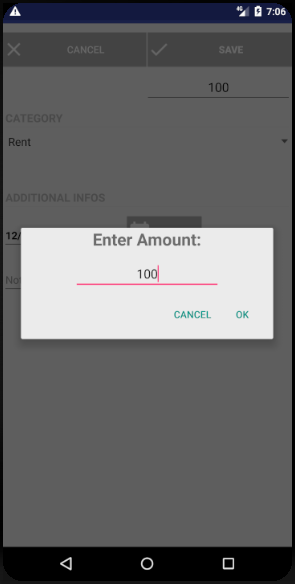


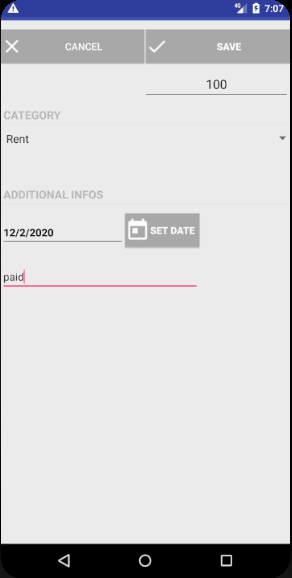
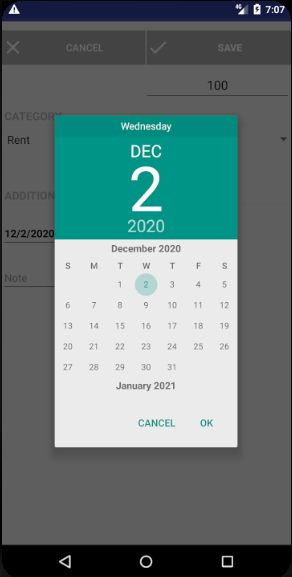
Delete category



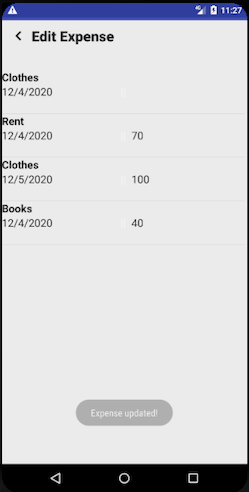
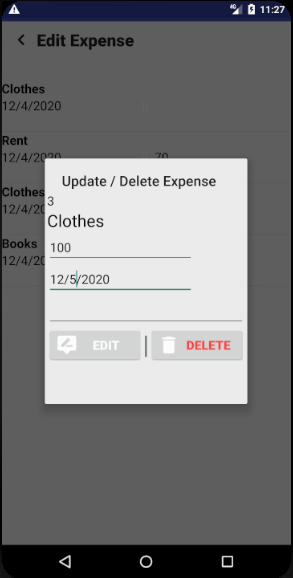
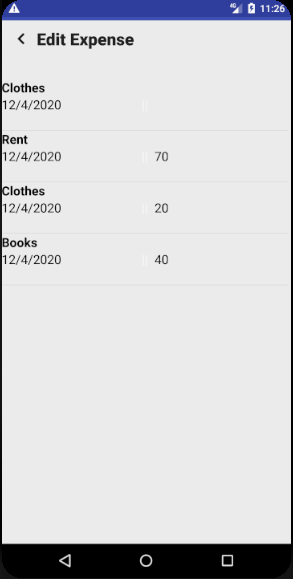


Add expense entry

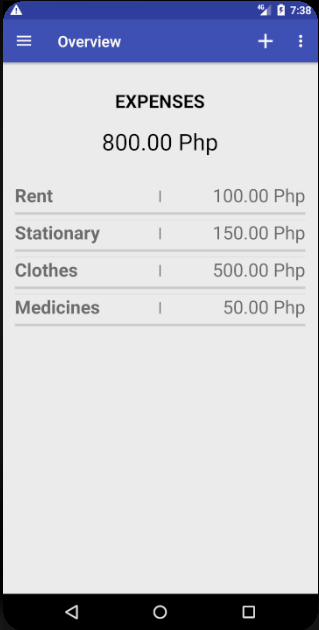
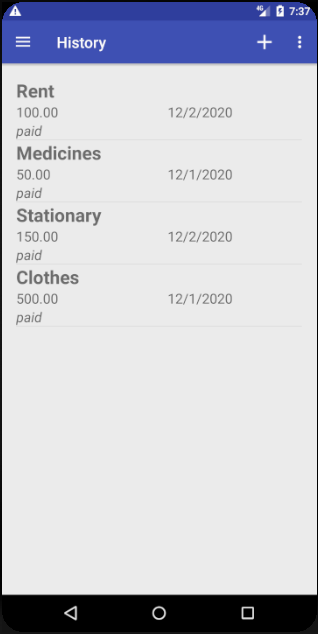




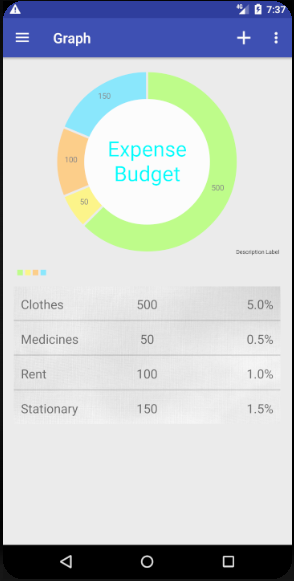
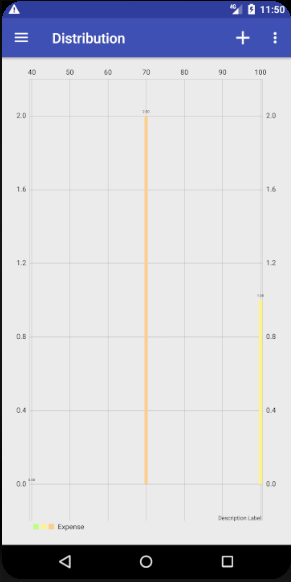
Update expense entry



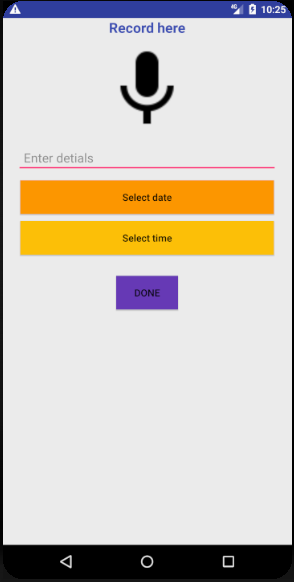
Overview of expense entries History of expense entries

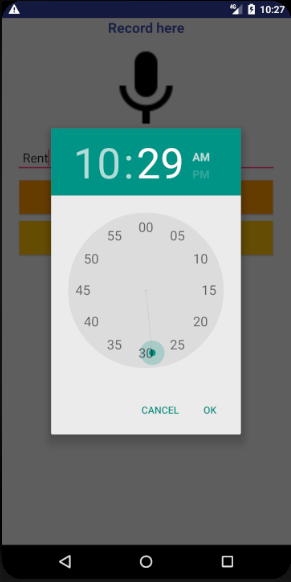
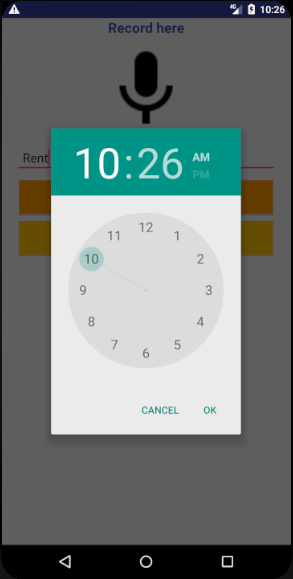
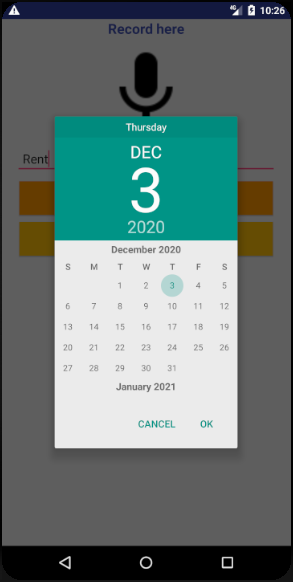
 

Display pie chart Display distribution chart

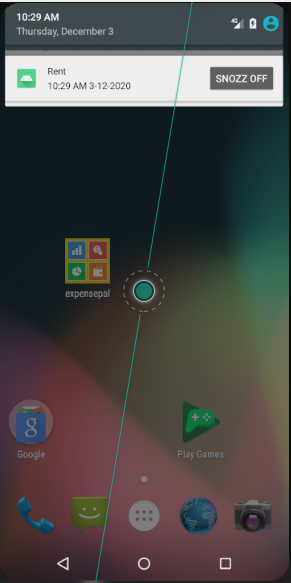
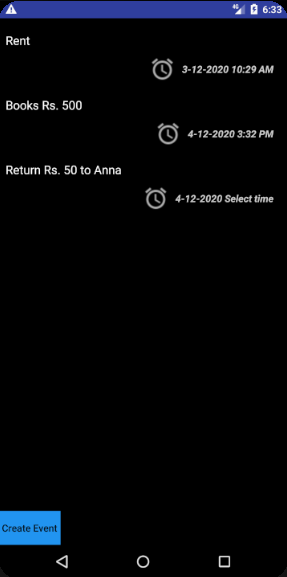
 

Create payment reminder

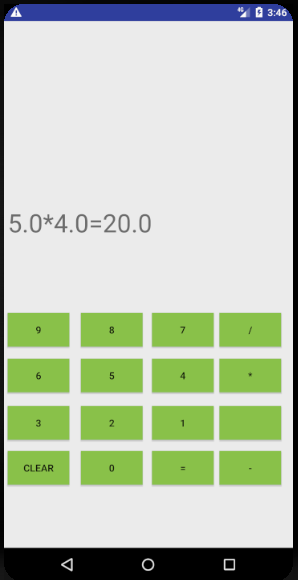


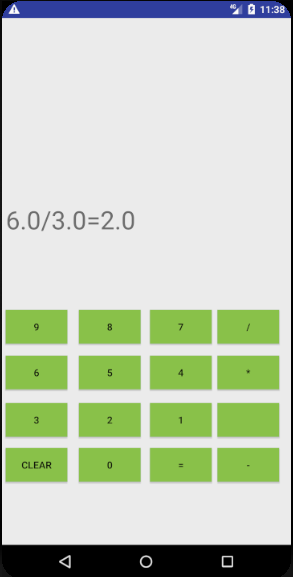


Payment reminder notification message

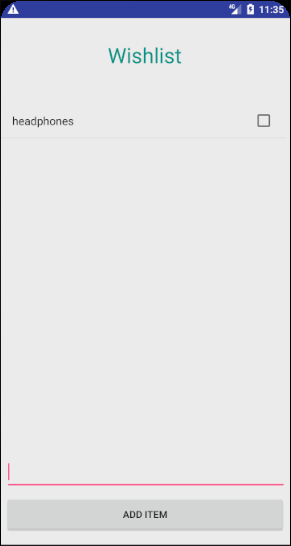
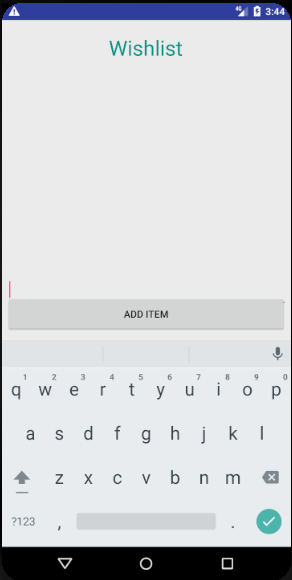


Calculator - addition, subtraction, multiplication, division

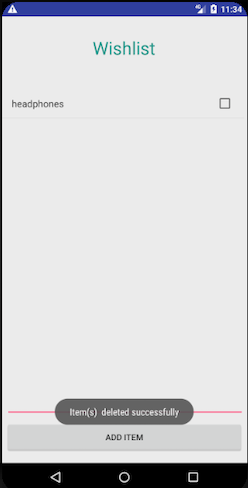
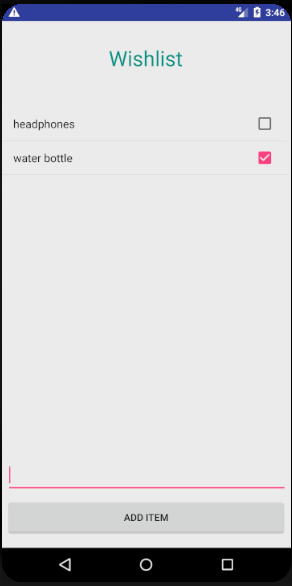




Add a wishlist item

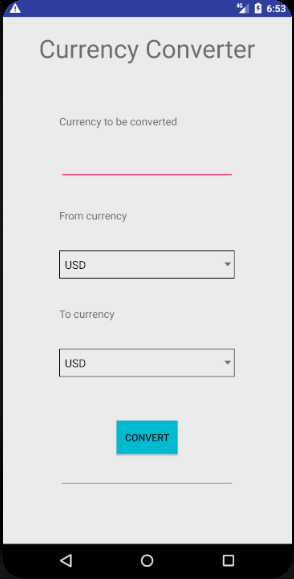
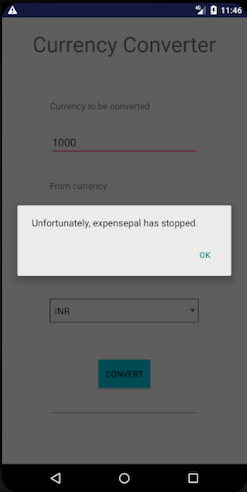


Delete a wishlist item

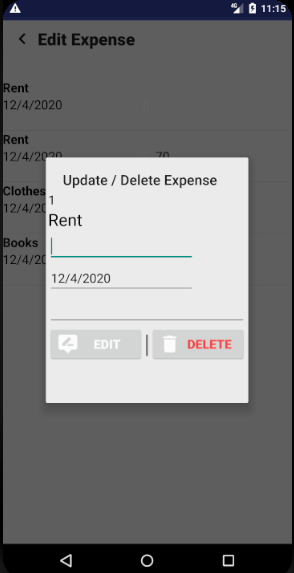
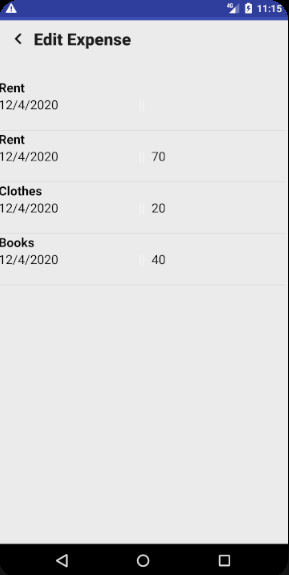
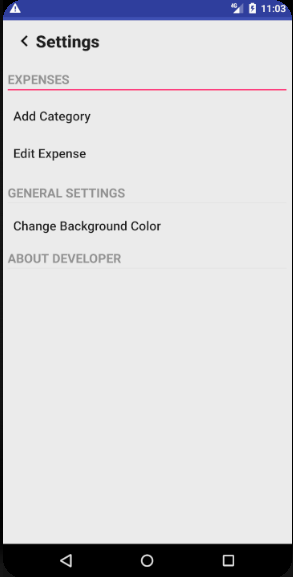


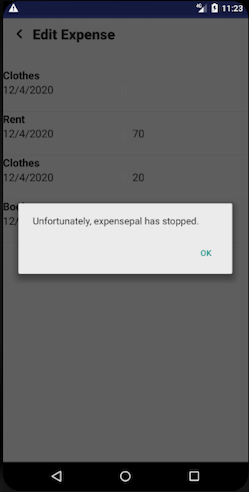
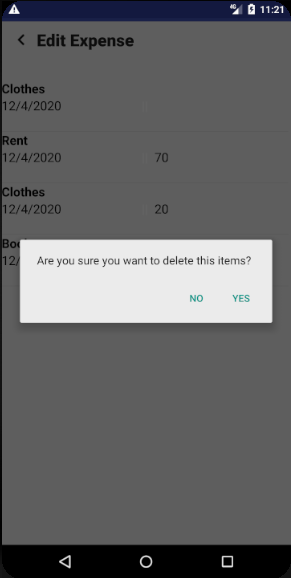
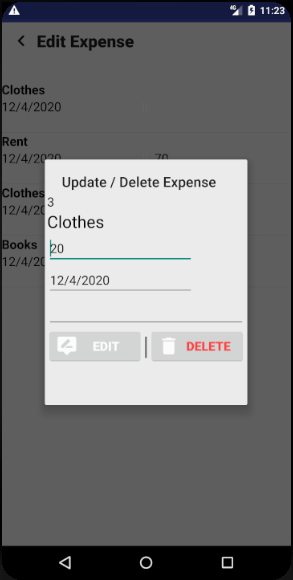
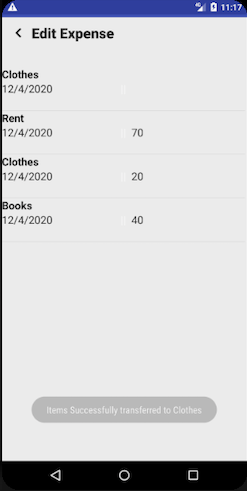
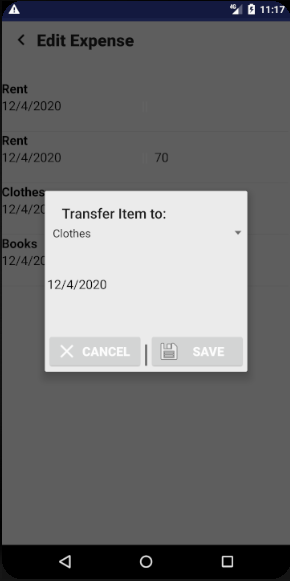
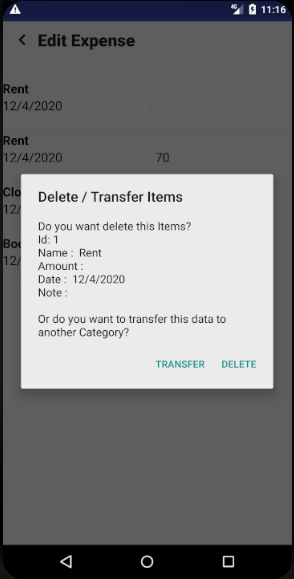
*11.2.2 Negative test cases*

Convert a currency value to another currency value

Delete an expense entry





1. **Validations**

In this project, most of the functional requirements are fulfilled. Our high priority requirements have been met properly. The app allows the users to register themselves using email ID and then allows the users to login using email ID and password. At the time of registering, the app asks the user to enter a password that satisfies the requirement of password based security. The user is allowed to add, delete, and update category names in the app. The app allows the user to add, and update any expense entry in different categories made by the user who is logged in. But the app is not able to delete any expense entry. The app displays list of expense entries in overview and history and category list in settings that become convenient for the user to view their expenditures. Also, the app displays pie charts in the budget page and bar charts in the distribution page.

Meanwhile, the medium priority requirements have somehow been met.

The app has a calculator facility where the user can perform arithmetic operations only and no advanced operations. The app has the facility to add payment reminders and get notifications at the specific time. The app has a currency converter but it does not display the converted value. Let's not forget that some low priorities have been met. The mobile app displays the welcome message once it is opened and the user clicks on the continue button to go to the login page. The mobile app has a wishlist where the user can add or delete items. However, the mobile app is not able to change background color.

Some of the non-functional requirements of this project have been met. The mobile app asks the user to enter email ID and password in order to log in to the app to retrieve restricted information therefore, security requirements are met. Anyone can download this app and use it to track their expenses and make wise decisions on their spending. Also, after logging in users would be able to add, delete, and update expense entries and categories plus, use payment reminder, calculator, and wishlist but would not be able to use currency converter and delete expenses. Thus, the usability requirement is somewhat met. It can be used on android phones readily if the user downloads this app from the app store, so portability requirement is met. The app is designed to send a notification message of payment dues to the users mobile phone on time. Plus, the app does display an updated pie chart and distribution chart after every added or updated expense entry. Here performance requirements are achieved. The mobile app is functional when the app is downloaded in the mobile; however, the login process requires network connection for the verification process. Therefore, the mobile app cannot work if the phone has no network connection for the login process but after the login process the app can be used in offline mode. Thus, reliability is not entirely achieved. The mobile app is available to all users but it is not available all the time because it requires network connection for the login process but not after the user is logged in. Thus, availability is not entirely achieved. The app is runnable on android phones. The mobile app is designed in a way for the user to view their expense entries and visualize how much of their expenditures are going in which categories. The colors and size of text set in this app makes it viewable to users. Here supportability is achieved.