Chapter 5

5. Android Data Storage

5.1. Shared Preferences

In android, **Shared Preferences** are used to save and retrieve the primitive data types (integer, float, Boolean, string, long) data in the form of key-value pair from a file within an apps file structure.The **Shared Preferences** object will point to a file that contains a key-value pairs and provides a simple read and write methods to save and retrieve the key-value pairs from a file. The Shared Preferences file is managed by an android framework and it can be accessed anywhere within the app to read or write data into the file, but it's not possible to access the file from any other app so it's secured. The Shared Preferences are useful to store the small collection of key-values such as user's login information, app preferences related to users, etc. to maintain the state of app, next time when they login again to the app.

Handle a Shared Preferences

In android, we can save the preferences data either in single or multiple files based on our requirements.

In case if we use single file to save the preferences, then we need to use **getPreferences()** method to get the values from Shared Preferences file and for multiple files we need to call a **getSharedPreferences()** method and pass a file name as a parameter.

Method	Description
getPreferences()	This method is for activity level preferences and each activity will have it's own preference file and by default this method retrieves a default shared preference file that belongs to the activity.
getSharedPreferences()	This method is useful to get the values from multiple shared preference files by passing the name as parameter to identify the file. We can call this from any Context in our app.

Following are the different ways to initialize the Shared Preferences in our application.

If we are using single shared preference file for our activity, then we need to initialize the **SharedPreferences** object by using **getPreferences()** method like as shown below.

```
SharedPreferences sharedPref = getPreferences(Context.MODE_PRIVATE);
```

In case, if we are using multiple shared preference files, then we need to initialize the **SharedPreferences** object by using **getSharedPreferences()** method like as shown below.

```
SharedPreferences sharedPref = getSharedPreferences("filename1",Context.MODE_PRIVATE);
```

Here, the name "**filename1**" is the preference file, which we want to read the values based on our requirements and the context mode **MODE_PRIVATE**, will make sure that the file can be accessed only within our application.

Write to Shared Preferences

To store a data in shared preference file, we need an **editor** to edit and save the changes in **SharedPreferences** object. Following is the code snippet to store the data in shared preference file using **editor**.

```
SharedPreferences sharedPref = getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putBoolean("keyname",true);editor.putString("keyname","string value");
editor.putInt("keyname","int value");editor.putFloat("keyname","float value");
editor.putLong("keyname","long value");editor.commit();
```

If you observe above code snippet, we created a **SharedPreferences.Editor** by calling the **edit()** method of **SharedPreferences** object. We added a primitive data type values such as integer, float, long, string and Boolean by passing the keys and values to the methods **putInt()**, **putString()**, etc. based on our requirements. After that, to save all the changes we are calling **commit()** method.

Read from Shared Preferences

To read or retrieve a values from Shared Preferences file, we need to call a methods such as **getInt()**, **getString()**, etc. by providing the key for the value which we want to get like as shown below.

```
SharedPreferences pref = getPreferences(Context.MODE_PRIVATE);
pref.getString("keyname",null);pref.getInt("keyname",0);pref.getFloat("keyname",0);
pref.getBoolean("keyname",true); pref.getLong("keyname",0);
```

If you observe above code snippet, we are getting the values from shared preferences using a methods such as **getInt()**, **getFloat()**, etc. by providing the key for the value which we want to get.

Deleting from Shared Preferences

To delete a value from Shared Preferences file, we need to call a **remove()** method by providing the key for the value which we want to delete like as shown below.

```
SharedPreferences pref = getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.remove("keyname");editor.commit();
```

If you observe above code snippet, we are deleting the values from shared preferences using a method called **remove()** by providing the key for the value which we want to delete and committing the changes to shared preferences file using **commit()** method.

Clearing from Shared Preferences

We can clear all the data from Shared Preferences file using **clear()** method like as shown below.

```
SharedPreferences pref = getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.clear();editor.commit();
```

If you observe above code snippet, we are clearing all the values from shared preferences using a method called **clear()** and committing the changes to shared preferences file using **commit()** method.

Now we will see how to store and retrieve primitive data type key-value pairs in shared preferences file using **SharedPreferences** object in android application with examples.

Android Shared Preferences Example

Following is the example of storing and retrieving the primitive data type values from shared preferences file using **SharedPreferences**.

Create a new android application using android studio and give names as **SharedPreferencesExample**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

Once we create an application, open **activity_main.xml** file from **\res\layout** folder path and write the code like as shown below.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/fstTxt"      android:layout_width="wrap_content"
    android:layout_height="wrap_content"      android:layout_marginLeft="100dp"
    android:layout_marginTop="150dp"      android:text="UserName" />
  <EditText
    android:id="@+id/txtName"      android:layout_width="wrap_content"
    android:layout_height="wrap_content"      android:layout_marginLeft="100dp"      android:ems="10"/>
  <TextView
    android:id="@+id/secTxt"      android:layout_width="wrap_content"
    android:layout_height="wrap_content"  android:text="Password"  android:layout_marginLeft="100dp" />
  <EditText
    android:id="@+id/txtPwd"      android:inputType="textPassword"
    android:layout_width="wrap_content"      android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"      android:ems="10" />
  <Button
    android:id="@+id/btnLogin"      android:layout_width="wrap_content"
    android:layout_height="wrap_content"    android:layout_marginLeft="100dp"      android:text="Login" />
</LinearLayout>
```

Now we will create another layout resource file **details.xml** in **\res\layout** path to get the first activity (**activity_main.xml**) details in second activity file for that right click on your **layout** folder Go to **New** select **Layout Resource File** and give name as **details.xml**.

Once we create a new layout resource file **details.xml**, open it and write the code like as shown below

details.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:layout_width="wrap_content"       android:layout_height="wrap_content"
      android:id="@+id/resultView"        android:layout_gravity="center"
      android:layout_marginTop="170dp"        android:textSize="20dp"/>
   <Button
      android:id="@+id/btnLogOut"        android:layout_width="wrap_content"
      android:layout_height="wrap_content"        android:layout_gravity="center"
      android:layout_marginTop="20dp"        android:text="Log Out" />
</LinearLayout>
```

Now open your main activity file **MainActivity.java** from **\java\com.example.sharedpreferencesexample** path and write the code like as shown below

MainActivity.java

```
import android.content.*;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
public class MainActivity extends AppCompatActivity {
   EditText uname, pwd;
   Button loginBtn;
   SharedPreferences pref;
   Intent intent;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      uname = (EditText)findViewById(R.id.txtName);
      pwd = (EditText)findViewById(R.id.txtPwd);
      loginBtn = (Button)findViewById(R.id.btnLogin);
      pref = getSharedPreferences("user_details",MODE_PRIVATE);
      intent = new Intent(MainActivity.this,DetailsActivity.class);
      if(pref.contains("username") && pref.contains("password")){
         startActivity(intent);
      }
      loginBtn.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            String username = uname.getText().toString();
            String password = pwd.getText().toString();
            if(username.equals("abebe") && password.equals("abc123")){
               SharedPreferences.Editor editor = pref.edit();
               editor.putString("username",username);
               editor.putString("password",password);
               editor.commit();
```

```
            Toast.makeText(getApplicationContext(), "Login Successful",Toast.LENGTH_SHORT).show();
            startActivity(intent);
        }
        else    {
            Toast.makeText(getApplicationContext(),"Credentials are not valid",Toast.LENGTH_SHORT).show();
        }  }   });
  }
}
```

If you observe above code, we are checking whether the entered username and password details matching or not based on that we are saving the details in shared preferences file and redirecting the user to another activity.

Now we will create another activity file **DetailsActivity.java** in **\java\com.examplw.sharedpreferencesexample** path to show the details from shared preference file for that right click on your application folder Go to **New** select **JavaClass** and give name as **DetailsActivity.java**.

Once we create a new activity file **DetailsActivity.java**, open it and write the code like as shown below

DetailsActivity.java

```
import android.content.*;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.*;
public class DetailsActivity extends AppCompatActivity {
  SharedPreferences prf;
  Intent intent;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.details);
    TextView result = (TextView)findViewById(R.id.resultView);
    Button btnLogOut = (Button)findViewById(R.id.btnLogOut);
    prf = getSharedPreferences("user_details",MODE_PRIVATE);
    intent = new Intent(DetailsActivity.this,MainActivity.class);
    result.setText("Hello, "+prf.getString("username",null));
    btnLogOut.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        SharedPreferences.Editor editor = prf.edit();
        editor.clear();
        editor.commit();
        startActivity(intent);
      }
    });
  }
}
```

Now we need to add this newly created activity in **AndroidManifest.xml** file in like as shown below.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.sharedpreferencesexample">
  <application
    android:allowBackup="true"  android:icon="@mipmap/ic_launcher" android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"  android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity android:name=".DetailsActivity" android:label="Shared Preferences - Details"></activity>
  </application>
</manifest>
```

If you observe above example, we are checking whether the entered user details matching or not based on that we are saving the user details in shared preferences file and redirecting the user to another activity file (**DetailsActivity.java**) to show the users details and added all the activities in **AndroidManifest.xml** file.

Android Session Management

In android, **Session Management** is a process which is used to maintain the required values in a session to use it in application. In android we can manage the logged in user details in session either by storing it in global variables or in Shared Preferences. In case, if we store the values in global variables, those will be lost whenever the user closes the application but if we store the values in Shared Preferences, those will be persisted even if the application is closed by a user.

In android, Shared Preferences are used to save and retrieve the primitive data types (integer, float, Boolean, string, long) in the form of key-value pair from a file within an apps file structure.

To know more about Shared Preferences, check this Android Shared Preferences with Examples.

Now we will see how to store and retrieve logged in user details from shared preferences file using SharedPreferences object and clear or delete the stored session values from Shared Preferences file whenever the user clicks on logout button in android application with examples.

Android Session Management Example

Following is the example of storing and retrieving the logged in user details from shared preferences file using SharedPreferences and clear the stored session values on logout button click.

Create a new android application using android studio and give names as **SharedPreferencesExample**. Once we create an application, open **activity_main.xml** file from **\res\layout** folder path and write the code like as shown below.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:id="@+id/fstTxt" android:layout_width="wrap_content"   android:layout_height="wrap_content"
      android:layout_marginLeft="100dp" android:layout_marginTop="150dp"    android:text="UserName" />
   <EditText
      android:id="@+id/txtName"    android:layout_width="wrap_content"
      android:layout_height="wrap_content"  android:layout_marginLeft="100dp"     android:ems="10"/>
   <TextView
      android:id="@+id/secTxt"   android:layout_width="wrap_content"
      android:layout_height="wrap_content"  android:text="Password"  android:layout_marginLeft="100dp" />
   <EditText
      android:id="@+id/txtPwd"  android:inputType="textPassword"  android:layout_width="wrap_content"
android:layout_height="wrap_content"    android:layout_marginLeft="100dp"  android:ems="10" />
   <Button
      android:id="@+id/btnLogin" android:layout_width="wrap_content"ndroid:layout_height="wrap_content"
      android:layout_marginLeft="100dp"     android:text="Login" />
</LinearLayout>
```

Now we will create another layout resource file **details.xml** in **\res\layout** path to get the first activity (**activity_main.xml**) details in second activity file for that right click on your **layout** folder Go to **New** select **Layout Resource File** and give name as **details.xml**.

Once we create a new layout resource file **details.xml**, open it and write the code like as shown below

details.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:layout_width="wrap_content"  android:layout_height="wrap_content"android:textSize="20dp"
      android:id="@+id/resultView"  android:layout_gravity="center"  android:layout_marginTop="170dp"/>
   <Button
      android:id="@+id/btnLogOut"      android:layout_width="wrap_content"
      android:layout_height="wrap_content"       android:layout_gravity="center"
      android:layout_marginTop="20dp"      android:text="Log Out" />
</LinearLayout>
```

Now open your main activity file **MainActivity.java** from **\java\com.example.sharedpreferencesexample** path and write the code like as shown below

MainActivity.java

```
import android.content.*;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

```
import android.view.View;
import android.widget.*;
public class MainActivity extends AppCompatActivity {
    EditText uname, pwd;
    Button loginBtn;
    SharedPreferences pref;
    Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        uname = (EditText)findViewById(R.id.txtName);
        pwd = (EditText)findViewById(R.id.txtPwd);
        loginBtn = (Button)findViewById(R.id.btnLogin);
        pref = getSharedPreferences("user_details",MODE_PRIVATE);
        intent = new Intent(MainActivity.this,DetailsActivity.class);
        if(pref.contains("username") && pref.contains("password")){
            startActivity(intent);
        }
        loginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = uname.getText().toString();
                String password = pwd.getText().toString();
                if(username.equals("abebe") && password.equals("abc123")){
                    SharedPreferences.Editor editor = pref.edit();
                    editor.putString("username",username);
                    editor.putString("password",password);
                    editor.commit();
                    Toast.makeText(getApplicationContext(), "Login Successful",Toast.LENGTH_SHORT).show();
                    startActivity(intent);
                }
                else    {
                    Toast.makeText(getApplicationContext(),"Credentials are not valid",Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

If you observe above code, we are checking whether the entered username and password details matching or not based on that we are saving the details in shared preferences file and redirecting the user to another activity.Now we will create another activity file **DetailsActivity.java** in **\java\com.example.sharedpreferencesexample** path to show the details from shared preference file for that right click on your application folder → Go to **New** → select **JavaClass** and give name as **DetailsActivity.java**.

Once we create a new activity file **DetailsActivity.java**, open it and write the code like as shown below

DetailsActivity.java

```
import android.content.*;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.*;
public class DetailsActivity extends AppCompatActivity {
    SharedPreferences prf;
    Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.details);
        TextView result = (TextView)findViewById(R.id.resultView);
        Button btnLogOut = (Button)findViewById(R.id.btnLogOut);
        prf = getSharedPreferences("user_details",MODE_PRIVATE);
        intent = new Intent(DetailsActivity.this,MainActivity.class);
        result.setText("Hello, "+prf.getString("username",null));
        btnLogOut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SharedPreferences.Editor editor = prf.edit();
                editor.clear();
                editor.commit();
                startActivity(intent);
            }
        });
    }
}
```

Now we need to add this newly created activity in **AndroidManifest.xml** file in like as shown below.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sharedpreferencesexample">
    <application
        android:allowBackup="true"   android:icon="@mipmap/ic_launcher" android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"   android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".DetailsActivity" android:label="Shared Preferences - Details"></activity>
    </application>
</manifest>
```

If you observe above example, we are checking whether the entered user details matching or not based on that we are saving the user details in shared preferences file and redirecting the user to another activity file (**DetailsActivity.java**) to show the users details and added all the activities in **AndroidManifest.xml** file.

5.2. Internal Storage

In android, **Internal Storage** is useful to store the data files locally on the device's internal memory using **FileOutputStream** object. After storing the data files in device internal storage, we can read the data file from device using **FileInputStream** object.

The data files saved in internal storage is managed by an android framework and it can be accessed anywhere within the app to read or write data into the file, but it's not possible to access the file from any other app so it's secured. When the user uninstall the app, automatically these data files will be removed from the device internal storage.

Write a File to Internal Storage

By using android **FileOutputStream** object **openFileOutput** method, we can easily create and write a data to a file in device's internal storage.

Following is the code snippet to create and write a private file to the device internal memory.

```
String FILENAME = "user_details";
String name = "abebe";
FileOutputStream fstream = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fstream.write(name.getBytes());
fstream.close();
```

If you observe above code, we are creating and writing a file in device internal storage by using **FileOutputStream**object **openFileOutput** method with **file name** and **MODE_PRIVATE** mode to make the file private to our application. We used **write()** method to write the data in file and used **close()** method to close the stream.In android, we have a different mode such as **MODE_APPEND**, **MODE_WORLD_READBLE**, **MODE_WORLD_WRITEABLE**, etc. to use it in our application based on your requirements.

Apart from above methods **write()** and **close()**, **FileOutputStream** object is having other methods, those are

Method	Description
getChannel()	It returns the unique FileChannel object associated with this file output stream.
getFD()	It returns the file descriptor which is associated with the stream.
write(byte[] b, int off, int len)	It writes **len** bytes from the specified byte array starting at offset **off** to the file output stream.

Read a File from Internal Storage

By using android **FileInputStream** object **openFileInput** method, we can easily read the file from device's internal storage.

Following is the code snippet to read a data from file which is in device's internal memory.

```
String FILENAME = "user_details";
FileInputStream fstream = openFileInput(FILENAME);
StringBuffer sbuffer = new StringBuffer();
int i;
while ((i = fstream.read())!= -1){
   sbuffer.append((char)i);
}
fstream.close();
```

If you observe above code, we are reading a file from device internal storage by using FileInputStream object openFileInput method with file name. We used read() method to read one character at a time from the file and used close() method to close the stream.

Apart from above methods **read()** and **close()**, **FileInputStream** object is having other methods, those are

Method	Description
getChannel()	It returns the unique FileChannel object associated with this file output stream.
getFD()	It returns the file descriptor which is associated with the stream.
read(byte[] b, int off, int len)	It reads **len** bytes of data from the specified file input stream into an array of bytes.

Now we will see how to save files directly on the device's internal memory and read the data files from device internal memory by using **FileOutputStream** and **FileInputStream** objects in android application with examples.

Android Internal Storage Example

Following is the example of storing and retrieving the data files from device's internal memory by using **FileOutputStream** and **FileInputStream** objects.

Create a new android application using android studio and give names as **InternalStorageExample**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

Once we create an application, open **activity_main.xml** file from **\res\layout** folder path and write the code like as shown below.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/fstTxt"  android:layout_width="wrap_content"  android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"  android:layout_marginTop="150dp"  android:text="UserName" />
  <EditText
    android:id="@+id/txtName"  android:layout_width="wrap_content"
     android:layout_height="wrap_content"  android:layout_marginLeft="100dp"  android:ems="10"/>
  <TextView
    android:id="@+id/secTxt"  android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text="Password"  android:layout_marginLeft="100dp" />
  <EditText
    android:id="@+id/txtPwd"  android:inputType="textPassword"  android:layout_width="wrap_content"
    android:layout_height="wrap_content"  android:layout_marginLeft="100dp"  android:ems="10" />
  <Button
    android:id="@+id/btnSave"  android:layout_width="wrap_content"
    android:layout_height="wrap_content"  android:layout_marginLeft="100dp"  android:text="Save" />
</LinearLayout>
```

Now we will create another layout resource file **details.xml** in **\res\layout** path to get the first activity (**activity_main.xml**) details in second activity file for that right click on your **layout** folder → Go to **New** → select **Layout Resource File** and give name as **details.xml**.

Once we create a new layout resource file **details.xml**, open it and write the code like as shown below

details.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:layout_width="wrap_content"  android:layout_height="wrap_content"android:textSize="20dp"
    android:id="@+id/resultView"  android:layout_gravity="center"  android:layout_marginTop="170dp"/>
  <Button
    android:id="@+id/btnBack"  android:layout_width="wrap_content"
    android:layout_height="wrap_content"  android:layout_gravity="center"
```

```
android:layout_marginTop="20dp"       android:text="Back" />
</LinearLayout>
```

Now open your main activity file **MainActivity.java** from **\java\com.example.internalstorageexample** path and write the code like as shown below

MainActivity.java

```
import android.content.*;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
public class MainActivity extends AppCompatActivity {
   EditText uname, pwd;
   Button saveBtn;
   FileOutputStream fstream;
   Intent intent;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      uname = (EditText)findViewById(R.id.txtName);
      pwd = (EditText)findViewById(R.id.txtPwd);
      saveBtn = (Button)findViewById(R.id.btnSave);
      saveBtn.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            String username = uname.getText().toString()+"\n";
            String password = pwd.getText().toString();
            try {
               fstream = openFileOutput("user_details", Context.MODE_PRIVATE);
               fstream.write(username.getBytes());
               fstream.write(password.getBytes());
               fstream.close();
```

```
            Toast.makeText(getApplicationContext(), "Details Saved Successfully",Toast.LENGTH_SHORT).show();
                intent = new Intent(MainActivity.this,DetailsActivity.class);
                startActivity(intent);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    });
  }
}
```

If you observe above code, we are taking entered username and password details and saving it in device local file and redirecting the user to another activity.

Now we will create another activity file **DetailsActivity.java** in **\java\com.example.internalstorageexample** path to show the details from device internal storage for that right click on your application folder → Go to **New** → select **Java Class** and give name as **DetailsActivity.java**. Once we create a new activity file **DetailsActivity.java**, open it and write the code like as shown below

DetailsActivity.java

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
public class DetailsActivity extends AppCompatActivity {
  FileInputStream fstream;
  Intent intent;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.details);
        TextView result = (TextView)findViewById(R.id.resultView);
        Button back = (Button)findViewById(R.id.btnBack);
        try {
            fstream = openFileInput("user_details");
            StringBuffer sbuffer = new StringBuffer();
            int i;
            while ((i = fstream.read())!= -1){
                sbuffer.append((char)i);
            }
            fstream.close();
            String details[] = sbuffer.toString().split("\n");
            result.setText("Name: "+ details[0]+"\nPassword: "+details[1]);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                intent = new Intent(DetailsActivity.this,MainActivity.class);
                startActivity(intent);
            }       });    }
}
```

Now we need to add this newly created activity in **AndroidManifest.xml** file in like as shown below.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.internalstorageexample">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
```

```
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity android:name=".DetailsActivity" android:label="Internal Storage - Details"></activity>
  </application>
</manifest>
```

If you observe above example, we are saving entered details file and redirecting the user to another activity file (**DetailsActivity.java**) to show the users details and added all the activities in **AndroidManifest.xml** file.

5.3. External Storage

In android, **External Storage** is useful to store the data files publicly on the shared external storage using **FileOutputStream** object. After storing the data files on external storage, we can read the data file from external storage media using **FileInputStream** object. The data files saved in external storage are world-readable and can be modified by the user when they enable USB mass storage to transfer files on a computer.

Grant Access to External Storage

To read or write files on the external storage, our app must acquire the **WRITE_EXTERNAL_STORAGE** and **READ_EXTERNAL_STORAGE** system permissions. For that, we need to add following permissions in android manifest file like as shown below.

```
<manifest>
  ....
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
  ....
</manifest>
```

Checking External Storage Availability

Before we do any work with external storage, first we need to check whether the media is available or not by calling **getExternalStorageState()**. The media might be mounted to a computer, missing, read-only or in some other state. To get the media status, we need to write the code like as shown below.

```
boolean Available= false;
boolean Readable= false;
String state = Environment.getExternalStorageState();
if(Environment.MEDIA_MOUNTED.equals(state)){
   // Both Read and write operations available
   Available= true;
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
   // Only Read operation available
   Available= true;
   Readable= true;
} else {
   // SD card not mounted
   Available = false;
}
```

If you observe above code snippet, we used **getExternalStorageState()** method to know the status of media mounted to a computer, such as missing, read-only or in some other state.

In android, by using **getExternalStoragePublicDirectory()** method we can access the files from appropriate public directory by passing the type of directory we want, such as DIRECTORY_MUSIC, DIRECTORY_PICTURES, DIRECTORY_RINGTONES, etc. based on our requirements.

If we save our files to corresponding media-type public directory, the system's media scanner can properly categorize our files in the system.

Following is the example to create a directory for new photo album in the public pictures directory.

```
// Get the directory for the user's public pictures directory.
File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), albumName);
if (!file.mkdirs()) {
   Log.e(LOG_TAG, "Directory not created");
}
```

In case, if we are handling the files that are not intended for other apps to use, then we should use a private storage directory on the external storage by calling **getExternalFilesDir()**.

Write a File to External Storage

By using android **FileOutputStream** object and **getExternalStoragePublicDirectory** method, we can easily create and write a data to the file in external storage public folders.

Following is the code snippet to create and write a public file in device **Downloads** folder.

```
String FILENAME = "user_details";
String name = "abebe";
```

```
File folder = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File myFile = new File(folder, FILENAME);
FileOutputStream fstream = new FileOutputStream(myFile);

fstream.write(name.getBytes());
fstream.close();
```

If you observe above code, we are creating and writing a file in device public **Downloads** folder by using **getExternalStoragePublicDirectory** method. We used **write()** method to write the data in file and used **close()**method to close the stream.

Read a File from External Storage

By using android **FileInputStream** object and **getExternalStoragePublicDirectory** method, we can easily read the file from external storage.

Following is the code snippet to read a data from file which is in **downloads** folder.

```
String FILENAME = "user_details";
File folder = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File myFile = new File(folder, FILENAME);
FileInputStream fstream = new FileInputStream(myFile);
StringBuffer sbuffer = new StringBuffer();
int i;
while ((i = fstream.read())!= -1){
   sbuffer.append((char)i);
}
fstream.close();
```

If you observe above code, we are reading a file from device Downloads folder storage by using **FileInputStream**object **getExternalStoragePublicDirectory** method with **file name**. We used **read()** method to read one character at a time from the file and used **close()** method to close the stream.

Now we will see how to save or write file to external memory and read the file data from external storage using android **FileInputStream** and **FileOutputStream** objects in android applications with examples.

Android External Storage Example

Following is the example of storing and retrieving the data files from external memory by using **FileOutputStream** and **FileInputStream** objects.

Create a new android application using android studio and give names as **ExternalStorageExample**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

Once we create an application, open **activity_main.xml** file from **\res\layout** folder path and write the code like as shown below.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/fstTxt" android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"  android:layout_marginTop="150dp"  android:text="UserName" />
  <EditText
  android:id="@+id/txtName"  android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"      android:ems="10"/>
  <TextView
    android:id="@+id/secTxt"  android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text="Password"    android:layout_marginLeft="100dp" />
  <EditText
    android:id="@+id/txtPwd" android:inputType="textPassword"    android:layout_width="wrap_content"
    android:layout_height="wrap_content"  android:layout_marginLeft="100dp"     android:ems="10" />
  <Button
   android:id="@+id/btnSave" android:layout_width="wrap_content"android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"      android:text="Save" />
</LinearLayout>
```

Now we will create another layout resource file **details.xml** in **\res\layout** path to get the first activity(**activity_main.xml**) details in second activity file for that right click on your **layout** folder → Go to **New** → select **Layout Resource File** and give name as **details.xml**.

Once we create a new layout resource file **details.xml**, open it and write the code like as shown below

details.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:layout_width="wrap_content"  android:layout_height="wrap_content"android:textSize="20dp"
    android:id="@+id/resultView"  android:layout_gravity="center" android:layout_marginTop="170dp" />
  <Button
   android:id="@+id/btnBack"  android:layout_width="wrap_content" android:layout_height="wrap_content"
```

```
    android:layout_gravity="center"    android:layout_marginTop="20dp"    android:text="Back" />
</LinearLayout>
```

Now open your main activity file **MainActivity.java** from **\java\com.example.externalstorageexample** path and write the code like as shown below

MainActivity.java

```
package com.example.externalstorageexample;
import android.content.Intent;
import android.os.Environment;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {
   EditText uname, pwd;
   Button saveBtn;
   FileOutputStream fstream;
   Intent intent;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      uname = (EditText)findViewById(R.id.txtName);
      pwd = (EditText)findViewById(R.id.txtPwd);
      saveBtn = (Button)findViewById(R.id.btnSave);
      saveBtn.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            String username = uname.getText().toString()+"\n";
            String password = pwd.getText().toString();
```

```
            try {

ActivityCompat.requestPermissions(MainActivity.this, new String[]{android.Manifest.permission.READ_EXTERN
AL_STORAGE},23);
                File folder =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
                File myFile = new File(folder,"user_details");
                fstream = new FileOutputStream(myFile);
                fstream.write(username.getBytes());
                fstream.write(password.getBytes());
                fstream.close();
                Toast.makeText(getApplicationContext(), "Details Saved in
"+myFile.getAbsolutePath(),Toast.LENGTH_SHORT).show();
                intent = new Intent(MainActivity.this,DetailsActivity.class);
                startActivity(intent);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    });
  }
}
```

If you observe above code, we are taking entered username and password details and saving it in device external memory and redirecting the user to another activity.

Now we will create another activity file DetailsActivity.java in \java\com.example.externalstorageexample path to show the details from external storage for that right click on your application folder → Go to **New** → select **Java Class** and give name as **DetailsActivity.java**.

Once we create a new activity file **DetailsActivity.java**, open it and write the code like as shown below

DetailsActivity.java

```
package com.example.externalstorageexample;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
```

```
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
public class DetailsActivity extends AppCompatActivity {
    FileInputStream fstream;
    Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.details);
        TextView result = (TextView)findViewById(R.id.resultView);
        Button back = (Button)findViewById(R.id.btnBack);
        try {
            File folder = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
            File myFile = new File(folder,"user_details");
            fstream = new FileInputStream(myFile);
            StringBuffer sbuffer = new StringBuffer();
            int i;
            while ((i = fstream.read())!= -1){
                sbuffer.append((char)i);
            }
            fstream.close();
            String details[] = sbuffer.toString().split("\n");
            result.setText("Name: "+ details[0]+"\nPassword: "+details[1]);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        back.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            intent = new Intent(DetailsActivity.this,MainActivity.class);
            startActivity(intent);
        }
    });
  }
}
```

Now we need to add this newly created activity in **AndroidManifest.xml** file in like as shown below.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.externalstorageexample">
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity android:name=".DetailsActivity" android:label="External Storage - Details"></activity>
  </application>
</manifest>
```

If you observe above example, we are saving entered details file and redirecting the user to another activity file (**DetailsActivity.java**) to show the users details and added all the activities in **AndroidManifest.xml** file.

5.4. SQLite

SQLite is an open source light weight relational database management system (RDBMS) used to perform database operations, such as storing, updating, retrieving data from database. In our android applications Shared Preferences, Internal Storage and External Storage options are useful to store and maintain the small amount of data. In case, if we want to deal with large amount of data, then **SQLite database** is the preferable option to store and maintain the data in structured format.

By default, Android comes with built-in SQLite Database support so we don't need to do any configurations.

Just like we save the files on device's **internal storage**, Android stores our database in a private disk space that's associated to our application and the data is secure, because by default this area is not accessible to other applications.

The package **android.database.sqlite** contains all the required API's to use SQLite database in our android applications.

Now we will see how to create database and required tables in SQLite and perform CRUD (insert, update, delete and select) operations in android applications.

Create Database and Tables using SQLite Helper

In android, by using **SQLiteOpenHelper** class we can easily create required database and tables for our application. To use **SQLiteOpenHelper**, we need to create a subclass that overrides the **onCreate()** and **onUpgrade()** call-back methods.

Following is the code snippet of creating database and tables using **SQLiteOpenHelper** class in our android application.

```
public class DbHandler extends SQLiteOpenHelper {
   private static final int DB_VERSION = 1;
   private static final String DB_NAME = "usersdb";
   private static final String TABLE_Users = "userdetails";
   private static final String KEY_ID = "id";
   private static final String KEY_NAME = "name";
   private static final String KEY_LOC = "location";
   private static final String KEY_DESG = "designation";
   public DbHandler(Context context){
      super(context,DB_NAME, null, DB_VERSION);
   }
   @Override
   public void onCreate(SQLiteDatabase db){
      String CREATE_TABLE = "CREATE TABLE " + TABLE_Users + "("
            + KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + KEY_NAME + " TEXT,"
            + KEY_LOC + " TEXT,"
            + KEY_DESG + " TEXT"+ ")";
      db.execSQL(CREATE_TABLE);
   }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
        // Drop older table if exist
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_Users);
        // Create tables again
        onCreate(db);
    }
}
```

If you observe above code snippet, we are creating database "**usersdb**" and table "**userdetails**" using **SQLiteOpenHelper** class by overriding **onCreate** and **onUpgrade** methods.

Method	Description
onCreate()	This method is called only once throughout the application after database is created and the table creation statements can be written in this method.
onUpgrade()	This method is called whenever there is an updation in database like modifying the table structure, adding constraints to database, etc.

Now we will see how to perform CRUD (create, read, delete and update) operations in android applications.

Insert Data into SQLite Database

In android, we can insert data into SQLite database by passing **ContentValues** to **insert()** method.The following code snippet shows how to insert data into SQLite database using **insert()** method.

```
//Get the Data Repository in write mode
SQLiteDatabase db = this.getWritableDatabase();
//Create a new map of values, where column names are the keys
ContentValues cValues = new ContentValues();
cValues.put(KEY_NAME, name);
cValues.put(KEY_LOC, location);
cValues.put(KEY_DESG, designation);
// Insert the new row, returning the primary key value of the new row
long newRowId = db.insert(TABLE_Users,null, cValues);
```

If you observe above code, we are getting the data repository in write mode and adding required values to columns and inserting into database.

Read the Data from SQLite Database

In android, we can read the data from SQLite database using **query()** method.The following code snippet shows how to read the data from SQLite Database using **query()** method.

```
//Get the Data Repository in write mode
SQLiteDatabase db = this.getWritableDatabase();
Cursor cursor =
db.query(TABLE_Users, new String[]{KEY_NAME, KEY_LOC, KEY_DESG}, KEY_ID+ "=?",new String[]{String.value
Of(userid)},null, null, null, null);
```

If you observe above code, we are getting the details from required table using **query()** method based on our requirements.

Update Data in SQLite Database

In android, we can update the data in SQLite database using **update()** method in android applications.The following code snippet shows how to update the data in SQLite database using **update()** method.

```
//Get the Data Repository in write mode
SQLiteDatabase db = this.getWritableDatabase();
ContentValues cVals = new ContentValues();
cVals.put(KEY_LOC, location);
cVals.put(KEY_DESG, designation);
int count = db.update(TABLE_Users, cVals, KEY_ID+" = ?",new String[]{String.valueOf(id)});
```

If you observe above code, we are updating the details using **update()** method based on our requirements.

Delete Data from SQLite Database

In android, we can delete data from SQLite database using **delete()** method.The following code snippet shows how to delete the data from SQLite database using **delete()** method.

```
//Get the Data Repository in write mode
SQLiteDatabase db = this.getWritableDatabase();
db.delete(TABLE_Users, KEY_ID+" = ?",new String[]{String.valueOf(userid)});
```

If you observe above code, we are deleting the details using **delete()** method based on our requirements.Now we will see how to create SQLite database and perform CRUD (insert, update, delete, select) operations on SQLite Database in android application with examples.

Android SQLite Database Example

Following is the example of creating the SQLite database, insert and show the details from SQLite database into android listview using **SQLiteOpenHelper** class.

Create a new android application using android studio and give names as **SQLiteExample**. Once we create an application, create a class file **DbHandler.java** in **\java\com.example.sqliteexample** path to implement SQLite

database related activities for that right click on your application folder → Go to **New** → select **Java Class** and give name as **DbHandler.java**.

Once we create a new class file **DbHandler.java**, open it and write the code like as shown below

DbHandler.java

```
import android.content.*;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import java.util.ArrayList;
import java.util.HashMap;
public class DbHandler extends SQLiteOpenHelper {
   private static final int DB_VERSION = 1;
   private static final String DB_NAME = "usersdb";
   private static final String TABLE_Users = "userdetails";
   private static final String KEY_ID = "id";
   private static final String KEY_NAME = "name";
   private static final String KEY_LOC = "location";
   private static final String KEY_DESG = "designation";
   public DbHandler(Context context){
      super(context,DB_NAME, null, DB_VERSION);
   }
   @Override
   public void onCreate(SQLiteDatabase db){
      String CREATE_TABLE = "CREATE TABLE " + TABLE_Users + "("
            + KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + KEY_NAME + " TEXT,"
            + KEY_LOC + " TEXT,"
            + KEY_DESG + " TEXT"+ ")";
      db.execSQL(CREATE_TABLE);
   }
   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
      // Drop older table if exist
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_Users);
      // Create tables again
      onCreate(db);
   }
   // **** CRUD (Create, Read, Update, Delete) Operations ***** //

   // Adding new User Details
   void insertUserDetails(String name, String location, String designation){
      //Get the Data Repository in write mode
      SQLiteDatabase db = this.getWritableDatabase();
      //Create a new map of values, where column names are the keys
      ContentValues cValues = new ContentValues();
      cValues.put(KEY_NAME, name);
      cValues.put(KEY_LOC, location);
      cValues.put(KEY_DESG, designation);
      // Insert the new row, returning the primary key value of the new row
```

```
        long newRowId = db.insert(TABLE_Users,null, cValues);
        db.close();
    }
    // Get User Details
    public ArrayList<HashMap<String, String>> GetUsers(){
        SQLiteDatabase db = this.getWritableDatabase();
        ArrayList<HashMap<String, String>> userList = new ArrayList<>();
        String query = "SELECT name, location, designation FROM "+ TABLE_Users;
        Cursor cursor = db.rawQuery(query,null);
        while (cursor.moveToNext()){
            HashMap<String,String> user = new HashMap<>();
            user.put("name",cursor.getString(cursor.getColumnIndex(KEY_NAME)));
            user.put("designation",cursor.getString(cursor.getColumnIndex(KEY_DESG)));
            user.put("location",cursor.getString(cursor.getColumnIndex(KEY_LOC)));
            userList.add(user);
        }
        return  userList;
    }
    // Get User Details based on userid
    public ArrayList<HashMap<String, String>> GetUserByUserId(int userid){
        SQLiteDatabase db = this.getWritableDatabase();
        ArrayList<HashMap<String, String>> userList = new ArrayList<>();
        String query = "SELECT name, location, designation FROM "+ TABLE_Users;
        Cursor cursor =
db.query(TABLE_Users, new String[]{KEY_NAME, KEY_LOC, KEY_DESG}, KEY_ID+ "=?",new String[]{String.value
Of(userid)},null, null, null, null);
        if (cursor.moveToNext()){
            HashMap<String,String> user = new HashMap<>();
            user.put("name",cursor.getString(cursor.getColumnIndex(KEY_NAME)));
            user.put("designation",cursor.getString(cursor.getColumnIndex(KEY_DESG)));
            user.put("location",cursor.getString(cursor.getColumnIndex(KEY_LOC)));
            userList.add(user);
        }
        return  userList;
    }
    // Delete User Details
    public void DeleteUser(int userid){
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_Users, KEY_ID+" = ?",new String[]{String.valueOf(userid)});
        db.close();
    }
    // Update User Details
    public int UpdateUserDetails(String location, String designation, int id){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues cVals = new ContentValues();
        cVals.put(KEY_LOC, location);
        cVals.put(KEY_DESG, designation);
        int count = db.update(TABLE_Users, cVals, KEY_ID+" = ?",new String[]{String.valueOf(id)});
        return  count;
```

```
    }
}
```

If you observe above code, we implemented all SQLite Database related activities to perform CRUD operations in android application.Now open **activity_main.xml** file from **\res\layout** folder path and write the code like as shown below.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/fstTxt"  android:layout_width="wrap_content"
        android:layout_height="wrap_content"    android:layout_marginLeft="100dp"
        android:layout_marginTop="150dp"     android:text="Name" />
    <EditText
        android:id="@+id/txtName"       android:layout_width="wrap_content"
        android:layout_height="wrap_content"   android:layout_marginLeft="100dp"      android:ems="10"/>
    <TextView
        android:id="@+id/secTxt"     android:layout_width="wrap_content"
        android:layout_height="wrap_content"   android:text="Location" android:layout_marginLeft="100dp" />
    <EditText
        android:id="@+id/txtLocation"  android:layout_width="wrap_content"
        android:layout_height="wrap_content"    android:layout_marginLeft="100dp"       android:ems="10" />
    <TextView
        android:id="@+id/thirdTxt"   android:layout_width="wrap_content"
  android:layout_height="wrap_content"   android:text="Designation" android:layout_marginLeft="100dp" />
    <EditText
        android:id="@+id/txtDesignation"      android:layout_width="wrap_content"
        android:layout_height="wrap_content"  android:layout_marginLeft="100dp"       android:ems="10" />
    <Button
        android:id="@+id/btnSave"  android:layout_width="wrap_content"
        android:layout_height="wrap_content"   android:layout_marginLeft="100dp"      android:text="Save" />
</LinearLayout>
```

Now we will create another layout resource file **details.xml** in **\res\layout** path to show the details in custom listviewfrom SQLite Database for that right click on your layout folder → Go to **New** → select **Layout Resource File** and give name as **details.xml**.Once we create a new layout resource file details.xml, open it and write the code like as shown below

details.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/user_list"   android:layout_width="fill_parent"
        android:layout_height="wrap_content"    android:dividerHeight="1dp" />
```

```
    <Button
        android:id="@+id/btnBack"     android:layout_width="wrap_content"android:text="Back"
 android:layout_height="wrap_content" android:layout_gravity="center"android:layout_marginTop="20dp"/>
</LinearLayout>
```

Create an another layout file (**list_row.xml**) in **/res/layout** folder to show the data in listview, for that right click on **layout** folder → add new **Layout resource file** → Give name as **list_row.xml** and write the code like as shown below.

list_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"    android:layout_height="wrap_content"
    android:orientation="horizontal"    android:padding="5dip" >
    <TextView
        android:id="@+id/name"  android:layout_width="wrap_content"  android:layout_height="wrap_content"
        android:textStyle="bold"   android:textSize="17dp" />
    <TextView
        android:id="@+id/designation"  android:layout_width="wrap_content"
        android:layout_height="wrap_content"   android:layout_below="@id/name"
        android:layout_marginTop="7dp" android:textColor="#343434"   android:textSize="14dp" />
    <TextView
        android:id="@+id/location" android:layout_width="wrap_content"
        android:layout_height="wrap_content"  android:layout_alignBaseline="@+id/designation"
        android:layout_alignBottom="@+id/designation"   android:layout_alignParentRight="true"
        android:textColor="#343434"      android:textSize="14dp" />
</RelativeLayout>
```

Now open your main activity file **MainActivity.java** from **\java\com.example.sqliteexample** path and write the code like as shown below

MainActivity.java

```
package com.example.sqliteexample;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText name, loc, desig;
    Button saveBtn;
    Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        name = (EditText)findViewById(R.id.txtName);
        loc = (EditText)findViewById(R.id.txtLocation);
```

```
        desig = (EditText)findViewById(R.id.txtDesignation);
        saveBtn = (Button)findViewById(R.id.btnSave);
        saveBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = name.getText().toString()+"\n";
                String location = loc.getText().toString();
                String designation = desig.getText().toString();
                DbHandler dbHandler = new DbHandler(MainActivity.this);
                dbHandler.insertUserDetails(username,location,designation);
                intent = new Intent(MainActivity.this,DetailsActivity.class);
                startActivity(intent);
            Toast.makeText(getApplicationContext(), "Details Inserted Successfully",Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

If you observe above code, we are taking entered user details and inserting into SQLite database and redirecting the user to another activity.Now we will create another activity file **DetailsActivity.java** in **\java\com.example.sqliteexample** path to show the details from SQLite database for that right click on your application folder → Go to **New** → select **Java Class** and give name as **DetailsActivity.java**.Once we create a new activity file **DetailsActivity.java**, open it and write the code like as shown below

DetailsActivity.java

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.*;
import java.util.ArrayList;
import java.util.HashMap;

public class DetailsActivity extends AppCompatActivity {
    Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.details);
        DbHandler db = new DbHandler(this);
        ArrayList<HashMap<String, String>> userList = db.GetUsers();
        ListView lv = (ListView) findViewById(R.id.user_list);
        ListAdapter adapter = new SimpleAdapter(DetailsActivity.this, userList,
R.layout.list_row,new String[]{"name","designation","location"}, new int[]{R.id.name, R.id.designation,
R.id.location});
        lv.setAdapter(adapter);
        Button back = (Button)findViewById(R.id.btnBack);
        back.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            intent = new Intent(DetailsActivity.this,MainActivity.class);
            startActivity(intent);
        }
    });
  }
}
```

If you observe above code, we are getting the details from SQLite database and binding the details to android listview. Now we need to add this newly created activity in **AndroidManifest.xml** file in like as shown below.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.sqliteexample">
  <application
    android:allowBackup="true"      android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"      android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"      android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity android:name=".DetailsActivity" android:label="SQLite Example - Details"></activity>
  </application>
</manifest>
```

If you observe above example, we are saving entered details in SQLite database and redirecting the user to another activity file (**DetailsActivity.java**) to show the users details and added all the activities in **AndroidManifest.xml** file

5.5. Web Service with PHP/Java &MySQL

External Databases are also the data repository for android apps for example apps that provide weather information, exchange rates, world news etc. MySQL is one of the database tool. Connection between an android app and mysql can be done through

- Web Services
- JDBC without web services

A web service is a standard used for exchanging information between applications or systems of heterogeneous type. Software applications written in various programming languages and running on various platforms can use web services to exchange information over Internet using http protocol.This inter-operability can be achieved between Java and Dot net applications, or PHP and Java applications.

A WebService is a Consumer_Machine-to-Provider_Machinecollaboration schema that operates over a computer network. The data exchanges occur independently of the OS, browser, platform,and programming languages used by the provider and the consumers.A provider may expose multiple **EndPoints** (sets of WebServices), each offering any number of typically related functions. WebServices expect the computer network to support standard Web protocols such as XML, HTTP, HTTPS, FTP, and SMTP. **Example:** Weather information, money exchange rates, world news, stock market quotation are examples of applications that can be modeledaround the notion of a remote data-services provider surrounded bycountless consumers tapping on the server's resources.

Why Web Service?

Here are the reasons for using Web Service:

- **Expose method as a service over network:** Web service is a chunk of code written in some programming language (Say C# or Java) that can be invoked remotely through http protocol. Once the Web methods are exposed publically, any applications can invoke it and use the functionality of the web methods.
- **Application Inter-operability – Connect heterogeneous applications:** With the help of web service, heterogeneos applications (Java and Dot Net / Java and PHP application) can communicate with each other. Web service written in Java can be invoked from PHP by passing the required parameters to web methods. This makes the applications platform and technology independant.
- **Standardized protocol:** Web Services uses standardized industry standard protocol for the communication which must be adhered and followed by the applications creating Web Service.
- **Cost effective communication:** Web service is using SOAP over HTTP protocol for communication which is much cheaper than systems like B2B.

What is the need of using Web Service in Android applications?

Existing Web applications are in a need of creating mobile applications to show their presence in mobile platform as well. Almost all web applications are having their Mobile applications created in Android, iOS or Windows platform.

Exposing the existing functionalities of the applications is bit tough as all the functionalities have to re-written in the respective platforms.

But it can be easily achieved with much ease by creating Web Service and expose the existing functionalities as web methods to Mobile platforms.

Here are the few advantages of using Web Service in Android:

- **Make client more lightweight:**Adding a web service layer makes the client more lightweight, both in terms of the required CPU power and the bandwidth used during the processing. Most of

the processing to be done in client end can be separated and put inside a web service layer which will be extremely helpful for end-users in terms of:

1. Using less CPU increases the battery life
2. Using less bandwidth reduces monthly payments over data charge

- **Re-usage of existing functionalities:**While designing the web service, we could also get significant benefits by reusing the existing functionalities by exposing them as web methods.
- **Remote DB hit made simple:** DB residing remotely can be hit from inside Android applications through Web Service calls. Making Web Service call from Android applications allows us to add functionality outside the scope of a DB like caching data, applying business rules over the data etc.,

Why should the Android developer learn how to create a WebService?

- Simple apps are usually self-contained and do not need to collaborate with other parties to obtain additional data or services (for instance, think of a scientific calculator)
- However, there are many cases in which the data needed to work with is very extensive, or changes very often and cannot (should not) be hardcoded into the app. Instead, this kind of data should be requested from a reliable external source (for instance, what is the Euro-to-Dollar rate of change right now?)
- Another class of apps requires a very high computing power perhaps not available in the mobile device (think of problems such as finding the shortest/fastest route between to mapped locations, or best air-fare & route selection for a traveler)
- It is wise for an Android developer to learn how to solve typical problems that exceed the capacities of the handheld devices. Understanding the possibilities offered by the client-server computing model will make the developer be a more complete and better professional.

Web Service Architecture

An ideal *Webservice* provider is designed around four logical layers which define the ways in which data is to be transported, encoded, exposed and discovered by the users.

Layers	Responsibility
Transport	Move messages through the network, using HTTP, SMTP, FTP, ...
Messaging	Encoding of data to be exchanged (XML)
Description	WSDL (Web Service Desc. Lang) used for describing public methods available from the endpoint
Discovery	UDDI (Universal Description & Discovery Integration) facilitates location and publishing of services through a common registry

The Client Side - Consuming WebServices

There are two widely used forms of invoking and consuming Web Services:

Representational State Transfer (REST)

Closely tie to the HTTP protocol by associating its operation to the common methods: GET, POST, PUT, DELETE for HTTP/HTTPS.

This model has a simple invocation mode and little overhead. Service calls rely on a URL which may also carry arguments. Sender & receiver must understand how they pass data items from one another. As an example: Google Maps API uses the REST model.

Remote Procedure Call (RPC).

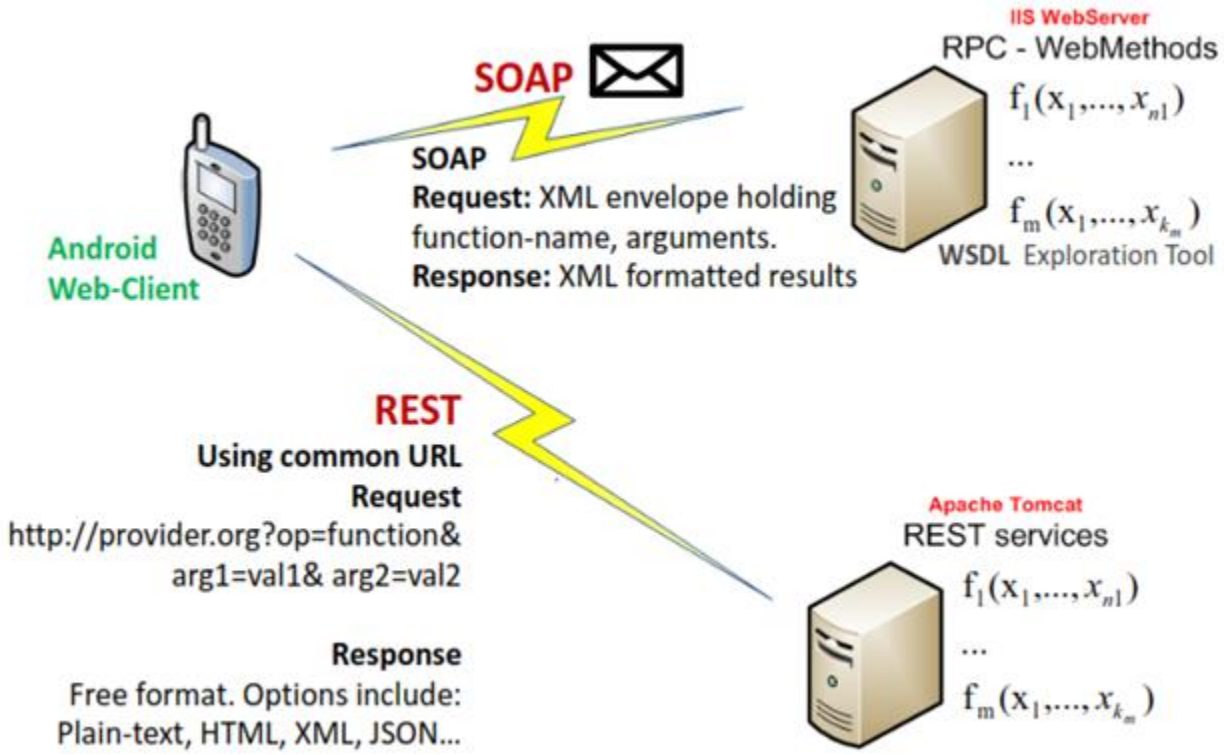
Remote services are seen as coherent collections of discoverable functions (or method calls) stored and exposed by EndPoint providers. Some implementations of this category include: Simple Object Access Protocol (SOAP), Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) and Sun Microsystems's Java/Remote Method Invocation (RMI).

REST vs. SOAP

Although SOAP and REST technologies accomplish the same final goal, that is request and receive a service, there are various differences between them.

- REST users refer to their remote services through a conventional URL that commonly includes the location of the (stateless) server, the service name, the function to be executed and the parameters needed by the function to operate (if any). Data is transported using HTTP/HTTPS.
- SOAP requires some scripting effort to create an XML envelop in which data travels. An additional overhead on the receiving end is needed to extract data from the XML envelope. SOAP accepts a variety of transport

mechanisms, among them HTTP, HTTPS, FTP, SMTP, etc. SOAP uses WSDL (Web Service Description Language) for exposing the format and operation of the services. REST lacks an equivalent exploratory tool.



Example 1:Android User Registration with PHP and MySQL (RESTful Web Service)

1. Create the backend web server. Download xampp or wamp.
2. Test your server by opening http://localhost and check mysql database server by opening http://localhost/phpmyadmin
3. Open the mysql database server and create a database called androidtest and table called users.

#	Name	Type	Collation	Attributes
1	name	varchar(30)	latin1_swedish_ci	
2	fname	varchar(30)	latin1_swedish_ci	
3	email	varchar(30)	latin1_swedish_ci	
4	password	varchar(30)	latin1_swedish_ci	

4. Create a project folder named as test-android under xampp root directory i.e. c:\xampp\htdocs and create the following php file under your project folder
   a. Create database connection using php and save it as db-connect.php

```
<?php
$host="localhost";
$user="root";
$password="";
$db="androidtest";
$con=mysqli_connect($host,$user,$password,$db);
if(!$con) {
  echo '<h1>MySQL Server is not connected</h1>';
```

```
    }
     ?>
```

b. Create a php file that allow you to insert data to users table and name it register.php.

```
<?php
include("db-connect.php");
if(!$con) {
  echo '<h1>MySQL Server is not connected</h1>'.mysqli_connect_error($con);
}else{
                $name = $_POST["name"];
                $fname = $_POST["fname"];
                $email = $_POST["email"];
                $password = $_POST["password"];
   // Registration
   if(!empty($name) && !empty($fname) && !empty($email)&& !empty($password)){
      $json_registration=createNewRegisterUser($name,$fname,$email,$password);
      echo json_encode($json_registration);
   }
}
function createNewRegisterUser($name,$fname,$email,$password){
include("db-connect.php");
          $isExisting=isEmailExist($email);
          if($isExisting){
             $json['success'] = 0;
             $json['message'] = "Error in registering. Probably the username/email already exists";
}else{
             $query="insert into users(name,fname,email,password) values ('$name', '$fname',
'$email','$password')";
             $inserted = mysqli_query($con, $query);
if($inserted == 1){
                $json['success'] = 1;
                $json['message'] = "Successfully registered the user";
}else{
                $json['success'] = 0;
                $json['message'] = "Unable to register the user";
             }
             mysqli_close($con);
          }
          return $json;
       }
```

```
function isEmailExist($email){
include("db-connect.php");
    $query="select * from users where email='$email'";
   $result=mysqli_query($con, $query);
   if(mysqli_num_rows($result) >0){
      mysqli_close($con);
       return true;
   }
   return false;
 }
?>
```

5. Open Android Studio and add the following to your app's build gradle file

```
android{
…
useLibrary 'org.apache.http.legacy'

}
dependencies{ …
compile "org.apache.httpcomponents:httpcore:4.3.2"
}
```

6. Allow remote connection permission in AndroidManifest.xml file.

```
<manifest ...>

<uses-permission android:name="android.permission.INTERNET" />
```

7. Create an activity with the name **activity_jsonregistration.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<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"
tools:context=".JSONREgistration"
android:orientation="vertical">

<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/dbresultmysql"/>
<EditText
android:id="@+id/usernameidmysql"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textPersonName"
android:hint="Name" />

<EditText
android:id="@+id/userfathernameidmysql"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textPersonName"
android:hint="Father Name" />
```

```
<EditText
android:id="@+id/useremailidmysql"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textEmailAddress"
android:hint="Email"/>

<EditText
android:id="@+id/userpasswordidmysql"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textPassword"
android:hint="PAssword"/>

<Button
android:id="@+id/registeruseridmysql"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Register User"
tools:layout_editor_absoluteX="81dp"
tools:layout_editor_absoluteY="73dp" />

<Button
android:id="@+id/viewuseridmysql"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="View User Profile"
tools:layout_editor_absoluteX="81dp"
tools:layout_editor_absoluteY="151dp" />
<Button
android:id="@+id/updateuseridmysql"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Update User Profile"
tools:layout_editor_absoluteX="81dp"
tools:layout_editor_absoluteY="151dp" />
<Button
android:id="@+id/deleteuseridmysql"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Delete User Profile"
tools:layout_editor_absoluteX="81dp"
tools:layout_editor_absoluteY="151dp" />
</LinearLayout>
```

JSONREgistration.java

```
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;

public class JSONRegistration extends AppCompatActivity {
    Button registerbtn;
    EditText emailmysql,namemysql,fnamemysql,passwordmysql;
```

```
    String name,fname,email,password;
    String URL= "http://10.0.2.2/test-android/register.php";
    JSONParser jsonParser=new JSONParser();
int i=0;

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

registerbtn = findViewById(R.id.registeruseridmysql);
emailmysql = findViewById(R.id.useremailidmysql);
namemysql = findViewById(R.id.usernameidmysql);
fnamemysql = findViewById(R.id.userfathernameidmysql);
passwordmysql = findViewById(R.id.userpasswordidmysql);
registerbtn.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
name = namemysql.getText().toString();
fname = fnamemysql.getText().toString();
email = emailmysql.getText().toString();
password = passwordmysql.getText().toString();
                AttemptRegistration attemptRegistration= new AttemptRegistration();
                    attemptRegistration.execute(name,fname,email,password);

                }

        });
    }
private class AttemptRegistration extends AsyncTask<String, String, JSONObject> {
@Override
protected void onPreExecute() {
super.onPreExecute();
        }
@Override
protected JSONObject doInBackground(String... args) {
            String name = args[0];
            String fname=args[1];
            String email= args[2];
            String password= args[3];
            ArrayList params = new ArrayList();
            params.add(new BasicNameValuePair("name", name));
            params.add(new BasicNameValuePair("fname", fname));
            params.add(new BasicNameValuePair("email", email));
            params.add(new BasicNameValuePair("password", password));
            JSONObject json = jsonParser.makeHttpRequest(URL, "POST", params);
return json;
        }
protected void onPostExecute(JSONObject result) {
try {
if (result != null) {

Toast.makeText(getApplicationContext(),result.getString("message"),Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(getApplicationContext(), "Unable to retrieve any data from
server", Toast.LENGTH_LONG).show();
                }
            } catch (JSONException e) {
                Toast.makeText(JSONRegistration.this, e.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        }

    }

}
```

8. Create JSONParser class (JSONParser.java)

```
import android.util.Log;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
public class JSONParser {

static InputStream is = null;
static JSONObject jObj = null;
static JSONArray jArr = null;
static String json = "";
static String error = "";

// constructor
public JSONParser() {

    }

// function get json from url
    // by making HTTP POST or GET mehtod
public JSONObject makeHttpRequest(String url, String method,
                                  ArrayList params) {

// Making HTTP request
try {

// check for request method
if(method.equals("POST")){
// request method is POST
                // defaultHttpClient
HttpClient httpClient = new DefaultHttpClient();
                HttpPost httpPost = new HttpPost(url);
                httpPost.setEntity(new UrlEncodedFormEntity(params));
try {
                    Log.e("API123", " "
+convertStreamToString(httpPost.getEntity().getContent()));
                    Log.e("API123",httpPost.getURI().toString());
                } catch (Exception e) {
                    e.printStackTrace();
                }

                HttpResponse httpResponse = httpClient.execute(httpPost);
                Log.e("API123",""+httpResponse.getStatusLine().getStatusCode());
error= String.valueOf(httpResponse.getStatusLine().getStatusCode());
                HttpEntity httpEntity = httpResponse.getEntity();
is = httpEntity.getContent();

            }else if(method.equals("GET")){
// request method is GET
```

```
DefaultHttpClient httpClient = new DefaultHttpClient();
                String paramString = URLEncodedUtils.format(params, "utf-8");
                url += "?" + paramString;
                HttpGet httpGet = new HttpGet(url);

                HttpResponse httpResponse = httpClient.execute(httpGet);
                HttpEntity httpEntity = httpResponse.getEntity();
is = httpEntity.getContent();
            }

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

try {
            BufferedReader reader = new BufferedReader(new InputStreamReader(
is, "utf-8"), 8);
            StringBuilder sb = new StringBuilder();
            String line = null;
while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
is.close();
json = sb.toString();
            Log.d("API123",json);
        } catch (Exception e) {
            Log.e("Buffer Error", "Error converting result " + e.toString());
        }

// try to parse the string to a JSON object
try {
jObj = new JSONObject(json);
jObj.put("error_code",error);
        } catch (JSONException e) {
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }

// return JSON String
return jObj;

    }

private String convertStreamToString(InputStream is) throws Exception {
        BufferedReader reader = new BufferedReader(new InputStreamReader(is));
        StringBuilder sb = new StringBuilder();
        String line = null;
while ((line = reader.readLine()) != null) {
            sb.append(line);
        }
        is.close();
return sb.toString();
    }
}
```

9. Run the application and test the result.

Note:

- ***Use 10.0.2.2 address in the url(JSONREgistration.java) if you want to test the application using AVD.***

- ***Use the ip address of the server in the url(JSONREgistration.java) if you want to test the app on your device(mobile).***

Example2: Android User Registration with JDBC (No Web Service)

1. **Create an activity called UserRegistrationWithoutWebService**

activity_user_registration_without_web_service.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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"
tools:context=".UserRegistrationWithoutWebService"
android:orientation="vertical">

<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/dbresultmysqlnophp"/>
<EditText
android:id="@+id/usernameidmysqlnophp"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textPersonName"
android:hint="Name" />

<EditText
android:id="@+id/userfathernameidmysqlnophp"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textPersonName"
android:hint="Father Name" />

<EditText
android:id="@+id/useremailidmysqlnophp"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textEmailAddress"
android:hint="Email"/>

<EditText
android:id="@+id/userpasswordidmysqlnophp"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textPassword"
android:hint="Password"/>

<Button
android:id="@+id/registeruseridmysqlnophp"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Register User"
tools:layout_editor_absoluteX="81dp"
tools:layout_editor_absoluteY="73dp" />
</LinearLayout>
```

UserRegistrationWithoutWebService.java

```
import android.os.AsyncTask;
import android.os.Bundle;
```

```
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;

public class UserRegistrationWithoutWebServiceextends AppCompatActivity {

private static final String url = "jdbc:mysql://10.0.2.2:3306/androidtest";
private static final String user = "root";
private static final String pass = "";
    Button btnregister;
    EditText nametxt,fnametxt,emailtxt,passwordtxt;
    TextView resulttv;
    String name,fname,email,password;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_user_registration_without_web_service);
resulttv = findViewById(R.id.dbresultmysqlnophp);
btnregister = findViewById(R.id.registeruseridmysqlnophp);
nametxt = findViewById(R.id.usernameidmysqlnophp);
fnametxt = findViewById(R.id.userfathernameidmysqlnophp);
emailtxt = findViewById(R.id.useremailidmysqlnophp);
passwordtxt = findViewById(R.id.userpasswordidmysqlnophp);
btnregister.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub
name=nametxt.getText().toString();
fname=fnametxt.getText().toString();
email=emailtxt.getText().toString();
password=passwordtxt.getText().toString();
                RegisterNewUser registernewuser = new RegisterNewUser();
                registernewuser.execute("");
            }
        });
    }

private class RegisterNewUser extends AsyncTask<String, Void, String> {
        String res = "";

@Override
protected void onPreExecute() {
super.onPreExecute();
           Toast.makeText(getApplicationContext(), "Please wait...",
Toast.LENGTH_SHORT).show();
          }

@Override
protected String doInBackground(String... params) {
try {
                Class.forName("com.mysql.jdbc.Driver");
                Connection con = DriverManager.getConnection(url, user, pass);
                System.out.println("Database Connected!");
                String result = "Database Connection Successful\n";
                String sql="insert into users values(?,?,?,?)";
                PreparedStatement pstmt = con.prepareStatement(sql);
                pstmt.setString(1,name);
```

```
                pstmt.setString(2,fname);
                pstmt.setString(3,email);
                pstmt.setString(4,password);
if(pstmt.executeUpdate()>0)
res="User Registered Successfully";
else
res="User not registered!";
            } catch (Exception e) {
                e.printStackTrace();
res = e.toString();
            }
return res;
        }

@Override
protected void onPostExecute(String result) {
resulttv.setText(result);
        }
    }

}
```