TATA CONSULTANCY SERVICES
Experience certainty.

Basics of Android Programming

06/07/2015

CITIGROUP- Infoscape

Developer

Payel Sarkar

Banking and Financial Services
payel2.s@tcs.com

knowmax
Our Intelligence Hub

Confidentiality Statement
Include the confidentiality statement within the box provided. This has to be legally approved **Confidentiality and Non-Disclosure Notice** The information contained in this document is confidential and proprietary to TATA Consultancy Services. This information may not be disclosed, duplicated or used for any other purposes. The information contained in this document may not be released in whole or in part outside TCS for any purpose without the express written permission of TATA Consultancy Services.

Tata Code of Conduct
We, in our dealings, are self-regulated by a Code of Conduct as enshrined in the Tata Code of Conduct. We request your support in helping us adhere to the Code in letter and spirit. We request that any violation or potential violation of the Code by any person be promptly brought to the notice of the Local Ethics Counselor or the Principal Ethics Counselor or the CEO of TCS. All communication received in this regard will be treated and kept as confidential.

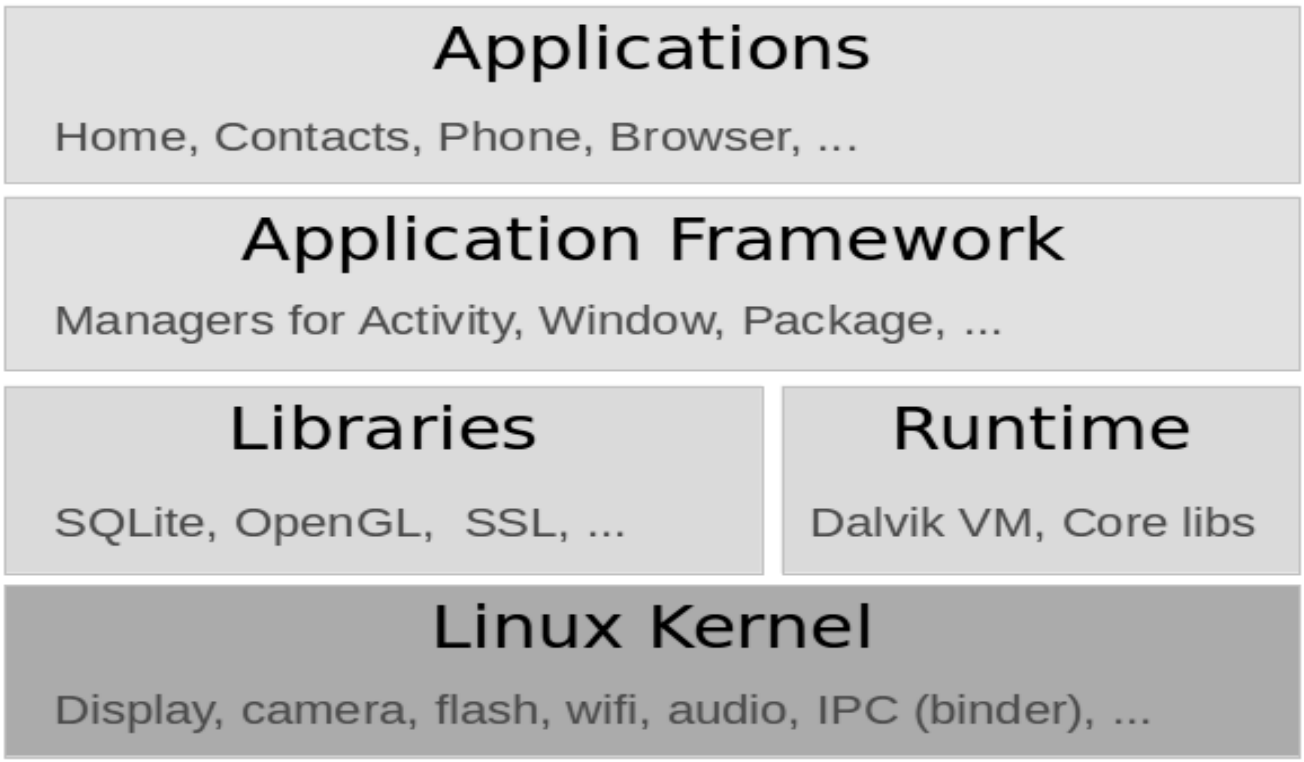
Table of Content

1. Introduction

Android is a Java-based operating system. The system is very lightweight and full featured. The Android operating system is open platform, meaning that it's not tied to one hardware manufacturer and/or one provider. The openness of Android is allowing it to gain market share quickly. Android was developed by the Open Handset Alliance, led by Google, and other companies. Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android. The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008. The Android system supports background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the OpenGL-ES (short OpenGL) standard and grants access to the file system as well as an embedded SQLite database.An Android application typically consists of different visual and non visual components and can reuse components of other applications.

2. Android platform architecture

The Android system is a full software stack, which is typically divided into the four areas as depicted in the following graphic.



The levels can be described as:

2.1 Linux kernel - At the bottom of the layers is Linux. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Android was created on top of the open-source Linux kernel. The Android team chose to use this kernel because it provided proven core features to develop the Android operating system on. The features of the Linux kernel include the following:

- **Security model:** The Linux kernel handles security between the application and the system.
- **Memory management:** The kernel handles memory management, which helps us to develop our app.
- **Process management:** The Linux kernel manages processes well, allocating resources to processes as they need them.
- **Network stack:** The Linux kernel also handles network communication.
- **Driver model:** The goal of Linux is to ensure that everything works. Hardware manufacturers can build their drivers into the Linux build.

2.2 Libraries - On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc. A summary of some key core Android libraries available to the Android developer is as follows –

- **android.app** – Provides access to the application model and is the cornerstone of all Android applications.
- **android. content** – Facilitates content access, publishing and messaging between applications and application components.
- **android. database** – Used to access data published by content providers and includes SQLite database management classes.
- **android.opengl** – A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** – Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android. text** – Used to render and manipulate text on a device display.
- **android. View** – The fundamental building blocks of application user interfaces.
- **android. widget** – A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** – A set of classes intended to allow web-browsing capabilities to be built into applications.

2.3 Android Runtime - This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core features like

memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance.

2.4 Application Framework - The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications. The Android framework includes the following key services –

- **Activity Manager** – Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** – Allows applications to publish and share data with other applications.
- **Resource Manager** – Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** – Allows applications to display alerts and notifications to the user.
- **View System** – an extensible set of views used to create application user interfaces.

2.5 Applications - The Android Open Source Project contains several default application, like the Browser, Camera, Gallery, Music, Phone and more

3. Setting up the environment

- latest version of Java JDK from Oracle's Java site and PATH and JAVA_HOME environment variables have to be set to refer to the directory which contain Java complier javac.
- download the latest version of android studio from **http://developer.android.com/sdk/index.html**
- Install the Android studio

- After Android Studio launch JDK path has to be mentioned.
- Android Studio, Android SDK, Android Virtual Machine and performance (Intel chip) components need to be selected to install.
- Need to specify the location of Android studio and Android SDK
- specify the ram space for Android emulator by default it would take 512MB of local machine RAM
- At final stage, it would extract SDK packages; it would take 2626MB of Hard disk space. Need to click finish after that.
- After completion of the installation the below window will open to create an android project.

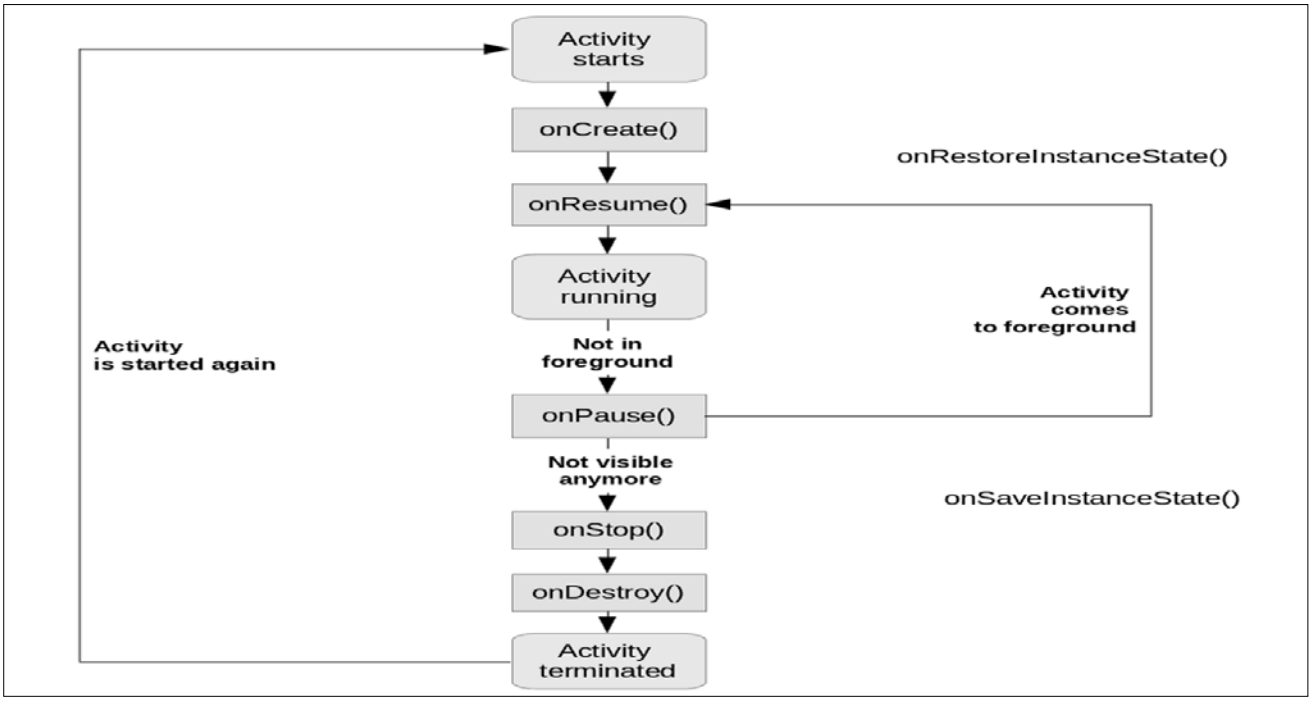
- We need to create an Android virtual device by clicking on AVD_Manager icon
- After Click on a virtual device icon, it going to be shown by default virtual devices which are present on the SDK, or else need to create a virtual device by clicking Create new Virtual device button

4. Application Component and Life Cycle

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact. Applications are defined to Android via the android manifest file, located in the root of the Eclipse project definition (AndroidManifest.xml) Double clicking on the AndroidManifest.xml file in the Eclipse project will open the Manifest editor. The manifest editor is the normal way of creating and modifying the manifest file. Following are the components of an Android application:

4.1 Activites

This is the first component of Android Application. This is a Java class file where we write the logic. An android application typically consists of several activities. An activity interacts with the user to do one thing, only ONE, such as Unlock screen, sending email, view calendar etc. An application consists of multiple activities that are loosely bound together. But only one activity can be specified as main activity which is displayed while launching the application. Every time an activity starts previous activity is stopped but the data is preserved. Every screen in an Android app is an Activity. Activity will always have a User Interface. It is not possible to have an Activity without a UI. Every Android app has a Manifest.xml where all Activites are defined and one Activity is marked as Main Activity. The Activity will show up when the app starts. This is usually referred to as the launcher Activity or the main Activity too. Every Activity has its own lifecycle. The life cycle of an activity with its most important methods is displayed in the following diagram.



4.2 Views

An activity contains views and viewgroups. Views in Android are UI components that could be used to build user interface. Examples are buttons, label, and text boxes. One or more views can be grouped together into a ViewGroup. Following are the types of View in android.

- Basic Views
- List Views
- Picker Views
- Menus
- Display Views
- Additional Views

ViewGroup provides a layout where we can order the appearance of views. Android support the following view groups.

- LinearLayout
- TableLayout
- AbsoluteLayout
- FrameLayout
- RelativeLayout
- ScrollView

The Layout primarily defines the pattern in which the View element should show up. Eg. LinearLayout mandates that the View elements inside it either stack up horizontally or vertically while RelativeLayout lets each View element position itself relative to the parent or a sibling element. UI is defined as XML. The top XML element is a Layout element. Inside it, there can be either View elements or Layout elements. An example XML file is shown below.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk
/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal" >
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Button 1" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Button 2" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/a
ndroid"
  xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
  android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_ma
rgin"

android:paddingRight="@dimen/activity_horizontal_
margin"

android:paddingTop="@dimen/activity_vertical_margi
n"

android:paddingBottom="@dimen/activity_vertical_m
argin" tools:context=".MainActivity">

  <TextView android:text="@string/hello_world"
    android:layout_width="550dp"
    android:layout_height="wrap_content" />
```

4.3 Services

Services are the Android way of keeping an operation going on in the background but do not get initiated without user invocation. Another component such as an activity should invoke. To be more technical, service does not have an independent thread. When user need to have long running tasks like playing music, downloading data or uploading photos; it is achieved through Service. Service doesn't have any UI. To show any information to the user from Service, Notifications are used. There are two ways of creating a Service. One way is to tie the Service with an Activity. In this case, the Service will end once Activity stops. The other way is to run a Service independent of any Application. This way, the Service keeps running in the background even after the application is stopped. All Android services are implemented as a subclass of Service class defined in Android SDK. There are two types of services in Android. They are:

- Unbound Services - It's a type of service which is not bounded to any components. Once started, it will run in the background even after the component that started the service gets killed. It can be run in the background indefinitely and should stop by itself after the operation it's intended to carry out is completed.
- Bound Services - It's bound to other components and runs till the component to which it is bounded runs.

A service is implemented as a subclass of Service class as follows –

```
public class TestService extends Service
{
}
```

4.4 Broadcast Receivers

A broadcast receiver is a component that responds to system-wide broadcast announcements. It is used to receive messages that are broadcasted by the Android system or other Android applications. There are many broadcasts that are initiated by the Android system itself and other applications can receive by using Broadcast receiver. Examples of broadcasts initiated by the system are:

- Warning that the battery is getting low
- Screen turned off
- Change of time zone
- When SMS / Call is received

Broadcast originates from the system as well as applications. Instance for broadcast originating from the system is 'low battery notification'. Application level is, like when user downloads an mp3 file, Mp3 player gets notified about it, and gets added to player list. For this action to take place, mp3 player has to register for this event. There

are two ways to register Android broadcastreceiver.

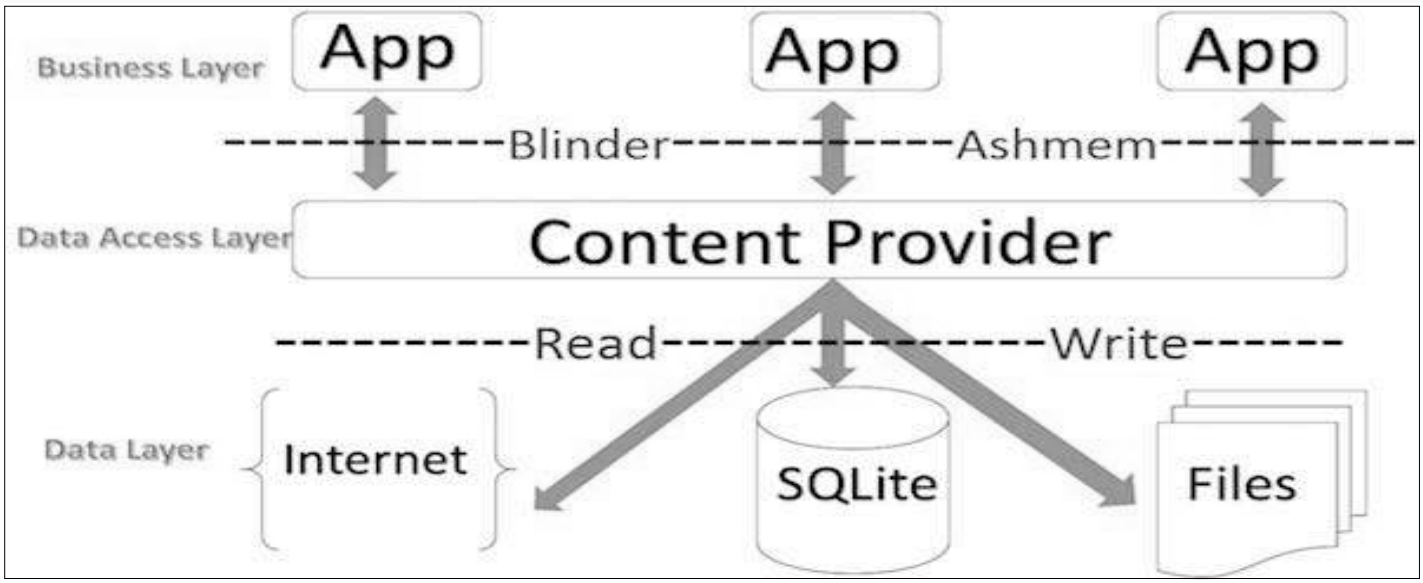
- One is static way in which the broadcast receiver is registered in an android application via AndroidManifest.xml file.
- Another way of registering the broadcast receiver is dynamic, which is done using Context.registerReceiver () method. Dynamically registered broadcast receivers can be unregistered using Context.unregisterReceiver () method.

```
public class TestReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    Toast.makeText(context, "Hello.", Toast.LENGTH_LONG).show();
  }
}
```

4.5 Content Providers

Android is embedded with SQLite Database where all data gets stored. Content providers in Android manage data that is being shared by more than one application. The Content provider of contacts database allows other applications to query, read, modify, and write the contacts info. Android comes with several other built in Content providers that we can use in our application. A content provider is implemented as a subclass of ContentProvider and must implement a standard set of APIs that enable other apps to perform transactions.

```
public class TestApplication extends  ContentProvider
{
}
```



4.6 Intents

Actually intents are not one of Android application components; instead it is the component activating mechanism in Android. It constitutes the core message system in Android and defines a message to activate a particular component. For example, if we want to invoke a new activity from current activity, we need to fire intent. That is by firing intent, we are telling the Android system to make something happen.

There are two types of Intents in Android:

Explicit Intents: In explicit Intent, we are highly specific. We specify which activity should get active on receiving the intent. These are usually used for application's internal communications.

Implicit Intents: In implicit Intent we are sending a message to the Android system to find a suitable Activity that can respond to the intent. For example, to send an e-mail, we can use intent. We will also specify the data to be operated on, with the intent. On receiving the Intent, Android system will invoke an Activity which is able to send e-mail messages with the data that we specified. If there is more than one activity is capable of receiving the Intent, the system presents a chooser to the user so that he can select which Activity/Application should handle it.

5. Create First Android Application

Below is the step to create a basic android application of Login

- Launch Android Development IDE.
- File -> New -> Project-> New Application
- Application Name is the app name that appears to users. -> LoginApp
- Company domain provides a qualifier that will be appended to the package name; -> com.test
- Package name is the fully qualified name for the project - >com.test.loginApp
- Project location is the directory on system that contains the project files.
- All other settings will be default and click Next. ->
- Add a Blank Activity to the mobile
- Activity Name: **MainActivity -** The main activity code is a Java file MainActivity.java. This is the actual application file

```
package com.test.loginApp;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.io.FileInputStream;
import java.io.FileOutputStream;

public class MainActivity extends Activity
 {
  Button b1,b2;
  EditText ed1,ed2;
  TextView tx1;
  int counter = 3;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    b1=(Button)findViewById(R.id.button);
    ed1=(EditText)findViewById(R.id.editText);
    ed2=(EditText)findViewById(R.id.editText2);

    b2=(Button)findViewById(R.id.button2);
    tx1=(TextView)findViewById(R.id.textView3);
    tx1.setVisibility(View.GONE);

    b1.setOnClickListener(new View.OnClickListener()
   {
      @Override
      public void onClick(View v) {
```

```
        if(ed1.getText().toString().equals("Test") && ed2.getText().toString().equals("Test")) {
          Toast.makeText(getApplicationContext(), "Redirecting...",Toast.LENGTH_SHORT).show();
        }
        else{
          Toast.makeText(getApplicationContext(), "Invalid User ID and Password",Toast.LENGTH_SHORT).show();
          tx1.setVisibility(View.VISIBLE);
          tx1.setBackgroundColor(Color.RED);
          counter--;
          tx1.setText(Integer.toString(counter));
          if (counter == 0) {
            b1.setEnabled(false);
          }
        }
      }
    });
    b2.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        finish();
      }
    });
  }
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
  }
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
      return true;
    }
    return super.onOptionsItemSelected(item);
  }
}
```

- Add activity_main to add respective XML components. Following is code of the activity_main.xml.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
  android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

  <TextView android:text="Login" android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textview"
    android:textSize="35dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView"
    android:layout_below="@+id/textview"
    android:layout_centerHorizontal="true"
    android:textColor="#ff7aff24"
    android:textSize="35dp" />

  <EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText"
    android:hint="Enter User ID"
    android:focusable="true"
    android:textColorHighlight="#ff7eff15"
    android:textColorHint="#ffff25e6"
    android:layout_marginTop="46dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
    <EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:ems="10"
    android:id="@+id/editText2"
```

```
    android:layout_below="@+id/editText"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignRight="@+id/editText"
    android:layout_alignEnd="@+id/editText"
    android:textColorHint="#ffff299f"
    android:hint="Password" />
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="login"
    android:id="@+id/button"
    android:layout_alignParentBottom="true"
    android:layout_toLeftOf="@+id/textview"
    android:layout_toStartOf="@+id/textview" />
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Cancel"
    android:id="@+id/button2"
    android:layout_alignParentBottom="true"
    android:layout_toRightOf="@+id/textview"
    android:layout_toEndOf="@+id/textview" />
</RelativeLayout>
```

- Modify AndroidManifest.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"package=" com.test.loginApp " >
  <uses-permission android:name="android.permission.INTERNET" />
  <application android:allowBackup="true" android:icon="@mipmap/ic_launcher"
android:label="@string/app_name" android:theme="@style/AppTheme" >
    <activity android:name=".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>
</application>
</manifest>
```

Acknowledgement:

Sincere Thanks to

http://developer.android.com/
http://www.tutorialspoint.com/android/

Thank You