SMART TRACKER - AN ANDROID APPLICATION TO TRACK SHOPPING INFORMATION

By

PRIYANKA MASURAM

B.E, Osmania University, India, 2009

A REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Approved by:

Major Professor
Daniel Andresen

Abstract

The world is becoming smarter with an increase in efficiency of wireless communication resulting in an accelerated use of smart phones. Open source of Android market gave a chance to individuals to freely develop their own applications which could be run easily on Android smart phones. The purpose of this project is to develop an Android application for managing and tracking a user's shopping information.

The Smart tracker is about tracking the user's purchase of clothes and accessories. The main advantage of this Android app is that it helps to remember the size of clothing purchased. When a user shops at a retail store, he feeds the information pertaining to the fitting of the clothes purchased brand wise, whether it is a correct fit or a bit loose/tight. The data thus entered can be retrieved when he decides to make a purchase in the same store in future. This tremendously decreases the user's shopping time and makes the experience easier as the right size is already known and hence there is no hassle of using trial rooms to see if the clothes fit.

In addition to size, Smart tracker also stores the price at which the item was bought and any additional user comments along with the store location. When data pertaining to a purchased clothing item is entered, the app provides the option to append an image of the item either by capturing a picture through the smart phone's camera or by uploading an image from the gallery. This helps the user recount the item to which the information corresponds. Another important feature is that tracking of purchases for a family can be personalized, i.e., the shopping information of each family member can be stored separately (especially for kids who can't track their own expense). Additionally, the user can also keep track of money spent on each item/brand/family member. Smart tracker also enables user to create a wish list to remember what they (or a friend) liked but didn't buy in a particular store.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude to my major professor, Dr. Daniel Andresen, for his guidance and constant encouragement throughout my project development. He has been a source of immense knowledge and provoked me to think innovatively.

Also, I would like to thank my committee members, Dr. Mitchell L. Neilsen and Dr. Torben Amtoft, for serving on my committee. I would like to extend my thanks to Professors of Computing and Information Sciences at Kansas State University for giving me the opportunity to take various courses under them, which helped me in this project. I would also like to acknowledge the academic and technical staff of the CIS department for their constant support and help in completion my graduate study.

Finally, I would like to express my love and gratitude to my family and friends for their endless support and motivation.

Chapter 1 - Introduction

1.1 Project Description

Smart Tracker is an Android application for managing personal shopping information of an individual. The application has various features like storing the information pertaining to the piece of clothing purchased such as name of the brand, type of clothing, size, price, location of store at which the item was bought, a picture of the item and any additional user comments. This helps the user know the right size in various brands thus eliminating the tedious process of going to the trial room to know if the clothing fits right or not.

The user is also provided an option to create several different user accounts for family members and friends thus enabling the separate storage of each member's shopping information. This helps the user to not only remember his shopping data but also that of his friends and family. Smart tracker also helps the user keep a tab on the total expense made by each family member or friend towards clothes shopping. Also, there is a provision for each user account to filter the list of items bought based on the clothing brand or the type of item (top, bottom, etc.). Users can also create a wish list to store information regarding items of clothing that were liked by them or friends but not purchased. This list can be retrieved whenever necessary.

1.2 Motivation

The main purpose of this project is to create an application for Android smart phones and tablets to keep a record of the list of items purchased by an individual and his family members. The app also stores information of each item like when and where it was bought, size, price and fitting. This is extremely useful to evaluate different stores, compare pricing, and decrease shopping time. Also, it enables the user to monitor each family member’s total shopping expense, thus helping the user track his finances.

Chapter 2 - Background & Related Work

2.1 Android OS

Android is a Linux based operating system that can be used on mobile devices such as smart phones and tablets. It is open source, in the sense that developers can not only create new applications but also modify existing applications and distribute them to other Android users all over the community. Also, there is a lot of documentation available online which makes it easy to develop an application in the Android OS.

2.2 Android Architecture

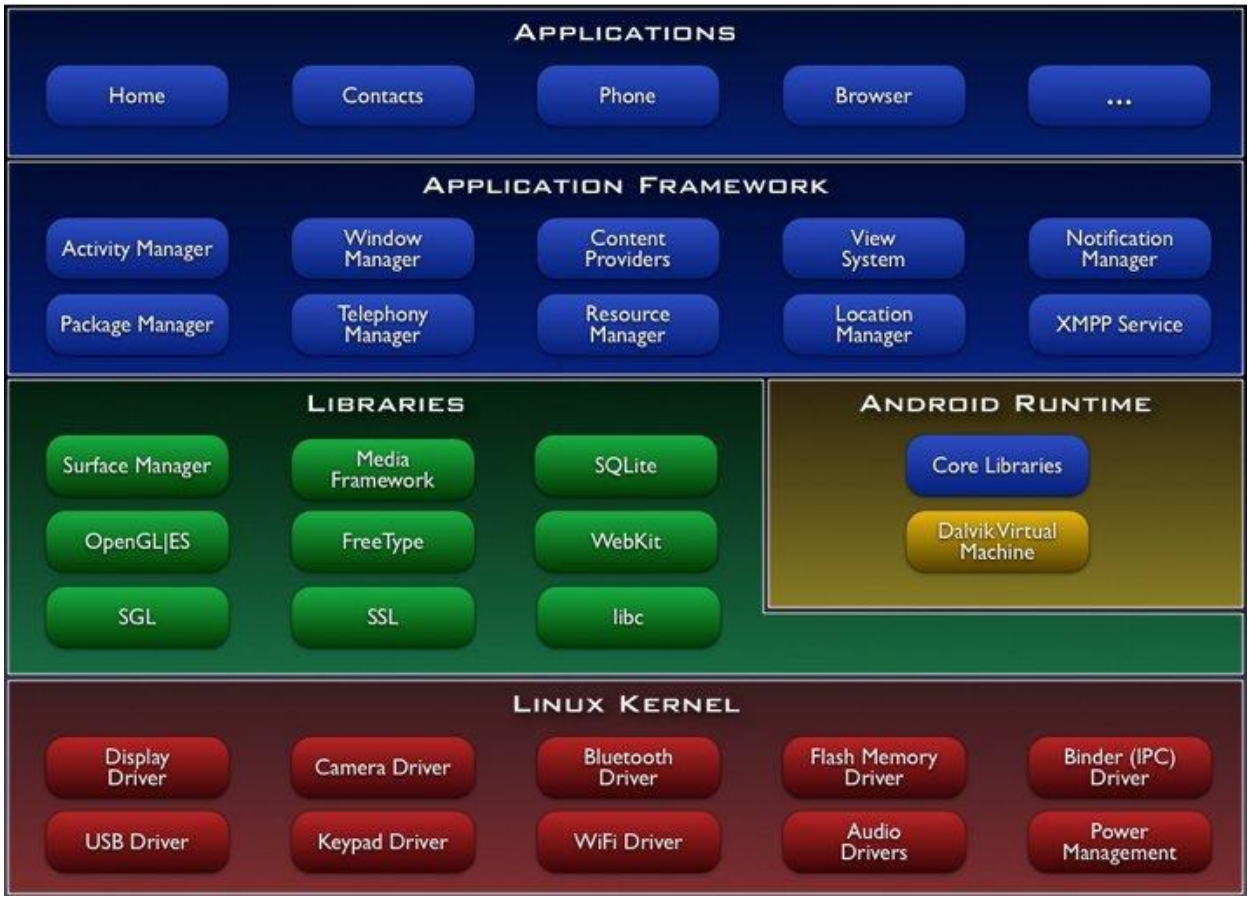

Figure 2.1 Android Architecture (1)([1])

Figure 2.1 above depicts the different layers and component of the Android operating system. Each layer offers varied services to the layer above it. The different layers in the architecture are:

- Linux Kernel - Interacts with the device hardware and consists of different drivers to communicate with the internal hardware.
- Libraries and Android Runtime - This layer includes different libraries required for the Android application development along with the SQLite database package libraries. Android runtime contains libraries native to Android and virtual machine to run the native code.
- Application framework - This is the main layer where the application can communicate with different classes and other applications or services
- Applications - It is the layer where the Android application resides and can be used by the user.

2.3 Data Storage

There are different types of storage available for an Android application. The data for android applications are normally stored using one of the below storage types:

- Direct Internal Storage (On device)
- External Storage Databases
- Shared preferences
- SQLite Database
- On Networks locations

This app uses SQLite database as a storage mode for all of its data.

2.4 Related work

There are many applications in the Android market which are available to users through Google Play services. Android developers can freely develop and host their application through Google play. There are apps like note taker to scribble the necessary data and save but it is not efficient and user friendly for storing and searching the data. Yelp is another famous app where you can find the nearest address and other information for the given location name but it does not suffice the user's need of storing the data. Hence, it is observed that there are not many apps where user can keep track of their shopping information as in Smart Tracker.

Chapter 3 - Requirement Analysis

3.1 Requirement Gathering

The development of the project started with gathering the requirements to build an effective and useful app with good user interface. Requirement gathering started with brain storming with fellow students and drawing the skeleton of requirements which included major features. The next step in requirement gathering was getting an insight into current applications in the Android marketplace where there is a wide variety of apps with different Graphical User Interface (GUI). This helped me decide what common features are required for application development in general and also enlightened me to distinguish good user interface from bad.

The fine graining of requirements was obtained from my major professor Dr. Daniel Andresen, with whom I met regularly. All these sources of requirements played a vital role in developing a simple and user friendly application. The major requirements of the application are:

- Providing a tab based view for user accounts, brands and wish list.
- Tab based view to view brands of each user, items, add items, edit user for a single user account.
- Enable swipe based navigation for different tabs and focus on the page being viewed.
- Display user friendly dialogs for date picking, location look up and image loading.
- Viewing total price along with accounts list and brands list to track costs.
- Offering expandable view which hides additional content and displays only required content on the header.
- Data must be retrieved without any loss (i.e., in case of images).

Other important requirement is to include Android platform version compatibility for different devices. The application has to be designed to support both Android smart phones and tablets, and has to be usable on devices with different screen sizes. Smart Tracker was developed on Android 4.3 and is targeted at Android versions 3.0 and higher.

3.2 Requirement Specification

3.2.1Software Requirements

There are different software requirements for development and running of the Android application.

For development of the application:

Operating System: Windows XP or higher / Mac OS X 10.5.8 or later / Linux

Platform: Android SDK Framework 10 or higher

Java Database: SQLite Database

Tools: Eclipse SDK 3.5, ADT plug-in for eclipse

Technologies used: Java, SQLite, Android, Google Maps V2 API

Debugger: Android Dalvik Debug Monitor Service (DDMS)

Android Emulator: SDK Version 3.0 or higher

For running the application on Android device:

Operating System: Android 3.0 or higher versions

Network: Wi-Fi Internet or cellular Network

3.2.2Hardware Requirement

There are distinct hardware requirements for developing and running an Android application.

For developing the application:

Processor: Intel Pentium IV or higher

RAM: 256 MB

Space on disk: minimum 250MB

Hardware Requirements for running the application:

Device: Smart Phone or Android Tablet with Android version 3.0 or higher

Camera: 1.3MP rear or front camera

Minimum space to execute: 5.0MB

Chapter 4 - System Architecture and Design

4.1 System Architecture

Following is the system architecture for the Smart tracker application.

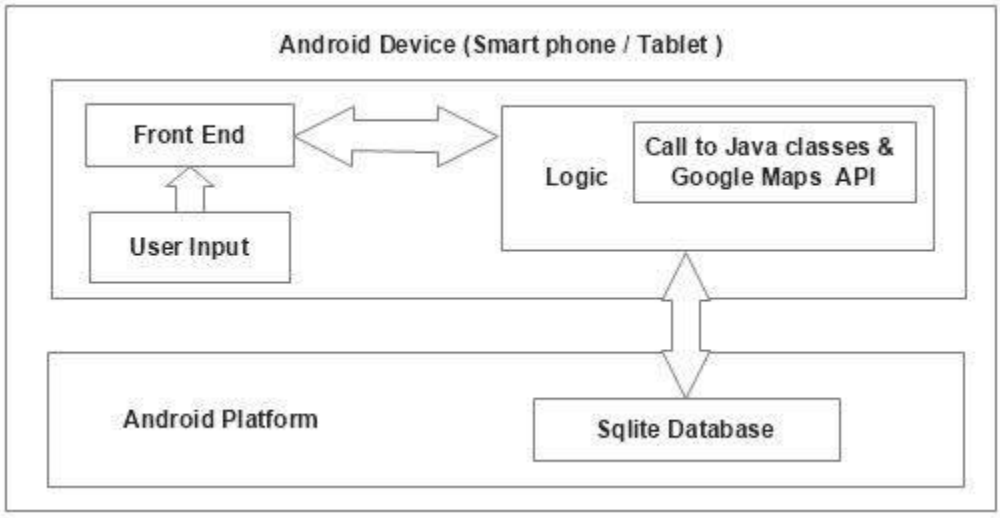

Figure 4.1 System Architecture

From Figure 4.1, we can see the system architecture of this application. There are 3 major screens which require user input - Add user account, Add user item, and Add wish list item. The user input is filled in the form of text fields, date and location picker and button clicks. Based on the touch event on the text boxes, the focus changes to the touched widget and action is performed accordingly. The on touch events on the links navigate to the Map view where the application interacts with Google Maps V2 API. Another important feature of this application is the capability to store images. Camera and Gallery are the applications with which the Smart Tracker communicates to load images. The action events performed in the background will post back to the front end GUI. For example, the selected image from the Camera/Gallery posts back to Image view on the front end screen.

The data given from the input screens are passed to the java classes which in turn calls the SQLite database created on the device's internal memory. Data is stored in the form of tables like any other relational database systems.

4.2 System Design

UML diagrams are used to explain how the system is designed. After the requirements are gathered, design of the system is created using UML, Unified Modelling Language. The views of this project are captured in Use Case and Class Diagrams. The actions and entities are identified and structured into these Use Case and Class Diagrams.

4.2.1 Use case diagram

The use case diagram is a graphical representation of the interaction between different elements of the system. It is a behavioral view of the application. The user of the system is the Android device user who wishes to use the Smart Tracker. Figure 4.2 depicts the use case diagram for this application.

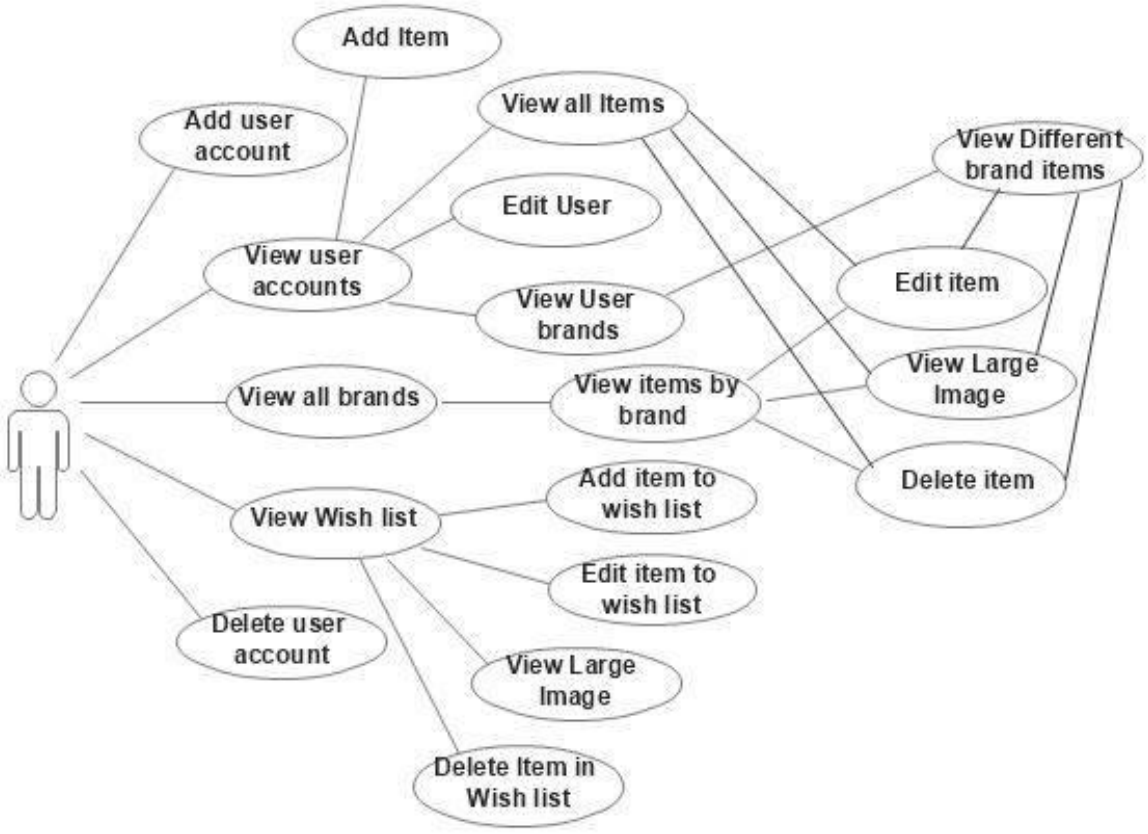

Figure 4.2 Use case diagram

4.2.2 Class Diagram

A class diagram is used to describes the structure of a system with help of different classes, attributes, operations (or methods) and the relationships among those classes. The class diagram can be considered as the main building block of object oriented modelling. Each activity and fragment in the android application is depicted as a class and the interaction (navigation) between them is explained as the relationship between the classes.

The figure 4.3 and figure 4.4 illustrates the system design in the form of a class diagram. The MainActivity is the home screen of the application and the navigation starts from this activity. Difference types of relationship are represented by the connections between classes.

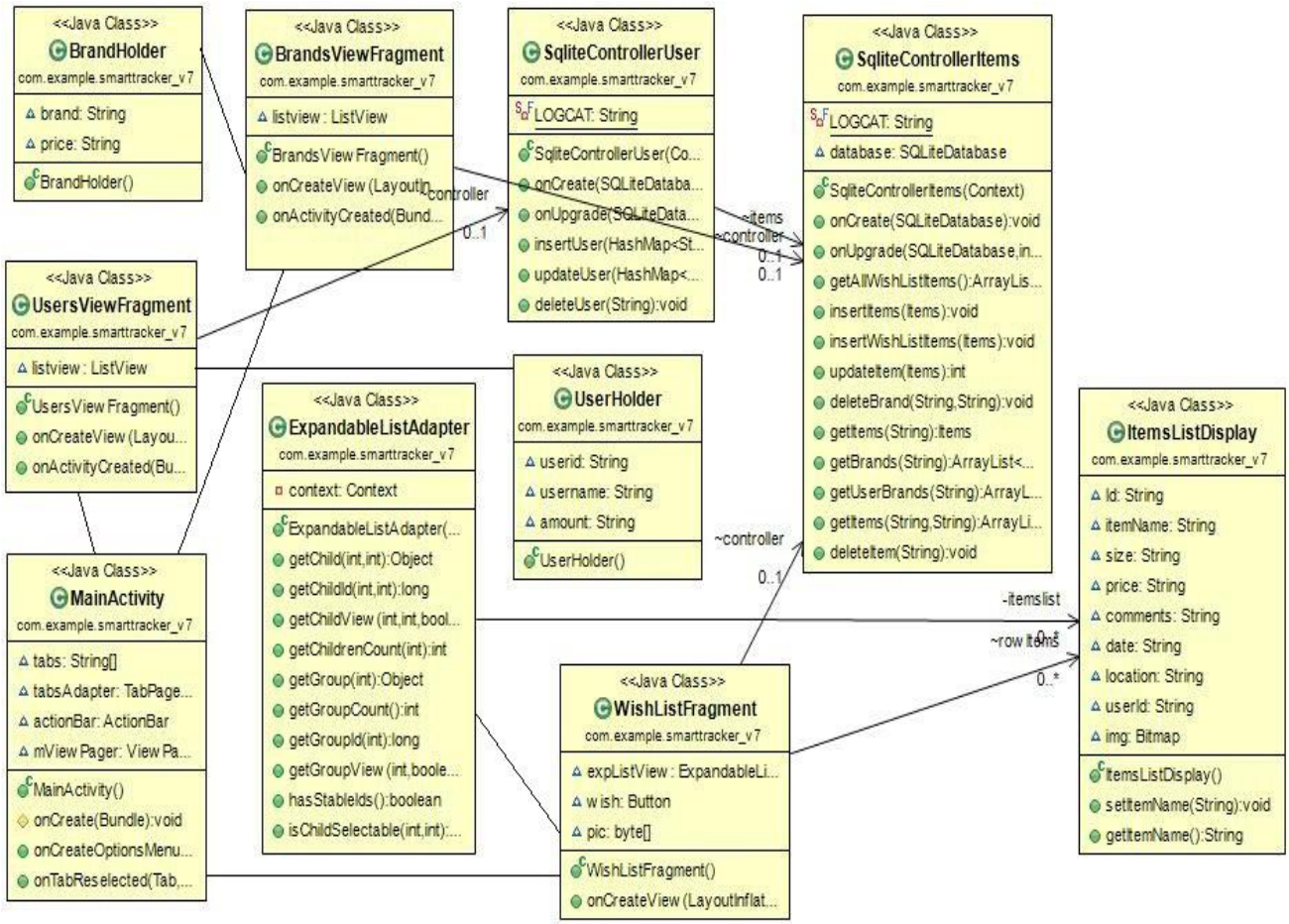

Figure 4.3 Class Diagram 1

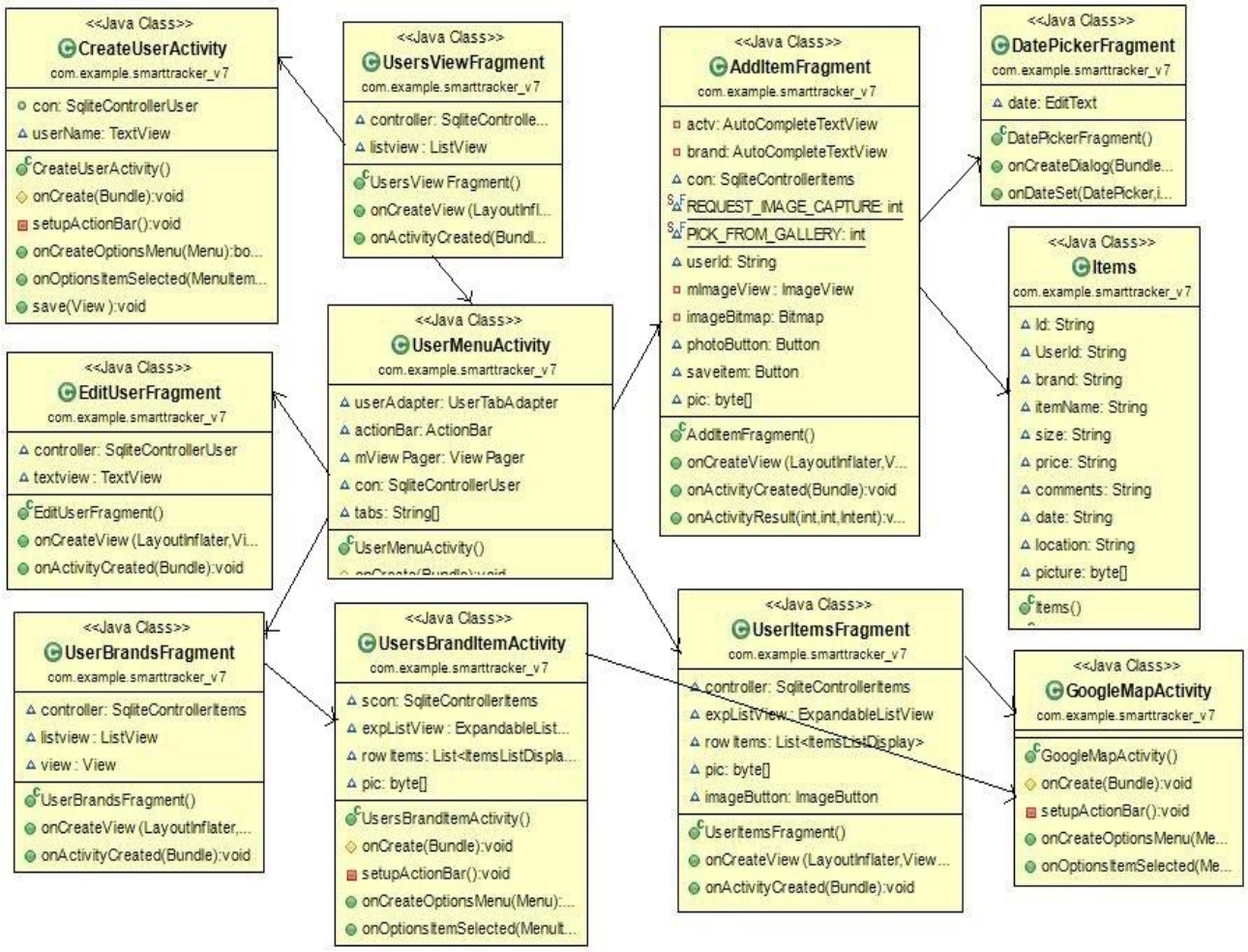

Figure 4.4 Class Diagram 2

Chapter 5 - Android Framework Components

Android application is developed in Java using Android SDK (Software development kit). The application can be developed in various IDEs (Integrated development environment). The IDE used with the Smart tracker application is Eclipse and an ADT-plug in for eclipse is used to support Android programming. Android SDK has specific modules which compile the java code along with the resource files into an Android Package with '.apk' extension. When the application is run on an Android device, this is the package which is installed on the smart phone or tablet. Various components of the Android framework are discussed in this section.

5.1AndroidManifest.xml

The manifest contains vital information about the application which is required for the Android system to run the application code. AndroidManifest.xml is present in the root directory of every Android application.

This file consists of the description of different Android components like activities, services, content providers along with the intents and intent filters. Manifest also contains the minimum and target SDK versions which the application supports. This xml file also includes the permissions required to access different APIs (Application programming interface) like maps, internet, network services etc., in addition to permission to communicate with and access other Android applications on the device like camera and gallery. Android manifest also defines the hierarchy of the activities in the application.

Below is the detailed AndroidManifest.xml file used for the Smart tracker application.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.smarttracker_v7"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="12"
        android:targetSdkVersion="18" />

    <uses-permission android:name="com.google.android.maps.permission.MAPS_RECEIVE"
/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

```
    <!-- Required to show current location -->
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <!-- Required OpenGL ES 2.0. for Maps V2 -->
    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.smarttracker_v7.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.example.smarttracker_v7.CreateUserActivity"
            android:label="@string/title_activity_create_user"
            android:parentActivityName="com.example.smarttracker_v7.MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.example.smarttracker_v7.MainActivity" />
        </activity>
        <activity
            android:name="com.example.smarttracker_v7.UserMenuActivity"
            android:label="@string/title_activity_user_menu"
            android:parentActivityName="com.example.smarttracker_v7.MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.example.smarttracker_v7.MainActivity" />
        </activity>
        <activity
            android:name="com.example.smarttracker_v7.UsersBrandItemActivity"
            android:label="@string/title_activity_users_brand_item"
            android:parentActivityName="com.example.smarttracker_v7.UserMenuActivity"
>
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.example.smarttracker_v7.UserMenuActivity" />
        </activity>
         <activity
            android:name="com.example.smarttracker_v7.GoogleMapActivity"
            android:label="@string/title_activity_map"
            android:parentActivityName="com.example.smarttracker_v7.UserMenuActivity"
>
            <meta-data
```

```
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.smarttracker_v7.UserMenuActivity" />
        </activity>

        <uses-library android:name="com.google.android.maps" />

        <meta-data android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />

        <meta-data android:name="com.google.android.maps.v2.API_KEY"
     android:value="AIzaSyBgrXDrzkVmPVA3rzY5jhd1trZ5iDvhJvc" />


    </application>

</manifest>
```

The AndroidManifest.xml of this application includes different activities, activity hierarchy, user permissions and libraries required for the application. The minimum SDK used for this application is 12. Due to the latest features used in this application like fragmentation and Google Maps, the Android powered devices which run on versions lower than Android 3.0 (Honeycomb) cannot run this application. The targeted SDK version for this application is 18. Smart tracker application can be run on any Android devices with version Android 3.0 (Honeycomb) and higher. The different user permissions required for this application include internet, network, maps, coarse location, and fine location for displaying remote location and device location on Google Maps. Google Android Maps library and Google play services are used to open Google maps. Google Maps V2 API key is generated for the application to access the Google Map addresses.

There are different activities defined in the manifest which are used by this application. The intent-filter MAIN indicates the startup activity for the application. Application icon for the project is also mapped in the manifest file.

5.2 Activities

An Activity is the component of an Android application which is responsible for the interaction with user. It consists of back bone logic for front end GUI (Graphical User Interface) which is displayed to the user. Each activity is associated with a view built in XML which contains user interface components like text boxes, buttons, labels, images etc., that are directly viewed by user.

During its life cycle, an Android activity undergoes different call back methods like Create, Start, Resume, Pause, Resume, Stop and Destroy. Activity alters its display on the state change. Figure 5.1 below shows the life cycle of an activity.

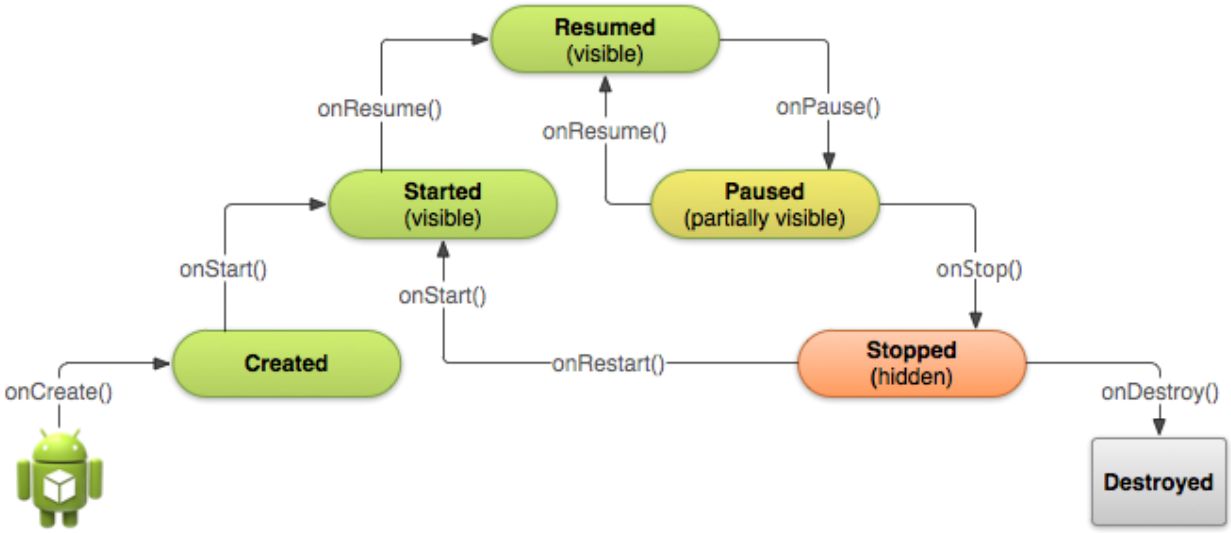

Figure 5.1 Activity Life Cycle (2)

An activity is initialized in onCreate(Bundle) where UI components are accessed with findViewById(R.id.<viewname>). Views are bound to the activity by setContentView(R.layout.<layoutname>). Below is the sample code for activity initialization.

```
@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_create_user);
		// Show the Up button in the action bar.
		setupActionBar();
		TextView userName = (TextView) findViewById(R.id.newuser);
		. . .
	}
```

5.3 Fragments

A fragment is an android component which represents the partial behavior or portion of user interface in an Activity. Fragments are used when there is a need of similar portion of interface in multiple activities so it can be said that fragments can be reusable. In addition to the reusable feature of fragments they can also be combined in single activity to build a multi-pane user interface. It can be considered as a sub-activity which can be added or removed while the

activity is running. Although fragment has a separate life cycle, it cannot exist independently so it should always be embedded in an activity. Fragments can also exist without an associated UI.

5.4 Intents

An Intent is an abstract description of the operation to be performed by an activity. Intents are used to communicate between different activities and fragments. They provide a runtime binding between different activities. The significance of intent is to launch new activities. Intent contains the information necessary for the new activity to perform the requested action.

Intent is started using startActivity() function. It has different components like action, data, category, extras and flags which can be set before starting the intent using startActivity(). There are two major types of intents - implicit intents and explicit intents. Implicit intents are those which are communicating with other external applications instead of components specified in the application. Explicit intents are those which calls the components in the application by its activity name. Explicit intents uses the activity names registered in the AndroidManifest.xml. Few implicit intents used in this project are: *ACTION_GET_CONTENT, EXTRA_INITIAL_INTENTS, ACTION_IMAGE_CAPTURE* etc.

Intent(Implicit) to view to open camera for taking a picture:

```
Intent cameraIntent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
```

Intent(Explicit) for calling another activity in the application:

```
Intent intent = new Intent(getApplicationContext(), CreateUserActivity.class);
```

5.5 SQLite Database

Android applications which require relational database can use many different databases. When the data is specific to the user/device and needs to be stored on the Android device, we need a database which can manipulate the local memory. Android platform has a built-in SQLite database engine which is self-contained, server less and requires no configuration.

The database is created by extending SQLiteOpenHelper class and the created database is stored in device memory due to which the database is local to that application on the device. Database is created by overriding onCreate() function of the SQLiteOpenHelper class. The SQLite database objects can be used for transactions on the database.

Chapter 6 - Implementation

The Smart Tracker application is designed to make tracking of an individual's shopping easier. The main objective of the application is to provide a simple user friendly interface where the user can easily store and retrieve his shopping information. The main features of the application are:

- Create user account - Owner of the device can create any number of user accounts. This is useful to create a separate account for each family member or friend. The created account is stored in the database.
- View user account - All the user accounts are displayed as a list from which the application user can select an account to be viewed. Smart Tracker provides different views of user account where user can view all the brands purchased by a particular user, list of all items, list of items under each brand, add an item or edit the account.
- View all brands - This displays all the available brands stored in the database irrespective of the total accounts. Items under each brand will be displayed irrespective of user account.
- Add items for users - This is the main feature of the application where the user can add the purchased item under an account. There are different fields like Brand, Item Name, Size, Price, Date, Location, Comments and an option to insert an image. All this data is stored in the SQLite database created on the device's memory.
- Add items to wish list - Wish list is for the items which are chosen but not yet purchased by the user. This helps the user during a possible future purchase. The user can directly add the item with the necessary fields under wish list. The wish list items are directly visible under wish list tab.

The Smart Tracker Android application is developed on Android SDK 4.4 using Eclipse Juno IDE. ADT plug for eclipse is installed for Android development environment along with the Java 1.6. The user interface is developed using XML and the business logic is written in Java. Business logic connects to the SQLite database for transactional data. The application is debugged using Dalvik Debug Monitor Server (DDMS) in eclipse. LogCat is used to print the stack traces when necessary. The total lines of code written in this application are 3912 which includes Java and XML files. Table 6.1 shows the breakdown of lines of code.

Language	LOC
Java	2325
Xml	1587
Total	3912

Table 6.1 Lines of Code (LOC)

Across the span of 3 months, around 320 hours of effort was put in for analysis, design, development and testing of this application. Additional time was invested in device environment set-up, Android development environment set-up and documentation.

6.1 Graphical User Interface

The graphical user interface of the Smart Tracker Android application is developed using XML. This is designed to keep the interface for the user simple, understandable, user friendly and easy to navigate from between different tasks.

6.1.1 Splash Screen

The figure 6.1 shows the splash screen which is seen for 3000 milliseconds while the application is launched.

Figure 6.1 Splash Screen

6.1.2 Home Page

The home page of the Smart tracker shows 3 tabs:

- User Accounts view
- Brands view
- Wish list view

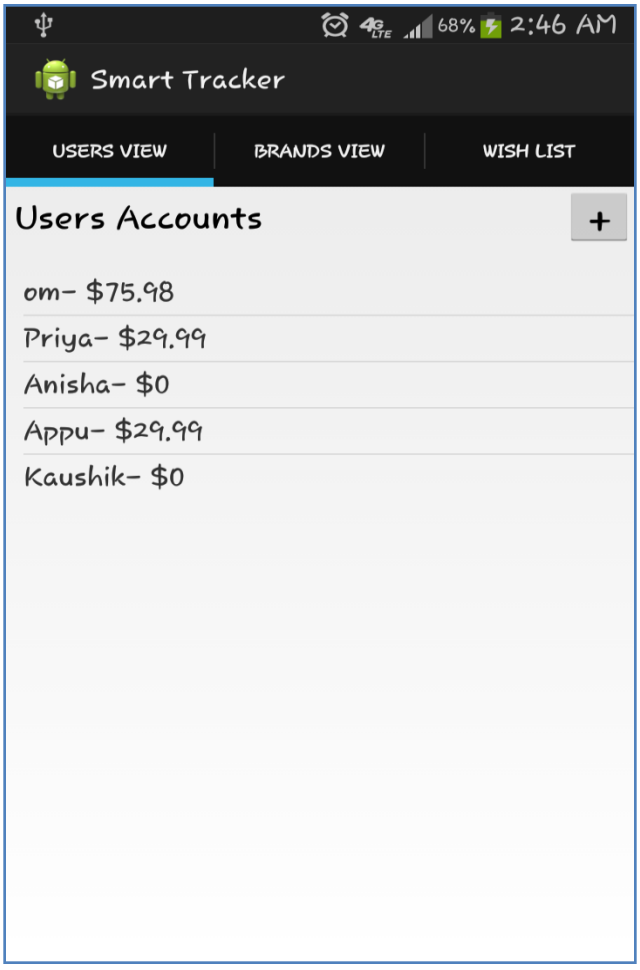

Figure 6.2 User Accounts Screen

Figure 6.1 shows the main screen which users will be able to view when they launch the application. This shows the list of user accounts stored in the application. It can also be seen that there is a cumulative amount of all the items added under that user account. The list displayed is an XML list which is retrieved from User and Items tables from SQLite database stored on the device memory. The list items are clickable and open the detailed view for each user.

6.1.3 Create User

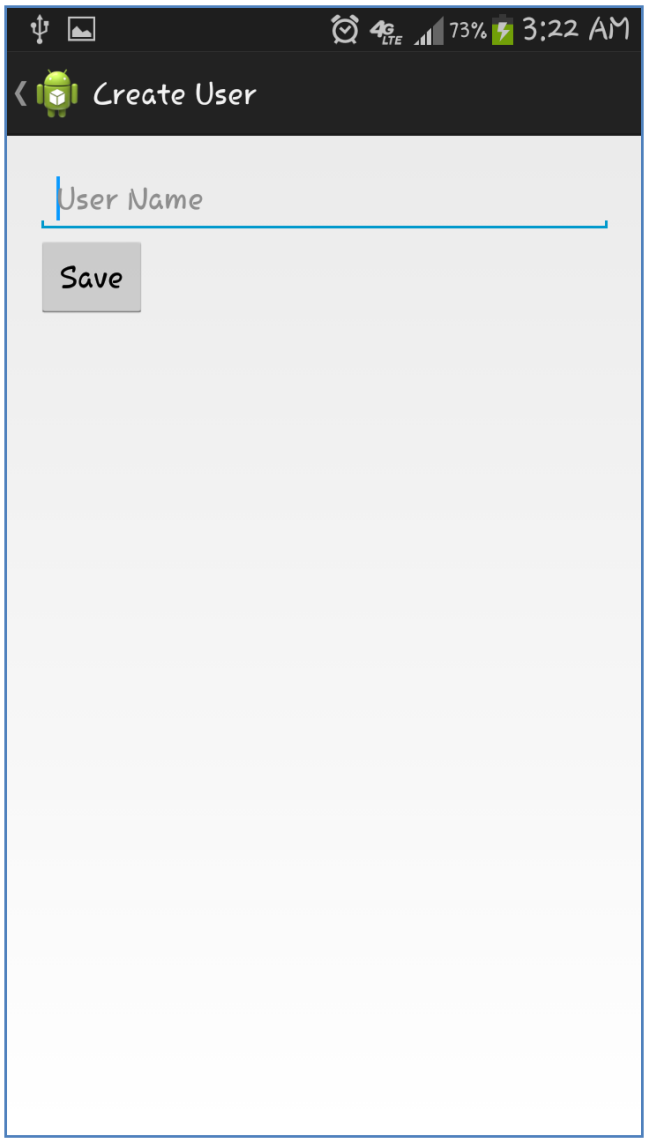

Figure 6.3 Create User View

On the User Account view, there is a "+" button on the top right corner of the screen. This button is to create a new user account. The above activity will be opened when the "+" button is clicked. User has to enter the username to create a new user. A blank user name will throw a validation error on the screen.

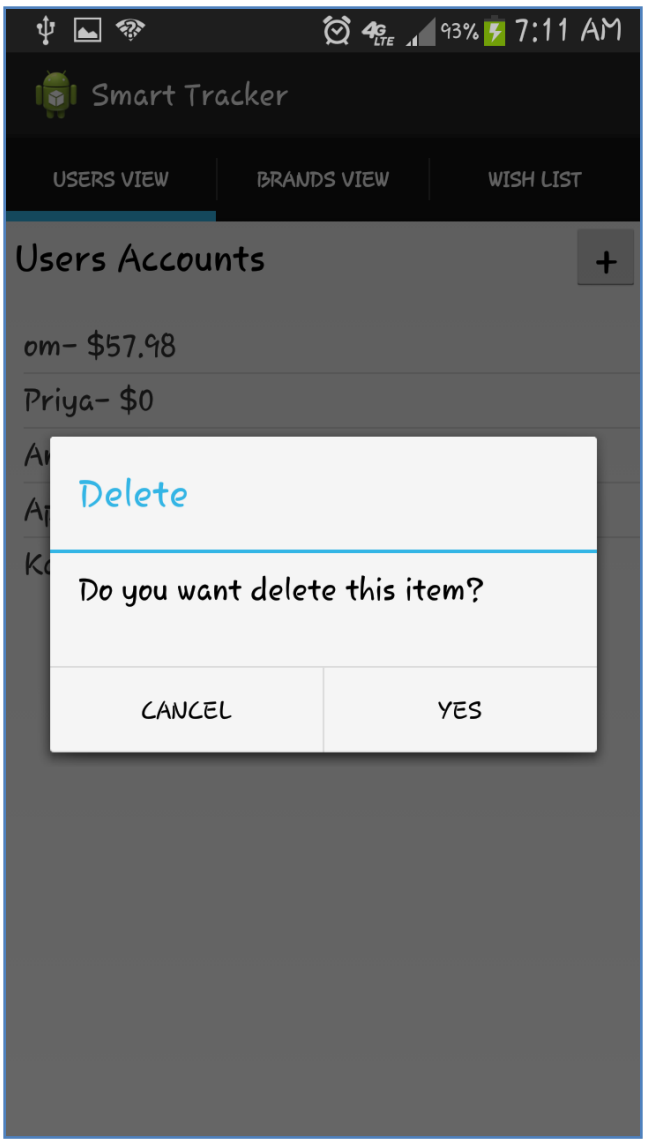

Figure 6.4 User Account Delete Screen

The user can not only create but also remove a user account when it is no longer in use. On the long press of the list item displayed on the Users view screen, you can see a pop up window where user can select the appropriate option. The user can click on 'YES' to delete the user or on 'CANCEL' to dismiss the alert dialog. This alert dialog is handled on the item click event of the user account.

6.1.4Brands View Screen

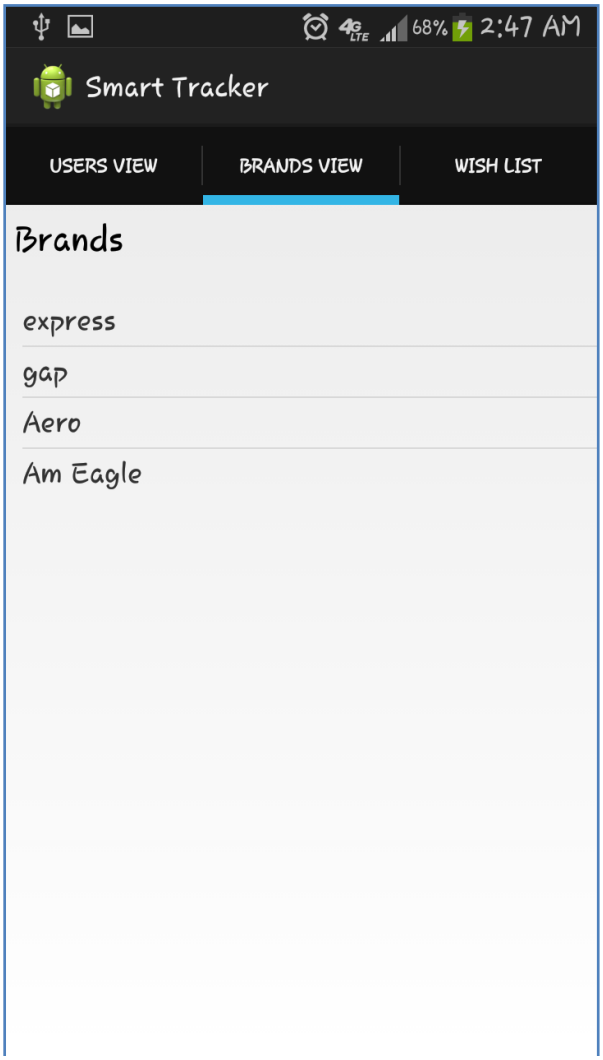

Figure 6.5 Brands View

This is the brands view where user of the application can view the brands added for all the user accounts. The main motive of this screen is to provide a view where everyone's brands can be viewed simultaneously. This is a list view which displays distinct brands of all the accounts.

6.1.5Items – Brands View Screen

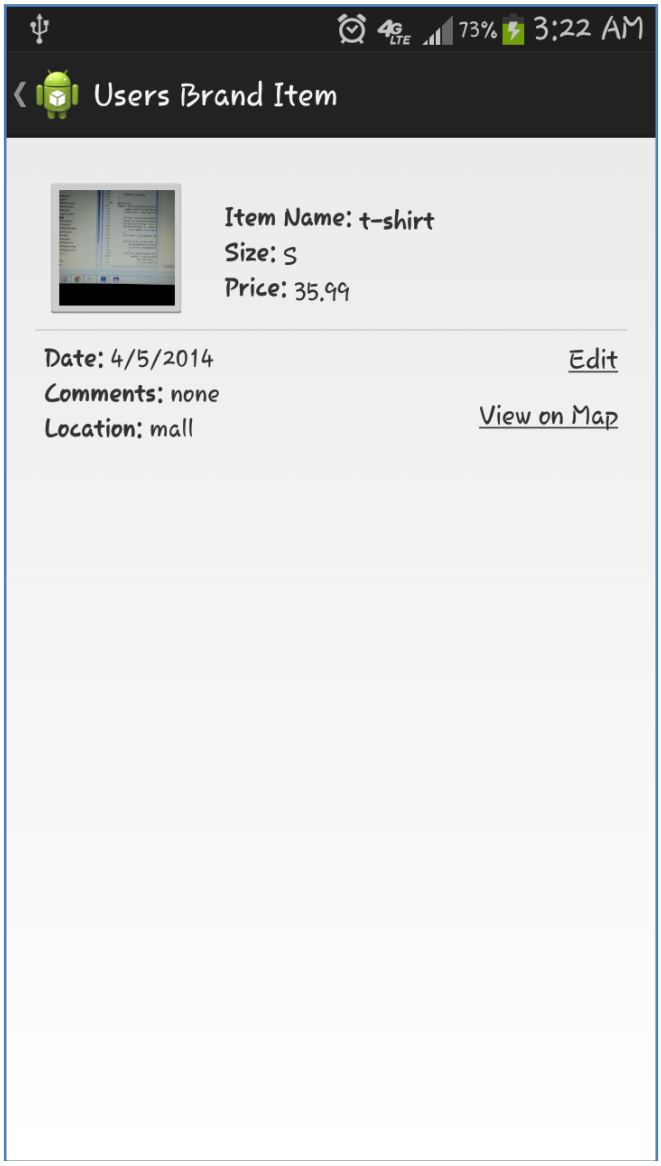

Figure 6.6 Brands View Items

This is the item list displayed when one of the list of brands on the Brands View screen is clicked. This screen displays all the items under the selected brand. The items displayed on this page are all the items of different users under that brand.

6.1.6 Wish List Screen

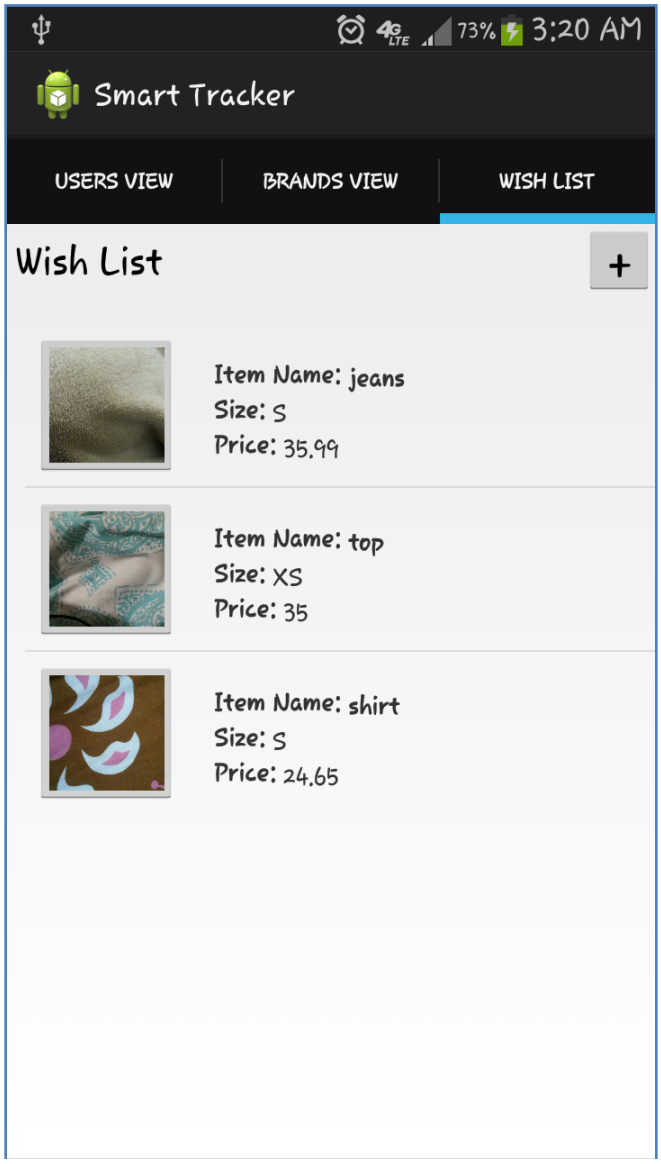

Figure 6.7 Wish List View

If the user wants to store an item which is not purchased, he can do that in the wish list view. The wish list is not associated with any brand or user. Everyone's wish item can be stored and retrieved at once. These wish list items are expandable views where user can expand and collapse on demand. User can add the wish item by clicking on the "+" button on the top right corner of the screen. Clicking the "+" button opens another activity screen where user can add all the details.

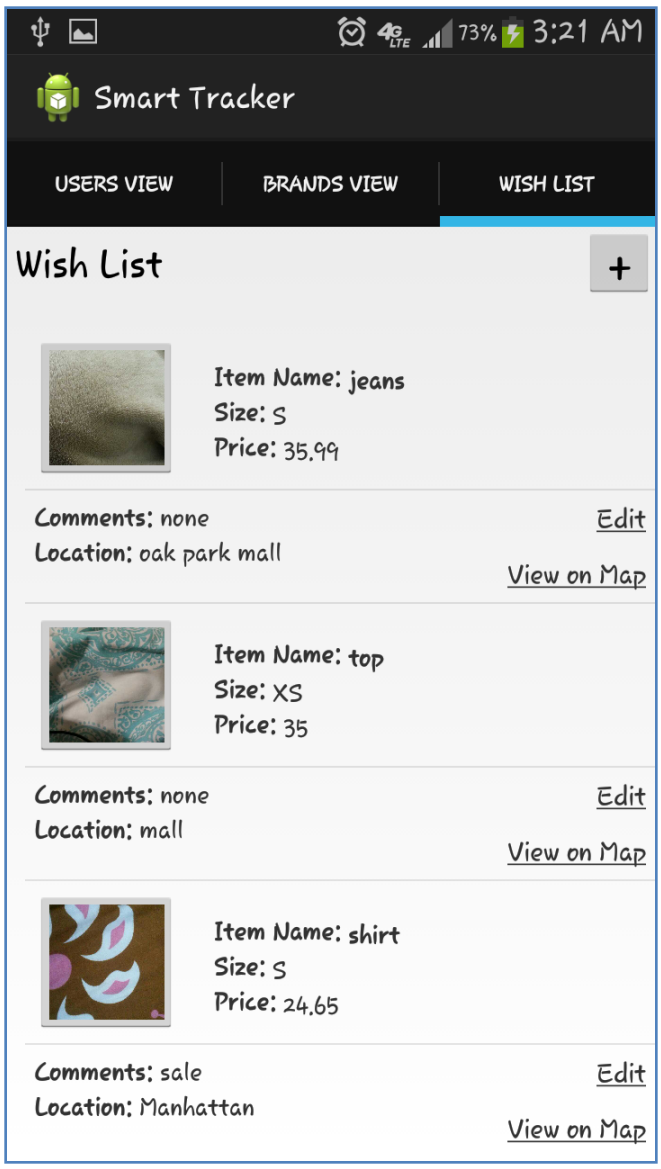

Figure 6.8 Wish List Expanded View

This is the expanded view of the Wish list item. On click of the item, the list expands and collapses. In addition to the extra fields displayed in the expanded view, user has a choice to select edit or View on Map option. On clicking the Edit link, user will navigate to Edit Item activity where the fields will be populated. User can change the necessary fields and update the item. Moreover, user can also view the location on the map by selecting the 'View on Map link'. By clicking this link, a Map activity will be opened and Google Maps will be loaded onto the screen.

6.1.7 User Brands View

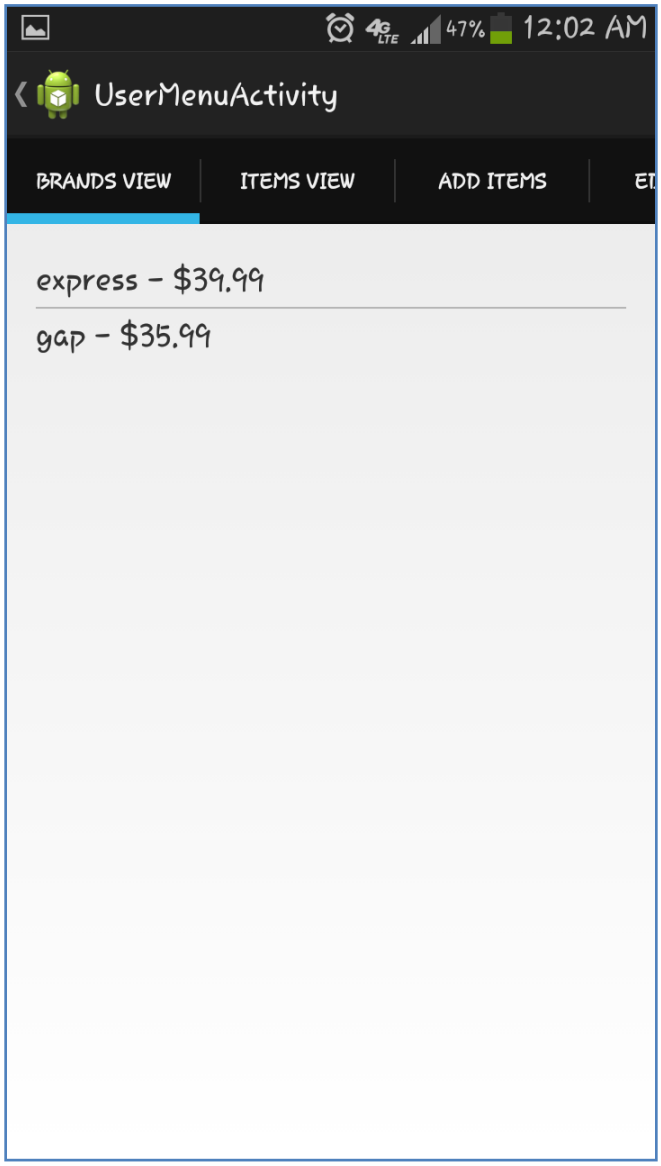

Figure 6.9 User Brands View

This is the user brands screen which is displayed by clicking on the user account. On user account click, the user menu will be displayed where brands view is the first fragment called in this activity.

6.1.8 All Items of an Account

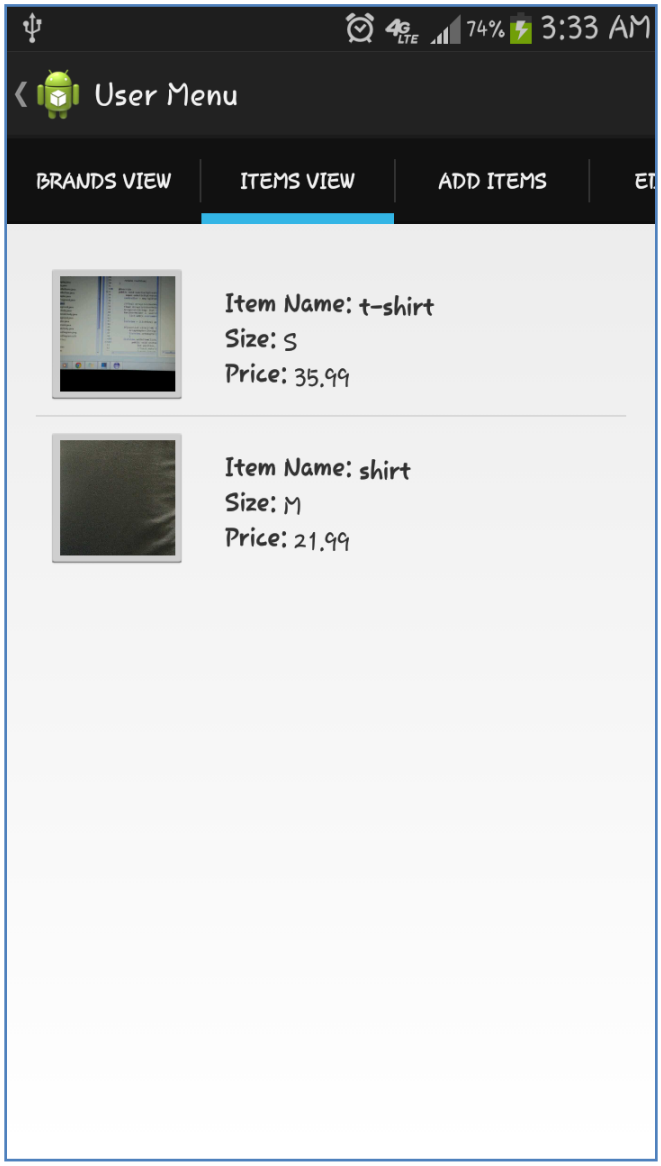

Figure 6.10 User Account Items View

Figure 6.8 shows the user account items view where user will be able to view all the items under particular user. The items displayed in this view are not grouped by brand. This view is also an Expandable list view where user can also expand and contract the item.

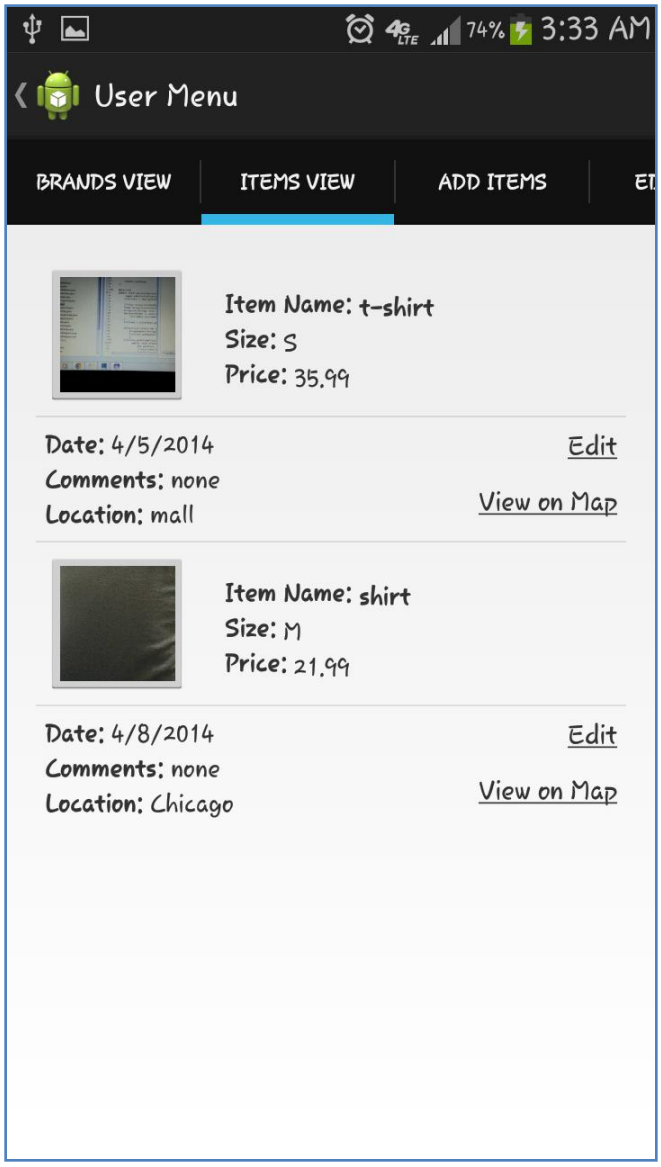

Figure 6.11 User Account Items – Expandable View

This expandable view is similar to the wish list items expandable view. The expanded part of the list contains Edit and View on Map link. The Edit link is for editing the item if the user wishes to. View on the Map is the Google Map view of the location where the item is shopped. This expandable view can also be compressed which make the view user friendly.

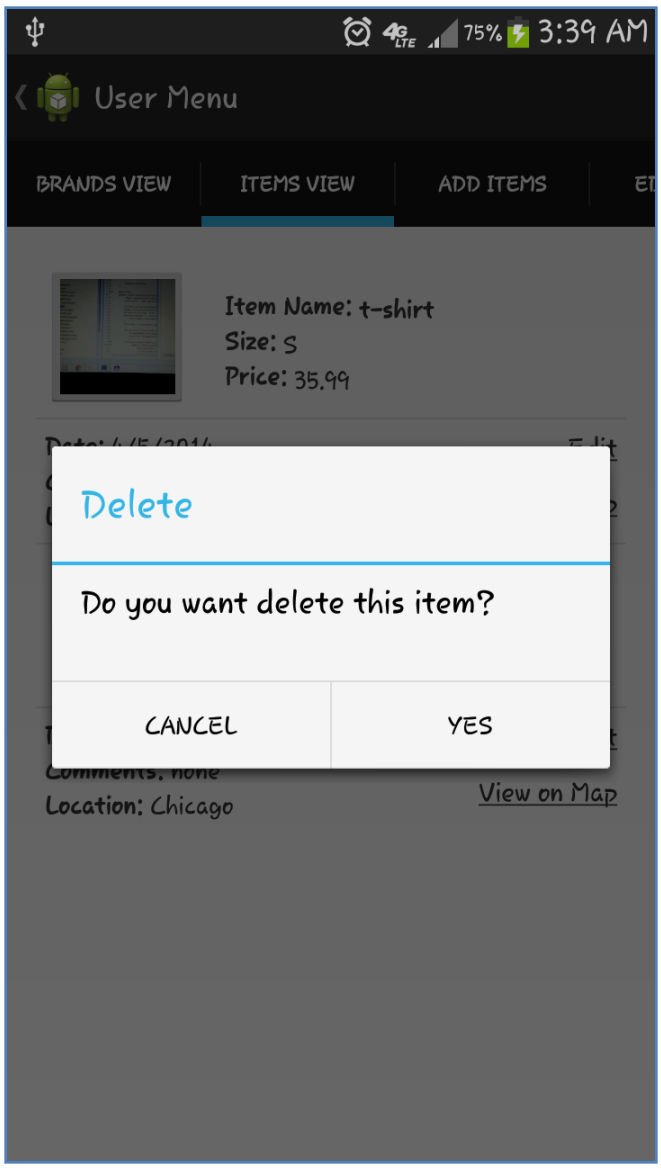

Figure 6.12 User Item Delete View

If the user wants to delete an item from the list, he can delete it in the view. This can be achieved by a long click on the item header. The alert dialog is displayed as in Figure 6.10. User needs to choose one of the options on the alert dialog. The positive button 'YES' deletes the respective item and the negative option 'CANCEL' cancels the delete action.

6.1.9 Add Item Screen

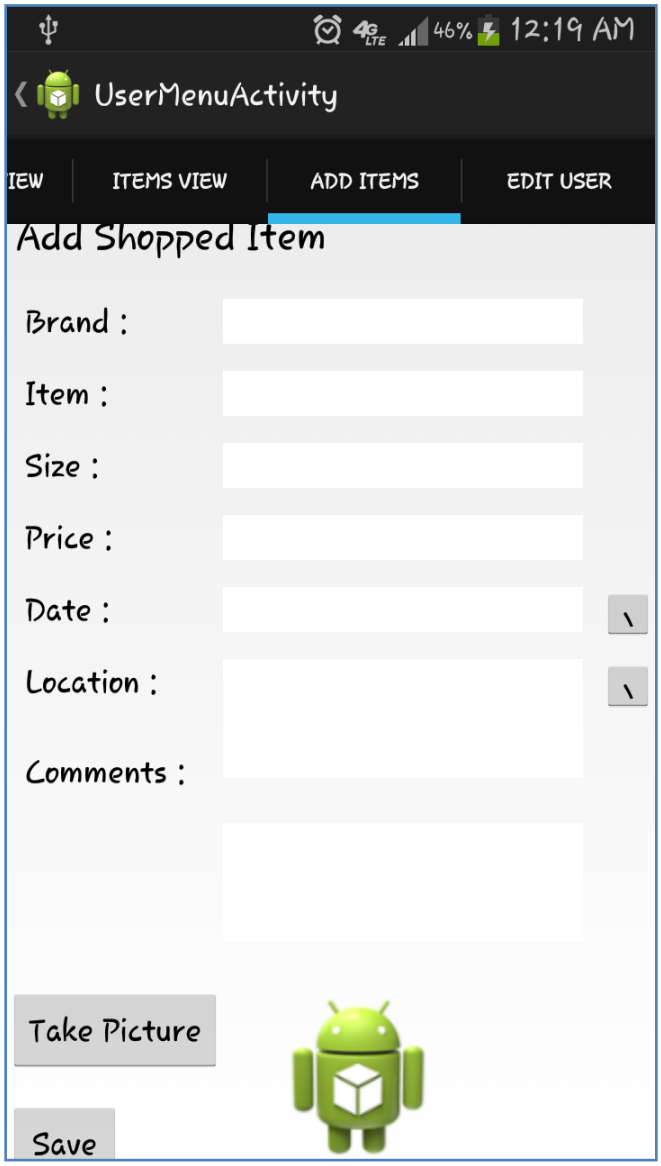

Figure 6.13 Add Item View

This screen contains the input widgets for items whose information needs to be stored into the database. The input fields include Brand, Item name, Size, Price, Date, Location and Comments. User can also load picture and save it along with the other data. The database functions will be called on click of save button.

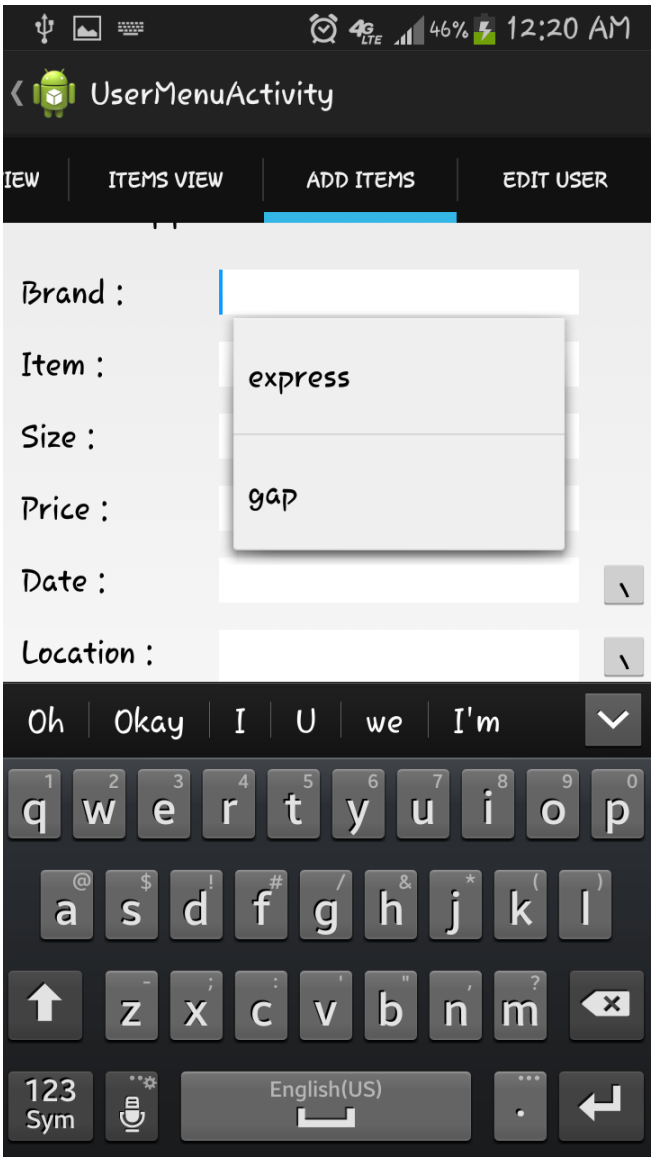

Figure 6.14 Add Item – Brand Auto Complete

In the Add Item Screen, there are different input fields which are to be filled by the user. Brand is one field which user needs to fill. It is difficult for the user to enter long names of the brands each time they enter an item. Hence, an auto complete drop down is implemented. The data for this auto complete dropdown is populated from the existing brands which are stored by the user itself. The dropdown is displayed on focus.

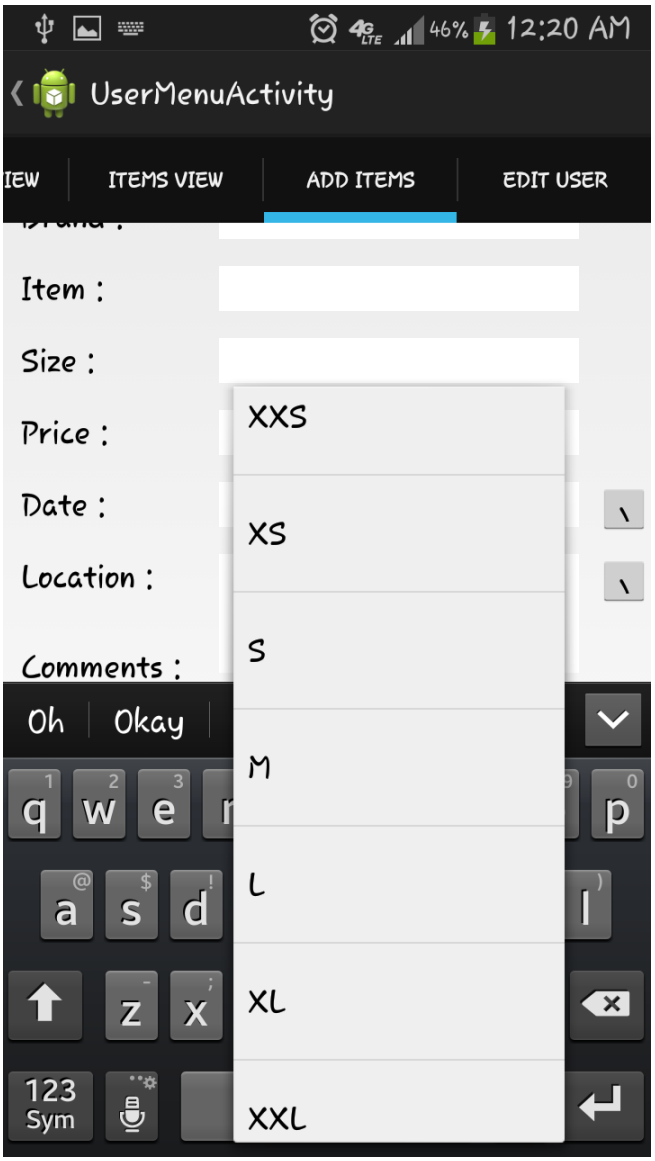

Figure 6.15 Add Item-Size

Size is another column in Add Item view where user enters the size of the Item purchased (shirt/top/jeans). It has an auto complete enabled to make the input fields user friendly. The basic sizes are saved in an array and populated automatically when the user starts typing the size. The pop up is displayed when the focus is changed from one input field to another.

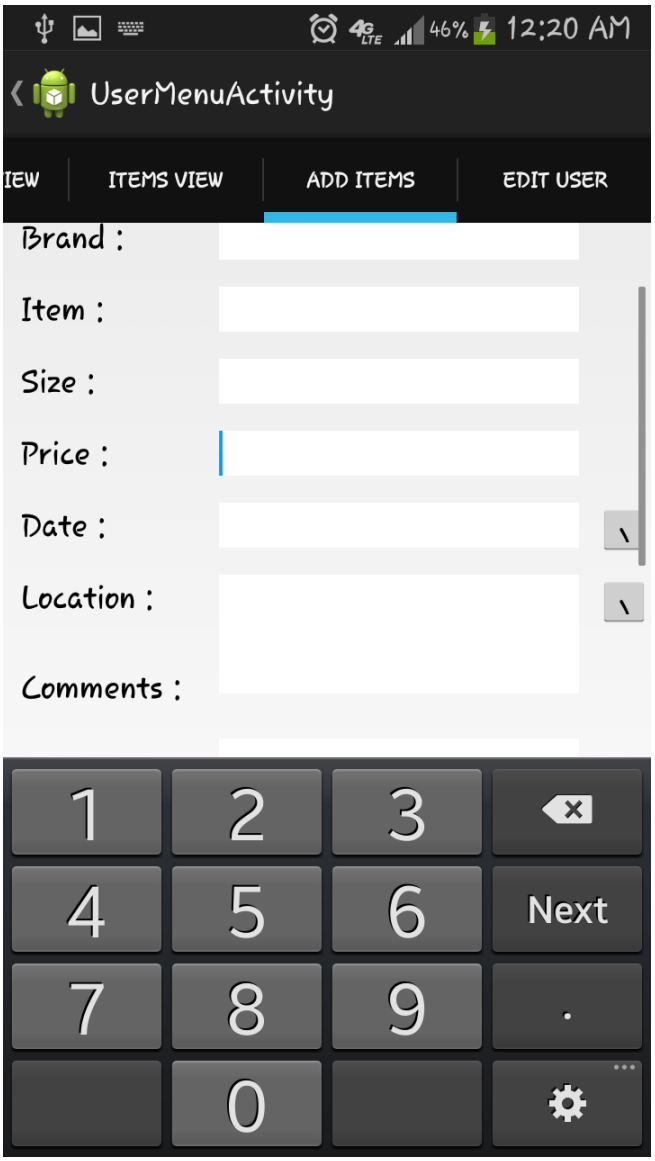

Figure 6.16 Add Item – Price

Another user input for the Add Item view is price. Smart Tracker also keeps track of price for each item. As the price cannot be in the form of string, we have limited the input type of the application. User will be able to enter only numeric values i.e., integral values or decimal values.

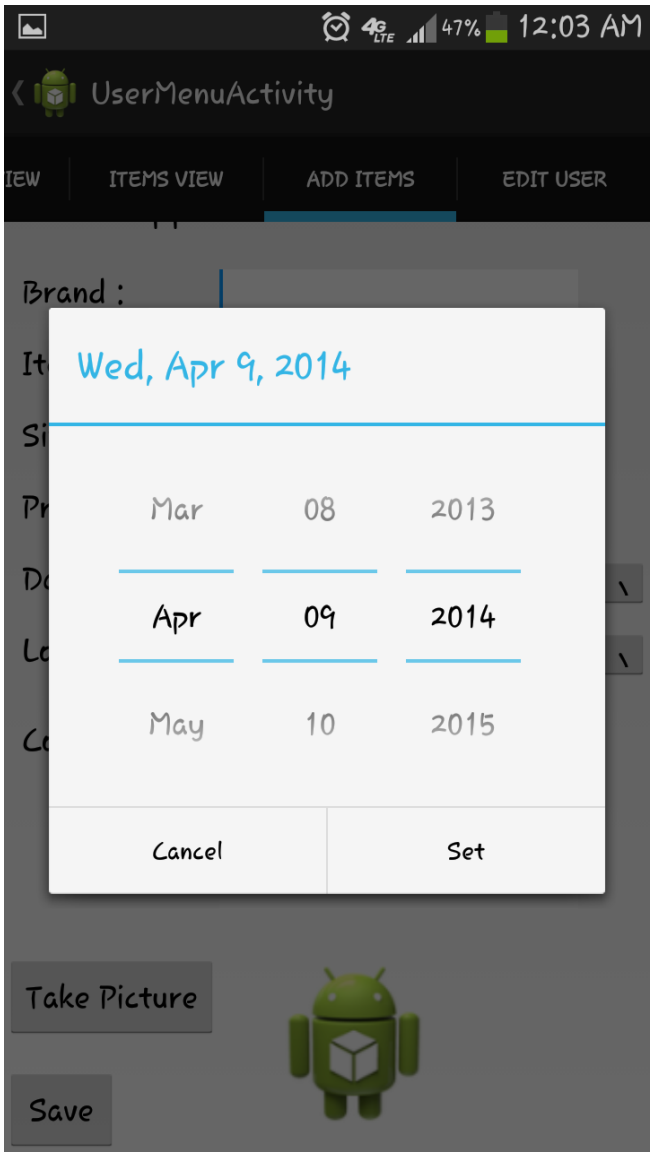

Figure 6.17 Add Item – Date picker

Add Item has a field which stores the date on which the item is bought. Entering the date picker is best option to provide an easier and understandable interface. The above displayed dialog in the project was very useful for date picker. Calendar is called inside dialog to display the date picker. Use can scroll over the months, dates and years view and select the required date. Set button on the dialog is used to set the time on the dialog. If the user wishes to manually enter the date instead, he can do so too.

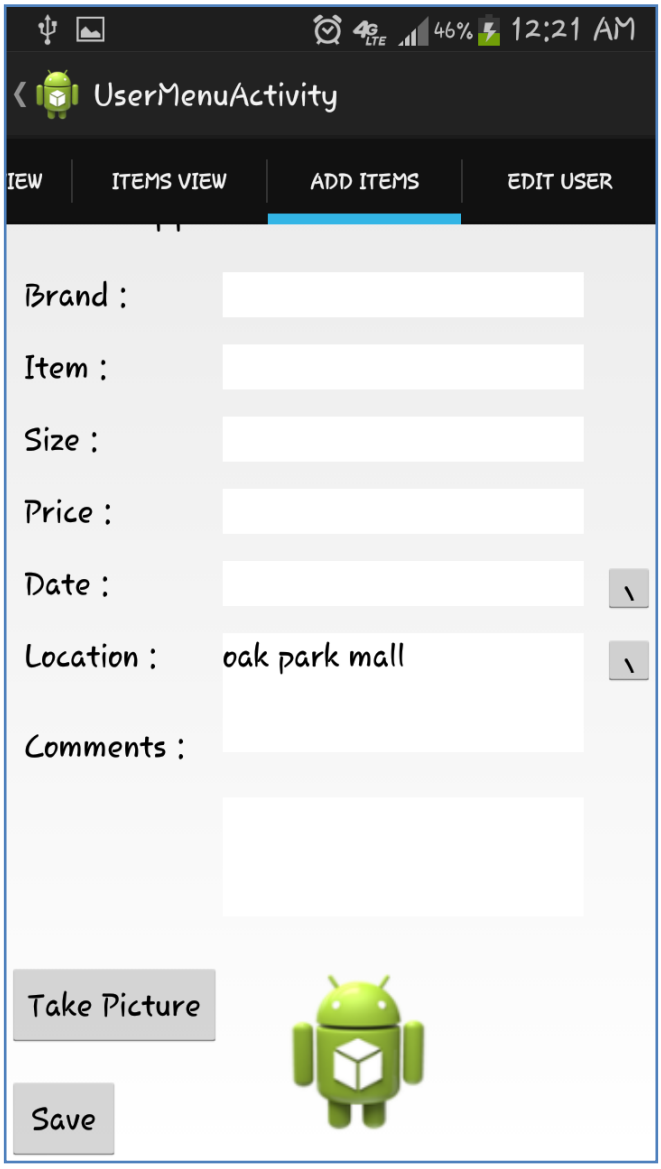

Figure 6.18 Add Item – Location

Location is another input which user can store in the database. This is the location of the store in which the item was bought. User can type the key work and press the button for look up of location through Google API services.

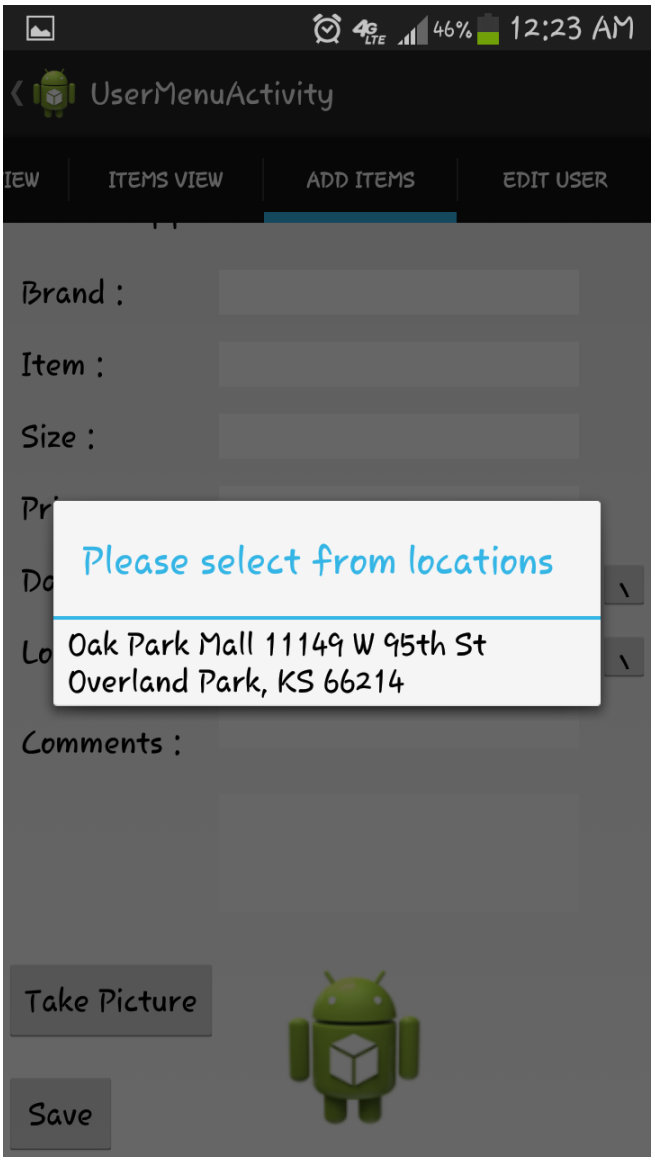

Figure 6.19 Add Item – Location Lookup

When the user enters the location keyword and clicks on lookup button, the string will be used to get the widely known locations with the entered keyword. These locations are list of addresses which will be displayed in the dialog as shown in Figure 6.17. User needs to choose the appropriate location from the list displayed. By this, the user can save the complete address of the location.

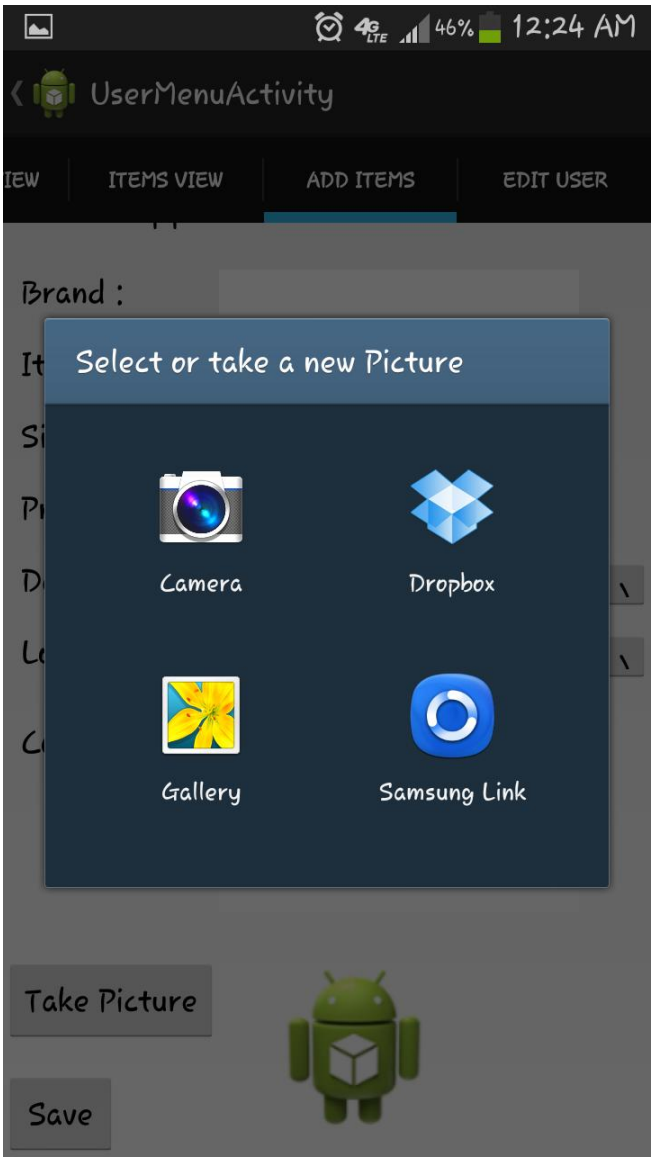

Figure 6.20 Add Item – Picture Chooser

Smart Tracker also provides a facility to store the images in the database so that it can be easily remembered. To add a picture, user can choose one of the available sources. User can directly take a picture using the device camera or can select an existing image from the gallery. The images are scaled and compressed to display it on the screen as a bitmap image. Images are converted to byte array and are stored in the database as BLOB type. After the blob is retrieved from the database, it is re-converted into image bitmap and displayed on the list view to the user.

6.1.10 Edit User

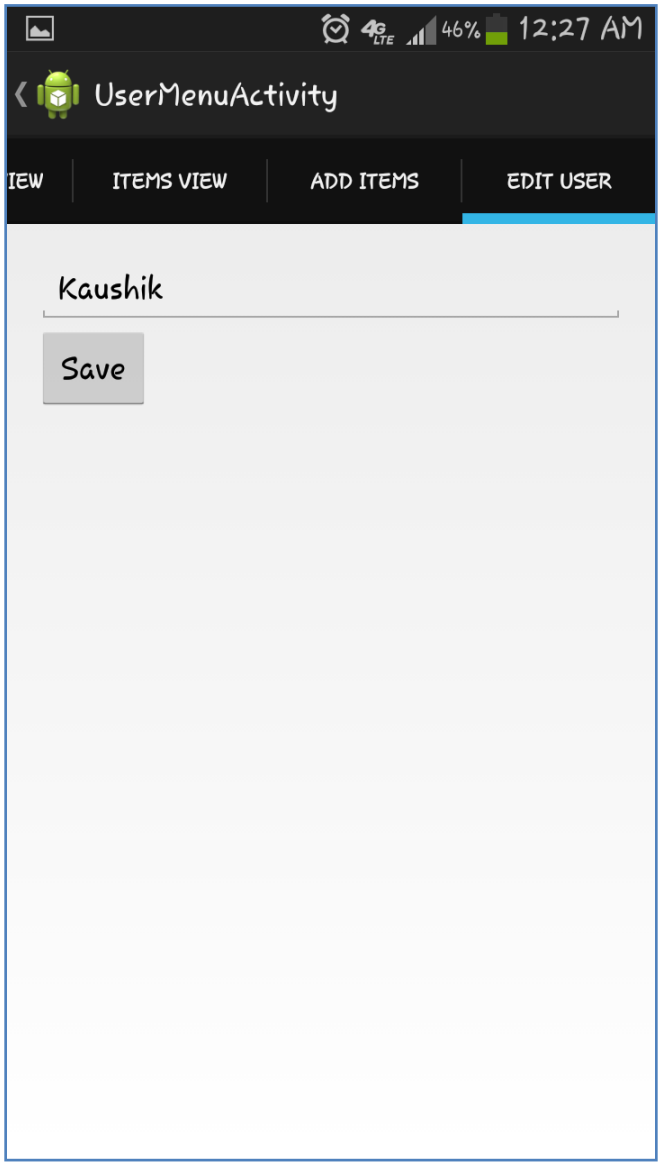

Figure 6.21 Edit User View

It is where the user can edit the name of the user account. The name will be pre populated for convenience and user can edit this if required.

6.1.11Maps

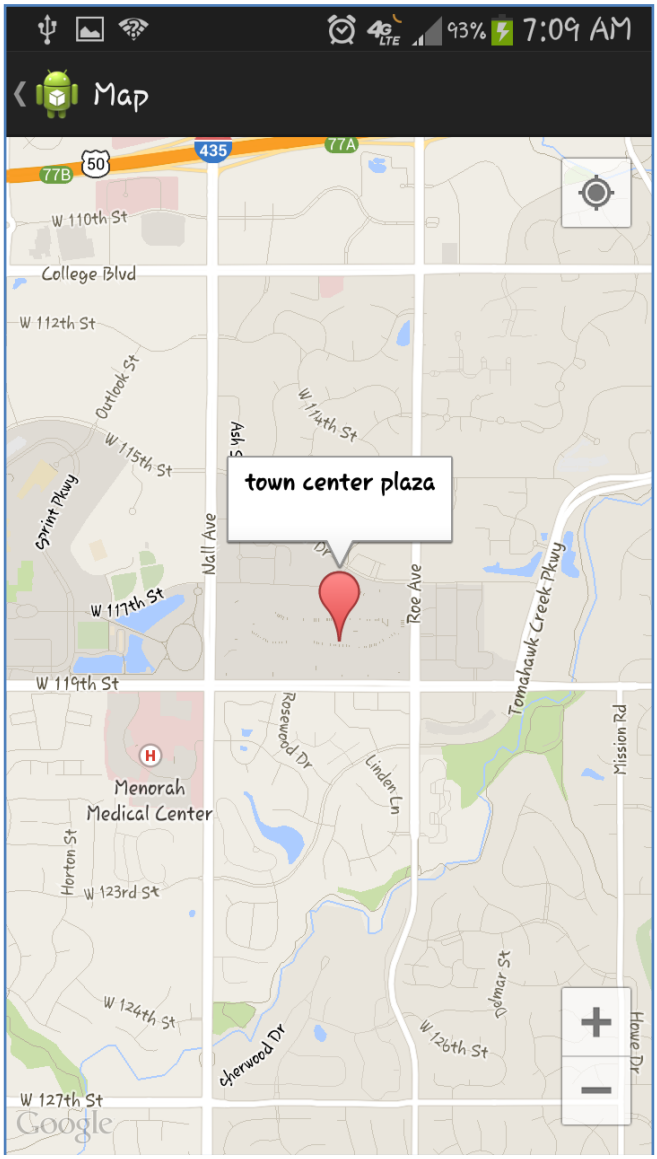

Figure 6.22 Google Maps

Whenever the items are displayed in the expanded list view, the header contains important information and any additional information is available on expanding the item. View on Map is the link available on expanded list and can be used to view the location address on the Google Maps. When the link is clicked, the address is evaluated to calculate latitude and longitude and then displayed on the map with the marker.

Chapter 7 - Testing

As we know that testing ensures the correctness of the application, this android project is tested in different aspects which are discussed in this chapter.

Android framework provides a default logging and debugging tool – Dalvik Debug Monitor Server (DDMS) which logs the errors and other information in the Logcat. DDMS posts the system messages like printing statements from the code, stack traces, errors and warnings on to the Logcat which allows the developer to debug the application effectively.

7.1 Unit Testing

Unit testing is about testing each unit/part of the application independently. Different fragments and activities in this android application are tested for its functionality during this phase. The following unit test cases are run manually on Samsung galaxy s3 for its correctness.

Sr. no	Test Case	Expected Result	Result
1	On load of startup screen	Display the list of existing user accounts	Pass
2	On click of Brands tab	Display the collective list of brands stored for all users	Pass
3	On click of Wish list tab	Displays all the wish list items	Pass
4	On click of wish list item	Expands the wish list item to display additional information	Pass
5	On click of expanded wish list item	Collapses the wish list item to hide the addition details	Pass
6	On click of items under user account	Expands/collapse the additional details	Pass
7	On click of add user tab	Displays fields to be filled by the user	Pass
8	On focus of Brand name text	Displays auto complete drop	Pass

	box	down of brands of that user	
9	On focus of item name	Displays auto complete drop down of items of that user	Pass
10	On focus of size	Displays auto complete drop down of sizes of that user	Pass
11	On focus of price	Displays keyboard with only for numeric and decimal entry	Pass
12	On click of '>' button beside date text box (Smart phone view)	Displays date dialog	Pass
13	On click of '>' button beside date text box (Tablet view)	Displays date dialog along with calendar	Pass
14	On click of '>' button Beside location text box	Displays error message if no location is entered	Pass
		Displays list of available address for the given location name	
15	On click of take picture button	Opens chooser for camera and gallery	Pass
16	On click of camera button in the chooser	Opens camera	Pass
17	On click of gallery in the chooser	Opens gallery to select image	Pass
18	On Click of edit user tab	Displays the name of the user account which is editable	Pass

Table 7.1 Unit Test Cases

7.2 Integration Testing

After unit testing each module individually, the code is tested for inter module navigation. Below are the test cases which test the interaction between different activities and fragments in the project.

Sr. no	Test Case	Expected Result	Result
1	On swipe right to left from Users view screen	Navigates to brands view screen	Pass
2	On swipe right to left from brands view screen	Navigates to wish list screen	Pass
3	On swipe left to right from wish list screen	Navigates to brands view screen	Pass
4	On swipe left to right from brands view screen	Navigates to users view screen	Pass
5	On click of '+' button on users view screen	Navigates to create user activity	Pass
6	On click of save button in create user activity	Displays error message if no name (name field is empty) is entered.	Pass
		Saves the user if name is entered and navigates back to user view screen.	
7	On click of '+' button on wish list screen	Navigates to add user activity	Pass
8	On click of user account	Opens the user activity screen to displays all the brands under that user account	Pass
9	On swipe right to left from	Displays list of all the items	Pass

	brands fragment in user activity screen	under all the brands (combined view)	
10	On swipe right to left from all items fragment in user activity screen	Displays add user screen	Pass
11	On swipe right to left from add user fragment in user activity screen	Displays edit user screen	Pass
12	On click of Edit item on expanded items list view	Navigates to edit user screen, editable fields filled with the existing data	Pass
13	On click of View on map link in expanded view	Navigates to Map activity where the location is marked on the map	Pass

Table 7.2 Integration Test Cases

7.3 Compatibility Testing

The Smart Tracker android application is developed to support diverse range of android devices with different screen sizes. Application is tested for compatibility on Samsung galaxy S3 smart phone and Samsung Tab 10.1. Application is also tested for both portrait and landscape modes to verify the orientation. There are few changes in the layout display for smart phone and tablets. The date dialog displays only the scrollable data picker for the android smart phone where as it displays both scrollable view and calendar for tablets to make it user friendly.

The following figures show the difference in the layout for date dialog

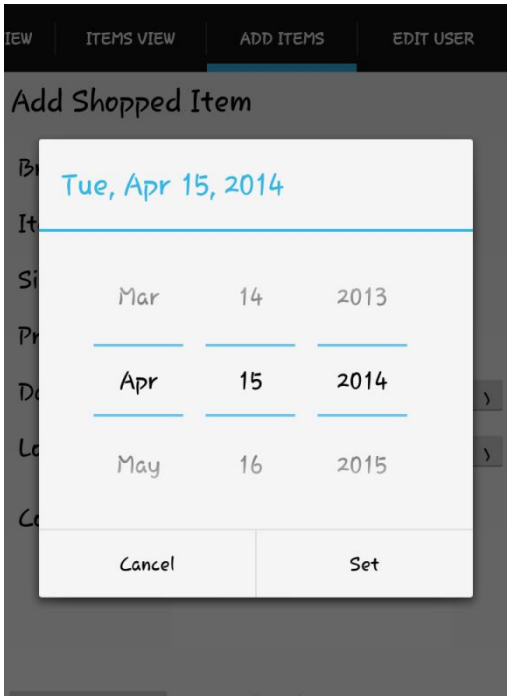

Figure 7.1 Date Dialog on Smart Phone

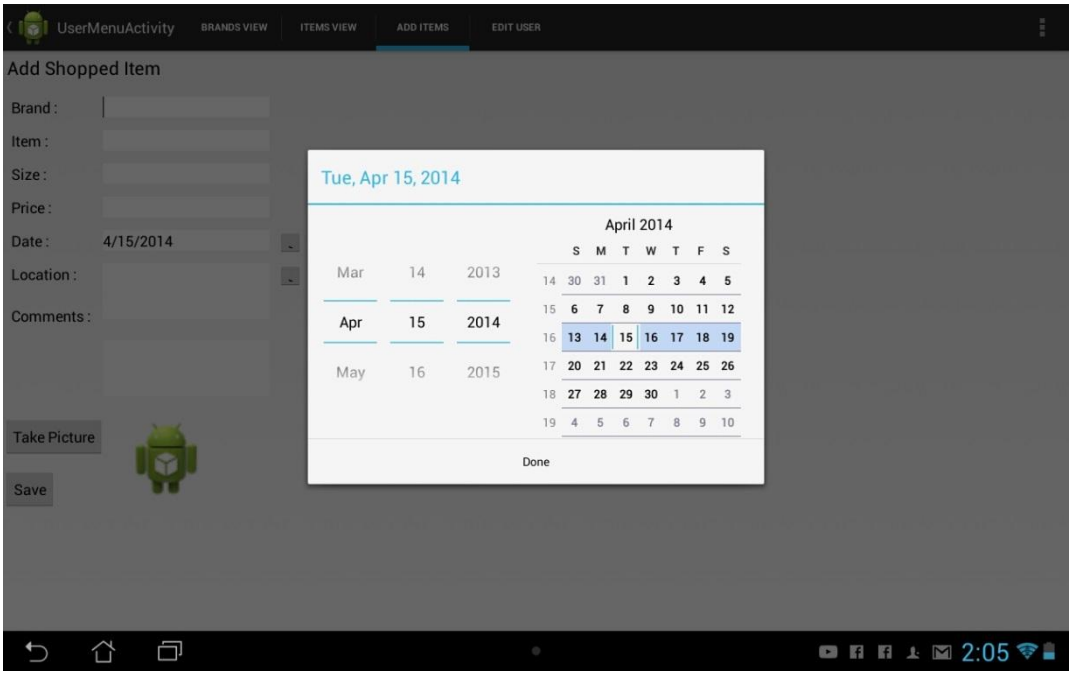

Figure 7.2 Date Dialog on Tablet

7.4 Performance Testing

7.4.1 Screen Transition Performance

The performance of the application is tested for each screen using Traceview tool. Traceview is a graphical viewer which is used to profile the performance of the application. The below table depicts the performance of the application with respect to different screens.

Screen	Response Time (ms)
User Accounts View	0.24
Brands View	0.28
Items view	0.59
Add Item Screen	0.34
Create User	0.21
Edit User	0.23
Edit Item	0.45
Wish list view	0.56
Add wish list item	0.34
Location look up	0.61
Mark location on map	1.2

Table 7.3 Screen and Response times

7.4.2 Number of Items for Display

The application is also tested for performance with respect to number of items retrieved from the database. As the shopped items may contain the images which are stored as blob in the database tables, the time required to retrieve the images increases with the number of images. Performance is measured using JMeter holding the following assumptions:

- User may not add more than 10 user accounts (to track shopping of family and friends)
- Under normal circumstances, user may not add more than 500 pieces of clothing and other accessories to track their shopping.

Below is the graph showing the response times with respect to number of items being displayed as expandable list view when the number of users 5 and 10.

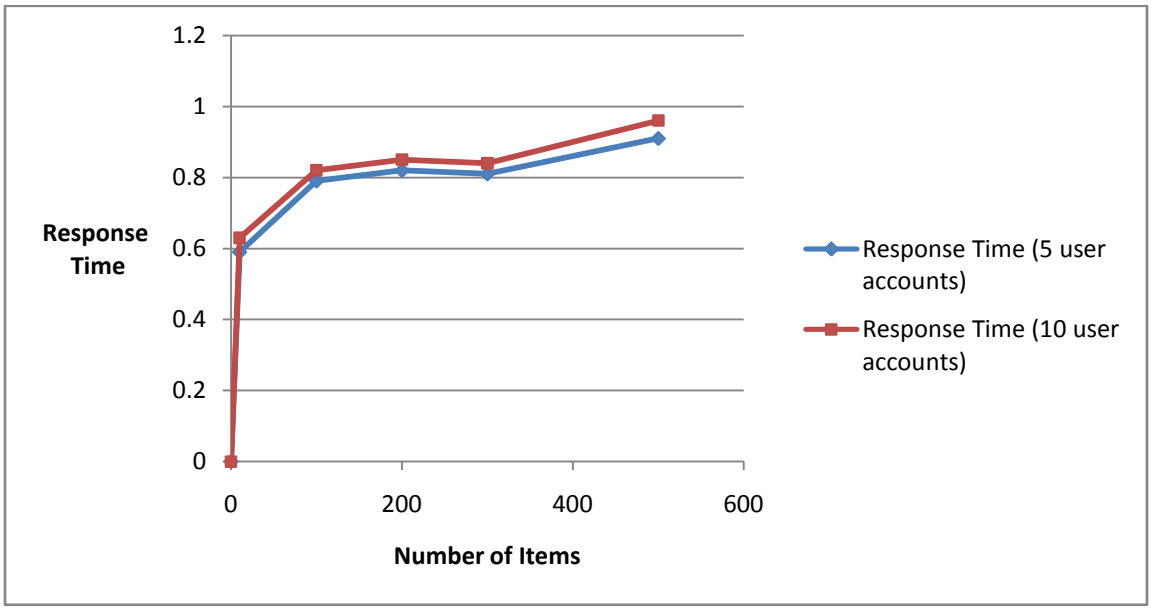

Figure 7.3 Performance graph

7.4.3Performance Analysis

From the performance testing, we can see that the overall performance of the application is good. The response time of the activity is increasing with the number of items stored by the user. The reason behind this is the time taken for retrieval of data from the SQLite database tables. The information of the item also contains the picture data stored in the form of blob which increases the response time. The performance can be increase by storing and retrieving the images from the local folder on the android device but this has a trade off of losing the associated when the images are deleted by the user. The persistent store of images into database helped in avoid such activities but increased the response time.

According to the statistical analysis, the response time required for any activity is less than 2sec after which the user goes for a duplication of click. As the response time is less than 2 sec for all the activities, it can be said that the performance of the application is good.

Chapter 8 - Conclusion

Android is an operating system that is widely used on smart phones and tablets. It is customizable and open source. So, a user can create a new application or develop an existing one to suit his requirements. Also, a lot of documentation about Android application development is available online which makes it relatively easier to design an Android app.

The objective of this report was to design an Android app called Smart Tracker which helps an individual maintain a database of information regarding the items of clothing purchased. The app was designed to work with not only individual purchases but also to track the shopping information of other family member and friends of the user. Also, a provision was made to calculate the total expense made by each user towards shopping.

The application development is done by following complete software development life cycle starting with requirement gathering which is followed by design, development and testing respectively. All the initial requirements are met and the application runs successfully on both Android smart phones and tablets. This project enabled me to learn android application development using latest versions android sdk which was required for implementation of new features like change page on swipe using fragments. It also helped me in understanding and working with SQLite Database and Google Maps V2 API for storing & retrieving the data and showing locations on Google Maps respectively. Android JUnit Testing is done with the help of android test framework which is embedded with the ADT bundle for eclipse. I have also learnt to measure the performance of an Android application using JMeter.

Chapter 9 - Future Work

The Smart Tracker application can be further enhanced to include following features:

- The application can be extended to include scanning of barcode on the price tag which decreases the effort of entering the data in the input fields.
- Features like sharing the content with other users can be added in case of wish lists so that users can view others' shopping.
- The application can be expanded to other items like regular grocery shopping and also to maintain record and rating of food ordered in restaurant etc.

Chapter 10 - Bibliography

1. *Android System Architecture.* **Smieh.** 2012, Anatomy Physiology of an Android.

2. *Activity Life cycle.* [Online] [Cited: February 25, 2014.] http://developer.android.com/training/basics/activity-lifecycle/starting.html.

3. *Android Architecture.* [Online] [Cited: March 15, 2014.] http://commons.wikimedia.org/wiki/File:Android-System-Architecture.svg.

4. *Android basics.* [Online] [Cited: February 15, 2014.] http://developer.android.com/training/basics/.

5. *Focus on List view items.* [Online] [Cited: April 1, 2014.] http://mylifewithandroid.blogspot.com/2011/08/focus-problems-with-list-rows-and.html.

6. *Activity stacking.* [Online] [Cited: March 20, 2014.] http://tips.androidhive.info/2013/10/how-to-clear-all-activity-stack-in-android/.

7. *Custom date Dialog.* [Online] [Cited: March 20, 2014.] http://www.helloandroid.com/tutorials/how-display-custom-dialog-your-android-application.

8. *Simple use of Google maps.* [Online] [Cited: March 28, 2014.] http://android-er.blogspot.com/2012/12/a-simple-example-using-google-maps.html.

9. *Location look up.* [Online] [Cited: March 26, 2014.] http://android-er.blogspot.com/2013/04/search-address-by-name-with-geocoder.html.

10. *Google Maps V2 API.* [Online] [Cited: March 25, 2014.] https://developers.google.com/maps/documentation/android/.

11. *Google Maps Key generation.* [Online] [Cited: March 19, 2014.] https://console.developers.google.com/project.

12. *Goolge Maps.* [Online] [Cited: March 27, 2014.] http://www.androidhive.info/2013/08/android-working-with-google-maps-v2/.