

UNIVERSITY OF NOTTINGHAM

**SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING**

**JAV Web Based Programming – H63JAV**

**Java Programming Project**

**EasyAccounting App**

**Student’s name: Xindi Huang**

**Student’s ID: 16518350**

**Submission date: May 10, 2019**

**Manual**

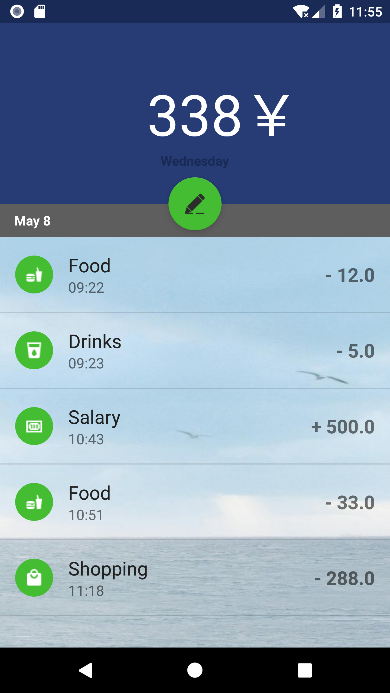
**System requirement:**

Android v.5.0 or higher

**Getting started**

This EasyAccounting app is designed for people who would like to record their daily costs easily. It can help users have a clear understanding of their daily consumption including the types and amount of income and expenditure.

The main menu of the application is shown below. The total amount of the expenditure of today will be displayed in the top. All the records of today will be displayed in a list. By swiping the list, user could check other date’s records.



**Day of the week**

**Date**

**Display that day’s records**

**Adding record button**

**Total amount of expenditure**

*Figure1 Main menu*

For a specific record, the type of expenditure of income, time and amount would be displayed.

**Type of expenditure or income**



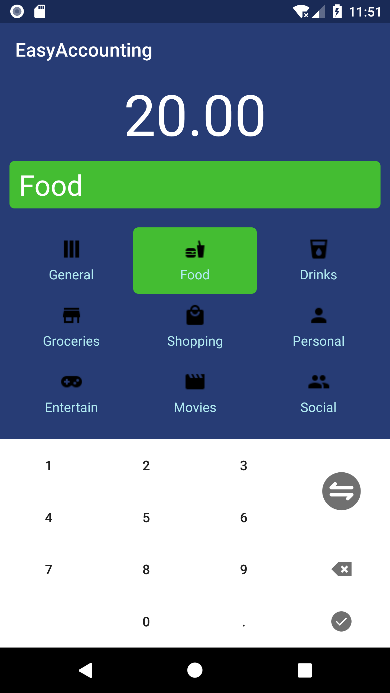
**Time**

**Amount**

*Figure2 Specific record*

**Add a record**

To add a record, user can click the adding record button in the main menu. Then, the interface is shown below. User could choose the type of expenditure or income in the type list. Then, type the amount through keyboard. User could switch to income or expenditure by clicking switching button. Finally, click the finish button, the interface will return to main menu.



**Finish**

**Backspace**

**Change to income or expenditure**

**Keyboard**

**Type list**

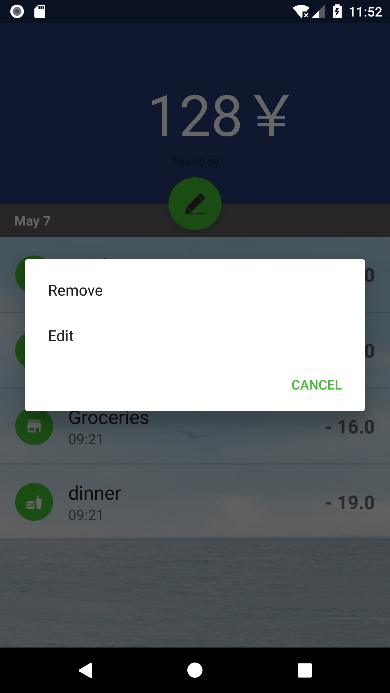
**Remark**

**Amount**

*Figure3 Adding record interface*

**Edit or Remove a record**

User could edit or remove a record by long clicking the record in the list. Then, a dialog will show up for user to choose removing or editing the record.



*Figure4 Dialog for choosing removing or editing*

**Testing and Discussions**

The testing could mainly be divided into unit testing and module testing. In the unit testing, the application could function properly and run smoothly. For the module testing, the database module, main menu module, and adding record module could function well separately.

The project could approximately meet the design goals. Firstly, it provides a friendly user interface which enable user to add a record conveniently. In the “adding records” page, a variety of consumption types are listed such as food, drinks, shopping. The user only need to choose the type in the list and type in the amount by keyboard and finally press the “finish” button. The whole process may only take about 10 seconds to add a new record.

Secondly, the app provides a convenient method to edit or delete the record. By long pressing the item, a dialog box will show up. The user could choose to edit that record or delete that record.

Thirdly, the app is based on SQLite database. All the records are stored in SQLite database and when the program ends, the data does not disappear. A part of database is displayed below.



*Figure5 Database*

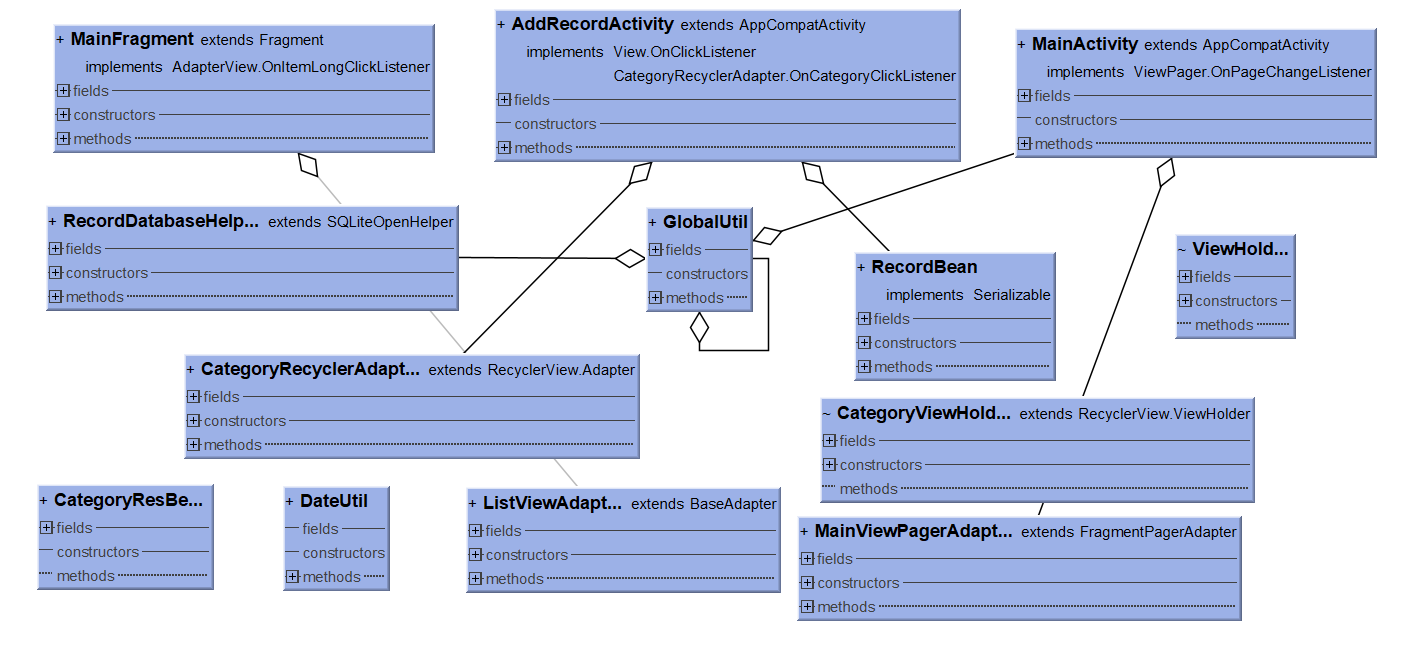
In addition, an open source Android UI component named Ticker is introduced into the project. It can realize displaying scrolling text which is used to display the total amount in the top of main menu. When the amount is changed, the scrolling text will show up.

In the main menu, all the records of that day are displayed in a listview. This listview is integrated in a fragment and the fragment is integrated in a viewpager. User could check other days’ record by swiping the list. Once a new record is added, the reload() method within viewpager is called first which would traverse all the fragments. Then, the reload() method within fragment is called to update the listview thus the new list with new added records is displayed.

Besides, a GlobalUtil class is included in the program whose job is to manage resources used in the program. With the help of this class, there is no need to initialize database helper every time once the data in the database is accessed.

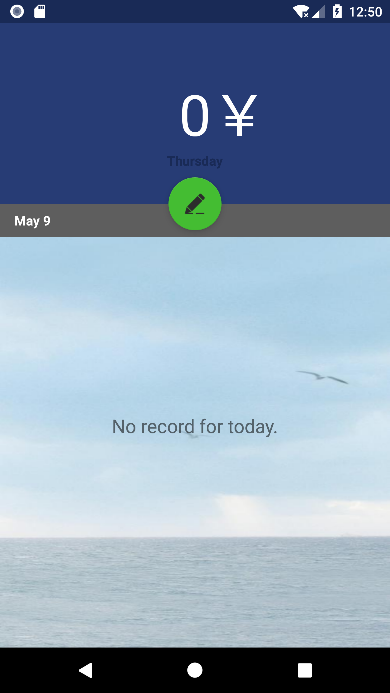
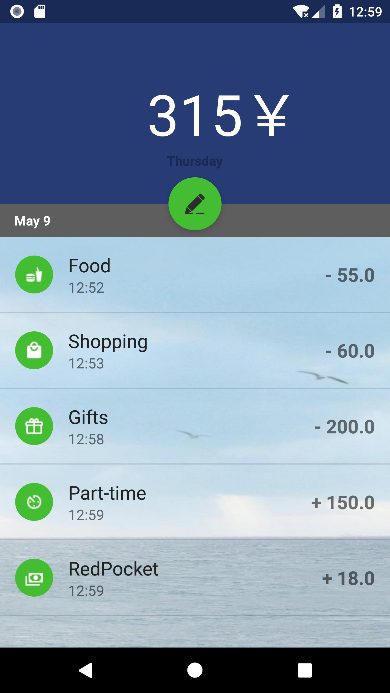
**Graphics**

**Class Diagram**

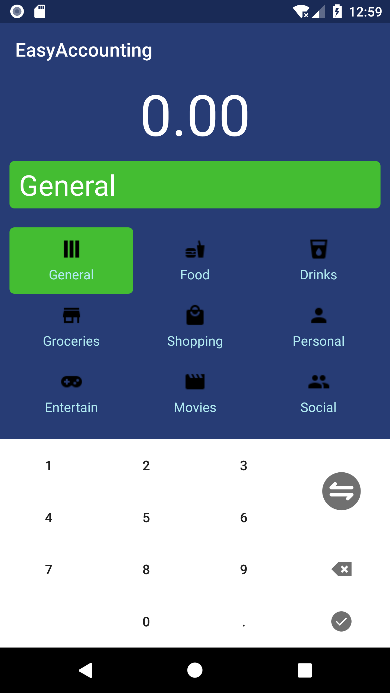
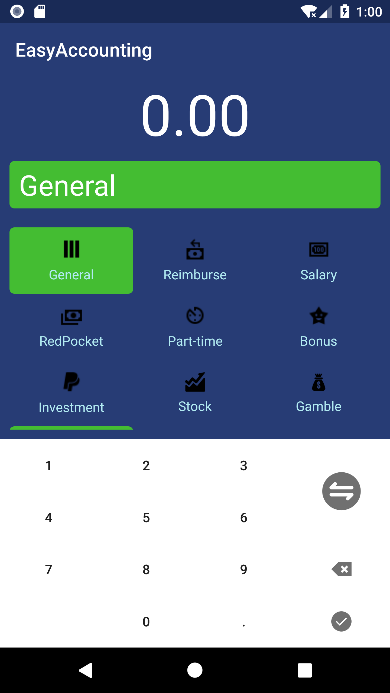


*Figure6 Class Diagram*

**GUI Design**

*Figure7. Main menu*  *Figure8. Main menu with records*

*Figure9. Adding records for expenditure Figure10. Adding records for income*

**Pseudo code**

This program will allow the user to add record, edit record and delete record for every day’s consumption.

**MainActivity.java**

Function onCreate(){

Set OnClickListener for adding record button

If (adding record button is clicked)

addRecordActivity started

Initialize viewpager, amountText

DateText ← DateUtil.java

end

}

**AddRecordActivity.java**

Function handleDot(){

Set OnclickListener

If (userput does not contain ‘.’)

Add ‘.’ to userInput

End

}

Function handleTypeChange(){

Set OnClickListener

If (type equals expense)

type←income

else

type←expense

end if

End

}

Function handleBackspace(){

Set OnClickListener

userInput←userInput.length -1

update AmountText

End

}

Function handleDone(){

Set OnClickListener

Change userInput to double

record←userInput

databaseHelper←record

reload viewpager

End

}

Function updateAmountText(){

If userInput contains ‘.’

If no digit after ‘.’

amountText←userInput+”00”

if 1 digit after ‘.’

amountText←userInput+”0”

else

amountText←userInput

if userInput does not contain ‘.’

if no input

amountText←”0.00”

else

amountText←userInput+”.00”

End

}

**RecordDatabaseHelper.java**

Function onCreat(SQLiteDatabase){

Create database with id, uuid, type, category, remark, amount, time, date

end

}

Function addRecord(Recordbean){

Create instance of database

Create ContentValues to pass data to database

ContentValues ← Recordbean.getUuid

ContentValues ← Recordbean.getType

ContentValues ← Recordbean.getCategory

ContentValues ← Recordbean.getRemark

ContentValues ← Recordbean.getAmount

ContentValues ← Recordbean.getDate

ContentValues ← Recordbean.getTime

Database ← ContentValues

end

}

Function removeRecord(uuid){

Delete data according to uuid

end

}

Function editRecord(uuid,RecordBean){

Delete data according to uuid

Set uuid to Recordbean

Call function addRecord

end

}

Function readRecords(date){

Create linkedlist for RecordBean

Query the database according to date

Do

Obtain data from database to recordBean

linkedlist ← recordBean

While (there is next record)

Return linkedlist

end

}

**MainFragment.java**

Constructor MainFragment(date){

Get data from database according to date

end

}

Function onCreateView(Layoutinflater){

Set layout for fragment

end

}

Function reload(){

Get data from database

If (listviewAdpater is null)

Create new ListViewAdapter

ListViewAdapter ← records

If (the number of records larger than 0)

Set “No record for today” invisible

end

}

Function getTotalCost(){

Total cost += record.getAmount()

end

}

Function onItemLongClick(){

If (long clicked)

Show dialog to choose to remove or edit

end

}

Function showDialog(index){

If (remove is chosen)

Get uuid

Databasehelp.remove(uuid)

Reload

Update the total cost

Else if (edit is chosen)

AddRecordActivity started

end

}

**MainViewPagerAdapter.java**

Function initFragment(){

Dates ← databaseHelper

Initialize the fragment ← Dates

end

}

Function getLastIndex(){

Return the size of fragments -1

}