

Project Proposal

SIT305 Android And iOS Mobile Programming



Deakin University

ZHENG LI

2019082963

**Directory**

[Overview 2](#_Toc41141454)

[Product Purpose 2](#_Toc41141455)

[ Target audience 2](#_Toc41141456)

[ Creativity 2](#_Toc41141457)

[Features 2](#_Toc41141458)

[Design 3](#_Toc41141459)

[ Screen & field 4](#_Toc41141460)

[ Navigation 5](#_Toc41141461)

[ Overall color scheme 6](#_Toc41141462)

[Data 6](#_Toc41141463)

[ Runtime variables 6](#_Toc41141464)

[ Permanent storage data 6](#_Toc41141465)

[API/Class structure 7](#_Toc41141466)

[Resources required 9](#_Toc41141467)

# Overview

The app I plan to develop is about the Account Book. This app can help us record the detail of each cash flow so that we can understand our current economic situation and manage funds reasonably. This app will display each fund according to chronological order and will visualize the proportion of each month's funds in each way in the form of a pie chart. It will be useful for us to modify our plan in the next month. The information of each user will be protected by their accounts and passwords.

# Product Purpose

## Target audience

The target group of this software is all people who have the funding or need to plan the funds reasonably. The account book app provides useful functions with a simple layout so that users can easily learn how to use this app and to operate these functions. Each step is reasonably and understandable. If I were a user, I would choose or buy an app that could provide powerful functions and also not so complicated.

## Creativity

The account book app provides a pie chart in the ‘chart’ page where the data selected by users will be visualized and the proportion of each part will be displayed. In this way, the user can efficiently understand and easily analyze the data by themselves instead of spending a considerable amount of time to calculate to get the result.

# Features

1. Data Visualization

Parts of data will be visualized in a form of Pie Chart. It will dynamically display the result (not only the data itself but also some information gotten from analysis) according to users’ choices. This feature can greatly help users understand the key information of data so that they can quickly modify the plan for future.

1. Data Storage (SQL Lite)

All the data is stored in the SQL Lite which is a local database. It can help users protect their personal information. Users can insert, update, or delete the data stored in the database by themselves. This database is bound with users’ devices.

1. Account management

This app provides an account management method that users can register and modify their personal information. It allows users to use their email or phone (email and phone should have been added to their account) to log in their account.

1. Services and broadcast receiver

To implement the dynamical update, I use the Broadcast receiver to filter and get the right command to refresh the recycler-View or refresh the data gotten from the database. Through this way, the app will run fluently and will not stay at a page and wait for a long time.

1. Text and Input (Spell checker)

This application provides the spell checker on the insert page so that users can conveniently check their spelling. It allows users to check more than a word at a time. The method will recommend some words that may be what you want.

1. Location

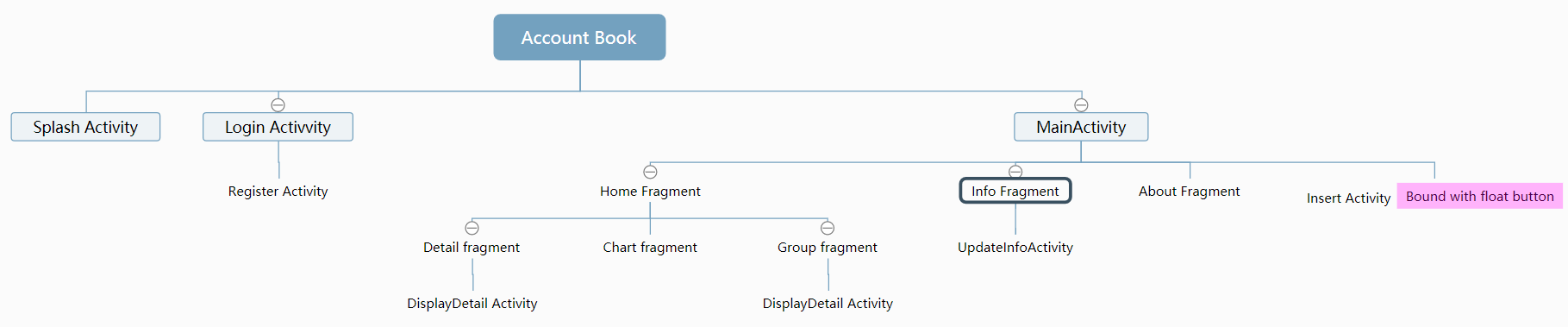
This app can get and record users’ locations if it has been authorized by users. It will be convenient for users to remember this fund flows and be useful for them to manage their plans.

1. Animation

Static pictures or static methods will make users boring if they have to frequently use this app, while animation will to some extent get users’ interest. The popping windows bound with a gradient effect, refresh method in recycler-View, and sliding effect in View Pager give users a better user experience.

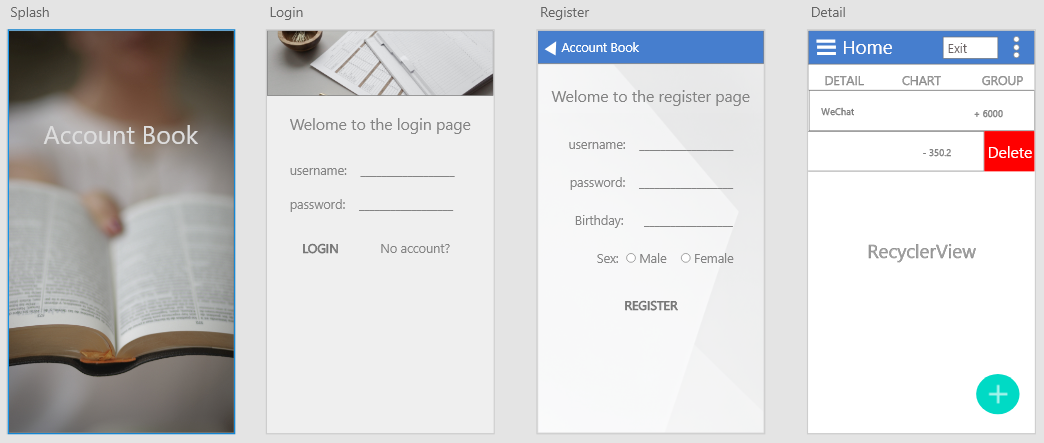
# Design

The UI design file is stored in the ‘design’ folder including the XD file.



## Screen & field

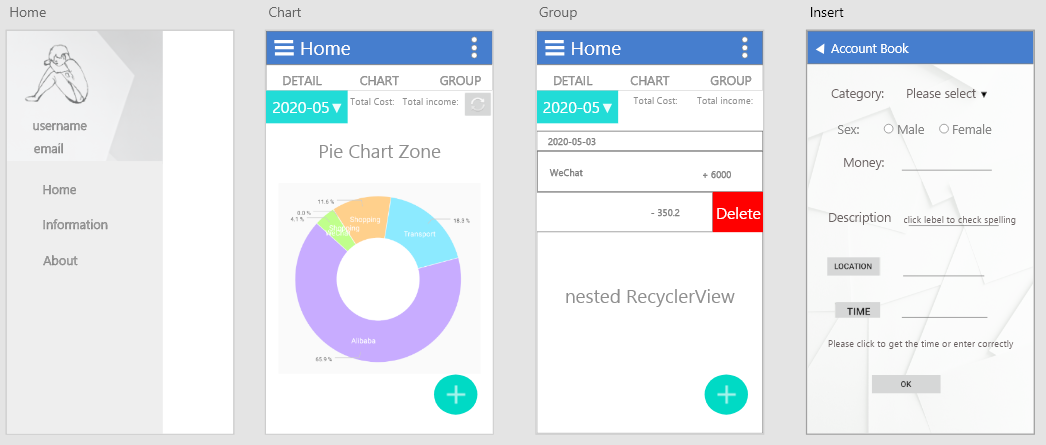
This app has 12 screens (some screens are bound with others). It can be mainly divided into 5 parts: login, register, display, user information, and about the setting. This application is designed based on the Navigation-View framework. The detail of each screen is followed:



‘Login’ screen mainly contains two buttons, three text-View, two Edit-Text, and an image-View. ‘No account’ button is linked to the 'Register' activity. It provides methods for the user to log in to their account and load the data. Users are allowed to use their username, email, or phone number to login.

‘Register’ screen contains two edit-text components, five text-view, one radio group, and one button component. Users should enter all the information to register an account. ‘Birthday’ is bound with a data picker function.

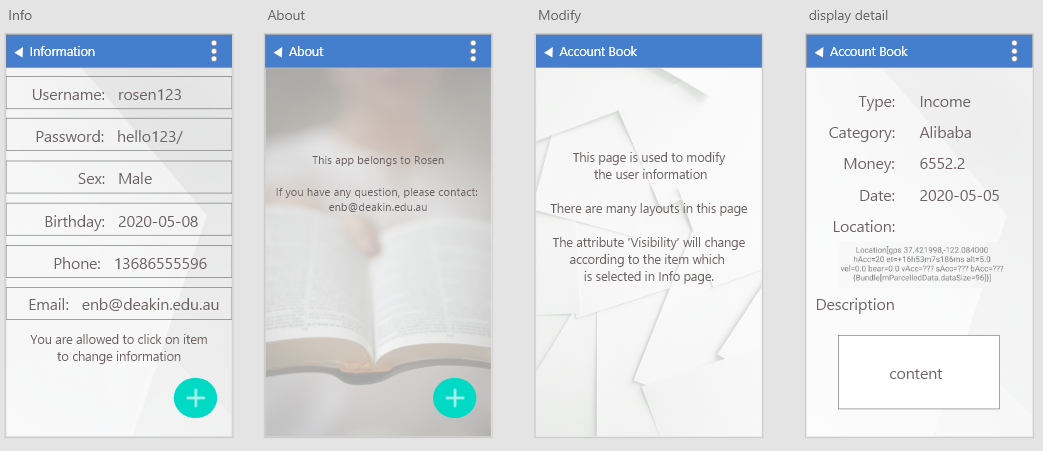
Fragment ‘detail’ only contains recycler-View which is redefined by myself to implement the slide delete function. The floating button ‘add’ is defined in the Main Activity, so it is available in a few pages.



‘Home’ is a navigation page that contains two groups: the first group includes a default image and two personal information which will change according to the users’ information; the second group contains three items and each item is linked to a fragment.

‘Chart’ and ‘Group’ fragment is similar to the ‘detail’ fragment: they all have the ‘date picker and total’ bar in the head of the page. ‘Chart’ contains a Pie chart component that can visualize the data. The pie chart component is referenced from another package. The ‘Group’ fragment contains a nested recycler-View: the inner view displays the content of each item and the outer view divides data into different groups according to the date.

‘Insert’ screen includes some text-view and edit-text components that is similar to the screen before. The ‘Description’ label is bound with a spell checker function. The ‘location’ and ‘time’ button are bound with geological sensor and date picker respectively.



‘Info’ screen contains six layouts and each layout includes two text-view components.

‘Display detail’ screen contains twelve text-view components. This screen is bound with the items in the ‘detail’ screen and ‘group’ screen.

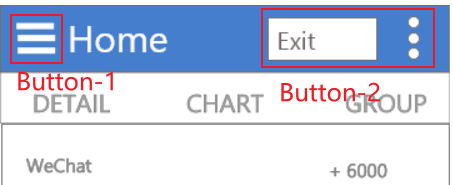
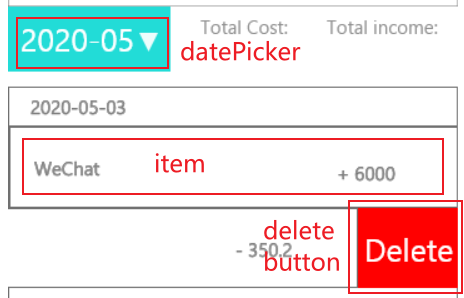
‘Modify’ contains five different layouts that include a few text-View and edit-text components and attribute ‘Visibility’ will change according to the item user selected in the ‘Info’ screen..

## Navigation

I select ‘Navigation-View’ in my design as it can easily show the function of each part on the screen. For example, when the user clicks the button-1 in the header bar on the screen. The ‘home’ screen will be displayed and the button-2 is bound with an Exit function. These types of icons are generally used by people so that they can quickly know the meaning of the button. Besides, the label bar below the header bar indicates that there are three different pages in this activity, so users will click or slide to change screen,

In the ‘chart’ and ‘group’ screen, I provide the on-click function and slide delete function as people always try to click on something or use gestures to test if there is a function.

Overall, I use many icons commonly used by people as key buttons for many functions. Therefore, users can easily learn and how to use the app to do some operations.

## Overall color scheme

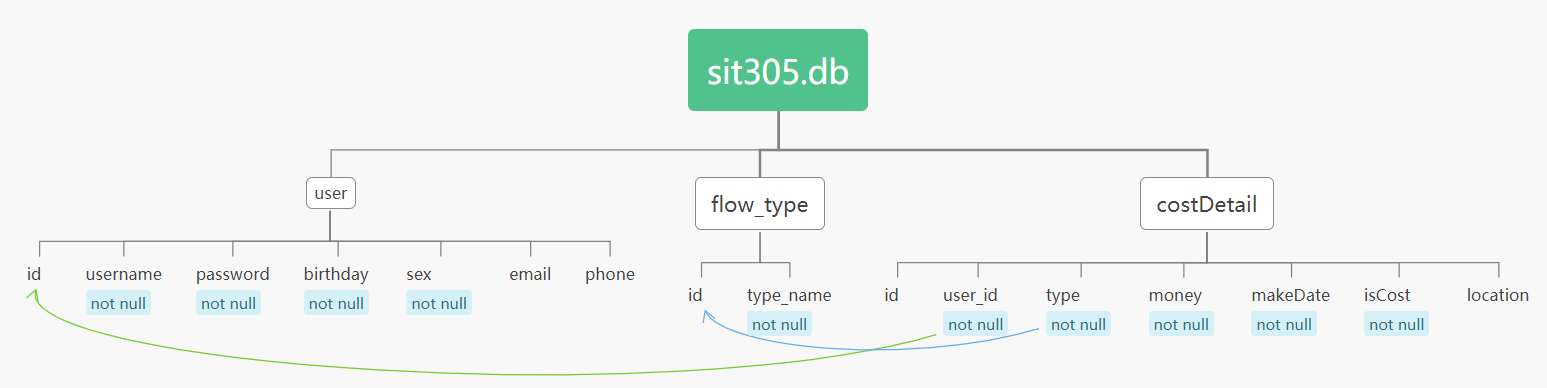
The theme colors of this software are gray and light-blue, gray will not make people feel too much pressure, the blue navigation bar will make the software feel a little office. The overall design style of the app is simple and practical, without a complicated design layout so that users can quickly learn and understand the function of each page or screen.

# Data

## Runtime variables

There will be a few runtime valuables while running this app including static tags which is used to filter broadcast. In this app, the runtime variables are mainly used to store the data received from the database and they will be released without possessing too much time. The other variables that are used to transmit the data between activities will be small enough. Thus, the fluency of the application will be promised.

## Permanent storage data



The database the app use contains three tables: 'user', 'flow\_type', and 'costDetail'. The table ‘user’ contains personal information. Table ‘flow\_type’ contains the pre-defined type of each flow. Table ‘costDetail’ contains the information of each cash flow.

The app will not load all the data at once but dynamically loads the data as required. Although the database may be called frequently, it takes up as little running memory as possible for as long as possible.

# API/Class structure

This app will contain about 13 classes excluding all the Activity.class and fragment.class files and the detail of each class will be displayed. The structure chart of some classes has been provided.

(The number of method and member have been displayed)

* Recycler-View Adapter
* Category Adapter

Category Adapter is planned to be used in the Insert-Activity to display the data of the table ‘flow\_type‘. This recycler-View will be bound with a popup window.

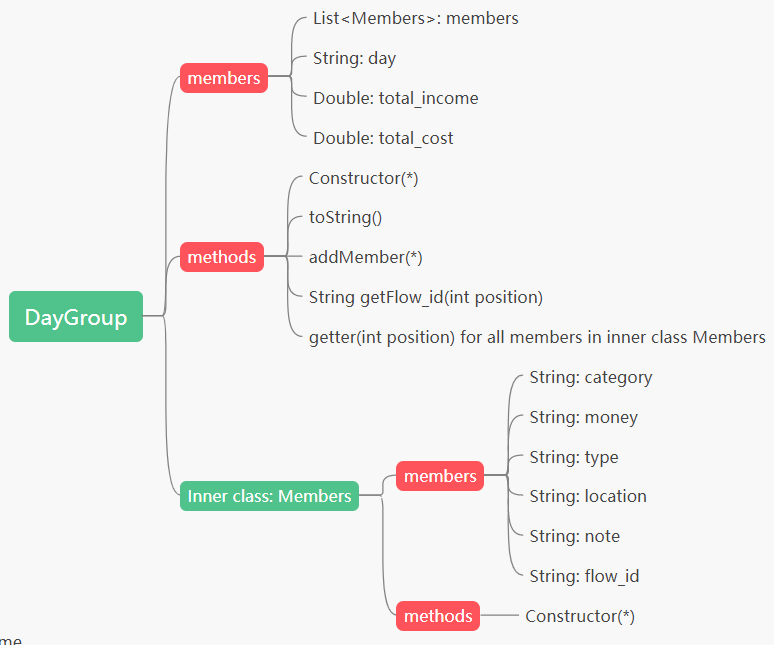
* Details Adapter

Detail Adapter will be used in the Fragment ‘Detail’ to display the data of each item that read from the table ‘costDetail‘. There will be on click function which is associated with the Activity ‘Display Detail’.

* Group Adapter & Group Item Adapter

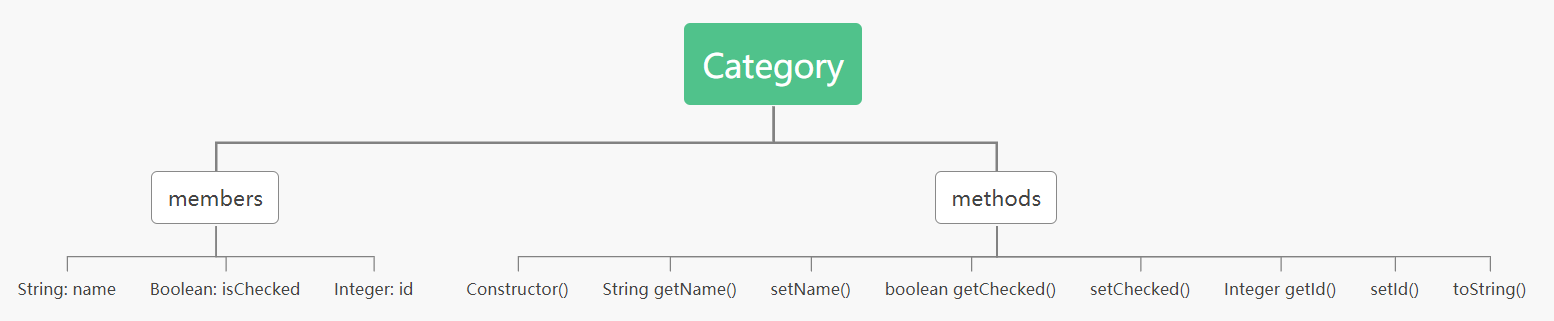
These two adapters will be used for the nested recycler-View in the ‘group’ screen.

* Model
* Day Group



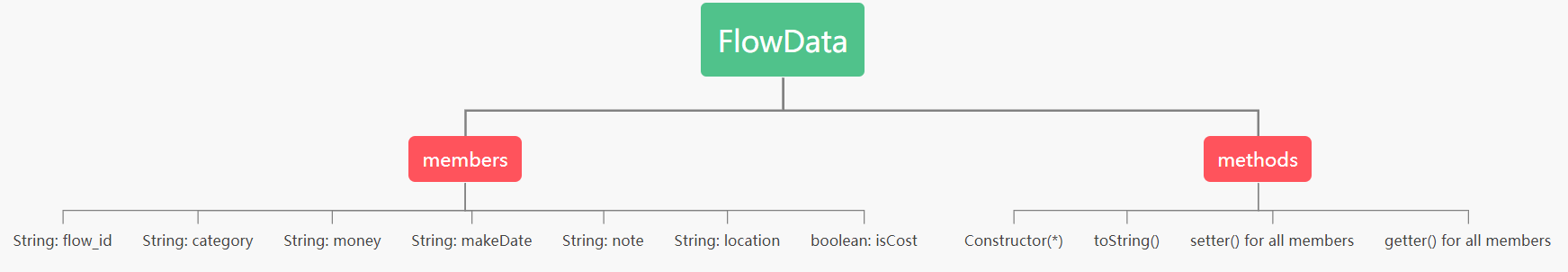
This class is design for the nested recycler-View according to the structure of the layout in the fragment ‘group’. The outer class contains the label of date and inner contains the detail information about each flow.

* Category



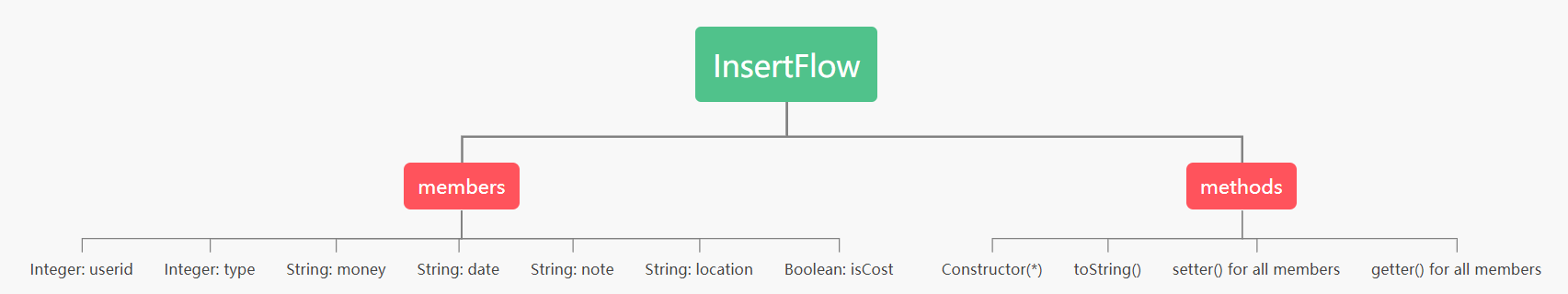
Class ‘Category’ is used to transmit data between the popups window and the database. It contains the setter and getter functions for each member.

* Flow Data



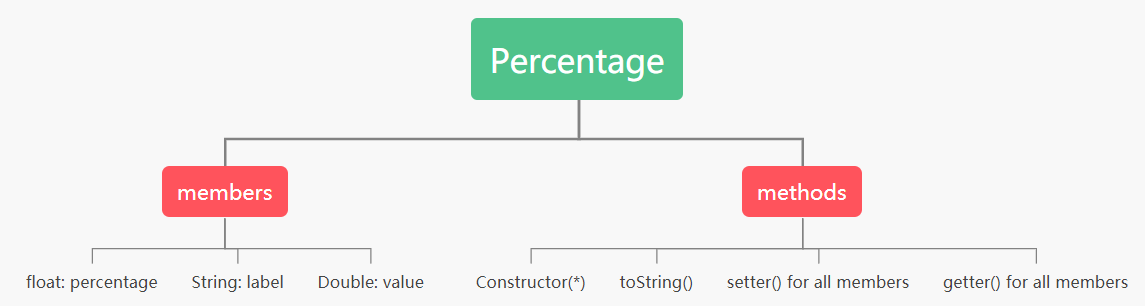
This class will be used in the fragment ‘detail’. It is similar to the class ‘DayGroup’ and ‘InsertFlow’, but some parameters are different.

* Insert Flow



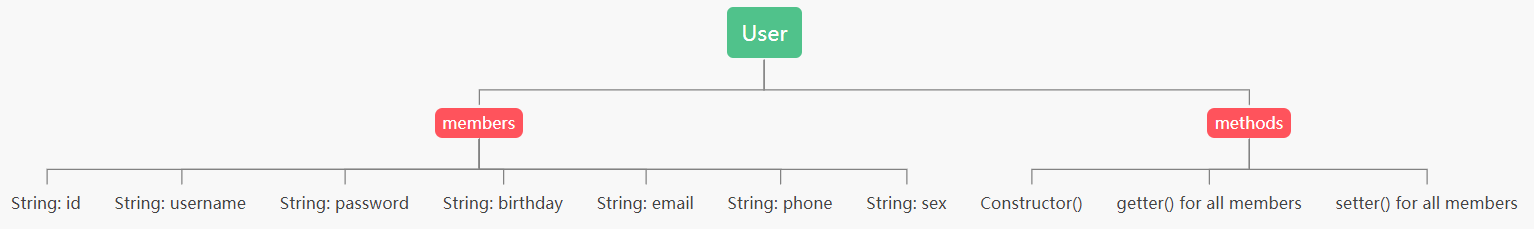
This class is designed for the function ‘insertData’. All the members in this are related to the data type in the table ‘costDetail’. The object of this class will be transmitted to the user-Service.

* Percentage



Class ‘Percentage’ is used to store the data which will be used in visualizing data in the fragment ‘chart’.

* User



Class ‘user’ is designed for account management. It will be used in the ‘LoginActivity’, ‘RegisterActivity’ and ‘UpdateInfoActivity’. All the members can be changed except ‘username’ and ‘user\_id’.

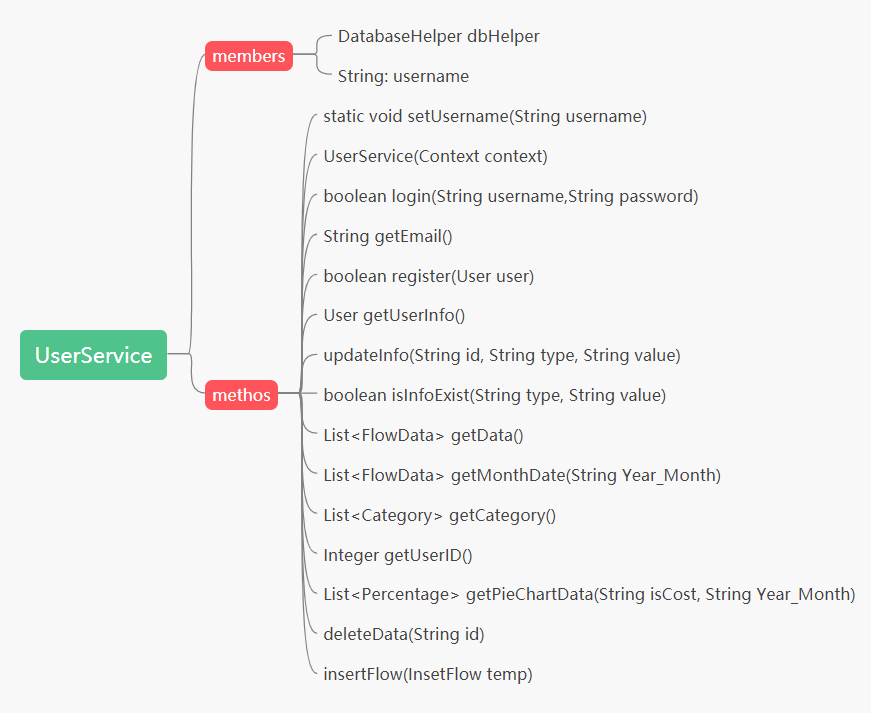
* Slide Recycler-View

In order to implement the ‘slide delete’ function, the class ‘RecyclerView’ will be extended and I will add some functions according to the requirement.

* Service
* Database Helper

This class extends the class ‘SQLOpenHelper’ to get the method to access the SQL Lite.

* User Service



The members ‘DbHelper’ is an object of class ‘DatabaseHepler’ and ‘username’ that will be initialized when user logins successfully (function ‘setUsername’).

The name of each method is associated with its function. These methods are related to the ‘search’, ‘insert’, ‘delete’, and ‘update’ of the database.

# Resources required

All quoted pictures are listed:

* splash.png —— Photo by Oscar Ivan Esquivel Arteaga on Unsplash
* login.png —— Photo by Jeremy Manoto on Unsplash
* temp1.png —— Photo by NORTHFOLK on Unsplash
* add.png —— Photo by Brandi Redd on Unsplash
* background\_2.png —— Photo by Viggo Jorgen on Unsplash
* background.png —— Photo by Bernard Hermant on Unsplash