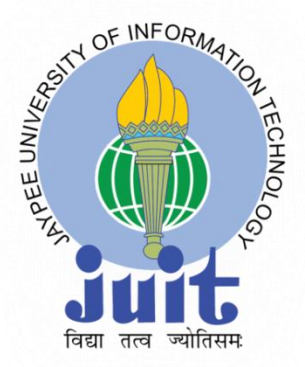
ACCOUNTS TRACKER ANDROID APPLICATION

ENROLLMENT NO. : 101298

NAME : YUGAL ARORA

PROJECT GUIDE : MS. ANSUYIA MAKROO



May 2014

Submitted in partial fulfillment of the Degree of

Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

Certificate

This is to certify that the work titled **— "Accounts Tracker Android Application**", submitted by **Yugal Arora** in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor

Name of Supervisor Ms. Ansuyia Makroo

Date

Acknowledgement

As I conclude our project with the God's grace, I have many people to thank; for all the help, guidance and support they lent me, throughout the course of my endeavor.

First and foremost, I am sincerely thankful to Ms. Ansuyia Makroo, my Project Guide, who has always encouraged me to put in my best efforts and deliver a quality and professional output. Her methodology of making the system strong from inside has taught me that output is not the END of project. I really thank her for her time & efforts.

I am deeply indebted to all those who provided reviews and suggestions for improving the materials and topics covered in my project, and I extend my apologies to anyone I may have failed to mention.

Enrollment number: 101298
Name : Yugal Arora

Signature

INDEX

ABSTRACT

This Android application allows user to keep track of current balance in different bank accounts held by the user and the transactions of those accounts Account tracker explains about implementing a app for android mobiles which will help users to know about bank balances in different banks and there transactions information. In present trend usage of apps had became a new trend because of availability of web services on mobiles. By considering these improvements in mobile technology knowing information of money transactions through mobile in less time can be useful application for users. In this application initially users need to install app and update details like listing out different banks and adding new bank accounts.

App provides following functionalities :

Add New Account: Using this feature we can add new account to list . User should enter customer number, account holder, bank name, branch Name, branch Address, IFSC code, MICR , Current Balance, Remarks.

Add New Transaction: Using this feature we can send money to account holder who is listed in our account. This account has account type, date, amount, cheque Number, cheque party, cheque details, Remarks.

Search Transactions: As we check bank transactions for every month and year here we have this option to select date from and to options which will help us to know transaction details along with from amount to amount.

I hope this project will be of great help to people from various age groups and professions as it will minimize the use of hard copies .

INTRODUCTION

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developers to write managed code in a Java-like language that utilizes Google-developed Java libraries, but does not support programs developed in native code.

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently.

The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

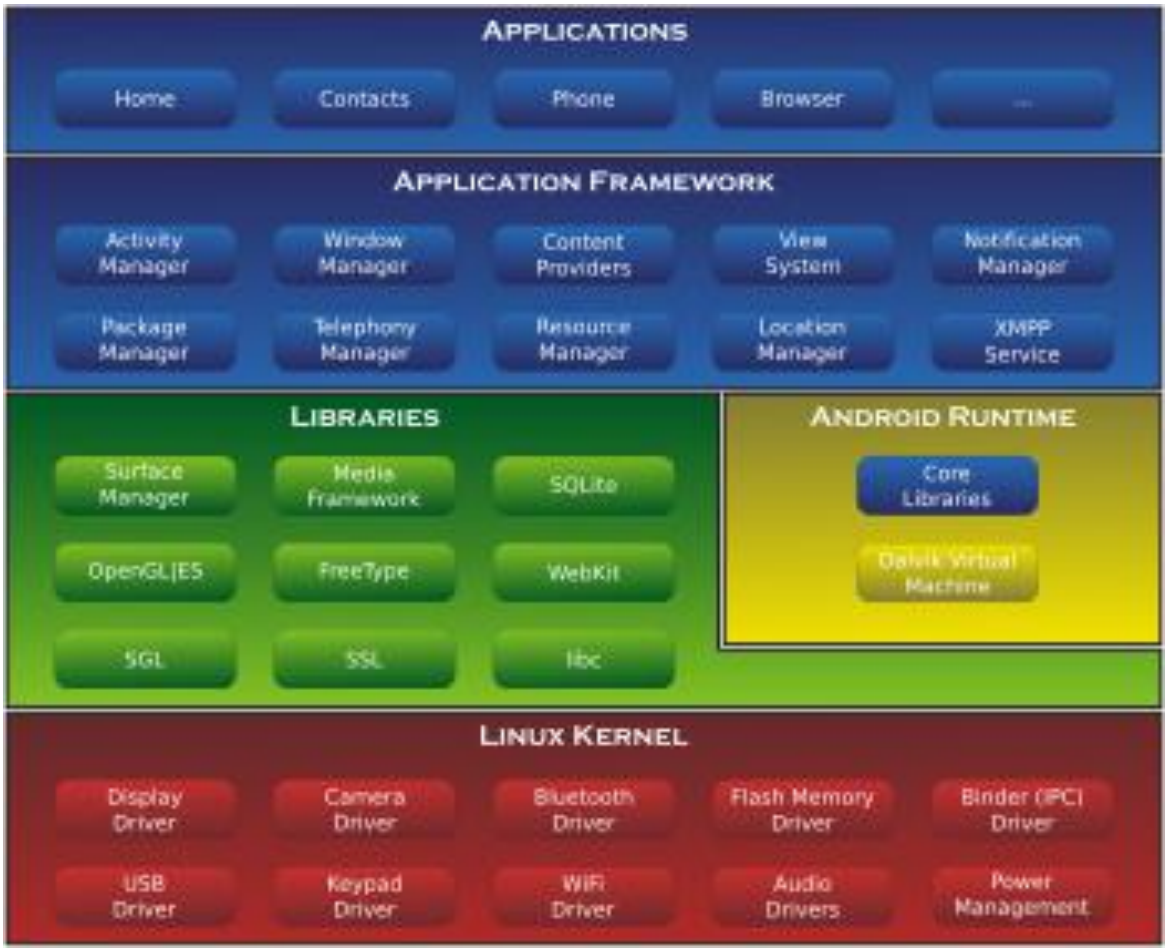
It helps to manage security, memory management, process management, network stack and other important issues. Therefore, the user should bring Linux in his mobile device as the main operating system and install all the drivers required in order to run it.

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.Underlying all applications is a set of services and systems.

CH. 1 : LITERATURE SURVEY

1.1 : ARCHITECTURE

The following diagram shows the major components of the Android operating system. Each section is described in more detail below



1.2 Libraries

The Android Support Library package is a set of code libraries that provide backward-compatible versions of Android framework APIs as well as features that are only available through the library APIs. Each Support Library is backward-compatible to a specific Android API level. This design means that your applications can use the libraries' features and still be compatible with devices running Android 1.6 (API level 4) and up.

This guide provides information about what features are enabled by the Support Libraries, how to use them in your development environment and information about library releases.

Overview

Including the Support Libraries in your Android project is considered a best practice for application developers, depending on the range of platform versions your app is targeting and the APIs that it uses. Using the features the libraries provide can help you improve the look of your application, increase performance and broaden the reach of your application to more users. If you use the Android code template_tools, you will notice that all the Android application templates include one or more of the Support Libraries by default.

The Support Libraries each target a base Android API level and each provides a different set of features. In order to effectively use the libraries, it is important to consider what features you want to support and understand what features are supported by each library at what Android API level. To get started, review the Support Library Features guide. After that, go to the Support Library Setup topic to learn how to incorporate the Support Libraries into your application. For more details about Support Library APIs, see the android.support packages in the API reference

Support Library Features

The Android Support Library package contains several libraries that can be included in your application. Each of these libraries supports a specific range of Android platform versions and set of features.

This guide explains the important features and version support provided by the Support Libraries to help you decide which of them you should include in your application. In general, we recommend including the v4 support and v7 appcompat libraries, because they support a wide range of Android versions and provide APIs for recommended user interface patterns.

In order to use any of the following libraries, you must download the library files to your Android SDK installation. Follow the directions for downloading the Support Libraries in Support Library Setup to complete this step. You must take additional steps to include a specific Support Library in your

application. See the end of each library section below for important information on how to include the library in your application

1.3 Android Runtime

At the same level there is Android Runtime, where the main component Dalvik Virtual Machine is located. It was designed specifically for Android running in limited environment, where the limited battery, CPU, memory and data storage are the main issues. Android gives an integrated tool "dx", which converts generated byte code from .jar to .dex file, after this byte code becomes much more efficient to run on the small processors.

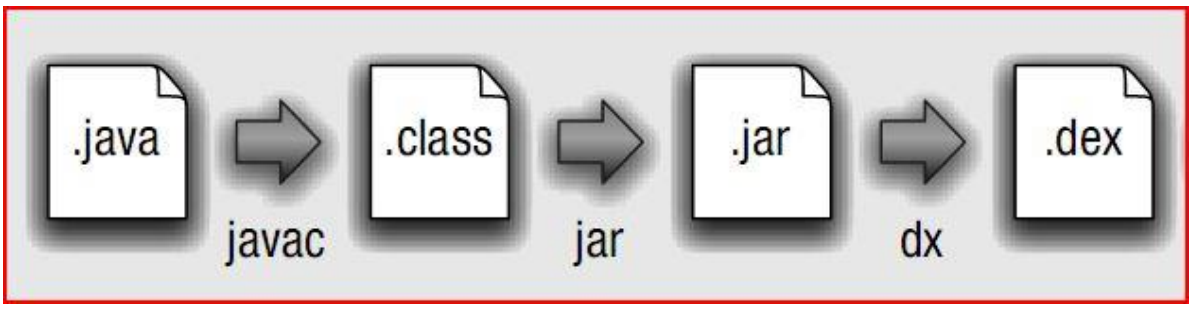


Figure 2.2 : Conversion from .java to .dex file

As the result, it is possible to have multiple instances of Dalvik virtual machine running on the single device at the same time. The Core libraries are written in Java language and contains of the collection classes, the utilities, IO and other tools.

1.4 MANAGING PROJECTS

Projects act as containers for storing things such as code and resource files. The SDK tools expect your projects to follow a specific structure so it can compile and package your application correctly, so it is highly recommended that you create them with Eclipse and ADT or with the android tool on the command line. There are three types of projects, and they all share the same general structure but differ in function:

Android Projects

Android projects are the projects that eventually get built into an .apk file that you install onto a device. They contain things such as application source code and resource files. Some are generated for you by default, while others should be created if required. The following directories and files comprise an Android project:

src/

Contains your stub Activity file, which is stored at src/*your/package/namespace/ActivityName*.java. All other source code files (such as .java or .aidl files) go here as well.

bin/

Output directory of the build. This is where you can find the final .apk file and other compiled resources.

jni/

Contains native code sources developed using the Android NDK. For more information, see the Android NDK documentation.

gen/

Contains the Java files generated by ADT, such as your R.java file and interfaces created from AIDL files.

assets/

This is empty. You can use it to store raw asset files. Files that you save here are compiled into an .apk file as-is, and the original filename is preserved. You can navigate this directory in the same way as a typical file system using URIs and read files as a stream of bytes using the AssetManager. For example, this is a good location for textures and game data.

res/

Contains application resources, such as drawable files, layout files, and string values. See Application Resources for more information.

anim/

For XML files that are compiled into animation objects. See the Animation resource type.

color/

For XML files that describe colors. See the Color Values resource type.

drawable/

For bitmap files (PNG, JPEG, or GIF), 9-Patch image files, and XML files that describe Drawable shapes or Drawable objects that contain multiple states (normal, pressed, or focused). See the Drawable resource type.

layout/

XML files that are compiled into screen layouts (or part of a screen). See the Layout resource type.

menu/

For XML files that define application menus. See the Menus resource type.

raw/

For arbitrary raw asset files. Saving asset files here instead of in the assets/ directory only differs in the way that you access them. These files are processed by aapt and must be referenced from the application using a resource identifier in the R class. For example, this is a good place for media, such as MP3 or Ogg files.

values/

For XML files that are compiled into many kinds of resource. Unlike other resources in the res/ directory, resources written to XML files in this folder are not referenced by the file name. Instead, the XML element type controls how the resources is defined within them are placed into the R class.

xml/

For miscellaneous XML files that configure application components. For example, an XML file that defines aPreferenceScreen, AppWidgetProviderInfo, or Searchability Metadata. See Application Resources for more information about configuring these application components.

libs/

Contains private libraries.

AndroidManifest.xml

The control file that describes the nature of the application and each of its components. For instance, it describes: certain qualities about the activities, services, intent receivers, and content providers; what permissions are requested; what external libraries are needed; what device features are required, what API Levels are supported or required; and others. See the AndroidManifest.xml documentation for more information

project.properties

This file contains project settings, such as the build target. This file is integral to the project, so maintain it in a source revision control system. To edit project properties in Eclipse, right-click the project folder and select**Properties**.

local.properties

Customizable computer-specific properties for the build system. If you use Ant to build the project, this contains the path to the SDK installation. Because the content of the file is specific to the local installation of the SDK, thelocal.properties should not be maintained in a source revision control system. If you use Eclipse, this file is not used.

ant.properties

Customizable properties for the build system. You can edit this file to override default build settings used by Ant and also provide the location of your keystore and key alias so that the build tools can sign your application when building in release mode. This file is integral to the project, so maintain it in a source revision control system. If you use Eclipse, this file is not used.

build.xml

The Ant build file for your project. This is only applicable for projects that you build with Ant.

1.5 Application Framework

After that, there is Application Framework, written in Java language. It is a toolkit that all applications use, ones which come with mobile device like Contacts or SMS box, or applications written by Google and any Android developer. It has several components.

The Activity Manager manages the life circle of the applications and provides a common navigation back stack for applications, which are running in different processes. The Package Manager keeps track of the applications, which are installed in the device. The Windows Manager is Java programming language abstraction on the top of lower level services that are provided by the Surface Manager.

The Telephony Manager contains of a set of API necessary for calling applications.

Content Providers was built for Android to share a data with other applications, for instance, the contacts of people in the address book can be used in other applications too. The Resource Manager is used to store localized strings, bitmaps, layout file descriptions and other external partsof the application. The View System generates a set of buttons and lists used in UI. Other components like Notification manager is used to customize display alerts and other functions.

1.6 Application Layer

At the top of Android Architecture we have all the applications, which are used by the final user. By installing different applications, the user can turn his mobile phone into the unique, optimized and smart mobile phone. All applications are written using the Java programming language.

1.7 DEVELOPING APPLICATIONS

Application Building Blocks

We can think of an Android application as a collection of components, of various kinds. These components are for the most part quite loosely coupled, to the degree where you can accurately describe them as a federation of components rather than a single cohesive application.

Generally, these components all run in the same system process. It's possible (and quite common) to create multiple threads within that process, and it's also possible to create completely separate child processes if you need to. Such cases are pretty uncommon though, because Android tries very hard to make processes transparent to your code.

Google provides three versions of SDK for Windows, for Mac OSX and one for Linux.The developer can use Android plugin for Eclipse IDE or other IDEs such as intelliJ.First step for Android developer is to decompose the prospective application into the components, which are supported by the platform. The major building blocks are these:

- Activity
- Intent Receiver
- Service
- Content Provider

Activity

User interface component, which corresponds to one screen at time. It means that for the simple application like Address Book, the developer should have one activity for displaying contacts, another activity component for displaying more detailed information of chosen name and etc.

Intent Receiver

Wakes up a predefined action through the external event. For example,for the application like Email Inbox, the developer should have intent receiver and register his code through XML to wake up an alarm notification, when the user receives email.

Service

A task, which is done in the background. It means that the user can start an application from the activity window and keep the service work, while browsing other applications. For instance, he can browse Google Maps application while holding a call or listening music while browsing other applications.

Content Provider

A component, which allows sharing some of the data with other processes and applications. It is the best way to communicate the applications between each other. Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

1.8 AndroidManifest.xml

The AndroidManifest.xml file is the control file that tells the system what to do with all the top-level components (specifically activities, services, intent receivers, and content providers described below) you've created. For instance, this is the "glue" that actually specifies which Intents your Activities receive.

A developer should predefine and list all components, which he wants to use in the specific AndroidManifest.xml file. It is a required file for all the applications and is located in the root folder. It is possible to specify all global values for the package, all the components and its classes used, intent filters, which describe where and when the certain activity should start, permissions and instrumentation like security control and testing.

Here is an example of AndroidManifest.xml file:

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3. package="dk.mdev.android.hello">
4. <application android:icon="@drawable/icon">
5. <activity class=".HelloAndroid" android:label="@string/app_name">
6. <intent-filter>
```

7. <action android:value="android.intent.action.MAIN" />
8. <category android:value="android.intent.category.LAUNCHER"/>
9. </intent-filter>
10. </activity>
11. </application>
12. </manifest>

The line 2 is a namespace declaration, which makes a standard Android attributes available for that application. In the line 4 there is a single <application> element, where the developer specifies all application level components and its properties used by the package. Activity class in the line 5 represents the initial screen the user sees and it may have one or more <intent-filter> elements to describe the actions that activity supports.

1.9 ECLIPSE

ANDROID SDK

Android Software development kit provides with API Libraries and tools necessary to build , test and debug applications for Android .

ACCOUNT TRACKER has been built in Eclipse which comes bundled in Android SDK .

After downloading android sdk tools we run eclipse .

The ADT plugin provides a *New Project Wizard* that you can use to quickly create a new Android project (or a project from existing code). To create a new project:

1. Select **File** > **New** > **Project**.
2. Select **Android** > **Android Application Project**, and click **Next**.
3. Enter the basic settings for the project:
   - Enter an **Application Name**. This name is used as the title of your application launcher icon when it is installed on a device.
   - Enter a **Project Name**. This text is used as the name of the folder where your project is created.
   - Enter a **Package Name**. This class package namespace creates the initial package structure for your applications code files and is added as the package attribute in your application's Android manifest file. This manifest value serves as the unique identifier for your application app when you distribute it to users. The package name must follow the same rules as packages in the Java programming language.

- Select a **Minimum Required SDK**. This setting indicates the lowest version of the Android platform that your application supports. This value sets the minSdkVersion attribute in the <uses-sdk> element of your manifest file.
- Select a **Target SDK**. This setting indicates the highest version of Android with which you have tested with your application and sets the targetSdkVersion attribute in your application's' manifest file.
- Select a **Compile With** API version. This setting specifies what version of the SDK to compile your project against. We strongly recommend using the most recent version of the API.
- Select a **Theme**. This setting specifies which standard Android visual style is applied to your application.
- Click **Next**.

4. In the **Configure Project** page, select the desired settings and click **Next**. Leave the **Create activity** option checked so you can start your application with some essential components.
5. In the **Configure Launcher Icon** page, create an icon and click **Next**.
6. In the **Create Activity** page, select activity template and click **Next**. For more information about Android code templates, see Using Code Templates.
7. Click **Finish** and the wizard creates a new project according to the options you have chosen.

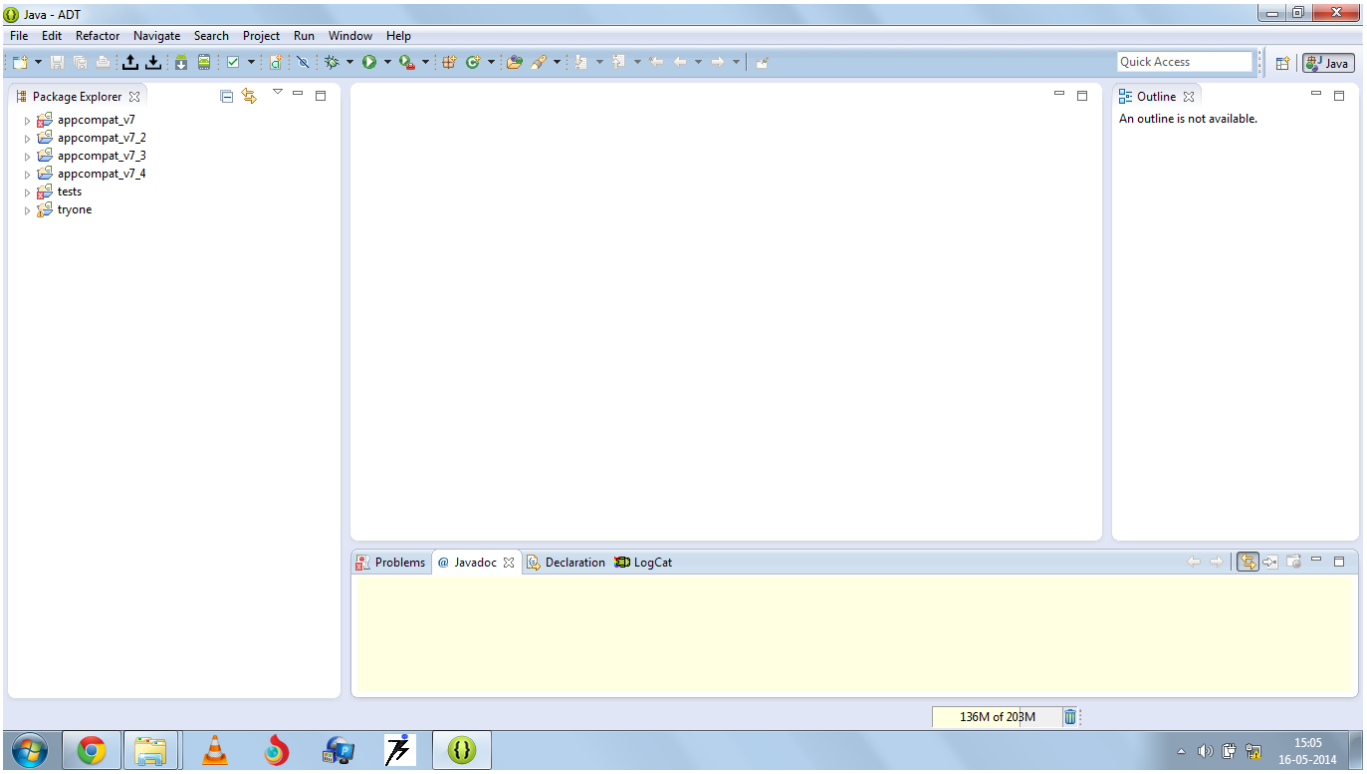


Fig. 1 : eclipse

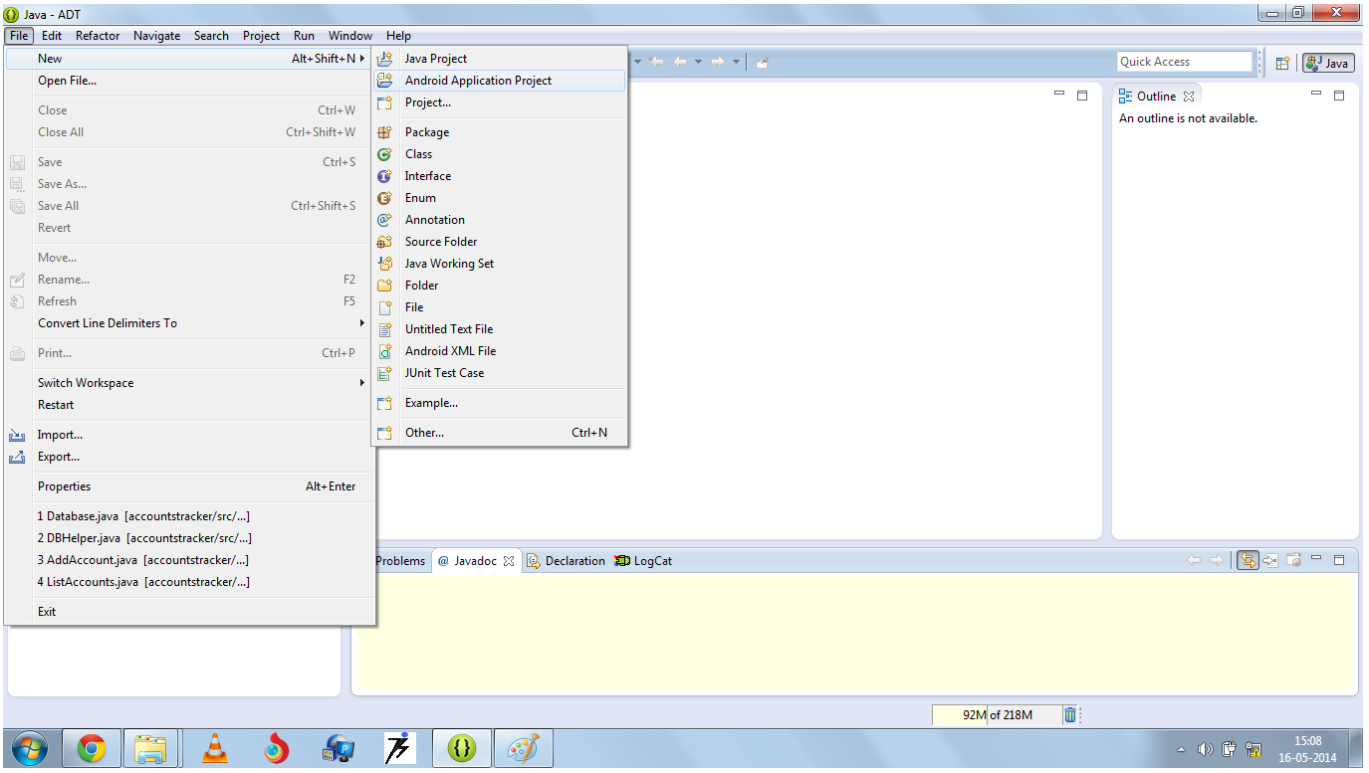


Fig 2 : creating new android project.

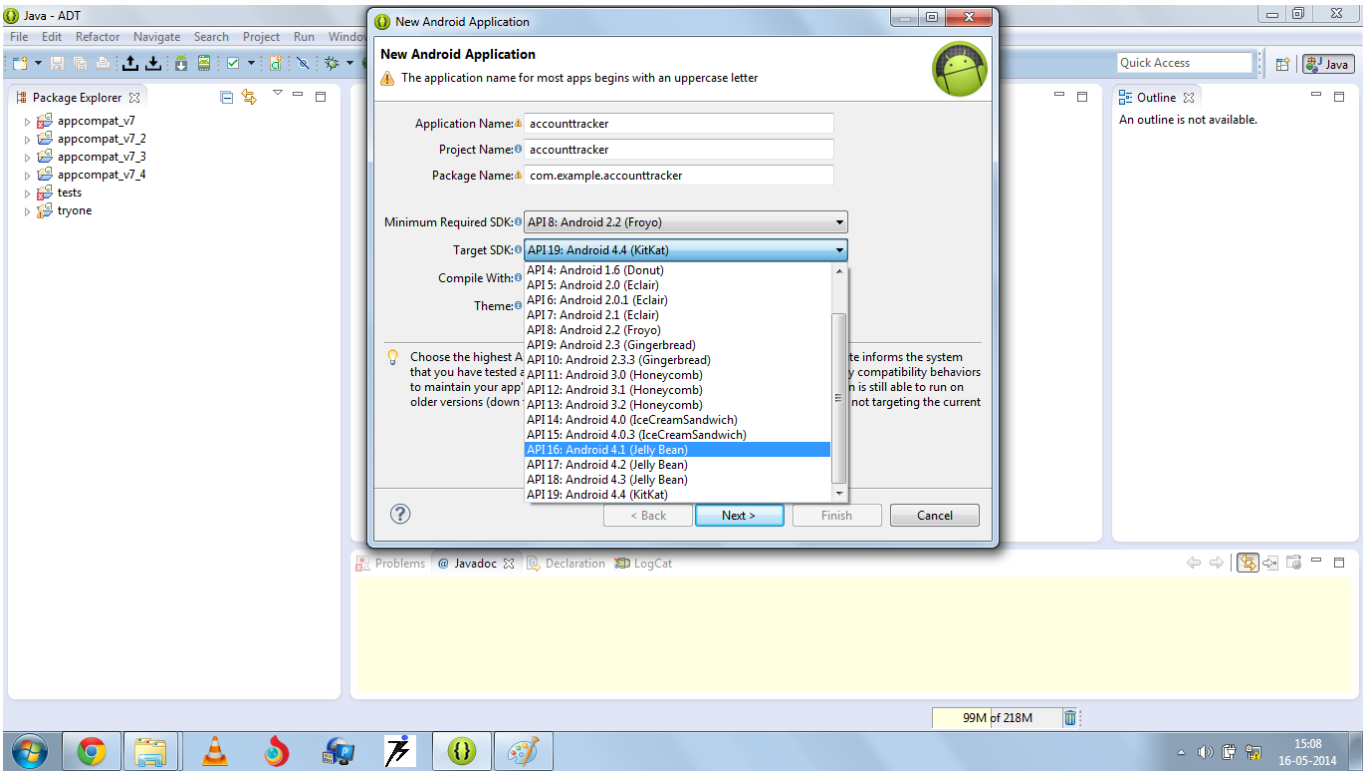


Fig 3 : we enter name of project and select the target api i.e 4.1.2 (jelly bean) for account tracker.

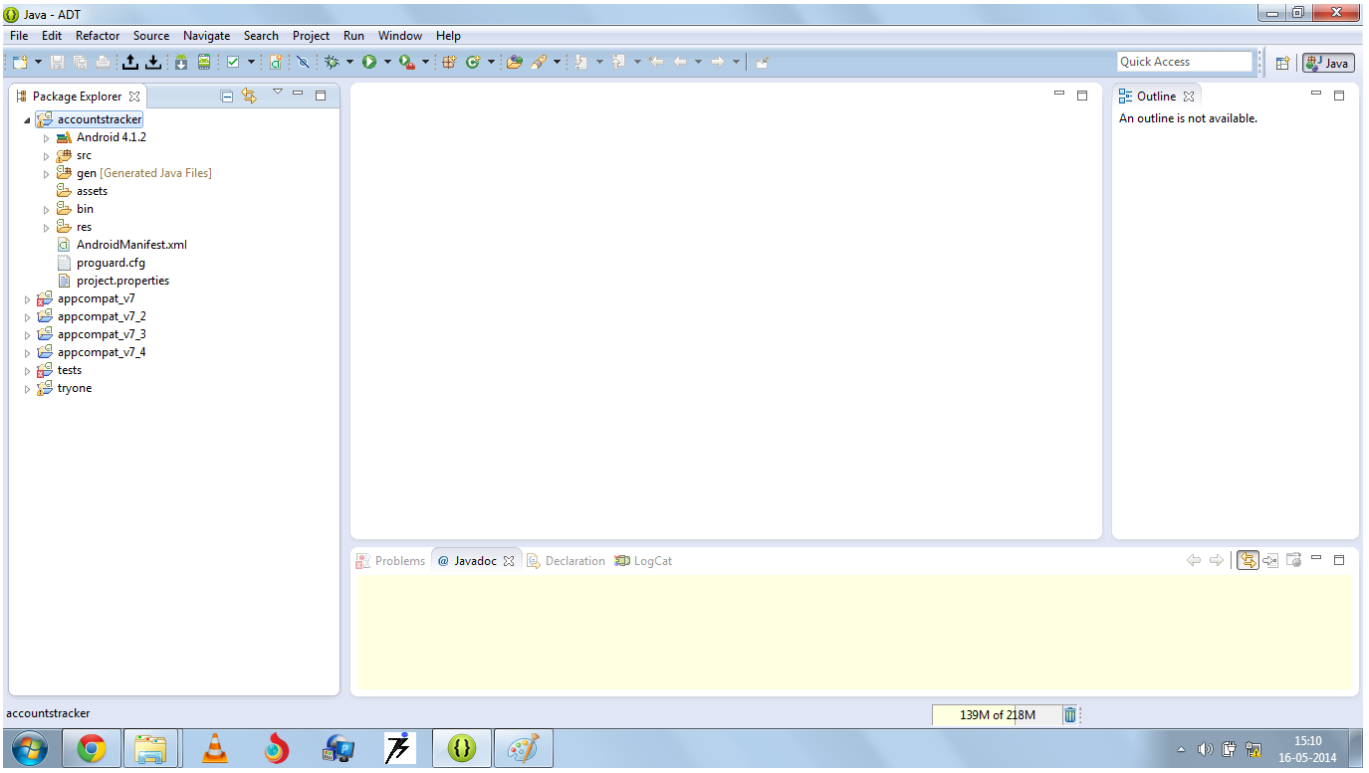


Fig 4 : now a new project has been created.

Under project name "accountstracker" we see various folders in package explorer , these are "src", "gen" , "assets" , "bin" , "res" and "AndroidManifest.xml" .

Now we expand each of these folders to see necessary files and edit them.

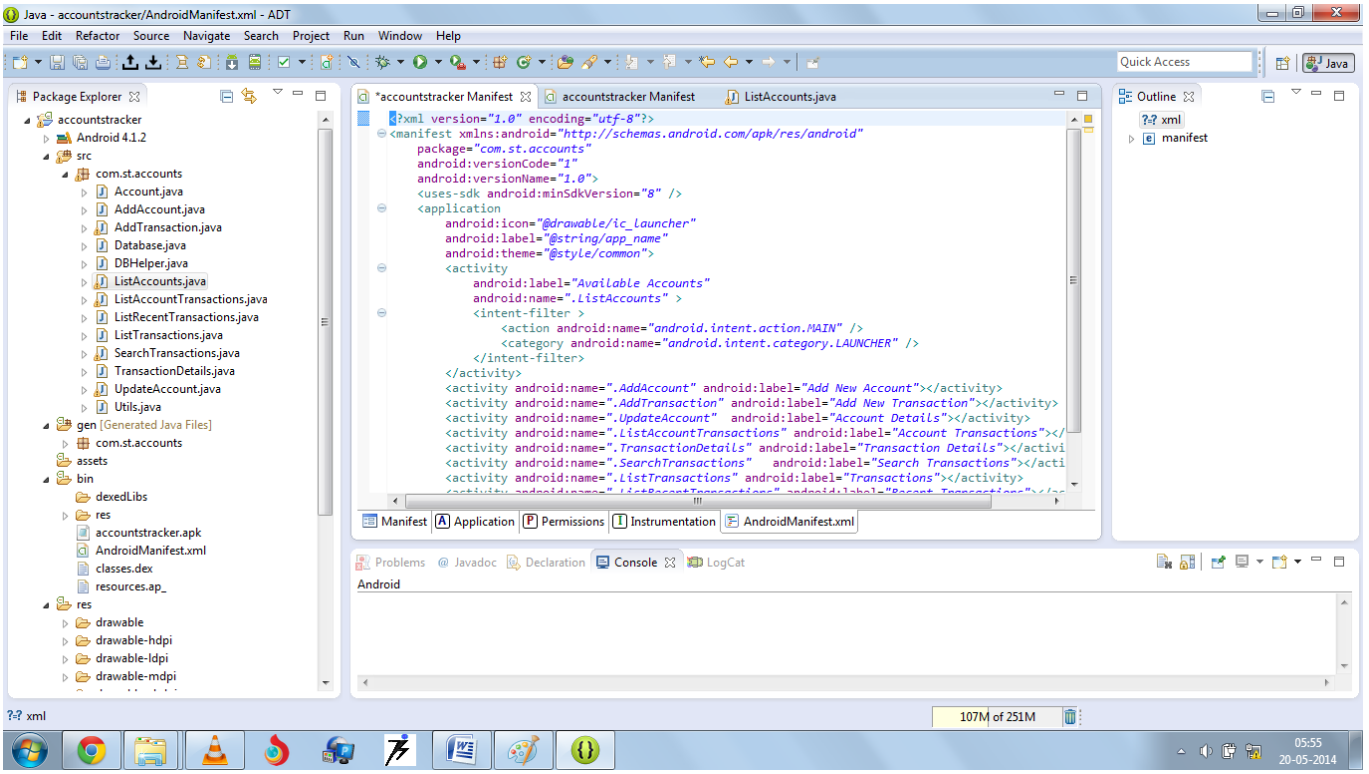


Fig 5 : we can see the files in src,gen,res folders .

CH 2 : DESIGN AND IMPLEMENTATION

2.1 ANDROID MANIFEST FILE

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.st.accounts"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/common">
        <activity
            android:label="Available Accounts"
            android:name=".ListAccounts" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".AddAccount" android:label="Add New Account"></activity>
        <activity android:name=".AddTransaction" android:label="Add New
Transaction"></activity>
        <activity android:name=".UpdateAccount"  android:label="Account Details"></activity>
        <activity android:name=".ListAccountTransactions" android:label="Account
Transactions"></activity>
        <activity android:name=".TransactionDetails" android:label="Transaction
Details"></activity>
        <activity android:name=".SearchTransactions"   android:label="Search
Transactions"></activity>
        <activity android:name=".ListTransactions" android:label="Transactions"></activity>
        <activity android:name=".ListRecentTransactions" android:label="Recent
Transactions"></activity>
    </application>

                                        </manifest>
```

2.2 LISTACCOUNTS.JAVA FILE

```
package com.st.accounts;

import android.app.Activity;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.os.Bundle;

import android.util.Log;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ListView;

import android.widget.SimpleCursorAdapter;

import android.widget.TextView;

import android.widget.Toast;

public class ListAccounts extends Activity {

ListView listAccounts;

@Override

public void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.listaccounts);
listAccounts = (ListView) this.findViewById(R.id.listAccounts);
listAccounts.setOnItemClickListener( new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View selectedView, int arg2,long arg3) {
TextView textAccountId = (TextView) selectedView.findViewById(R.id.textAccountId);
Log.d("Accounts", "Selected Account Id : " + textAccountId.getText().toString());
Intent intent = new Intent(ListAccounts.this, UpdateAccount.class);
intent.putExtra("accountid", textAccountId.getText().toString());
startActivity(intent);
}
});
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
return Utils.inflateMenu(this,menu);
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
return Utils.handleMenuOption(this,item);
}
@Override
public void onStart() {
```

```
super.onStart();

try {

DBHelper dbhelper = new DBHelper(this);

SQLiteDatabase db = dbhelper.getReadableDatabase();

Cursor accounts = db.query(
Database.ACCOUNTS_TABLE_NAME,null,null,null,null,null,null);

String from [] = { Database.ACCOUNTS_ID, Database.ACCOUNTS_BANK,
Database.ACCOUNTS_HOLDERS, Database.ACCOUNTS_BALANCE };

int to [] = { R.id.textAccountId,R.id.textBank, R.id.textHolder, R.id.textBalance};

SimpleCursorAdapter ca = new SimpleCursorAdapter(this,R.layout.account, accounts,from,to);

ListView listAccounts = (ListView) this.findViewById( R.id.listAccounts);

listAccounts.setAdapter(ca);

dbhelper.close();

} catch (Exception ex) {

Toast.makeText(this, ex.getMessage(), Toast.LENGTH_LONG).show();

}

}

public void addAccount(View v)

{

Intent intent = new Intent(this,AddAccount.class);

startActivity(intent);

}

public void addTransaction(View v)

{

Intent intent = new Intent(this,AddTransaction.class);
```

```
startActivity(intent);

}

public void recentTransactions(View v)

{

Intent intent = new Intent(this,ListRecentTransactions.class);

startActivity(intent);

}

}
```

2.3 COMMON_MENU.XML FILE :

```
<?xml version="1.0" encoding="utf-8" ?>
- <menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/optAddTransaction" android:icon="@drawable/ic_add_transaction"
      android:title="Add Transaction" />
  <item android:id="@+id/optSearchTransactions" android:icon="@drawable/ic_search_transactions"
      android:title="Search Transactions" />
  <item android:id="@+id/optAddAccount" android:icon="@drawable/ic_add_account" android:title="Add
      Account" />
  <item android:id="@+id/optListAccounts" android:icon="@drawable/ic_list_accounts"
      android:title="List Accounts" />
```

```
<item android:id="@+id/optRecentTransactions" android:icon="@drawable/ic_recent_transactions"
    android:title="Recent Transactions" />
  </menu>
```

2.4 LISTACCOUNTS.XML FILE :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
   <ListView
      android:id="@+id/listAccounts"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" >
   </ListView>
   <TextView
      android:textColor="#0000ff"
      android:textStyle="bold"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:text="Tap on the account to get details!" />
   <LinearLayout
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="horizontal">
<Button
   android:id="@+id/btnAddAccount"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Add  Account"
   android:onClick="addAccount" />

<Button
   android:id="@+id/btnAddTransaction"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Add  Trans"
   android:onClick="addTransaction" />

<Button
   android:id="@+id/btnRecentTransactions"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Recent  Trans"
   android:onClick="recentTransactions" />
```

</LinearLayout>

</LinearLayout>

2.5 RUNNING APP SCREENSHOTS

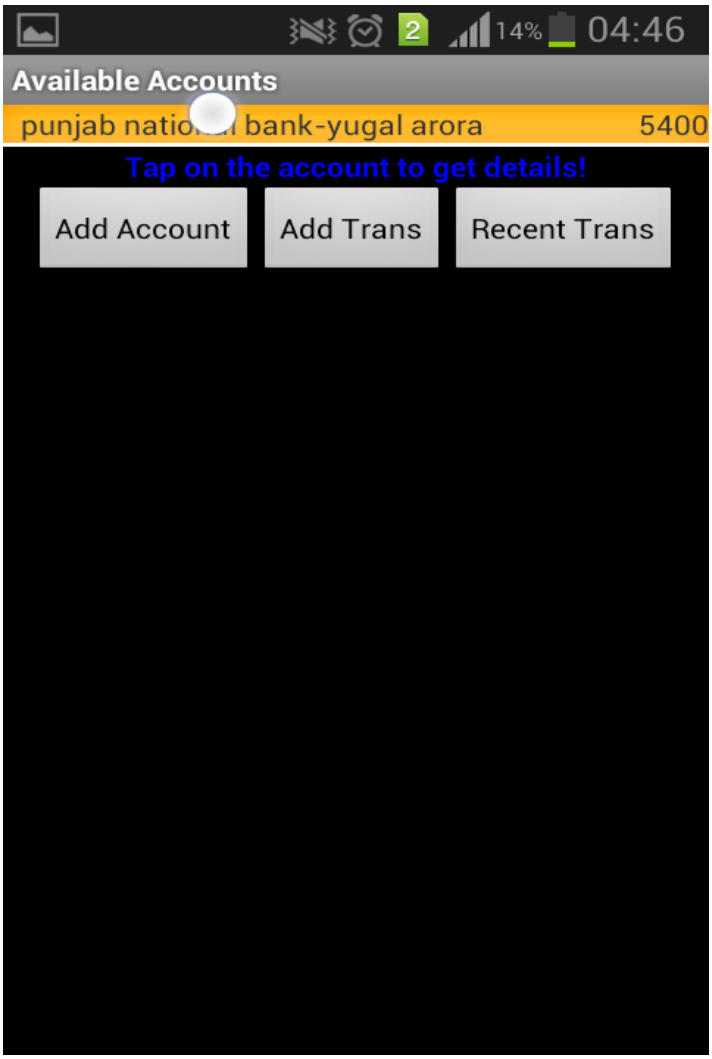


FIG 1 : ACCOUNTSTRACKER HOME SCREEN SHOWING A LIST OF ACCOUNTS IN AVAILABLE ACCOUNTS AND THREE BUTTONS TO PERFORM ADDITIONAL OPERATIONS.

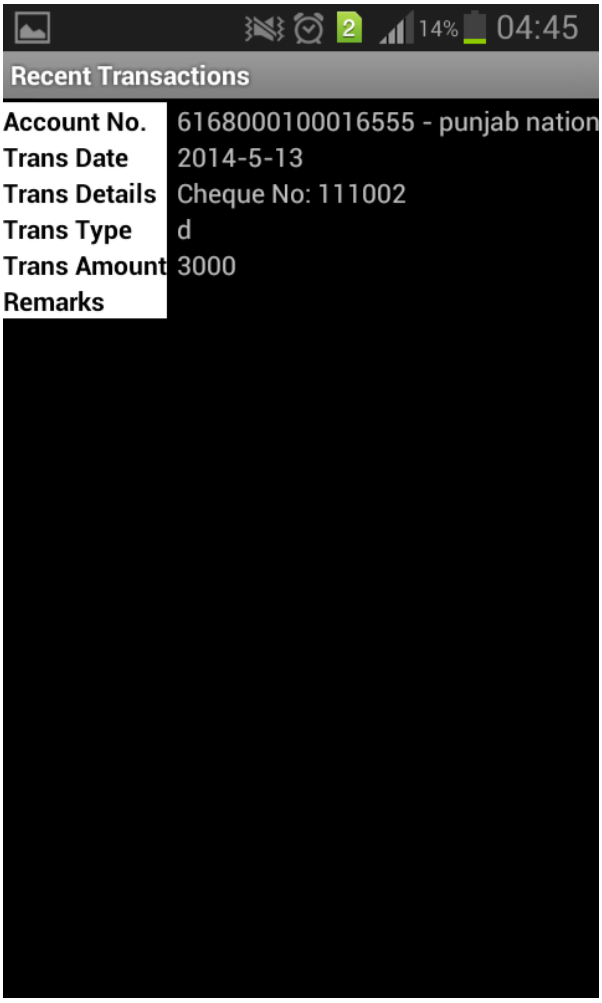


FIG 2 : IT SHOWS RECENT TANSACTIONS.THIS SCREEN OPENS WHEN USER SELECTS RECENT TRANSACTIONS OPTION FROM THE MENU.

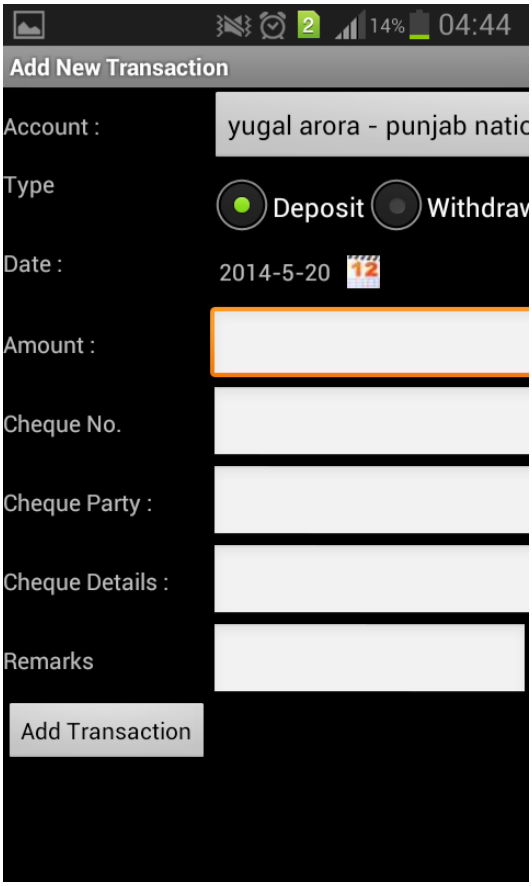


FIG 3 : IT SHOWS ADD NEW TRANSACTION SCREEN.

FIG 4 : IT SHOWS OPTION TO SELECT DATE IN ADD NEW TRANSACTION MENU.

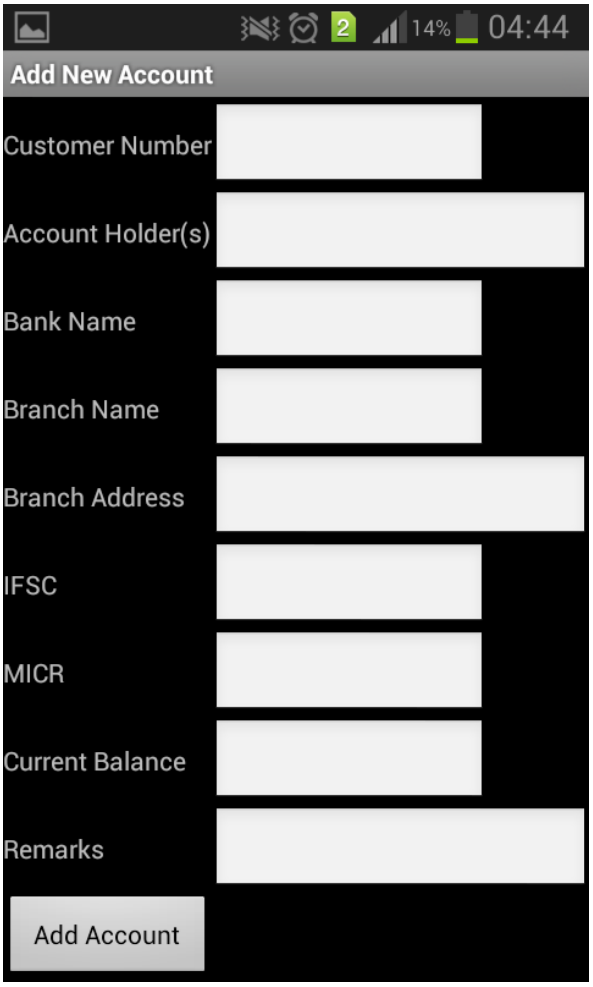


FIG 5 : IT SHOWS OPTION TO ADD NEW ACCOUNT .

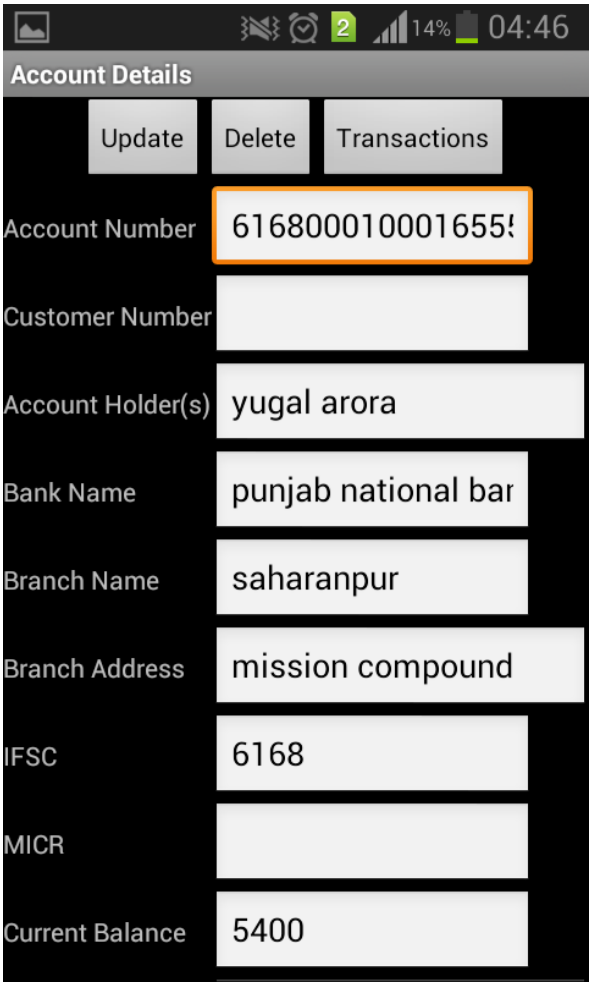
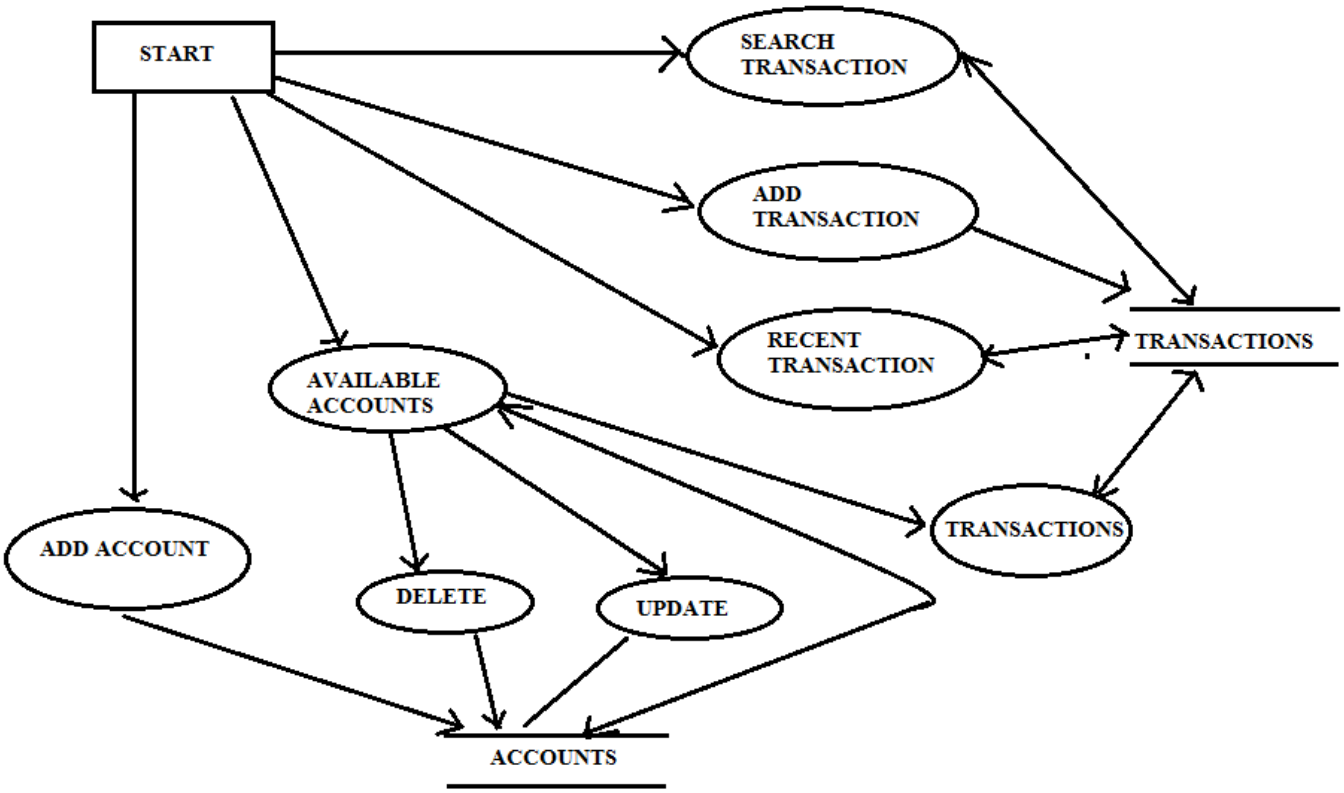
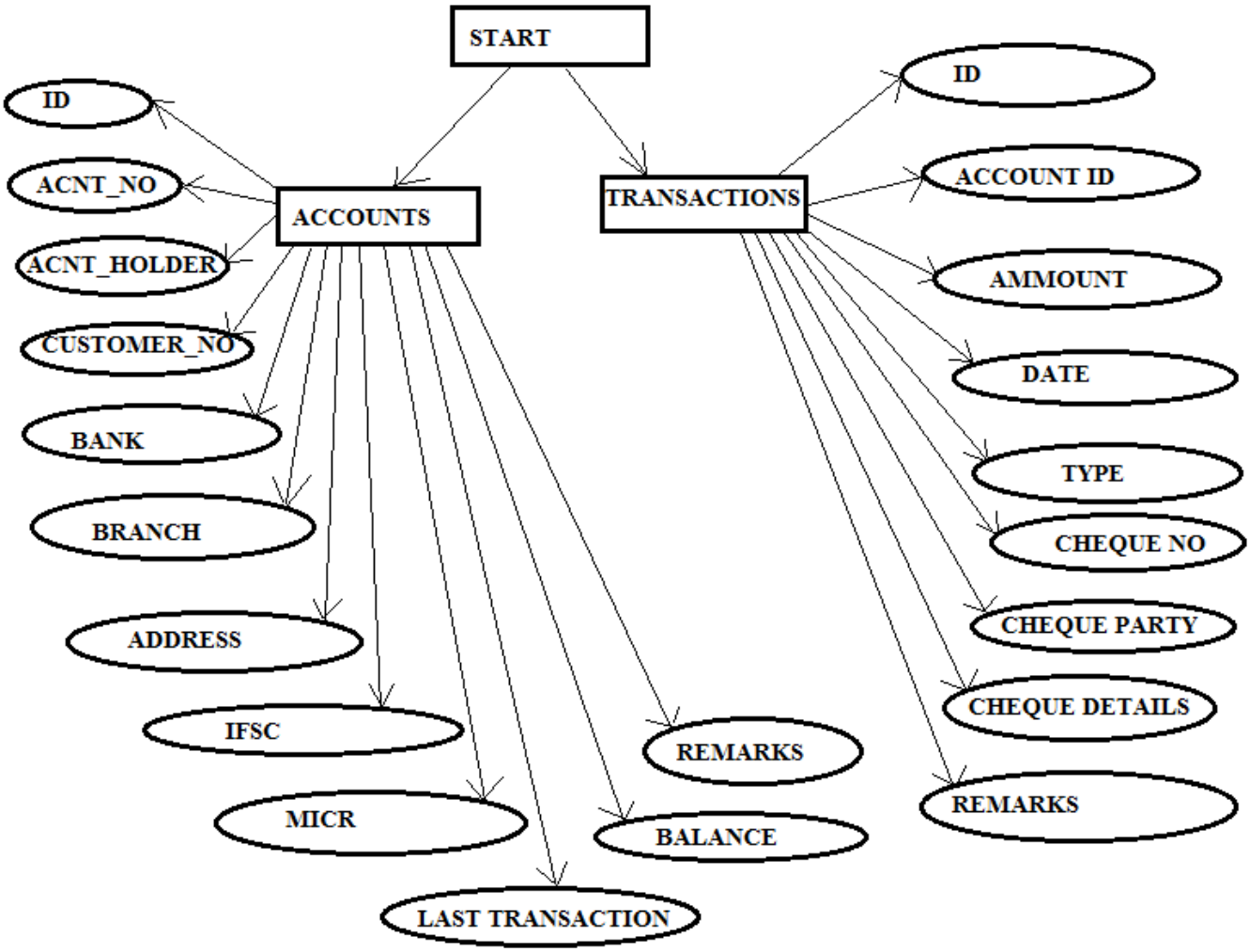


FIG 6 : IT SHOWS ACCOUNT DETAILS MENU.

CH 3.1 : DATA FLOW DIAGRAM :



3.2 :ENTITY RELATIONSHIP DIAGRAM :



CH 4 :FUTURE WORK

Just like all Btech Projects this app is not also perfect and it is intended to act as a stepping stone for further more advanced apps with better functionality.

1. App will be uploaded on android market so that users can download and rate it.

2. Certain other features will also be added to app like login facility with username and password option , upgrade support , synchronization between app data and users pc .

3. Skins , themes can also be added to the app to make it more attractive.

4. Advertisements can also be displayed in the app to generate revenue from it and app can also be made paid if users like it and it receives good rating.

5. A notifier for national holidays ahead can also be added to the app.

REFERENCES

- www.developer.android.com
- www.openhandsetalliance.com/**android**_overview.html
- www.**android**central.com
- https://**developer**.**android**.com/training/basics
- www.computerworld.com › Applications › App Development