Personal Budget Manager Android Application

K.P. Upekka Anuruddhi
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka
upekka.12@cse.mrt.ac.lk

Abstract— **People have unlimited needs and wants to be fulfilled with a limited income. It is important to manage the expenses to match the level of income. It is convenient for many people to record their transaction details in their personal smart-phones. This paper discusses the design and implementation of an Android application called "Personal Budget Manager" which can assist an individual to manage his/her daily transactions by recording and analyzing the transaction information. This system is developed based on the Rational Unified Process. Android Studio which is an integrated development environment for the Android platform was used for the development of the Personal Budget Manager application.**

Keywords- Budget Management; User Interface Design; Mobile applications

I. INTRODUCTION

Phone is no longer just a communication tool, but also an essential part of the people's communication and daily life. Android has dominated the smart-phone Operating System market with a share of 82.8% [1].

There is a big market base for any user friendly Android app that meets the user requirements. Personal transaction management is a requirement of many individuals. The individuals can easily use the app to record their daily transactions without any delay as a phone can be used almost anywhere. The app is user friendly and easily learn-able by any individual with basic knowledge.

The main purposes of the Personal Budget Manager Android app are recording and analyzing daily transactions of an individual. The user can get an overview of his/her transactions and the monthly balance of money remaining. The user can plan for the future and manage future expenses according to the calculated information provided by the application. The analysis tools in the app help the user to get the overall idea about the level of savings or losses while comparing with other time periods.

Android Studio 1.2.2 was used for the development of the Personal Budget Manager Android application. SQLite which is a relational database management system that is embedded into the end program was used to implement the underlying database of the system. Visual Paradigm, a software design tool tailored for agile software projects was used throughout the project. Robotium which is an Android test automation framework was used for testing the application.

The outcome of this project is the mobile application that can work in any Android device that meets the minimum Android API level. The app was developed with usability, response and accuracy in mind. The rest of this paper discusses and analyses the existing systems for personal budget management and the system models of the developed Personal Budget Manager Android Application.

II. LITERATURE REVIEW

Personal income/expense management is a problem faced by a large number of people around the world. So, many developers have tried to develop different expense management applications. 'Spending Tracker', 'Goodbudget ', 'Money Manager Expense & Budget' etc. are some of the Android applications which are available in the Google play app store.

Spending Tracker is an app developed by MH Riley Ltd which is used to keep track of expenses [2]. Goodbudget is a money manager for budget planning and personal finance management developed by Dayspring Technologies. Goodbudget is an expense & budget tracker that allows to pro-actively plan finances ahead of time with the Envelope system of budgeting [3]. Money Manager is a financial planning, review, expense tracking and personal asset management app developed by Realbyte Inc [4]. A common problem with these Android apps is the complexity. For example in Spending Tracker and Goodbudget the user has to fill more than six fields when doing a transaction entry. It takes a considerable amount of valuable time of the user to fill those complex forms. Money Manager app provides good functionality, but it does not provide quick access to important features. It is important that the user can easily record the transaction information within few minutes and quickly access the analysis tools from the home screen.

The project focused on developing an Android application which assists an individual to manage his personal transactions and budgets. Android applications are developed using the Java language. Java is an object-oriented language designed to be platform-independent and secure, using virtual machines. Android relies heavily on these Java fundamentals. The Android SDK (Software Development Kit) includes many standard Java libraries (data structure libraries, math libraries, graphics libraries, networking libraries) as well as special Android libraries that help develop Android applications [5].

Android system is a Linux-based system, as shown in Fig. 1, the Android architecture consists of four layers: Linux kernel, Libraries and Android runtime, Application framework and Applications. Each lower layer provides a sort of encapsulation, while providing call interface to the upper layers [6].

SQLite database was used to store data in the Personal Budget Manager Android application. SQLite is an open-source embedded database, which is written in C language by D. Richard Hipp. Compared with the usual databases such as SQL

Server, Oracle etc., SQLite is a lightweight database and it realizes a complete and embeddable database engine without additional components, it is especially suitable for embedded applications [7].

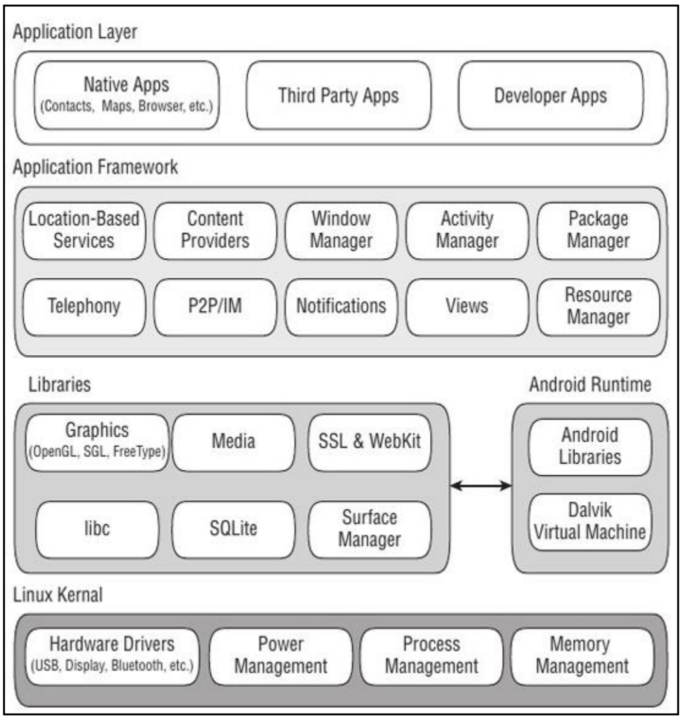


Fig. 1. Android Architecture

The Unified process of system development was followed in doing the project on developing the Personal Budget Manager Android application. Unified process (UP) is an architecture-centric, use-case driven, iterative and incremental development process that leverages unified modeling language [8]. Unified process can be applied to different software systems with different levels of technical and managerial complexity across various domains and organizational cultures. Unified process is an extensible framework that can be customized for specific projects. This process divides the development process into four phases: Inception, Elaboration, Conception and Transition.

III. SYSTEM MODELS

A. *System Requirement*

The main functionality of the Personal Budget Manager Android App included recording user's transactions and analyzing them to give useful information to the user to make important decisions. The user is able to insert his/her daily income and expenses under relevant categories. The user can categorize his/her transactions under the default categories such as salary (income), food, transport, health, household (expenses) etc. or add new categories. Also the app maintains transaction history. Graphical representations are used to compare budgets against real income/expenses. The user can get an overview of his/her income and expenses by looking at the statistical reports generated by the system. Graphical representations make it easy to be understood by many users. Also the user can sort and filter transactions according to different categories.

The overview of the system can be represented using the use case diagram shown in Fig. 2. The Use Case View describes the set of scenarios or use cases that represent significant functionality of the system.

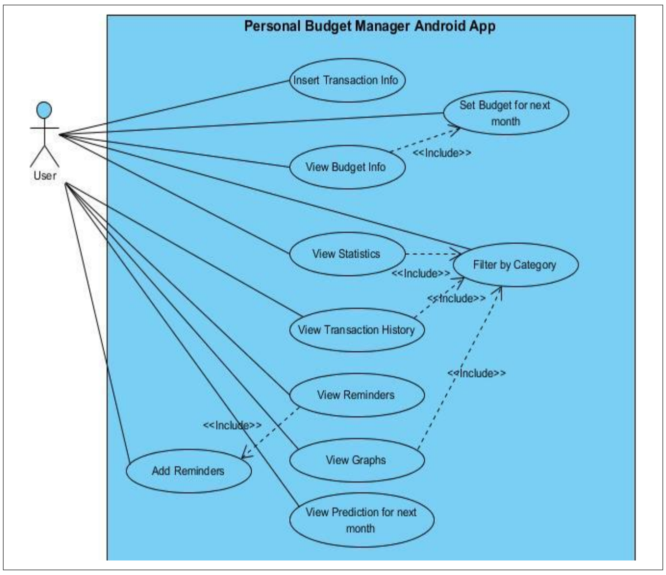


Fig. 2. Use case diagram

When considering usability requirements, user training time should be less, the app should provide quick access to more important features, and the app should be developed conforming to common usability standards given by the Google's Android Developer Guidelines [9]. Further, the system should have high availability, accuracy and Mean Time Between Failures (MTBF). The Performance of the app should be at a considerable level to give a quick response time.

B. *System Design*

Layered architecture was used in developing the Personal Budget Manager application. In layered architecture the system is organized into layers with related functionality associated with each layer. A layer provides services to the layer above it so the lowest-level layers represent core services that are likely to be used throughout the system.

The lowest layer includes database access. The classes such as TransactionDA, CategoryDA, BudgetDA etc. provide data access facilities specific to relevant Tables in the SQLite database. The DBHelper class establish the database connection and creates all the tables in database. The next layer is the application layer that includes the components concerned with the application functionality and utility components that are used by other application components. The top layer is providing user interface facilities. The classes BudgetManagerHome, CashIn, CashOut, Budget etc. provide services to the UI layer which is in XML (EXtensible Markup Language).

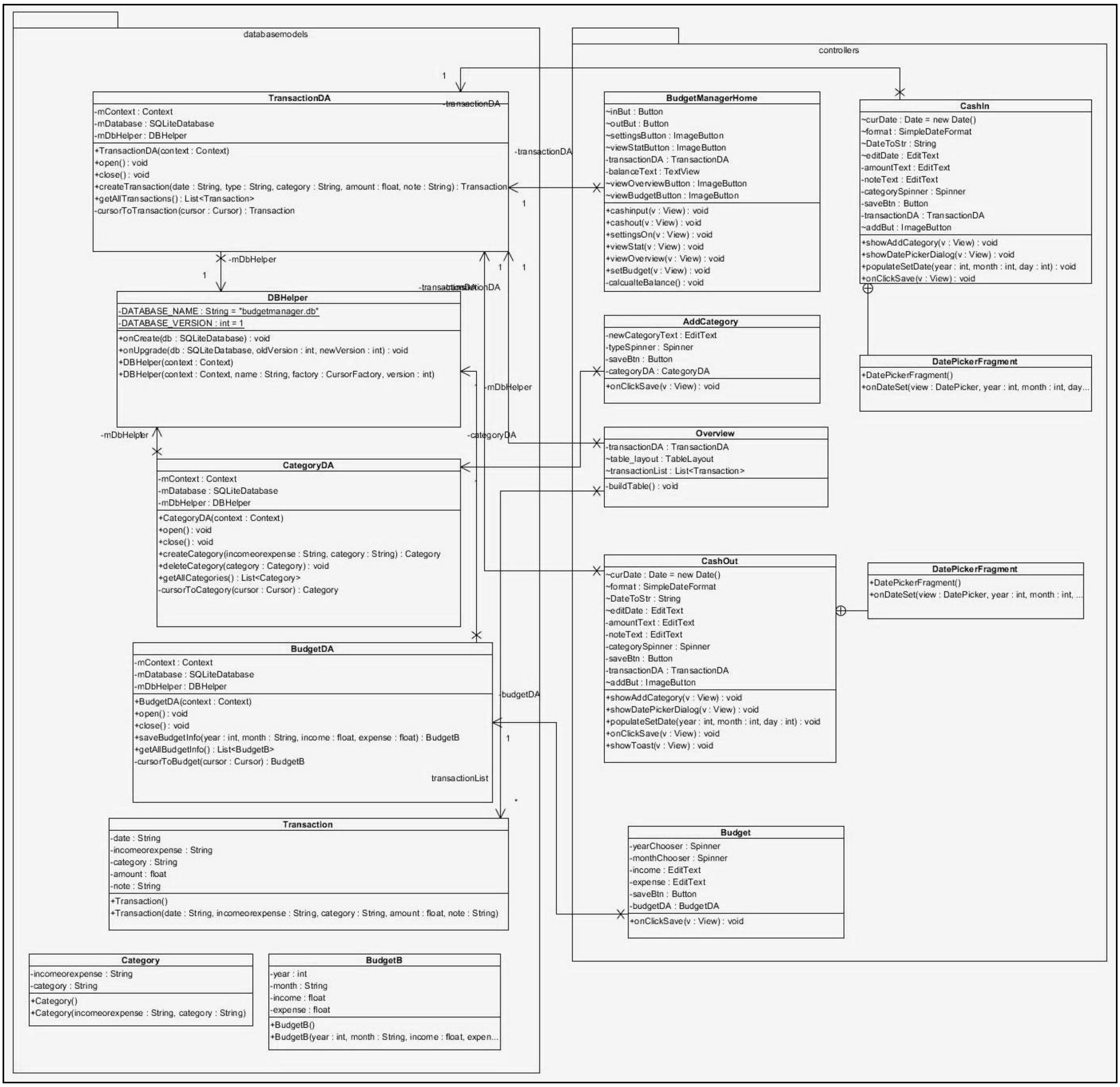

databasemodels
controllers
TransactionDA
-mContext : Context
-mDatabase : SQLiteDatabase
-mDbHelper : DBHelper
+TransactionDA(context : Context)
+open() : void
+close() : void
+createTransaction(date : String, type : String, category : String, amount : float, note : String) : Transaction
+getAllTransactions() : List<Transaction>
-cursorToTransaction(cursor : Cursor) : Transaction
DBHelper
-DATABASE_NAME : String = "budgetmanager.db"
-DATABASE_VERSION : int = 1
+onCreate(db : SQLiteDatabase) : void
+onUpgrade(db : SQLiteDatabase, oldVersion : int, newVersion : int) : void
+DBHelper(context : Context)
+DBHelper(context : Context, name : String, factory : CursorFactory, version : int)
CategoryDA
-mContext : Context
-mDatabase : SQLiteDatabase
-mDbHelper : DBHelper
+CategoryDA(context : Context)
+open() : void
+close() : void
+createCategory(incomeorexpense : String, category : String) : Category
+deleteCategory(category : Category) : void
+getAllCategories() : List<Category>
-cursorToCategory(cursor : Cursor) : Category
BudgetDA
-mContext : Context
-mDatabase : SQLiteDatabase
-mDbHelper : DBHelper
+BudgetDA(context : Context)
+open() : void
+close() : void
+saveBudgetInfo(year : int, month : String, income : float, expense : float) : BudgetB
+getAllBudgetInfo() : List<BudgetB>
-cursorToBudget(cursor : Cursor) : BudgetB
transactionList
Transaction
-date : String
-incomeorexpense : String
-category : String
-amount : float
-note : String
+Transaction()
+Transaction(date : String, incomeorexpense : String, category : String, amount : float, note : String)
Category
-incomeorexpense : String
-category : String
+Category()
+Category(incomeorexpense : String, category : String)
BudgetB
-year : int
-month : String
-income : float
-expense : float
+BudgetB()
+BudgetB(year : int, month : String, income : float, expen...
BudgetManagerHome
~inBut : Button
~outBut : Button
~settingsButton : ImageButton
~viewStatButton : ImageButton
-transactionDA : TransactionDA
-balanceText : TextView
~viewOverviewButton : ImageButton
~viewBudgetButton : ImageButton
+cashinput(v : View) : void
+cashout(v : View) : void
+settingsOn(v : View) : void
+viewStat(v : View) : void
+viewOverview(v : View) : void
+setBudget(v : View) : void
-calcualteBalance() : void
CashIn
~curDate : Date = new Date()
~format : SimpleDateFormat
~DateToStr : String
~editDate : EditText
-amountText : EditText
-noteText : EditText
-categorySpinner : Spinner
-saveBtn : Button
-transactionDA : TransactionDA
~addBut : ImageButton
+showAddCategory(v : View) : void
+showDatePickerDialog(v : View) : void
+populateSetDate(year : int, month : int, day : int) : void
+onClickSave(v : View) : void
DatePickerFragment
+DatePickerFragment()
+onDateSet(view : DatePicker, year : int, month : int, day...
AddCategory
-newCategoryText : EditText
-typeSpinner : Spinner
-saveBtn : Button
-categoryDA : CategoryDA
+onClickSave(v : View) : void
Overview
-transactionDA : TransactionDA
~table_layout : TableLayout
~transactionList : List<Transaction>
-buildTable() : void
CashOut
~curDate : Date = new Date()
~fomat : SimpleDateFormat
~DateToStr : String
~editDate : EditText
-amountText : EditText
-noteText : EditText
-categorySpinner : Spinner
-saveBtn : Button
-transactionDA : TransactionDA
~addBut : ImageButton
+showAddCategory(v : View) : void
+showDatePickerDialog(v : View) : void
+populateSetDate(year : int, month : int, day : int) : void
+onClickSave(v : View) : void
+showToast(v : View) : void
DatePickerFragment
+DatePickerFragment()
+onDateSet(view : DatePicker, year : int, month : int, ...
Budget
-yearChooser : Spinner
-monthChooser : Spinner
-income : EditText
-expense : EditText
-saveBtn : Button
-budgetDA : BudgetDA
+onClickSave(v : View) : void
-transactionDA
-transactionDA
-transactionDA
-mDbHelper
-mDbHelper
-mDbHelper
-categoryDA
-budgetDA
1
1
1
1
1
1
*

Fig. 3. Class Diagram

Fig. 4 and Fig. 5 show the activity diagrams for inserting transaction information and viewing statistical information, respectively. When the user presses the ‘INCOME’ button in the Home page, The ‘Cash In’ interface is displayed. The user can select the date by the date chooser. (Current date is set by default.) Then the user can select the category and enter the amount. He/she can add notes if required. Then if the ‘Save’ button is pressed the values will be inserted to the database. The current balance value will be automatically updated when any transaction is recorded. All the text fields are validated such that invalid data cannot be entered. When the user presses the ‘View Statistics’ button, he/she is navigated to the relevant interface. The user can filter the transaction information according to his/her requirement.

SQLite databases are used for saving data to a database on Android. The APIs needed to use a database on Android are available in the android.database.sqlite package. ER diagram for the system database is shown in Fig. 6 and the most important table in the database is Transaction Table. All the income and expense transactions are recorded in that table. Also it is very frequently used for calculations required in the application. Category table saves the list of categories belonging to incomes and expenses. The user can add new categories. This data is used to populate Spinners in the user interfaces. Budget Table data is used for comparing actuals with budgets.

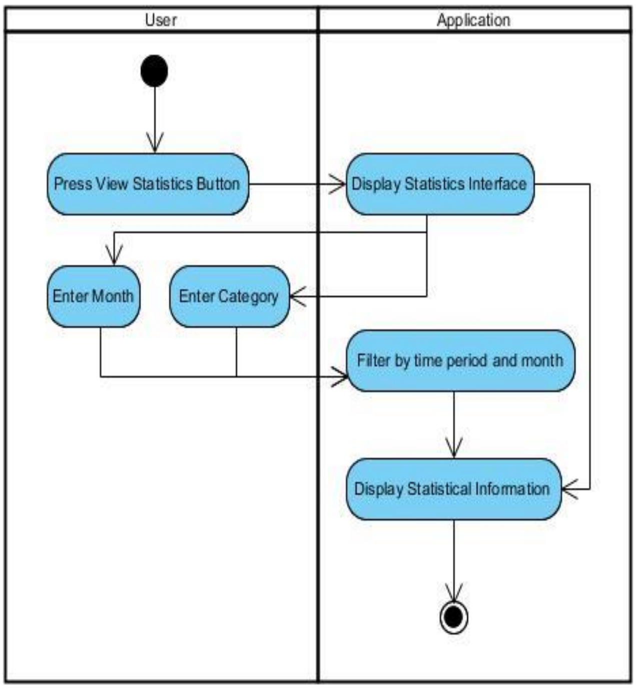

User
Application
Press View Statistics Button
Display Statistics Interface
Enter Month
Enter Category
Filter by time period and month
Display Statistical Information

Fig. 4. Activity diagram for inserting transaction information.

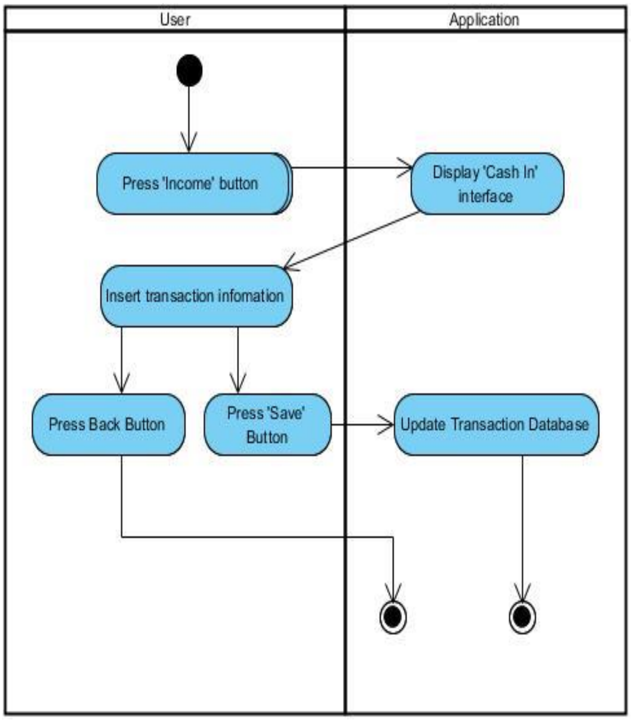

User
Application
Press 'Income' button
Display 'Cash In' interface
Insert transaction infomation
Press Back Button
Press 'Save' Button
Update Transaction Database

Fig. 5. Activity diagram for viewing statistical information.

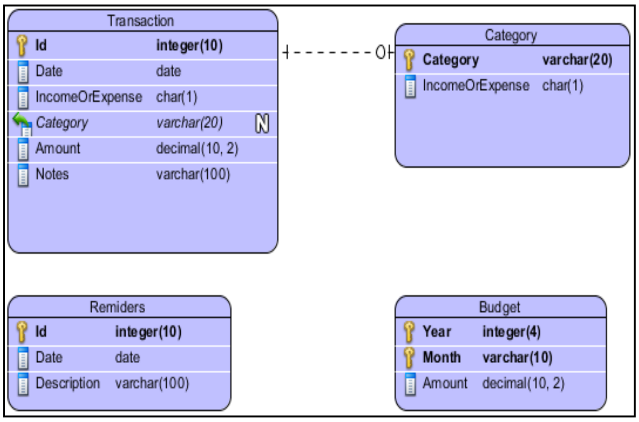


Fig. 6. Entity Relationship Diagram

IV. SYSTEM IMPLEMENTATION

A. *Implementation Procedure*

Android Studio 1.2.2 was used for the development of the Personal Budget Manager Android application. SQLite which is a relational database management system that is embedded into the end program was used to implement the underlying database of the system.

Visual Paradigm, a software design tool tailored for agile software projects was used throughout the project. All the designing activities related to the project were done by means of this tool. Visual Paradigm was used to draw Use Case Diagram, Activity Diagrams, Class Diagram, ER Diagram, etc. Smartsheet which is an intuitive on-line project management tool, was used to prepare the Gantt chart and Project Schedule [10]. Balsamiq Mockups which is a rapid wire framing tool was used to design the user interfaces [11]. It reproduced the experience of sketching on a white-board using a computer.

The Unified process of System Development Life Cycle (SDLC) was used in developing this project. There are four phases in the Unified process namely Inception, Elaboration, Construction and Transition [4]. There were two development iterations in the project. Implementation of the basic functionality and creating the required database was done in the first iteration. Then the additional features were implemented in the second iteration. The application was continuously tested from the beginning. Robotium which is an Android test automation framework was used for testing the application [12].

The default categories for income and expenses were saved in the database. The user has the ability to add new categories according to his/her choice after the app is installed. MPAndroidChart chart library [13] was used to draw the graphs in the application. MPAndroidChart is a powerful & easy to use chart library for Android, supporting line, bar, scatter and pie charts as well as scaling, dragging, selecting and animations. It works on Android 2.2 (API level 8) and upwards.

B. Algorithms

Fig. 7, shows the pseudocode for calculating current balance. The list of all relevant transactions is taken to an ArrayList. Then each transaction is checked whether it is an income or expense and added or subtracted from the balanceAmount accordingly.

```
Initialize balanceAmount to zero
Initialize transactionList as an ArrayList

FOR i=1 TO length of transactionList
        IF transactionList item is an income
                add amount to balanceAmount
        ELSE IF transactionList item is an expense
                subtract amount from balanceAmount
END of loop
Display amount
```

Fig.7. Pseudocode for calculating current balance.

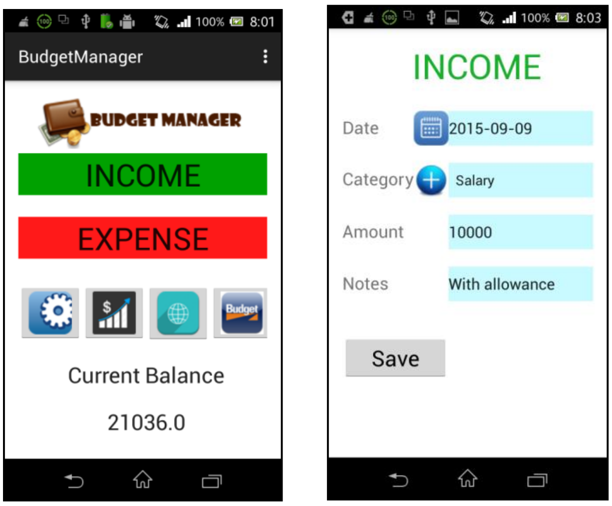
```
Initialize transactionList as an ArrayList
transactionList equals getAllTransactions()

FOR i=1 TO length of transactionList
        Create a new TableRow
        row Text = date + type + category + amount
        set row Text
        add TableRow to Table Layout
END of loop
```

Fig.8. pseudocode for displaying transaction history.

Fig. 8, shows the pseudocode for displaying transaction history. When displaying the list of transactions a new Table Row has to be created for each transaction. The values for the date, type, category and amount are retrieved from the transactionList which was taken from the database. Then the text is set to the Table Row and it is added to the Table Layout.

C. Main Interfaces

The screen-shots of the main interfaces are given in Fig.s 7 to 10. Budget Manager Home is the main activity in the application. There are quick access buttons to all the important functionalities provided to the user. There are two separate interfaces to add income and expenses. Date Choosers and Category selection Spinners are available for the convenience of the user. The user can get an overview of the income/expense behaviour by observing the transaction history. The Statistics interface gives access to graphs which are generated by analyzing the transaction information. The user can set budgets for the next time period using the Set Budget interface.



Current Balance

21036.0

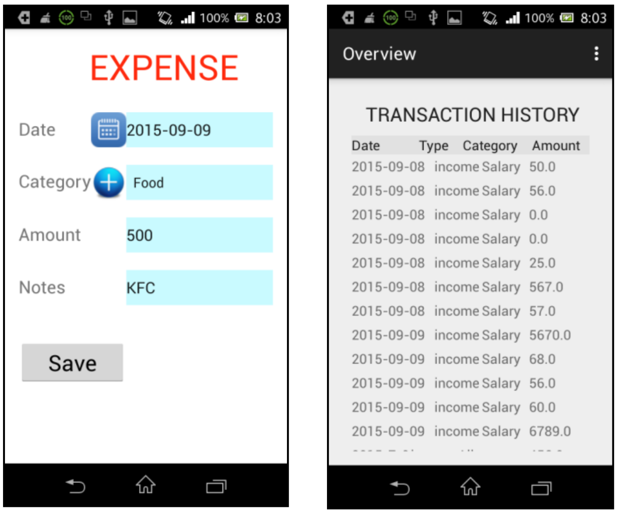
Date 2015-09-09

Category Salary

Amount 10000

Notes With allowance

Fig.9. (a). BudgetManagerHome (b). Add Income



Date 2015-09-09

Category Food

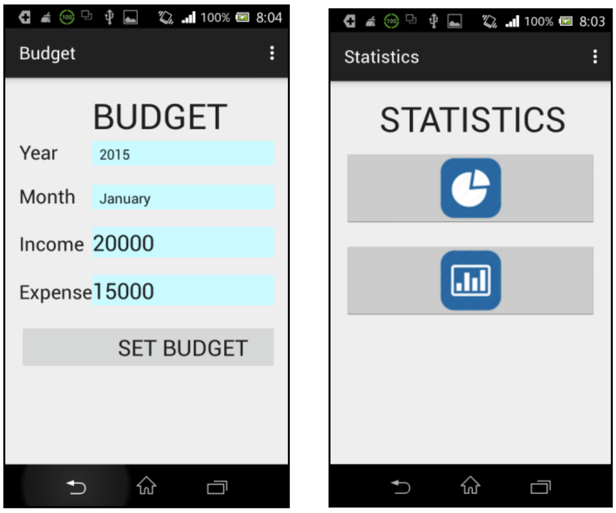
Amount 500

Notes KFC

TRANSACTION HISTORY

Date	Type	Category	Amount
2015-09-08	income	Salary	50.0
2015-09-08	income	Salary	56.0
2015-09-08	income	Salary	0.0
2015-09-08	income	Salary	0.0
2015-09-08	income	Salary	25.0
2015-09-08	income	Salary	567.0
2015-09-08	income	Salary	57.0
2015-09-09	income	Salary	5670.0
2015-09-09	income	Salary	68.0
2015-09-09	income	Salary	56.0
2015-09-09	income	Salary	60.0
2015-09-09	income	Salary	6789.0

Fig.10. (a). Add Expense (b). Transaction History



Year 2015

Month January

Income 20000

Expense 15000

Fig.11- (a) View Statistics (b). Set Budget

V. SYSTEM TESTING AND ANALYSIS

Testing is a critical software development activity because it helps to improve the quality of the applications, ensure better user satisfaction, and reduce overall development time spent on fixing defects. The Android testing framework, an integral part of the development environment, provides an architecture and powerful tools that helps to test every aspect of the application at every level from unit to framework [14]. The Android JUnit extensions provide component-specific test case classes. Robotium which is an Android test automation framework was used for testing the application [12]. The function testing was carried out to verify whether the core functional requirements expected from the application were achieved. All functions such as inserting transaction information, viewing history etc. were manually tested with sample data. Function testing covered everything in the application. User Interface (UI) testing verified the user's interaction with the application. UI testing was aimed to ensure that the UI provided the user with the appropriate access and navigation through the functions of the target-of-test.

Unit tests were done for different components of the application using JUnit framework. Unit tests on different components helped to discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification. The main performance criterion is response time. When the application was tested with sample data it was noticed that the app is capable of working efficiently with a quick response time.

Security and Access Control Testing focused on two key areas of security: Application-level security, including access to the Data or Business Functions and System-level Security, including logging into or remotely accessing to the system. As the Personal Budget Manager is an Android App which is used in a personal phone or tablet, there is no requirement of logging into the system. There is only one user for the app. None of the functions are restricted to the user. The system did not show any failures while testing.

VI. CONCLUSION AND FUTURE WORK

This paper described the design and implementation details of a budget management application. The outcome of the project is an Android application which can be used by any android user. The app contains many features which assist the user to manage his daily transactions. Analysis tools give an extra benefit to the user. User-friendliness and learn-ability of the app are very important as the app will be used by different users with diverse technological knowledge. The app can assist the user to mitigate the problem of managing limited income with unlimited needs as he can record and view his transactions regularly. He has an idea about the amount of money that is available to be spent as the current balance is shown in the home screen of the app. The drawbacks and issues seen in similar apps which were already available in the market were addressed in building this app. Personal Budget manager app is simple and efficient.

Further developments to the app include improving the analysis tools with more attractive and useful graphical representations. Also the budget prediction mechanism has to be improved for better accuracy. The app Back-up in the cloud has to be implemented. It will be very useful when the phone get misplaced or broken, as the user can reinstall the app in a new device and restore the past information.

REFERENCES

[1] IDC Corporate USA. (2015, September 15). Smartphone OS Market Share, 2015 Q2 [Online]. Available : http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[2] MH Riley Ltd. (2015, September 15). Spending Tracker. [Online]. Available : https://play.google.com/store/apps/details?id=com.mhriley.spendingtracker&hl=en

[3] Dayspring Technologies. (2015, September 15). Goodbudget: Budget & Finance. . [Online]. Available : https://play.google.com/store/apps/details?id=com.dayspringtech.envelopes

[4] Realbyte Inc. (2015, September 15). Money Manager Expense & Budget. [Online]. Available : https://play.google.com/store/apps/details?id=com.realbyteapps.moneymanagerfree

[5] Shane Conder & Lauren Darcey (2015, September 15). Mobile development. [Online]. Available : http://code.tutsplus.com/tutorials/learn-java-for-android-development-introduction-to-java--mobile-2604

[6] Ma, Li, Lei Gu, and Jin Wang, "Research and Development of Mobile Application for android Platform.", in International Journal of Multimedia and Ubiquitous Engineering, Vol.9, No.4 (2014), pp.187-198

[7] Chunyue Bi, "Research and Application of SQLite Embedded Database Technology", in Wseas Transactions on Computers, Jan. 2009, Issue 1, Vol.8

[8] Sinan Si Alhir, "Methods & Tools" , Spring 2002, Vol.10, pp 3

[9] (2015, September 15). Android Developer Guidelines [Online]. Available : http://developer.android.com/guide/index.html

[10] (2015, September 15). Smartsheet [Online]. Available https://www.smartsheet.com/product-tour/gantt-charts

[11] (2015, September 15). Balsamiq Mockups - rapid wireframing https://balsamiq.com/products/mockups/

[12] (2015, September 15). Robotium. [Online]. Available http://robotium.com/

[13] Philipp Jahoda. (2015, September 15). **MPAndroidChart .** [Online]. Available : https://github.com/PhilJay/MPAndroidChart

[14] (2015, September 15). Testing Fundamentals [Online]. Available : http://developer.android.com/tools/testing/testing_android.html