AN INDUSTRIAL TRAINING REPORT

On

Expensta

Submitted by

Vanshaj Bhatia
Roll No: 141500474

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA University
Mathura- 281406, INDIA
May, 2016

Contents

Abstract

The era has come where we are adapting to new conditions of exchanging currency, people are preferring to digitize their money and expenditures instead of having a chunk of money in their wallets. This change welcomes a vast possibility of development of the nation but at the same time it also welcomes a lot of vulnerabilities in managing and accounting the money in hand. 'Expensta' is an application that gives you facility for maintaining your daily expenditures and organizing them in simple and flawless manner. This app is made so that User will not have to go to his/her wallet each time when he/she needs an account of his money in hand and total expenditure, with having adequate knowledge of each particular expenditure. The project 'Expensta' is developed to replace the currently existing system, which is orthodox method of monitoring one's money on its own. In the present era where "time" proves to be the most important asset for an individual by replacing the current register system to fully computerize, it not only saves the precious asset that is time, but also accuracy, reliability and uniformity can be maintained. This project is useful for the people belonging to any age group irrespective of the profession of the person, either they are common man or a business man of a giant MNC. 'Expensta' helps them to account their money faster than the orthodox approach. This project will make the task of documenting one's expenditure in an easy manner.

CERTIFICATE

This is to certify that the project entitled **"Expensta"** carried out in Industrial Training is a bonafide work done by **Vanshaj Bhatia (141500474)** carried out under the guidance of **Mr. Praveen Mittal, Assistant Professor, Training Department** and is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

Signature of Supervisor:

Name of Supervisor:

Date:

ACKNOWLDGEMENT

I wish to record my heartfelt gratitude and sincere thanks to our project guide **Mr. Praveen Mittal,** Asst. Professor, GLA University, Mathura for his kind support and inspiration given to me till the end of my project.

I thank **Dr. Anand Singh Jalal,** Head & Assoc. Professor, Department Of Computer Engineering and Applications, for his constant support and encouragement given throughout the development of the project.

Last but not the least our sincere thanks to our parents, family members and friends for their continuous support, inspiration and encouragement without which this project would not have been success.

Chapter 1

Introduction

1.1 Motivation and Overview

Expensta is an application of using technology to account the money people have with the total expenditure, despite of the orthodox barriers. The technology of digitizing the money accountancy has been available for years but the acceptance of it was quite recent. It is made up of 1 single applications, which runs on any android mobile phone. To start the functioning, the user should first decide the type of expenditure then they can practice two kinds of operation, credit some money (this will increase the balance counter of application) and debit some money (this will be entertained when an expenditure occur) and during the last one security measures were taken.

This application makes use of android database SQLite to store user data and Shared Preferences also. This application can be used for large scale and long term money management since it features the capabilities to take account of expenditure on a particular day, week or year, this feature increases the standard of co-operation. In addition, it converts the complex concept of encrypting the user data in a user-friendly environment since the security of one's money is a matter of great concern. This application has further potentials, such as category wise money management, developer feedback panel, database backup and auto backup option, user defined categories and various other tools options.

This android application provides solution to most of the shortcomings of the traditional money monitoring systems. The business man, students and the firm are profited evenly by the suggested system. This application plays a pivotal role in decreasing the management gap, so it can be termed as very significant for the daily users.

1.2 Objective:

The objective of this project is to give a platform to a user so that he/she can manage his/her expenditures in an efficient and flawless manner which would cater the needs of a paperless money management economy. To develop an instant expense manager solution to enable users to seamlessly get the account for their money savings problems by tracking the flow of money. The project should be very easy to use and riches in user friendly graphics enabling even a novice person to use it.

Chapter 2

Software Requirement Analysis

2.0 Company Profile

Bhatia Technologies is an educational institute, which deals with the learning of different programming languages and computing solutions. All of which brought together the best resources and expertise of industry experts in the Information Technology. Our Strength is that we are not dependent on any other external resources for Technical Training.

We Offer Web Development in HTML 5 Using Java, J2EE, .NET or PHP technologies with Dynamic websites for entrepreneurs. We Also Develope Mobile Apps in Android with Dynamic & Web Interfaces

We Offer Technical Training from Core topics like C, C++, Data Structures, Linux to latest technologies like J2SE 1.8 ,J2EE, Android, JSP , Servlets, Struts ,Spring , Hibernate, JSTL, AJAX, PHP with mysql, Oracle (SQL & PL/SQL), Apache Tomcat , .net, C#, ASP.net, VB.net , Mobile Programing in .net etc.

2.1 Define the problem

Users would like to have an application software wherein they can manage flow of money within their expenditure. This system will be designed to maximize the user's way of monitoring upon the currency. The software uses a highly interactive GUI setup and one way substitution cypher encryption which makes it very secure from outside attacks. More specifically, this system is designed to allow the user to manage

and secure the transaction history in a more effective manner. The system also contains a file management system containing some important information and categories related to the application which a user accesses at every point of time. A user can only use the application if he intends to take the account of the money either debited or credited.

This software should enable its user either student or any other novice person to credit or debit the money accounted. The constraint imposed here is that user should enter an amount and the category the expenditure or earning belongs to. A note option should be facilitated with every transaction which could be left blank if not required. Each transaction should be associated with the date of transaction.

Both credit and debit option must include a set of predefined categories in addition to it the app must also enable the user to add their own categories according to their requirements.

The app should facilitate the option of backing up all the data and transactions done by user just in case if the application uninstalled, the valuable data is preserved. It should inculcate two additional feature of auto-backup and backup into readable format.

The format of backup of data should be in encrypted format to minimize the security loop holes making the data completely inaccessible from outside.

A balance window should be placed in the main activity where the user can instantly know about their total 'money in hand'.

A developer feedback application is entertained where the users can rate the application, report the bugs or compliment the developer.

2. 2 Define the modules and their functionalities (SRS)

2.2.1 Product Perspective

The orthodox money management application consists of basic money accounting controls like credit, debit etc. This project contains additional features like database backup, user categories, encrypted architecture etc. which makes it more useful and efficient in real world scenarios.

2.2.2 Product Functions

This project performs various functions which includes private money management, category view, transaction history, editing previous transactions and information, user memo for each transaction etc.

2.2.3 Functional Requirements

2.2.3.1 Functional Requirement of Credit Module

2.2.3.1.1 Introduction

Credit refers to accounting the earnings done. This is done with the help of credit module activity which acts as an intermediate between the user and the data warehouse. It also entertains the quick categories where money came and a memo for quick revision.

2.2.3.1.2 Inputs

The inputs of this module are date, amount, category, memo/note of the transaction which can be done by android built in keyboard or any other virtual keyboard software.

2.2.3.1.3 Processing

The inputs of this module date, amount, category, memo/note of the transaction are sent to the table in database and stored there by for future retrieval.

2.2.3.1.4 Outputs

After processing stage is completed the user return to the main activity flashing a short message of 'INR <amount> Credited'.

2.2.3.1.5 Error Handling

Generally, this kind of functionality does not face any type of error, but a common technical fault may occur when the amount so entered by the user is invalid. The amount must only include a decimal as a symbol '.', and should not entertain alphabets or any other symbol. The note/memo should not include any of the two symbols '~' or '`' which are used as delimiter in internal data management.

2.2.3.2 Functional Requirement of Debit Module

2.2.3.2.1 Introduction

Debit refers to accounting the expenditures. This is done with the help of debit module activity which acts as an intermediate between the user and the data warehouse. It also entertains the quick categories where money came and a memo for quick revision.

2.2.3.2.2 Inputs

The inputs of this module are date, amount, category, memo/note of the transaction which can be done by android built in keyboard or any other virtual keyboard software.

2.2.3.2.3 Processing

The inputs of this module date, amount, category, memo/note of the transaction are sent to the debit table in database and stored there by for future retrieval.

2.2.3.2.4 Outputs

After processing stage is completed the user return to the main activity flashing a short message of 'INR <amount> Debited'.

2.2.3.2.5 Error Handling

Generally, this kind of functionality does not face any type of error, but a common technical fault may occur when the amount so entered by the user is invalid. The amount must only include a decimal as a symbol '.', and should not entertain alphabets or any other symbol.

The note/memo should not include any of the two symbols '~' or '`' which are used as delimiter in internal data management.

2.2.3.4 Functional Requirement of Balance View Activity

2.2.3.4.1 Introduction

Balance View activity is used to take a quick throwback to the history of transaction either credit or debit done during a particular period. i.e. either day, week, month or year.

2.2.3.4.2 Inputs

The inputs of this module are date/duration of transactions to be viewed and category which is used to view expenditures on a specific category. By default, the date is set to today and the category is set to 'All'. Selection of category and date can be done by simply selecting from the pop up and drop down menu viz.

2.2.3.4.3 Processing

No processing of data is done in this activity. Although if a user wants to edit or delete the transaction it may do it so by redirecting the selected transaction to the 'Edit Transaction Activity'

2.2.3.4.4 Outputs

The balance activity generates the history of credit and debit done in a defined duration in a scrollable list view format.

2.2.3.4.5 Error Handling

Generally, this kind of functionality does not face any type of error, but a common technical fault may occur when the database holds no data to display. Thereby leading the list views of both credit and debit to remain empty. Hence flashing a message of no transactions found, please select the appropriate duration/date.

2.2.3.5 Functional Requirement of Setting Activity

2.2.3.5.1 Introduction

Settings activity is used to set the auto backup option and adding/removing categories.

2.2.3.5.2 Inputs

The inputs of this module are selection of categories to be added or removed and activating the android switch to facilitate the auto-backup.

2.2.3.5.3 Processing

No processing of data is done in this activity.

2.2.3.5.4 Outputs

The settings activity redirects to the category edit activity if selected.

2.2.3.6 Functional Requirement of Backup/Restore Database/Export to File Activity

2.2.3.6.1 Introduction

This activity is used to back up the transaction on user storage area.

2.2.3.6.2 Inputs

No inputs are required in this module.

2.2.3.6.3 Processing

The data from database is fetched, encrypted and stored to user storage area.

2.2.3.6.4 Outputs

For successful completion, it flashes a message of completion of task.

2.2.3.6.5 Error Handling

Generally, this kind of functionality faces an error when the database hold no data to be backed up. So, it instead halts the operation and notifies the user to add some data.

2.2.4 Non-Functional Requirements

2.2.4.1 Performance

The application is completely build up on the API of Android v5.0 which is light in weight as compare to the other old orthodox API's by Google. Hence it makes performance efficient, enhanced response times, lower transaction time, and higher throughput, even in the cases where the database is holding accounted transactions of over and year old.

2.2.4.2 Reliability

Since the software is built up on the Android Studio using SQLite database as provided in Android architecture which serves as a prior technology. Hence the rate of failure increases generally due to delay/fault in the system database architecture and the device capabilities only. These errors can be corrected by use of adequate and efficient android devices which met the minimum possible requirements of the application.

2.2.4.3 Security

The software is totally secured by substitution cypher algorithm. Hence no outsider can anonymously access or manipulate the existing data.

2.2.4.4 Maintainability

The application will be enacted with OTA feature in its future version which will enable the users to receive updates and mod fixes in real time without deploying man power to the user site.

2.2.4.5 Portability

The application itself is fully portable and is ready to be deployed on any android, provided that the device met the minimum requirements.

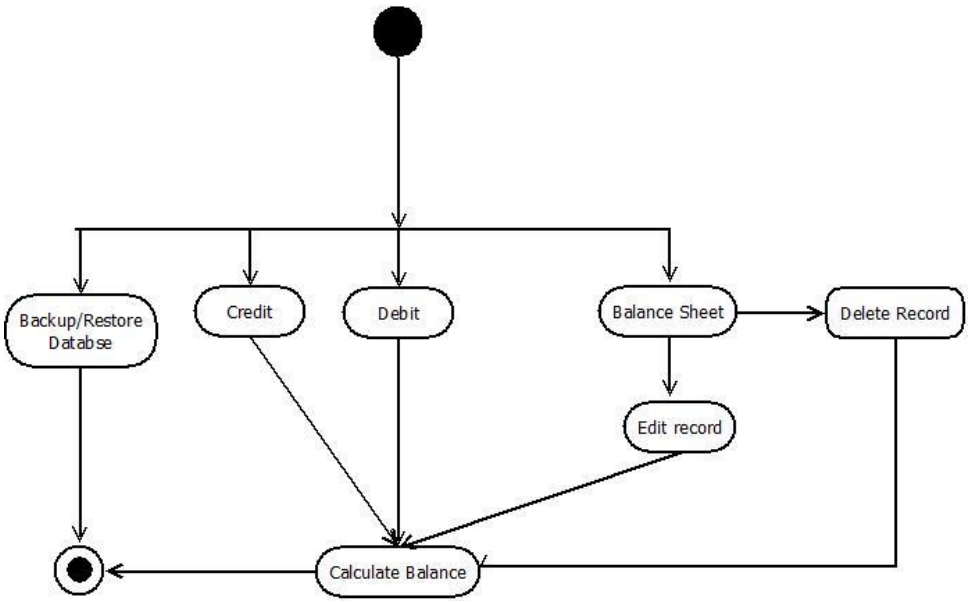
2.2.5 Design Constraints

There are no as such design constraints imposed by the authority. But certainly this application will be fully fledged with the best possible graphical units. Which would make the novice users experience more interactive.

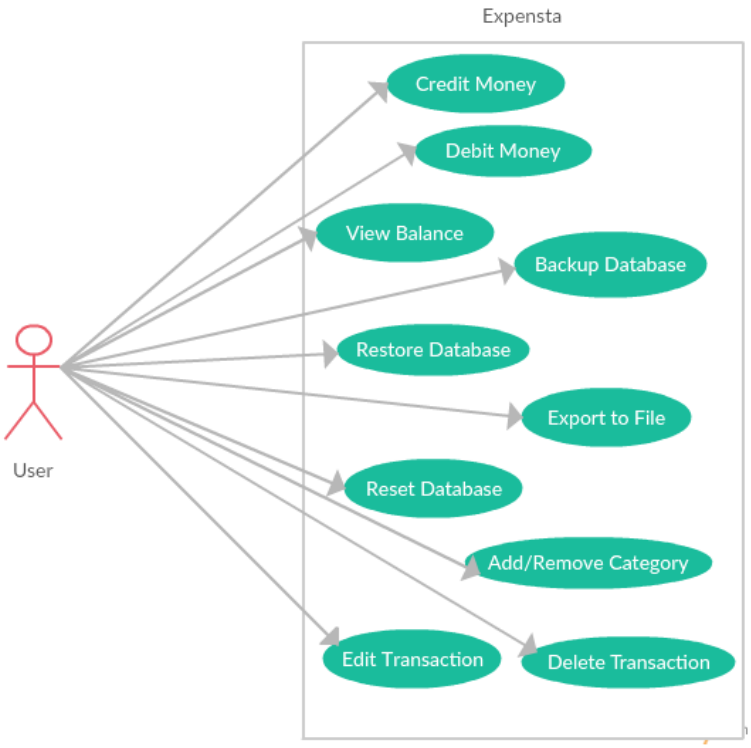
Chapter 3

Software Design

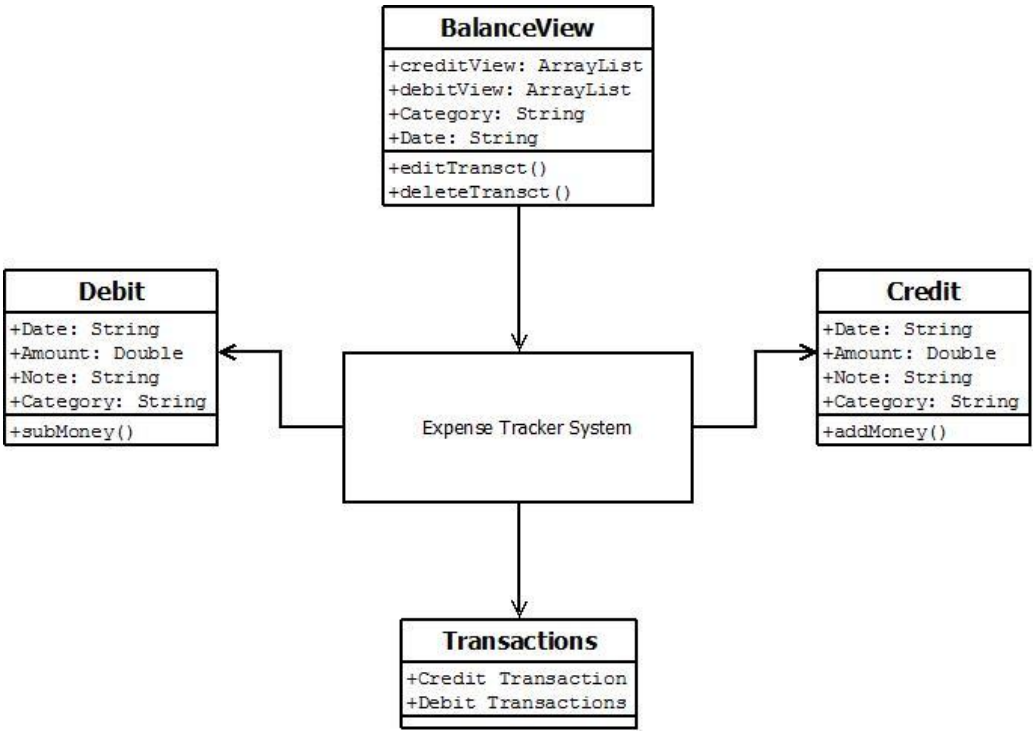
3.1 Flow Chart



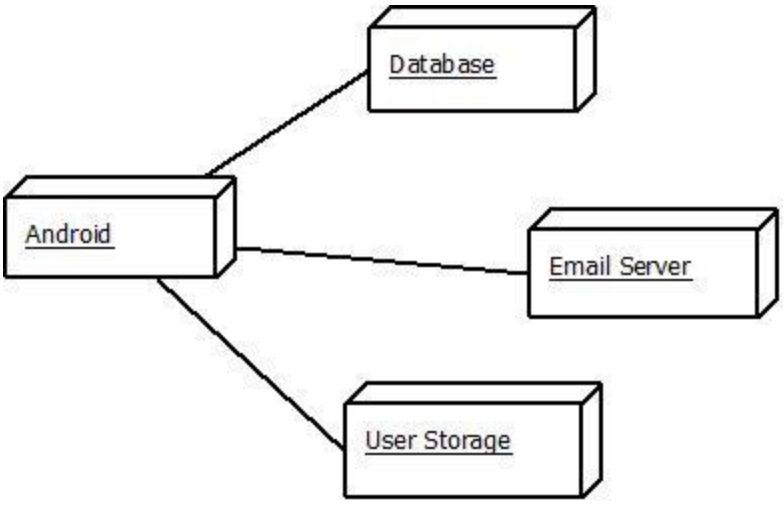
3.2 Use Case



3.3 Class Diagram



3.4 Deployment Diagram



Chapter 4

Testing

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code.

4.1 Testing Approaches

There are broadly two types of testing approaches:

Black-box testing

It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise.
In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.

Test-Cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily *functional* in nature, *non-functional* tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output, often with the help of an Oracle or a previous result that is known to be good, without any knowledge of the test object's internal structure.

White-box testing

It is conducted to test program and its implementation, in order to improve code efficiency or structure. It is also known as 'Structural' testing. In this testing method,

the design and structure of the code are known to the tester. Programmers of the code conduct this test on the code.

4.2 Testing Levels

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated and verified. Testing separately is done just to make sure that there are no hidden bugs or issues left in the software. Software is tested on various levels -

Unit Testing

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

Integration Testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passing and data updating etc.

System Testing

The software is compiled as product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- **Functionality testing** - Tests all functionalities of the software against the requirement.

- **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.

- **Security & Portability** - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

4.3 Test Cases

1. What type of currency are supported.

→ INR (Indian Rupees) are priory supported with initial version of application.

2. What is the error message received while trying to enter and invalid amount.

→ 'Invalid Amount'

3. Are the transactions done in shared format.

→ Yes, these can be backed up to user storage in readable format.

4. What happens while trying to back up empty database.

→ Error message of, 'No entries found'.

5. Is encryption also occurred in auto backup also.

→ Yes, thus making it unreadable from external sources.

6. Maximum length of memo.

→ 50 words can accommodate in transaction memo.

7. Are all transactions accounted editable.

→ Yes.

8. Can specific transactions be deleted.

→ Yes, it can be achieved in Balance View Activity.

9. Is there any option to backup data on Google Drive or OneDrive?

→ No, it occurs only on user storage.

10. Up to which date tractions can be made registered.

→ No restriction is imposed on date constraint. All dates supported by android system are entertained.

11. Maximum number of users accounts it can handle simultaneously.

→ A single account is handled at a time of operation only.

12. How feedback facility is provided?

→ The feedback of user is parsed into well-formed XML document and is then emailed to the developer automatically.

13.Error message if I enter symbols like '~' or '`' in memo or amount field.

→ " Invalid symbol '<symbol>' "

14. Are memo/notes also backed up.

→ No

15. Are the settings also backed up.

→ No, settings are not preserved.

16. Can I attach image of person also on the memo of transaction.

→ No, this facility is not indicted.

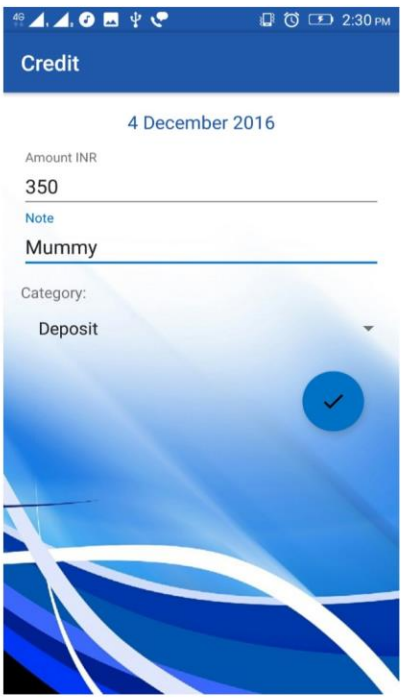
Chapter 5

Implementation and User Interface

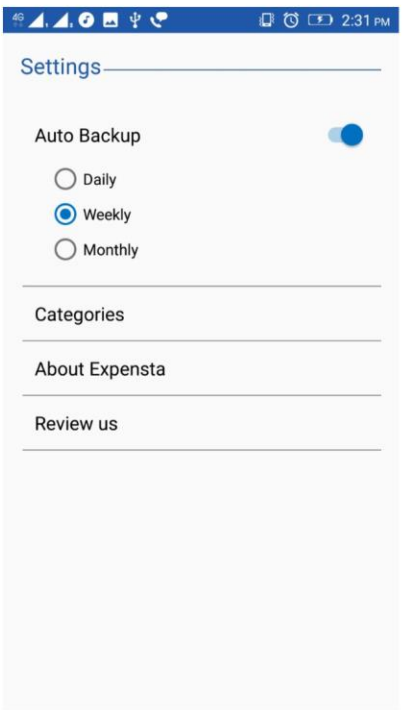
5.1 Screenshot of the project

Main Activity

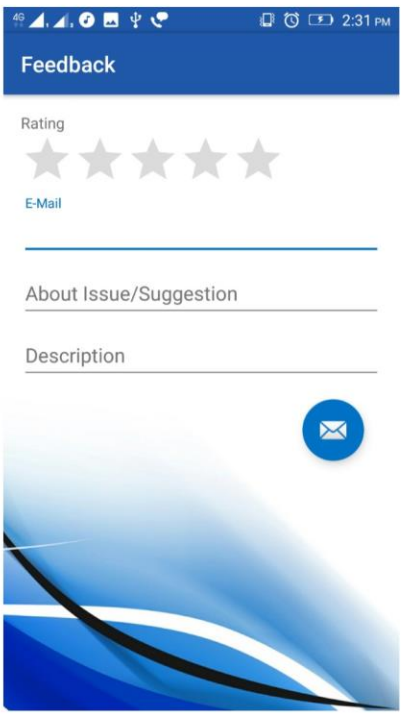
Credit Activity



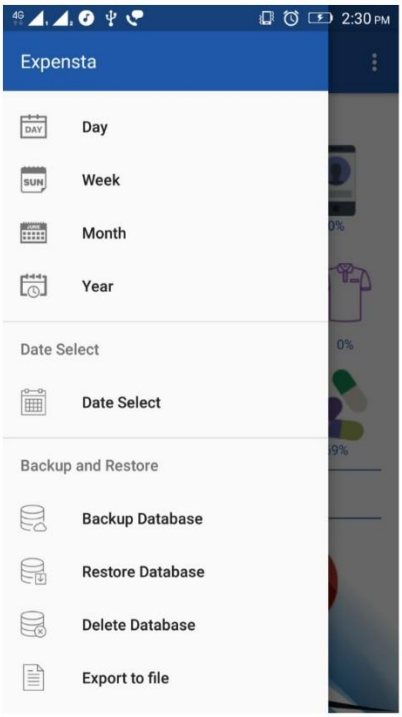
Settings Activity



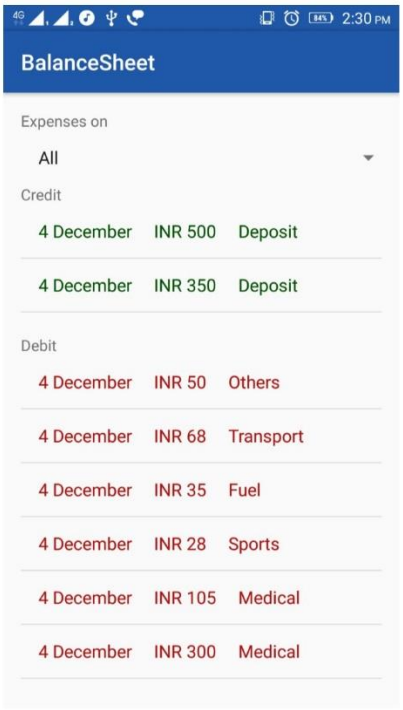
Feedback Activity



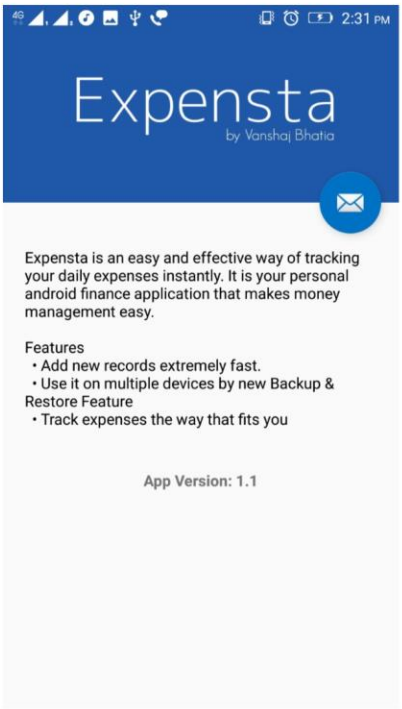
Navigation Bar



Balance View Activity



About Developer



5.2 Source code of the project

XML Files
content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:orientation="vertical"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="5dp"
    android:background="@drawable/back"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="apps.vanb.expensta.MainActivity"
    tools:showIn="@layout/app_bar_main">

    <LinearLayout
        android:layout_width="match_parent"
        android:orientation="vertical"
        android:layout_height="wrap_content">

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="#2158AA"
    android:id="@+id/lblDate"
    android:onClick="setDate"
    android:textAlignment="center"
    android:layout_marginBottom="15dp"
    android:textSize="20sp"
    />
<LinearLayout
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_height="wrap_content">
            <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:orientation="vertical">
        </LinearLayout>
                    <ImageView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="iconClickRent"
                android:id="@+id/rent"
                android:transitionName="Rent"
```

```
                android:src="@drawable/rent"/>
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/lblrent"
                style="@style/iconlbl"
                />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"

            android:orientation="vertical">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="iconClickCell"
                android:id="@+id/cell"
                android:transitionName="Cell"
                android:src="@drawable/cell_phone"/>
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/lblcellphone"
                style="@style/iconlbl"
                />
        </LinearLayout>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_marginBottom="15dp"
        android:layout_marginTop="15dp"
        >
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="2"

            android:orientation="vertical">
            <ImageView
                android:layout_width="match_parent"
                android:onClick="iconClickSports"
                android:layout_height="wrap_content"
                android:id="@+id/sports"
                android:padding="6dp"
                android:transitionName="Sports"
```

```
                android:src="@drawable/sports"/>
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/lblsports"
                style="@style/iconlbl"
                />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:orientation="vertical">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="iconClickClothes"
                android:transitionName="Clothes"
                android:id="@+id/clothes"
                android:padding="6dp"
                android:src="@drawable/clothes"/>
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/lblclothing"
                style="@style/iconlbl"
                />
        </LinearLayout>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        >
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:orientation="vertical">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="iconClickShopping"
                android:id="@+id/shopping"
                android:transitionName="Shopping"
                android:src="@drawable/shopping"/>
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/lblshopping"
```

```
                style="@style/iconlbl"
                />
        </LinearLayout>


            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/lblstudy"
                style="@style/iconlbl"
                />
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:orientation="vertical">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="iconClickMedical"
                android:id="@+id/medical"
                android:transitionName="Medical"
                android:src="@drawable/medical"/>
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/lblmedical"
                style="@style/iconlbl"
                />
        </LinearLayout>
    </LinearLayout>
</LinearLayout>

    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_marginTop="10dp"
        android:background="@color/colorPrimary"
        />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Balance: "
        android:layout_marginTop="10dp"
        android:layout_marginBottom="10dp"
        android:textAlignment="center"
        android:textSize="17sp"
        android:id="@+id/lblBalance"
        android:textColor="#2158AA"
```

```
        />
    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_marginBottom="10dp"
        android:background="@color/colorPrimary"
        />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_marginTop="5dp"
        >

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:id="@+id/credit"
            android:src="@drawable/plus"
            android:onClick="transct"
            />
        <ImageView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="transct"
            android:id="@+id/debit"
            android:src="@drawable/remove"/>
    </LinearLayout>
</LinearLayout>
    </ScrollView>
</LinearLayout>
```

content_credit_debit.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:orientation="vertical"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="@drawable/back2"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="apps.vanb.expensta.Credit"
    tools:showIn="@layout/activity_credit">
```

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        style="@style/toplbl"
        android:onClick="setDate"
        android:id="@+id/lblDate"
        />

    <android.support.design.widget.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Amount INR"
            android:inputType="numberDecimal"
            android:id="@+id/txtAmount"
            />
    </android.support.design.widget.TextInputLayout>

    <android.support.design.widget.TextInputLayout
        android:layout_width="match_parent"
        android:layout_marginBottom="10dp"
        android:layout_height="wrap_content">
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Note"
            android:id="@+id/txtNote"
            />
    </android.support.design.widget.TextInputLayout>


    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Category:"
        />
    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/sp1"
        >
    </Spinner>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
```

```
        android:layout_margin="@dimen/fab_margin"
        android:src="@drawable/ic_done" />

</LinearLayout>
```

content_balance_sheet.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:orientation="vertical"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="apps.vanb.expensta.BalanceSheet"
    tools:showIn="@layout/activity_balance_sheet">
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Expenses on"
    />
    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/spcat"
        ></Spinner>

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <LinearLayout
        android:layout_width="match_parent"
        android:orientation="vertical"
        android:layout_height="wrap_content">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Credit"
        />

    <ListView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:isScrollContainer="false"
        android:id="@+id/lv2">
    </ListView>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Debit"
        android:layout_marginTop="20dp"
        />
    <ListView
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:isScrollContainer="false"
        android:id="@+id/lv1">
    </ListView>
</LinearLayout>
    </ScrollView>
</LinearLayout>
```

Java Files:
MainActivity.java

```
package apps.vanb.expensta;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.AnimatorSet;
import android.annotation.TargetApi;
import android.app.AlarmManager;;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import java.sql.SQLClientInfoException;
import java.util.Calendar;

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {


    SQLiteDatabase db=null;
    ProgressBar mProgressView;
    TextView
lbltransport,lbleatingout,lblrent,lblcell,lblsports,lblclothes,lblshopping,
lblstudy, lblmedical,lblfuel,lblDate,lblBalance,lbltotCredit,lbltotDebit;
    View mainview;
    String[] d=new String[7];   String[] m=new String[7];   String[] y=new
String[7];       //for week calculation
    public String day,month,year,mode="day";
    ImageView clothes;


    @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR1)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
```

```
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        //User Coding here onwards
        clothes=(ImageView) findViewById(R.id.clothes);
        lbltransport=(TextView) findViewById(R.id.lbltransport);
        lbleatingout=(TextView) findViewById(R.id.lbleatingout);
        lblrent=(TextView) findViewById(R.id.lblrent);
        lblcell=(TextView) findViewById(R.id.lblcellphone);
        lblsports=(TextView) findViewById(R.id.lblsports);
        lblclothes =(TextView) findViewById(R.id.lblclothing);
        lblshopping=(TextView) findViewById(R.id.lblshopping);
        lblfuel=(TextView) findViewById(R.id.lblgasstation);
        lblstudy=(TextView) findViewById(R.id.lblstudy);
        lblmedical=(TextView) findViewById(R.id.lblmedical);
        lblDate=(TextView) findViewById(R.id.lblDate);
        lblBalance=(TextView) findViewById(R.id.lblBalance);
        lbltotCredit=(TextView) findViewById(R.id.lbltotCredit);
        lbltotDebit=(TextView) findViewById(R.id.lbltotDebit);
        mProgressView=(ProgressBar) findViewById(R.id.login_progress);
        mainview=(View) findViewById(R.id.mainview);
        try {

db=openOrCreateDatabase("apps.vanb.expensta.db",SQLiteDatabase.CREATE_IF_NE
CESSARY,null);
        }catch (Exception e)
        {
            Toast.makeText(MainActivity.this, ""+e.getMessage(),
Toast.LENGTH_SHORT).show();
        }


        String sql="create table if not exists credit(day text,month
text,year text, amount text, note text, category text, hash int, date
date)";
        db.execSQL(sql);
        sql="create table if not exists debit(day text,month text,year
text, amount text, note text, category text, hash int, date date)";
        db.execSQL(sql);
        sql="create table if not exists catdebit(category text)";
        db.execSQL(sql);


       // Toast.makeText(MainActivity.this, getHash(0)+" "+getHash(1) ,
Toast.LENGTH_SHORT).show();

        java.util.Calendar c = Calendar.getInstance();
        day = ""+c.get(Calendar.DAY_OF_MONTH);
        month =""+getMonth(c.get(Calendar.MONTH) + 1);
        year =""+c.get(Calendar.YEAR);
```

```
        Bundle b=getIntent().getExtras();
        try {
            String date = b.getString("date", null);
            mode=b.getString("mode",null);
            lblDate.setText(date);
        }catch(Exception e){}

        if(lblDate.getText().equals(""))
        {
            SharedPreferences prefs=getSharedPreferences("myprefs", MODE_
super.onBackPressed();
            SharedPreferences
prefs=getSharedPreferences("myprefs",MODE_PRIVATE);
            SharedPreferences.Editor ed=prefs.edit();
            ed.putString("mode",mode);
            ed.commit();
        }
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_feedback) {
            Intent i=new Intent(this, Feedback.class);
            startActivity(i);
        }
        if (id == R.id.action_devloper) {
            Intent i=new Intent(this, About.class);
            startActivity(i);
        }
        if (id == R.id.setting) {
            Intent i=new Intent(this, Settings.class);
            startActivity(i);
        }


        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
//navigation pane commandds here
        // Handle navigation view item clicks here.
```

```
        int id = item.getItemId();

       if (id == R.id.day) {
           mode="day";
           lblDate.setText(day+" "+month+" "+year);
           initIconLabel();

        }else if (id == R.id.week) {
           mode="week";
           initIconLabel();

       }else if (id == R.id.month) {
           mode="month";
           lblDate.setText(month+" "+year);
           initIconLabel();

        } else if (id == R.id.year) {
           mode = "year";
           lblDate.setText(year);
           initIconLabel();

       } else if (id == R.id.dateselect) {
           setDate(null);
        }else if (id == R.id.bkpdb) {
           backupDB();
        }else if (id == R.id.resdb) {
            restoreDB();
        }else if (id == R.id.deldb) {
           deleteDB();
       }else if (id == R.id.filedb) {
           fileDB();
       }


        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    //user funtions here onwards
    public void initIconLabel() {

        String date;
        String day,month,year;
        String r=null,sql=null;
        if(mode.equals("day")) {
            date= lblDate.getText().toString();
            String[] words = date.split(" ");
            day = words[0];
            month = words[1];
            year = words[2];
            r = "Select amount, category from debit where day='"+day+"' and
month='"+month+"' and year='"+year+"'";
            sql="select amount from credit where day='"+day+"' and
month='"+month+"' and year='"+year+"'";
        }

        else if(mode.equals("week"))
```

```
        {
            Calendar cal = Calendar.getInstance();
            int dw=1-cal.get(Calendar.DAY_OF_WEEK);
            //Toast.makeText(MainActivity.this, ""+dw,
            if (c.getString(1).equals("Fuel"))
                sumfuel += Double.parseDouble(c.getString(0));
            if (c.getString(1).equals("Study"))
                sumstudy += Double.parseDouble(c.getString(0));
            if (c.getString(1).equals("Medical"))
                summedical += Double.parseDouble(c.getString(0));
            sum += Double.parseDouble(c.getString(0));
        }
        lbltransport.setText("" + (int)(sumtransport * 100 / sum) + "%");
        lbleatingout.setText("" + (int)(sumeatingout * 100 / sum) + "%");
        lblrent.setText("" + (int)((sumrent * 100) / sum) + "%");
        lblcell.setText("" + (int)((sumcell * 100) / sum) + "%");
        lblsports.setText("" + (int)((sumsports * 100) / sum) + "%");
        lblclothes.setText("" + (int)((sumclothes * 100) / sum) + "%");
        lblshopping.setText("" + (int)((sumshopping * 100) / sum) + "%");
        lblfuel.setText("" + (int)((sumfuel * 100) / sum) + "%");
        lblstudy.setText("" + (int)((sumstudy * 100) / sum) + "%");
        lblmedical.setText("" + (int)((summedical * 100) / sum) + "%");

        Double sumDebit=sum;
        c=db.rawQuery(sql,null);
        while(c.moveToNext())
        {
            sumCredit+= Float.parseFloat(c.getString(0));
        }
        lbltotCredit.setText("INR "+sumCredit);
        lbltotDebit.setText("INR "+sumDebit);
        lblBalance.setText("Balance: INR "+(sumCredit-sumDebit));


    }

    public void setDate(View v)
    {
        DatePickerDialog.OnDateSetListener zz = new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view, int year, int
monthOfYear, int dayOfMonth) {
                monthOfYear++;
                if(mode.equals("day"))
                    lblDate.setText(dayOfMonth+" "+getMonth(monthOfYear)+"
"+year);
                else if(mode.equals("month"))
                    lblDate.setText(getMonth(monthOfYear)+" "+year);
                else if(mode.equals("year"))
                    lblDate.setText(""+year);
                else if(mode.equals("week"))
                {   mode="day";
                    lblDate.setText(dayOfMonth+" "+getMonth(monthOfYear)+"
"+year);
                }

                initIconLabel();
            }
```

```
        };
        java.util.Calendar c = Calendar.getInstance();
        int d = c.get(Calendar.DAY_OF_MONTH);
        int m = c.get(Calendar.MONTH);
        int y = c.get(Calendar.YEAR);
        DatePickerDialog dpp = new DatePickerDialog(MainActivity.this, zz,
y, m, d);
        dpp.show();


    }

    public int getHash(int i)
    {   String sql=null;
        if(i==0)    //debit
         sql="Select * from debit";
        else if(i==1)   //credit
            sql="Select * from credit";
        Cursor cu=db.rawQuery(sql,null);
        if(cu.moveToNext()==false)
            return 0;
        else
        {
            while(cu.moveToNext());
            cu.moveToPrevious();
            int j=Integer.parseInt(cu.getString(6));
            return ++j;
        }
    }

    public void transct(View v) {
        switch (v.getId()) {
            case R.id.credit:
                Intent i = new Intent(this, Credit.class);
                i.putExtra("date", lblDate.getText().toString());
                i.putExtra("mode", mode);
                startActivity(i);
                finish();
                break;
            case R.id.debit:
                Intent j = new Intent(this, Debit.class);
                j.putExtra("date", lblDate.getText().toString());
                j.putExtra("mode", mode);
                startActivity(j);
                finish();
                break;
            case R.id.lbltotCredit:
                Intent k = new Intent(this, BalanceSheet.class);
                k.putExtra("mode", mode);
                k.putExtra("date", lblDate.getText().toString());
                startActivity(k);
                finish();
                break;
            case R.id.lbltotDebit:
                Intent l = new Intent(this, BalanceSheet.class);
                l.putExtra("mode", mode);
                l.putExtra("date", lblDate.getText().toString());
                startActivity(l);
                finish();
```

```
                break;
            }
        }

show();
                        Intent i = new Intent(MainActivity.this,
MainActivity.class);
                        i.putExtra("date", lblDate.getText().toString());
                        i.putExtra("mode", mode);
                        startActivity(i);
                        MainActivity.this.finish();


                    }
                })
                .show();
    }

    public void deleteDB()
    {
        new AlertDialog.Builder(this)
            .setTitle("Caution")
            .setMessage("This will remove all your current data. Are you
sure you want to continue?")
            .setNegativeButton("No", null)
            .setPositiveButton("Yes", new DialogInterface.OnClickListener()
{
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    String sql;
                    sql = "delete from credit";
                    db.execSQL(sql);
                    sql = "delete from debit";
                    db.execSQL(sql);
                    Toast.makeText(MainActivity.this, "Database Reset
Completed", Toast.LENGTH_SHORT).show();
                    Intent i = new Intent(MainActivity.this,
MainActivity.class);
                    i.putExtra("date", lblDate.getText().toString());
                    i.putExtra("mode", mode);
                    startActivity(i);
                    MainActivity.this.finish();
                }
            })
            .show();
    }

    String date1,date2=null;
    public void fileDB()
    {

        int flag1=0,flag2=0;
        try{
            String sql, records = "";
            sql = "select * from credit";
            Cursor c = db.rawQuery(sql, null);
            if(c.moveToNext()==false)
                flag1=1;
            sql = "select * from debit";
```

```
            c=db.rawQuery(sql,null);
            if(c.moveToNext()==false)
                flag2=1;

            if(flag1==1 && flag2==1) {
                Toast.makeText(MainActivity.this, "No Entries Found :(",
Toast.LENGTH_SHORT).show();
                return;
            }

            File sd = Environment.getExternalStorageDirectory();
            File dir = new File(sd.getAbsolutePath() + "/Expensta");
            dir.mkdir();
            File file = new File(dir, "expensta_record.txt");
            FileOutputStream fos = new FileOutputStream(file);
            OutputStreamWriter osw = new OutputStreamWriter(fos);
            sql = "select * from credit";
            c = db.rawQuery(sql, null);
            records="Date\t\tAmount  Category   Note";
            records+="\nCredit\n";
            while (c.moveToNext()) {
                records += c.getString(0)+"-"+c.getString(1)+"-
"+c.getString(2)+"\t"+c.getString(3)+"\t"+c.getString(5)+"\t
"+c.getString(4);
                records = records + "\n";
            }

            records = records + "\nDebit\n";
            sql = "select * from debit";
            c = db.rawQuery(sql, null);
            while (c.moveToNext()) {
                records += c.getString(0)+"-"+c.getString(1)+"-
"+c.getString(2)+"\t"+c.getString(3)+"\t"+c.getString(5)+"\t
"+c.getString(4);
                records = records + "\n";
            }

            osw.write(records);
            osw.flush();
            osw.close();
            Toast.makeText(MainActivity.this, "File created at " + dir,
Toast.LENGTH_LONG).show();
        } catch (Exception e) {
            Toast.makeText(MainActivity.this, "" + e.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    }




}
```

Chapter 6

References

- http://www.tutorialpoints.com/ Swing/
- Standard SRS Template of IEEE Std 730-1998
- Mini Project format (GLA University)
- http://www.wikipedia.org/
- http://www.stackoverflow.com/
- www.theserveside.com/tutorial/