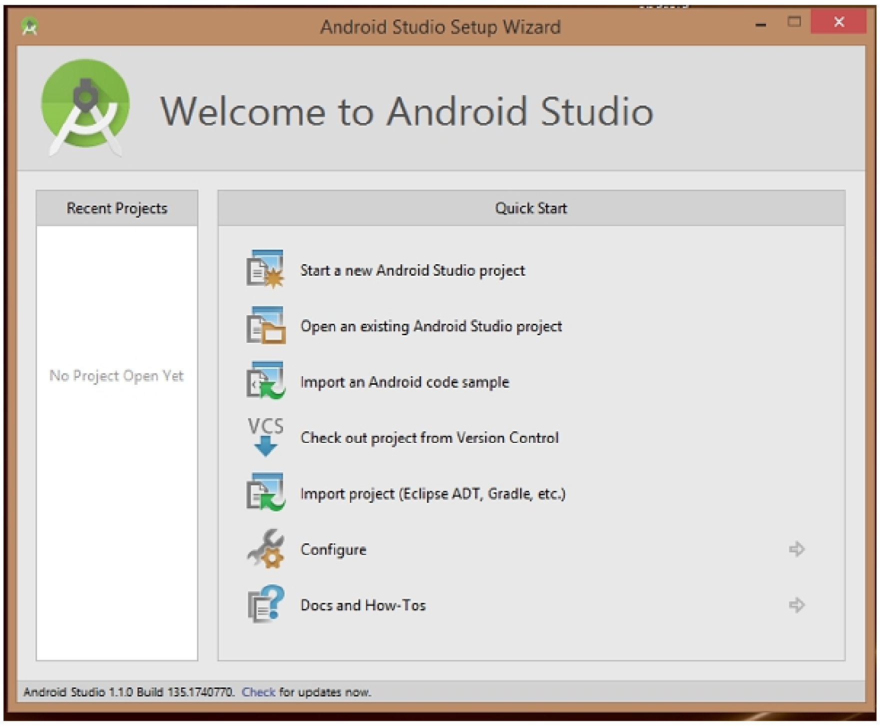
	Exp.No. **Date:**

Android - Hello World Example

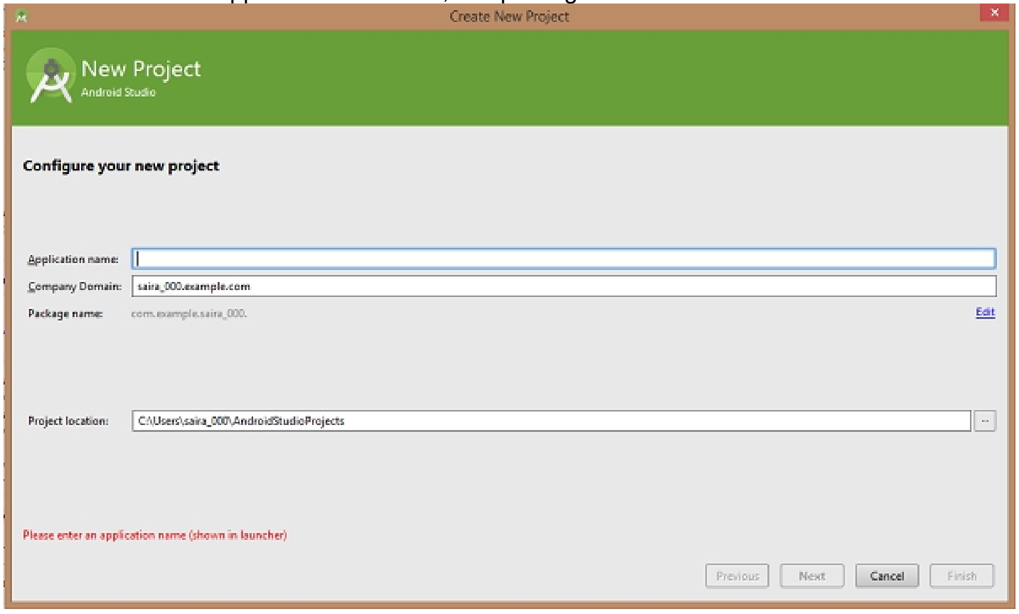
Create Android Application

The first step is to create a simple Android Application using Android studio. When you click on Android studio icon, it will show screen as shown below



Android Studio Setup Wizard
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos
Android Studio 1.1.0 Build 135.1740770. Check for updates now.

You can start your application development by calling start a new android studio project. in a new installation frame should ask Application name, package information and location of the project.−


Create New Project

New Project
Android Studio

Configure your new project

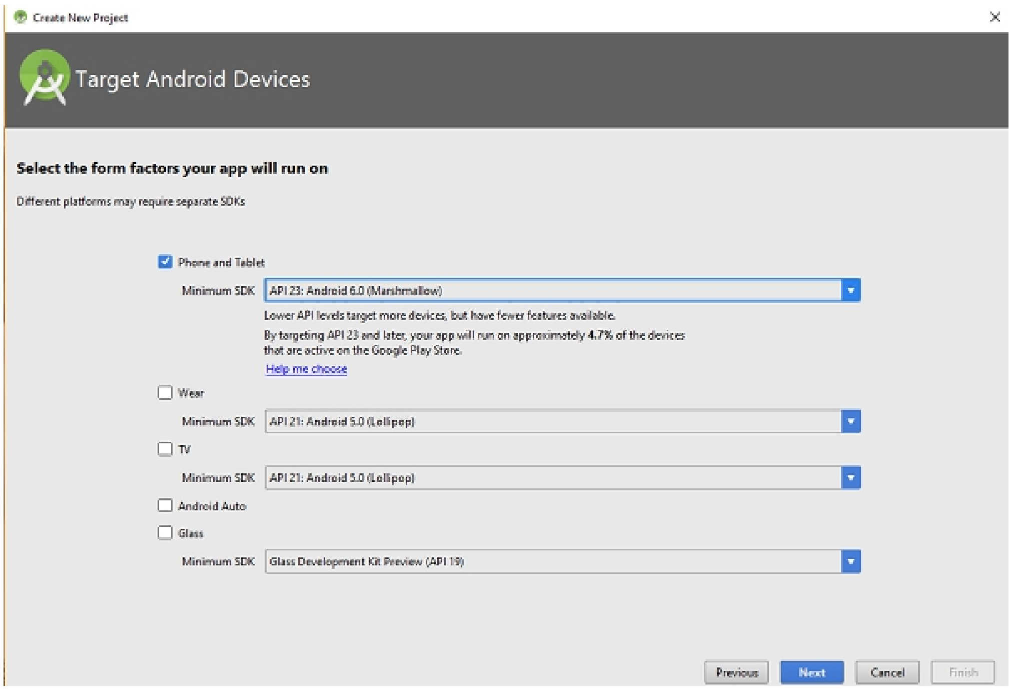
Application name:
Company Domain: saira_000.example.com
Package name: com.example.saira_000. Edit

Project location: C:\Users\saira_000\AndroidStudioProjects

Please enter an application name (shown in launcher)

	Exp.No. **Date:**

After entered application name, it going to be called select the form factors your application runs on, here need to specify Minimum SDK, in our tutorial, I have declared as API23: Android 6.0(Mashmallow) −


Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

[✓] Phone and Tablet

Minimum SDK: API 23: Android 6.0 (Marshmallow)

Lower API levels target more devices, but have fewer features available.
By targeting API 23 and later, your app will run on approximately **4.7%** of the devices that are active on the Google Play Store.
Help me choose

[ ] Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

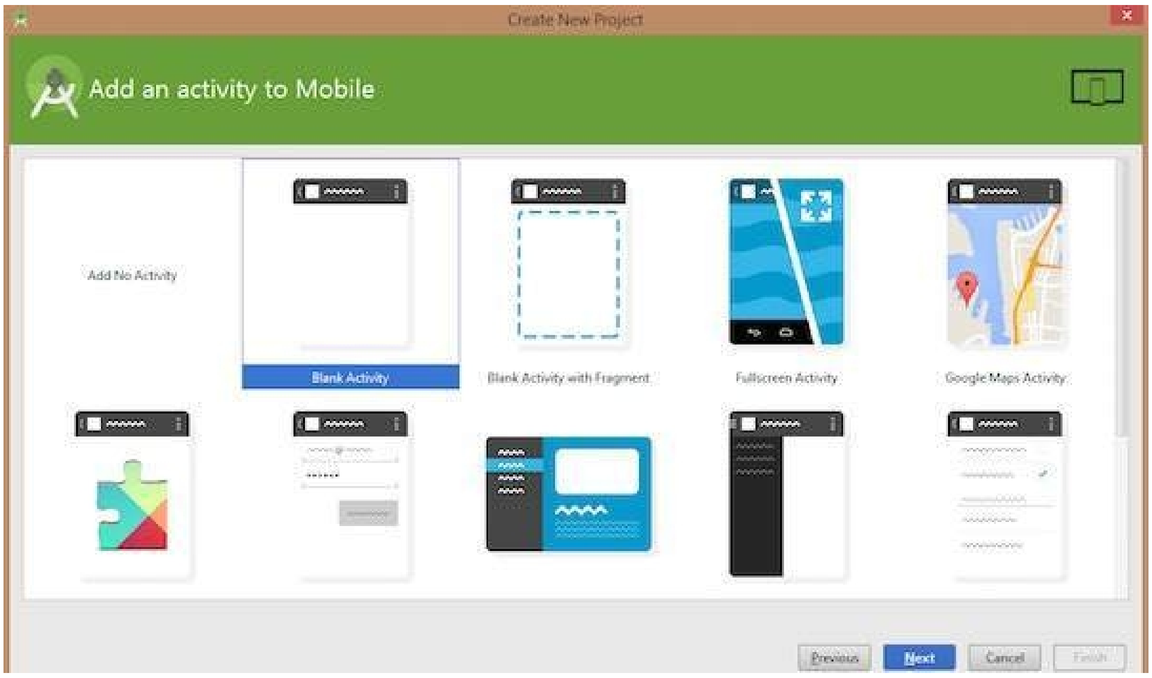
[ ] Android Auto

[ ] Glass

Minimum SDK: Glass Development Kit Preview (API 19)

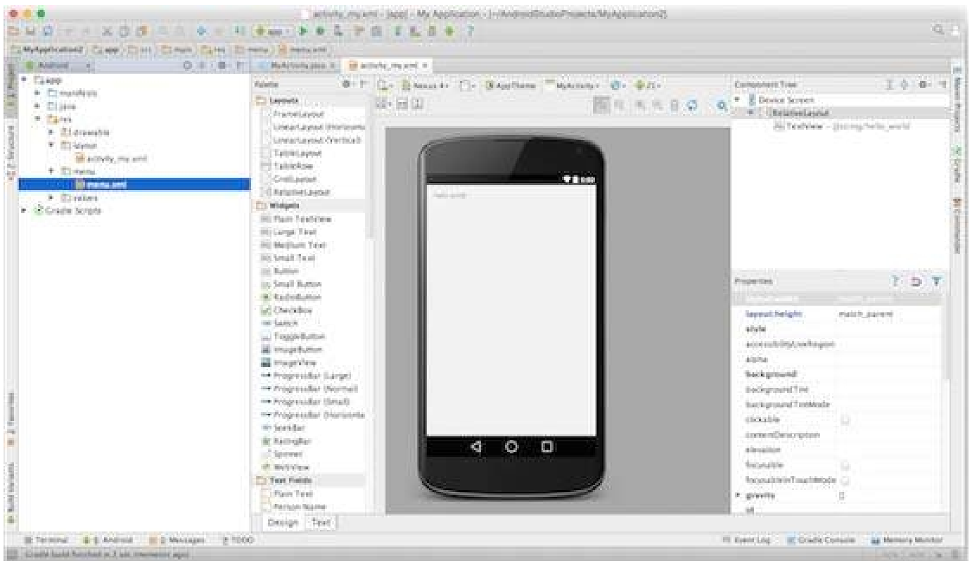
Previous | Next | Cancel | Finish

The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications.



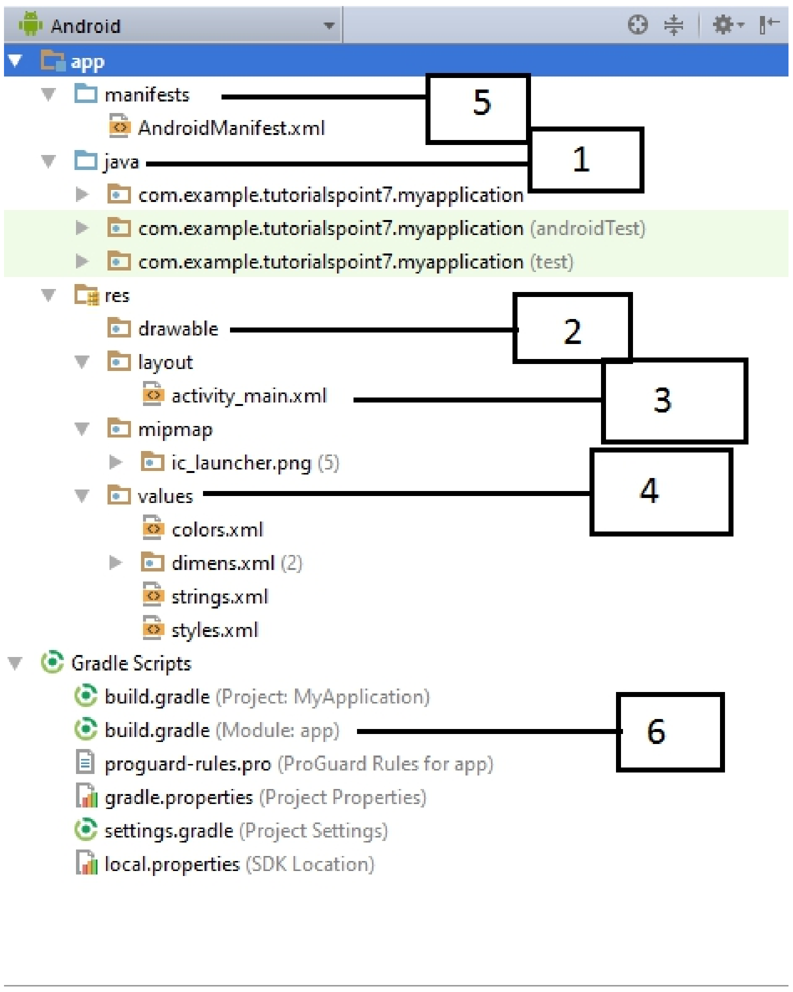
At the final stage it going to be open development tool to write the application code.

	Exp.No. **Date:**



Anatomy of Android Application

Before you run your app, you should be aware of a few directories and files in the Android project –



```
Android
▼ app
    ▼ manifests
        AndroidManifest.xml
    ▼ java
        ▶ com.example.tutorialspoint7.myapplication
        ▶ com.example.tutorialspoint7.myapplication (androidTest)
        ▶ com.example.tutorialspoint7.myapplication (test)
    ▼ res
        drawable
        ▼ layout
            activity_main.xml
        ▼ mipmap
            ▶ ic_launcher.png (5)
        ▼ values
            colors.xml
            ▶ dimens.xml (2)
            strings.xml
            styles.xml
▼ Gradle Scripts
    build.gradle (Project: MyApplication)
    build.gradle (Module: app)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

	Exp.No. **Date:**

Sr.No.	Folder, File & Description
1	**Java** This contains the **.java** source files for your project. By default, it includes an *MainActivity.java* source file having an activity class that runs when your app is launched using the app icon.
2	**res/drawable-hdpi** This is a directory for drawable objects that are designed for high-density screens.
3	**res/layout** This is a directory for files that define your app's user interface.
4	**res/values** This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
5	**AndroidManifest.xml** This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.
6	**Build.gradle** This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

Following section will give a brief overview of the important application files.

The Main Activity File

The main activity code is a Java file **MainActivity.java**. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application. Following is the default code generated by the application wizard for *Hello World!* application −

```
package com.example.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```

Here, *R.layout.activity_main* refers to the *activity_main.xml* file located in the *res/layout* folder. The *onCreate()* method is one of many methods that are figured when an activity is loaded.

The Manifest File

Whatever component you develop as a part of your application, you must declare all its components in a *manifest.xml* which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. For example, a default manifest file will look like as following file −

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

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      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>
   </application>
</manifest>
```

Here <application>...</application> tags enclosed the components related to the application. Attribute *android:icon* will point to the application icon available under *res/drawable-hdpi*. The application uses the image named ic_launcher.png located in the drawable folders

The <activity> tag is used to specify an activity and *android:name* attribute specifies the fully qualified class name of the *Activity* subclass and the *android:label* attributes specifies a string to use as the label for the activity. You can specify multiple activities using <activity> tags.

The **action** for the intent filter is named *android.intent.action.MAIN* to indicate that this activity serves as the entry point for the application. The **category** for the intent-filter is named *android.intent.category.LAUNCHER* to indicate that the application can be launched from the device's launcher icon.

The *@string* refers to the *strings.xml* file explained below. Hence, *@string/app_name* refers to the *app_name* string defined in the strings.xml file, which is "HelloWorld". Similar way, other strings get populated in the application.

Following is the list of tags which you will use in your manifest file to specify different Android application components −

- <activity>elements for activities
- <service> elements for services
- <receiver> elements for broadcast receivers
- <provider> elements for content providers

The Strings File

The **strings.xml** file is located in the *res/values* folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like as following file –

```
<resources>
   <string name="app_name">HelloWorld</string>
   <string name="hello_world">Hello world!</string>

<string name="menu_settings">Settings</string>
<string name="title_activity_main">MainActivity</string>
</resources>
```

The Layout File

The **activity_main.xml** is a layout file available in *res/layout* directory, that is referenced by your application when building its interface. You will modify this file very frequently to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout –

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true"
      android:padding="@dimen/padding_medium"
      android:text="@string/hello_world"
      tools:context=".MainActivity" />

</RelativeLayout>
```

This is an example of simple *RelativeLayout* which we will study in a separate chapter. The *TextView* is an Android control used to build the GUI and it have various attributes like *android:layout_width*, *android:layout_height* etc which are being used to set its width and height etc.. The *@string* refers to the strings.xml file located in the res/values folder. Hence, @string/hello_world refers to the hello string defined in the strings.xml file, which is "Hello World!".

Running the Application

Let's try to run our **Hello World!** application we just created. I assume you had created your **AVD** while doing environment set-up. To run the app from Android studio, open one of your project's activity files and click Run ▶ icon from the tool bar. Android studio installs the app on your AVD and starts it and if everything is fine with your set-up and application, it will display following Emulator window –

	Exp.No. **Date:**

Aim: **Creating the Application by using the Activity class Life Cycle** **Source code:**

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"

android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:text="@string/hello_world" />
</RelativeLayout>
```

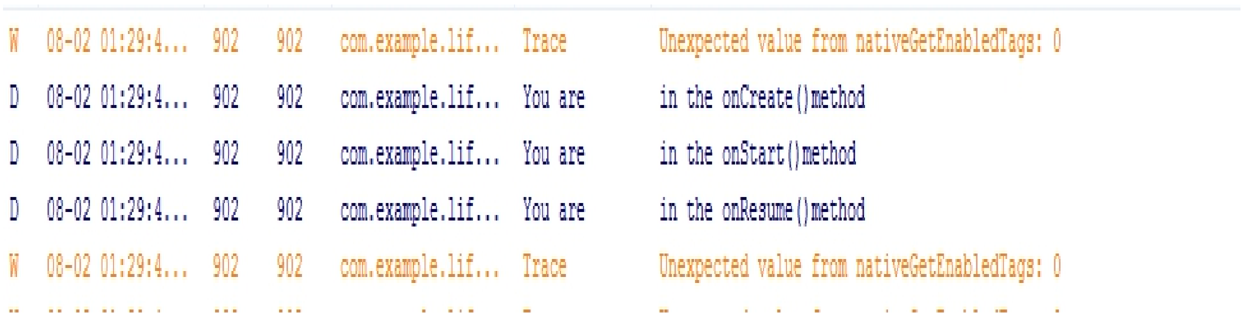
MainActivity.java

```
packagecom.example.lifecycle;
 importandroid.os.Bundle;
 importandroid.app.Activity;
importandroid.view.Menu;
 importandroid.util.Log;
public class MainActivity extends Activity { String tag="You are";
   @Override

protected void onCreate(Bundle savedInstanceState) {
```

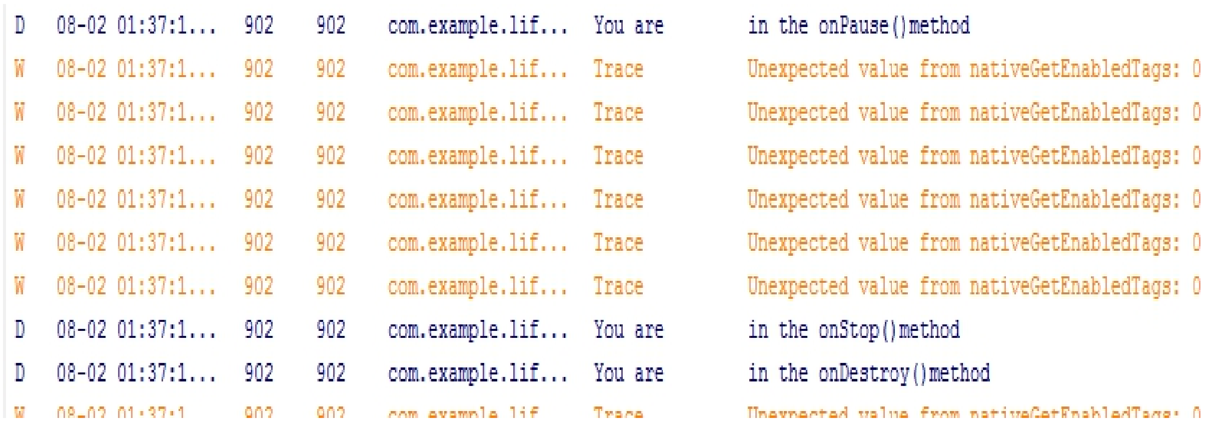
```
super.onCreate(savedInstanceState); setContentView(R.layout.activity_main); Log.d(tag,"in


the onCreate()method ");
   }

public void onStart()

   {

        super.onStart();

        Log.d(tag,"in the onStart()method ");

   }

public void onResume()

   {

        super.onResume();

        Log.d(tag,"in the onResume()method ");

   }

public void onPause()

   {

        super.onPause();

        Log.d(tag,"in the onPause()method ");

   }

public void onStop()

   {

        super.onStop();

        Log.d(tag,"in the onStop()method ");

   }

public void onDestroy()
```

```
        {
            super.onDestroy();

            Log.d(tag,"in the onDestroy()method ");

        }

        @Override

    publicbooleanonCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.activity_main, menu);

    return true;

        }

    }
```

Input&Output:



```
W  08-02 01:29:4...  902  902  com.example.lif...  Trace     Unexpected value from nativeGetEnabledTags: 0
D  08-02 01:29:4...  902  902  com.example.lif...  You are   in the onCreate()method
D  08-02 01:29:4...  902  902  com.example.lif...  You are   in the onStart()method
D  08-02 01:29:4...  902  902  com.example.lif...  You are   in the onResume()method
W  08-02 01:29:4...  902  902  com.example.lif...  Trace     Unexpected value from nativeGetEnabledTags: 0
```



```
D  08-02 01:37:1...  902  902  com.example.lif...  You are   in the onPause()method
W  08-02 01:37:1...  902  902  com.example.lif...  Trace     Unexpected value from nativeGetEnabledTags: 0
W  08-02 01:37:1...  902  902  com.example.lif...  Trace     Unexpected value from nativeGetEnabledTags: 0
W  08-02 01:37:1...  902  902  com.example.lif...  Trace     Unexpected value from nativeGetEnabledTags: 0
W  08-02 01:37:1...  902  902  com.example.lif...  Trace     Unexpected value from nativeGetEnabledTags: 0
W  08-02 01:37:1...  902  902  com.example.lif...  Trace     Unexpected value from nativeGetEnabledTags: 0
W  08-02 01:37:1...  902  902  com.example.lif...  Trace     Unexpected value from nativeGetEnabledTags: 0
D  08-02 01:37:1...  902  902  com.example.lif...  You are   in the onStop()method
D  08-02 01:37:1...  902  902  com.example.lif...  You are   in the onDestroy()method
```

	Exp.No. **Date:**

EXPERIMENT-2

Aim: Create an application that takes the name from a text box and shows hello message along with the name entered in text box, when the user clicks the OK button.

Source code:

activity_edit_text_app.xml:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">
<EditText
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Enter your name"
android:id="@+id/user_name"/>
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/response"/>
</LinearLayout>
```

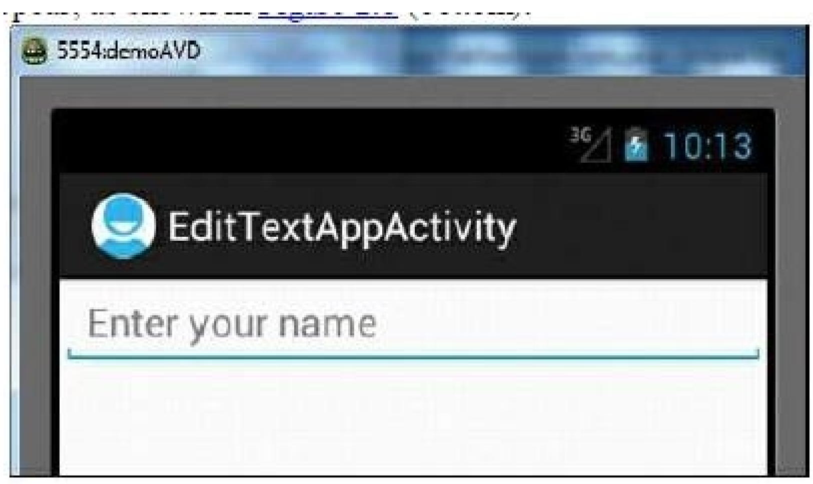
EditTextAppActivity.java:

```
Package com.androidunleashed.edittextapp;
import android.app.Activity;
    import android.os.Bundle;
    import android.widget.EditText;
    import android.widget.TextView;
    import android.view.View;
    import android.view.View.OnKeyListener;
    import android.view.KeyEvent;
    public class EditTextAppActivity
    extends Activity { @Override
    public void onCreate(Bundle savedInstanceState)
            { super.onCreate(savedInstanceState);
```

	Exp.No. Date:

```
setContentView(R.layout.activity_edit_text_app);
          final TextView resp =(TextView)this.findViewById(R.id.response);

final EditText username = (EditText)
findViewById(R.id.user_name);
username.setOnKeyListener(new
OnKeyListener() {
public boolean onKey(View v, int keyCode, KeyEvent event) {

if ((event.getAction() == KeyEvent.ACTION_UP) && (keyCode ==
KeyEvent.KEYCODE_ENTER))
{
resp.setText("Welcome
"+username.getText()+"!"); return true;
}
return false;
}
});
}
}
```

Output:



	Exp.No. **Date:**

EXPERIMENT-3.1 LINEAR LAYOUT

Aim: **Create Application by Using Building Blocks for Android Application Design by using Linear Layout**

Source code:

activity_linear_layout_app.xml

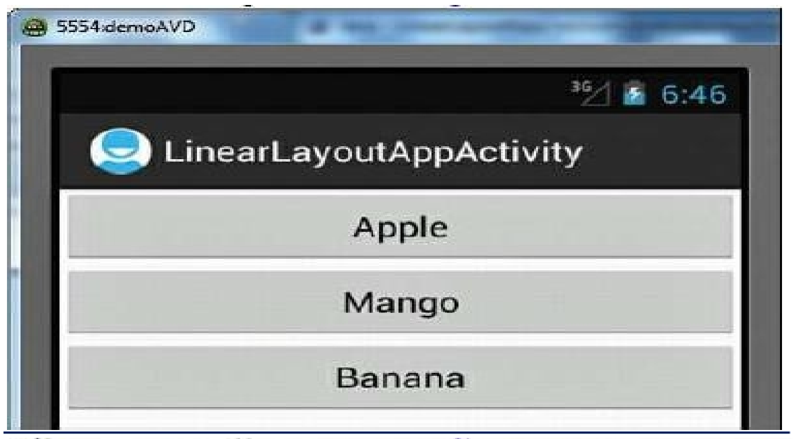
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<Button
android:id="@+id/Apple"
android:text="Apple"
android:layout_width="match_parent"
android:layout_height="wrap_content"
/>
<Button
android:id="@+id/Mango"
android:text="Mango"
android:layout_width="match_parent"
android:layout_height="wrap_content"
/>
<Button
android:id="@+id/Banana"
android:text="Banana"
android:layout_width="match_parent"
android:layout_height="wrap_content"
/>
</LinearLayout>
```

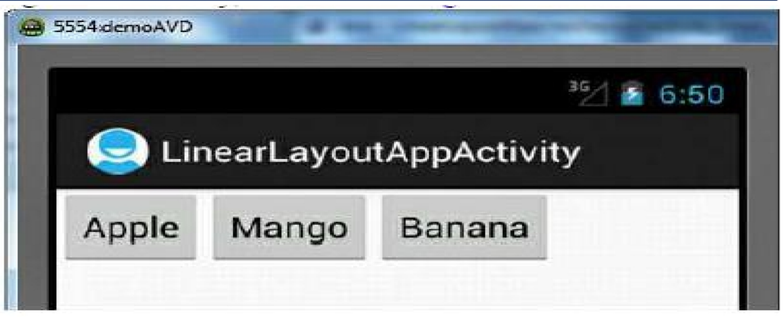
activity_linear_layout_app.xml File on Setting Horizontal

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal" >
```

```
    <Button
    android:id="@+id/Apple"
    android:text="Apple"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
    <Button android:id="@+id/Mango"
    android:text="Mango"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
    <Button
    android:id="@+id/Banana"
    android:text="Banana"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

Fig: Output on Setting vertical





	Exp.No. **Date:**

EXPERIMENT-3.2 Relative Layout

Aim: Create Application by Using Building Blocks for Android Application Design by using Relative Layout.

Source code:

activity_relative_layout_app.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">
<Button
android:id="@+id/Apple"
android:text="Apple"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="15dip"
android:layout_marginLeft="20dip" />
<Button
android:id="@+id/Mango"
android:text="Mango"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:padding="28dip"
android:layout_toRightOf="@id/Apple"
android:layout_marginLeft="15dip"
android:layout_marginRight="10dip"
android:layout_alignParentTop="true" />
<Button android:id="@+id/Banana"
android:text="Banana"
android:layout_width="200dip"
android:layout_height="50dip"
android:layout_marginTop="15dip"
android:layout_below="@id/Apple"
android:layout_alignParentLeft="true"
/>
```

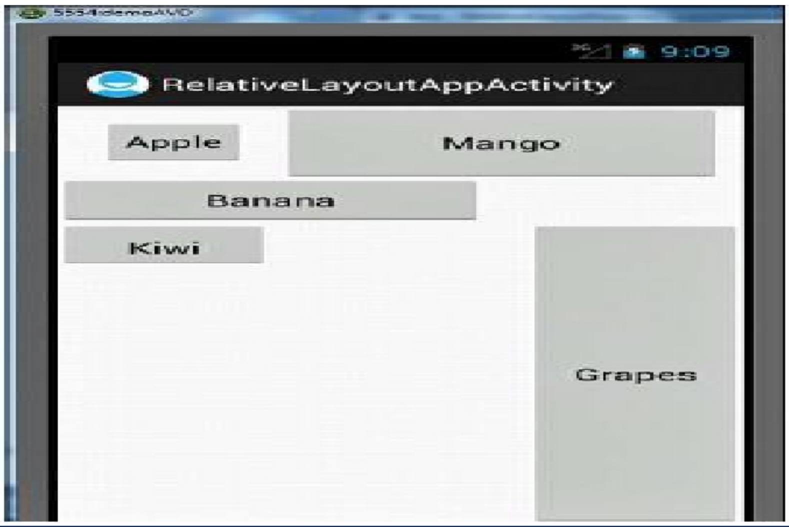
```
<Button
android:id="@+id/Grapes"

android:text="Grapes"
android:layout_width="wrap_content"
android:layout_height="match_parent"
android:minWidth="100dp"
android:layout_alignParentRight="true"
android:layout_below="@id/Banana" />
<Button
android:id="@+id/Kiwi"
android:text="Kiwi"
android:layout_width="100dip"
android:layout_height="wrap_content"
android:layout_below="@id/Banana"
android:paddingTop="15dip"
android:paddingLeft="25dip"
android:paddingRight="25dip" />
</RelativeLayout>
```

RelativeLayoutAppActivity.java

```
package
com.androidunleashed.relativelayoutapp;
import android.app.Activity;
import android.os.Bundle;
public class RelativeLayoutDemoActivity extends
Activity { @Override
        public void onCreate(Bundle savedInstanceState)
        { super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_relative_layout_app);
        }}
```

	Exp.No. **Date:**

OUTPUT:



Exp.No.
Date:

EXPERIMENT-3.3 Absolute Layout

Aim: **Create Application by Using Building Blocks for Android Application Design by using Absolute Layout.**

Source code:

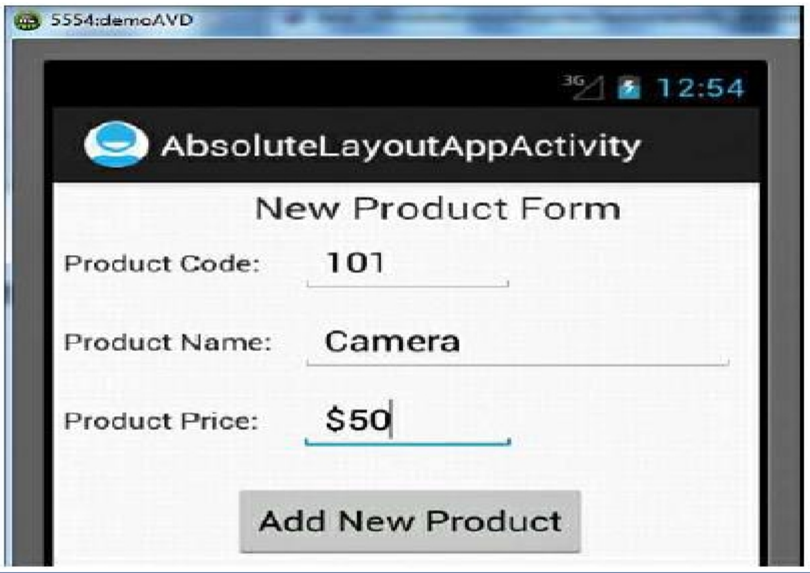
activity_absolute_layout_app.xml

```
<AbsoluteLayout
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:text="New Product Form"
 android:textSize="20sp" android.textStyle="bold"
android:layout_x="90dip" android:layout_y="2dip"/>
<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
 android:text="Product Code:" android:layout_x="5dip"
android:layout_y="40dip" />
<EditText android:id="@+id/product_code"
android:layout_width="wrap_content"
 android:layout_height="wrap_content"
android:minWidth="100dip"
android:layout_x="110dip" android:layout_y="30dip" />
<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Product Name:"
android:layout_x="5dip" android:layout_y="90dip"/>
<EditText android:id="@+id/product_name"
android:layout_width="200dip"
android:layout_height="wrap_content"
 android:minWidth="200dip" android:layout_x="110dip"
android:layout_y="80dip" android:scrollHorizontally="true"
/>
```

	Exp.No. **Date:**

```
<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"

 android:text="Product Price:"
 android:layout_x="5dip" android:layout_y="140dip" />
<EditText android:id="@+id/product_price"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:minWidth="100dip" android:layout_x="110dip"
android:layout_y="130dip" />
<Button android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/click_btn"
android:text="Add New Product" android:layout_x="80dip"
android:layout_y="190dip" />
</AbsoluteLayout>
```

Output:



New Product Form

Product Code: 101

Product Name: Camera

Product Price: $50

EXPERIMENT 3.4- UI-CHECK BOX

Aim: Creating the Application Choosing Options CheckBox

Source code:

activity_check_box_app.xml

```
<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="match_parent"
android:layout_height="wrap_content"
android:text="Select Items you want"/>
<CheckBox
android:id="@+id/checkbox_pizza"
android:layout_height="wrap_content"
android:text="Pizza $15"
android:layout_width="match_parent" />
<CheckBox
android:id="@+id/checkbox_hotdog"
android:layout_height="wrap_content"
android:text="Hot Dog $5"
android:layout_width="match_parent" />
<CheckBox
android:id="@+id/checkbox_burger"
android:layout_height="wrap_content"
android:text="Burger $10"
android:layout_width="match_parent"
/>
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/bill_btn"
android:text="Calculate Bill"/>
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

```
    android:id="@+id/amount"/>

    </LinearLayout>
```

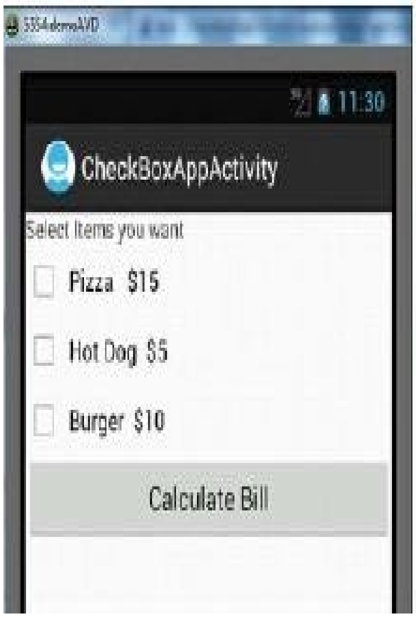
CheckBoxAppActivity.java

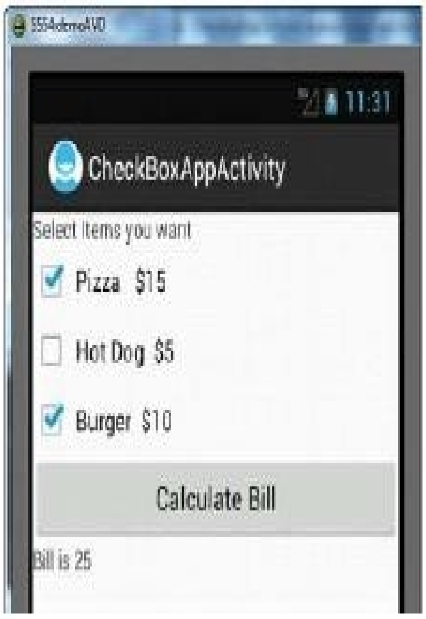
```
    package com.androidunleashed.checkboxapp;
    import android.app.Activity;
    import android.os.Bundle;
    import android.widget.Button;
    import   android.widget.TextView;
    import  android.widget.CheckBox;
    import android.view.View;
    import android.view.View.OnClickListener;
    public class CheckBoxAppActivity extends Activity implements
    OnClickListener {
    CheckBox
    c1,c2,c3;
    TextView
    resp;
    @Override
    public void onCreate(Bundle savedInstanceState)
            { super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_check_box_app);
            Button b = (Button)this.findViewById(R.id.bill_btn);
            resp = (TextView)this.findViewById(R.id.amount);
            c1 = (CheckBox)findViewById(R.id.checkbox_pizza);
            c2 = (CheckBox)findViewById(R.id.checkbox_hotdog);
            c3 = (CheckBox)findViewById(R.id.checkbox_burger);
            b.setOnClickListener(this);
    }
    public void
            onClick(View v)
            { int amt=0;
  if (c1.isChecked()) {
        amt=amt+15;
            }
            if (c2.isChecked()) {
                    amt=amt+5;
            }
            if (c3.isChecked()) {
                    amt=amt+10;
            }
            resp.setText("Bill is " +Integer.toString(amt));
```

	Exp.No. **Date:**

```
        }
}
```

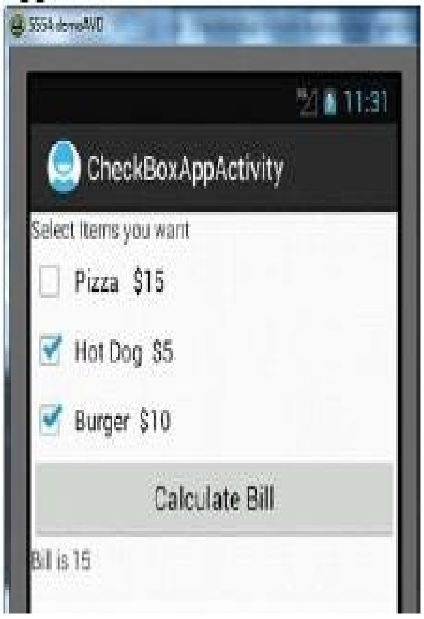
OUTPUT:





☑ Pizza $15
☐ Hot Dog $5
☑ Burger $10

```
Bill is 25
```



☐ Pizza $15
☑ Hot Dog $5
☑ Burger $10

```
Bill is 15
```

EXPERIMENT-3.5

Aim: Creating the Application Choosing Options RadioButton

Source code:

activity_radio_button_app.xml

```
<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="match_parent"
android:layout_height="wrap_content"
android:text="Select the type of hotel"/>
<RadioGroup
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical">
<RadioButton
android:id="@+id/radio_fivestar"
android:layout_width="wrap_content"
android:layout_height="wrap_content
" android:text="Five Star " />
<RadioButton
android:id="@+id/radio_threestar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Three Star" />
</RadioGroup>
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/hoteltype"/>
</LinearLayout>
```

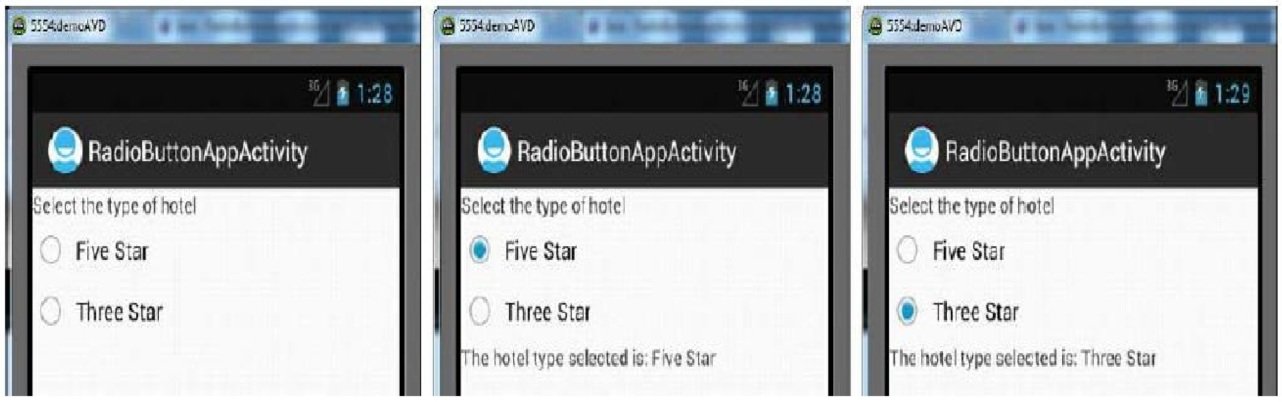
	Exp.No. **Date:**

RadioButtonAppActivity.java

```
package com.androidunleashed.radiobuttonapp;


import android.app.Activity;
    import android.os.Bundle;
    import android.widget.TextView;
    import android.widget.RadioButton;
    import android.view.View;
    import android.view.View.OnClickListener;
    public class RadioButtonAppActivity extends Activity
    { @Override
    public void onCreate(Bundle savedInstanceState)
    { super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_radio_button_app);
    RadioButton radioFivestar = (RadioButton)
    findViewById(R.id.radio_fivestar);
    RadioButton radioThreestar = (RadioButton)
    findViewById(R.id.radio_threestar);
    radioFivestar.setOnClickListener(radioListener);
    radioThreestar.setOnClickListener(radioListener);
    }
    private OnClickListener radioListener = new OnClickListener() {
    public void onClick(View v)
    { TextView selectedHotel =
    (TextView)
    findViewById(R.id.hoteltype);
    RadioButton rb = (RadioButton) v;
    selectedHotel.setText("The hotel type selected is: "+rb.getText());
    }};
    }
```

	Exp.No. **Date:**

Output:



Select the type of hotel
Five Star
Three Star

Select the type of hotel
Five Star
Three Star
The hotel type selected is: Five Star

Select the type of hotel
Five Star
Three Star
The hotel type selected is: Three Star

EXPERIMENT-3.6 UI RadioGroup

Aim: Creating the Application Choosing Options RadioGroup

Source code:

activity_radio_button_app.xml

```
<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="match_parent"
    android:layout_height="wrap_content"
    android:text="Select the type of hotel"/>
    <RadioGroup
    android:id="@+id/group_hotel"
    android:layout_width="match_parent"
    android:layout_height="wrap_content
    " android:orientation="vertical">
    <RadioButton
    android:id="@+id/radio_fivestar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content
    " android:text="Five Star " />
    <RadioButton
    android:id="@+id/radio_threestar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Three Star" />
    </RadioGroup>
    <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Select the type of room"/>
    <RadioGroup
    android:id="@+id/group_room"
```

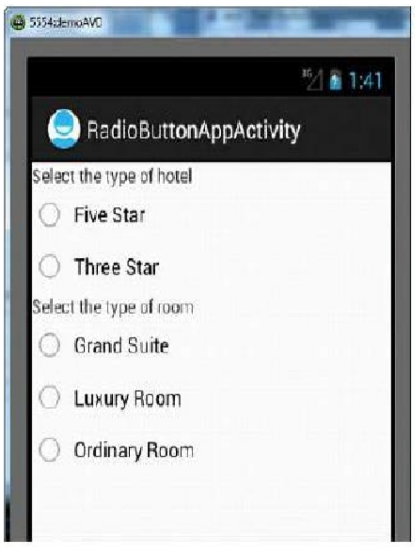
```
android:layout_width="match_parent"
android:layout_height="wrap_content

" android:orientation="vertical">

<RadioButton
android:id="@+id/radio_suite"
android:layout_width="wrap_content"
android:layout_height="wrap_content
" android:text="Grand Suite " />
<RadioButton android:id="@+id/radio_luxury"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Luxury Room" />
<RadioButton android:id="@+id/radio_ordinary"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Ordinary Room" />
</RadioGroup>
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/hoteltype" />
</LinearLayout>
```

RadioButtonAppActivity.java

```
package com.androidunleashed.radiobuttonapp;
import android.app.Activity;
    import android.os.Bundle;
    import android.widget.TextView; import
    android.widget.RadioButton; import
    android.view.View;
    import android.view.View.OnClickListener;
    public class RadioButtonAppActivity extends
    Activity { String str1="";
    String str2="";
    @Override
    public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_radio_button_app);
    RadioButton radioFivestar = (RadioButton) findViewById(R.id.radio_fivestar);
    RadioButton radioThreestar = (RadioButton) findViewById(R.id.radio_threestar);
    RadioButton radioSuite = (RadioButton) findViewById(R.id.radio_suite);

    RadioButton radioLuxury = (RadioButton) findViewById(R.id.radio_luxury);
     RadioButton radioOrdinary = (RadioButton) findViewById(R.id.radio_ordinary);

    radioFivestar.setOnClickListener(radioListener1);
    radioThreestar.setOnClickListener(radioListener1);
    radioSuite.setOnClickListener(radioListener2);
    radioLuxury.setOnClickListener(radioListener2);
    radioOrdinary.setOnClickListener(radioListener2);
    }
    private OnClickListener radioListener1 = new OnClickListener()
    { public void onClick(View v) {
    TextView selectedOptions = (TextView)
    findViewById(R.id.hoteltype);
    RadioButton rb = (RadioButton) v;
    str1="The hotel type selected is: " +rb.getText();
    selectedOptions.setText(str1+"\n"+str2);
    }
    };
    private OnClickListener radioListener2 = new OnClickListener() {
    public void onClick(View v)
    { TextView selectedOptions =
    (TextView)
    findViewById(R.id.hoteltype);
    RadioButton rb = (RadioButton) v;
    str2="Room type selected is: " +rb.getText();
    selectedOptions.setText(str1+"\n"+str2);
    }};
    }
```

	Exp.No. **Date:**

Output:


5554:demoAVD

1:41

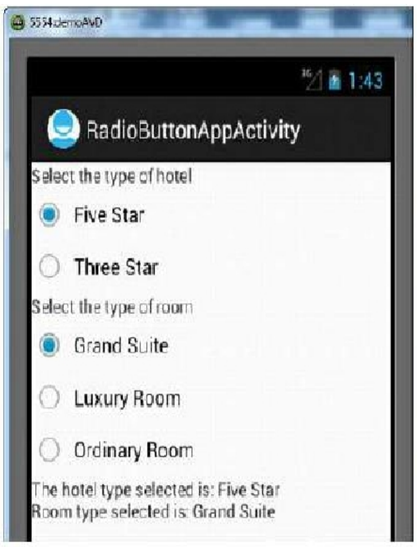
RadioButtonAppActivity

Select the type of hotel

- Five Star
- Three Star

Select the type of room

- Grand Suite
- Luxury Room
- Ordinary Room


5554:demoAVD

1:43

RadioButtonAppActivity

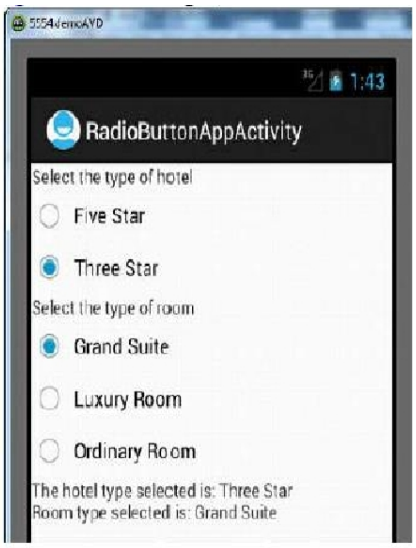
Select the type of hotel

- [✓ Five Star]
- Three Star

Select the type of room

- [✓ Grand Suite]
- Luxury Room
- Ordinary Room

```
The hotel type selected is: Five Star
Room type selected is: Grand Suite
```


5554:demoAVD

1:43

RadioButtonAppActivity

Select the type of hotel

- Five Star
- [✓ Three Star]

Select the type of room

- [✓ Grand Suite]
- Luxury Room
- Ordinary Room

```
The hotel type selected is: Three Star
Room type selected is: Grand Suite
```

	Exp.No. **Date:**

EXPERIMENT-3.6 UI Spinner

Aim: **Creating the Application Choosing Options Spinner**

Source code:

strings.xml

```
<resources>
<string  name="app_name">SpinnerApp</string>
<string  name="menu_settings">Settings</string>
<string  name="title_activity_spinner_app">SpinnerAppActivity</string>
<string name="choose_msg">Choose a fruit</string>
</resources>
```

arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string-array name="fruits">
<item>Apple</item>
<item>Mango</item>
<item>Orange</item>
<item>Grapes</item>
<item>Banana</item>
</string-array>
</resources>
```

activity_spinner_app.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">
<Spinner android:id="@+id/spinner"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:prompt="@string/choose_msg" android:entries="@array/fruits"/>
<TextView android:id="@+id/selectedopt"
android:layout_width="match_parent"
```

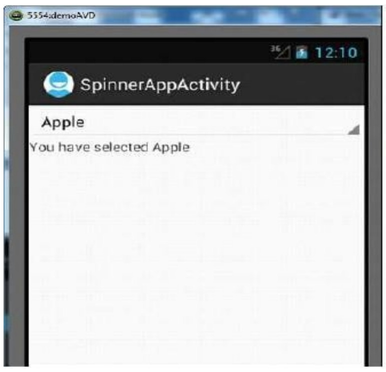
```
android:layout_height="wrap_content" />
</LinearLayout>
```

SpinnerAppActivity.java

```
package com.androidunleashed.spinnerapp;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
 import android.widget.Spinner;
import android.widget.AdapterView;
 import android.view.View;
import android.widget.AdapterView.OnItemSelectedListener;
public class SpinnerAppActivity extends Activity { @Override
public void onCreate(Bundle savedInstanceState)
{ super.onCreate(savedInstanceState);
setContentView(R.layout.activity_spinner_app);
final TextView selectedOpt= (TextView)findViewById(R.id.selectedopt);
Spinner spin=(Spinner)findViewById(R.id.spinner);
final String[] fruitsArray = getResources().getStringArray(R.array.fruits);
spin.setOnItemSelectedListener(new OnItemSelectedListener() {
public void onItemSelected(AdapterView<?> parent, View v, int position, long id)
 { selectedOpt.setText("You have selected " +fruitsArray[position]);
}
public void onNothingSelected(AdapterView<?> parent)
{
selectedOpt.setText("");
}
});
}
}
```

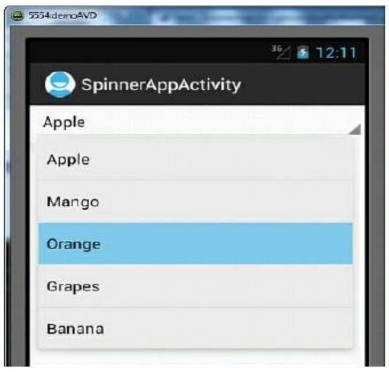
	Exp.No. **Date:**

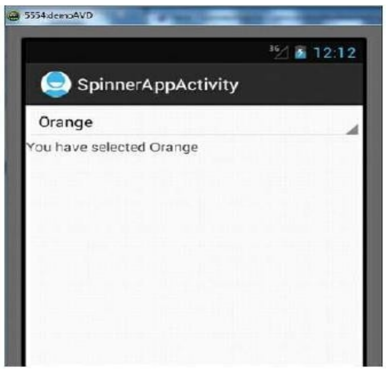
Output:



Apple

You have selected Apple





Orange

You have selected Orange

	Exp.No. Date:

EXPERIMENT-4 USING FRAGMENTS

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.fragapp1.MainActivity">
    <Button
        android:id="@+id/btn_load"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/str_btn_load" />
    <LinearLayout
        android:id="@+id/fragment_container"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"></LinearLayout>
</RelativeLayout>
```

Hello_fragment_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/str_tv_fragment"
        android:textColor="@color/purple_200"
        />
</LinearLayout>
```

MainActivity.java

```
package com.example.fragapp1;
import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

```
public class MainActivity extends Activity {
   /** Called when the activity is first created. */

@Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Button btnLoad = (Button) findViewById(R.id.btn_load);
      OnClickListener listener = new OnClickListener() {
         @Override
         public void onClick(View v) {
            FragmentManager fragmentManager = getFragmentManager();
            FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
            HelloFragment hello = new HelloFragment();
            fragmentTransaction.add(R.id.fragment_container, hello, "HELLO");
            fragmentTransaction.commit();
         }
      };
      btnLoad.setOnClickListener(listener);
   }
}
```

EXPERIMENT 5

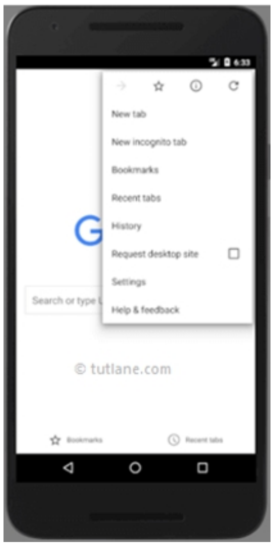
Develop an application that uses a menu with 3 options for dialing a number, opening a website and to send an SMS. On selecting an option, the appropriate action should be invoked using intents.

Android Menus (Options, Context, Popup)

In android, **Menu** is a part of the user interface (UI) component which is used to handle some common functionality around the application. By using Menus in our applications, we can provide better and consistent user experience throughout the application.

We can use Menu APIs to represent user actions and other options in our android application activities.

Following is the pictorial representation of using menus in the android application.



In android, we can define a Menu in separate XML file and use that file in our activities or fragments based on our requirements.

Define an Android Menu in XML File

For all menu types, Android provides a standard XML format to define menu items. Instead of building a menu in our activity's code, we should define a menu and all its items in an XML menu resource and load menu resource as a Menu object in our activity or fragment.

	Exp.No. **Date:**

In android, to define menu, we need to create a new folder **menu** inside of our project resource directory (**res/menu/**) and add a new XML file to build the menu with the following elements.

Element	Description
<menu>	It's a root element to define a Menu in XML file and it will hold one or more and elements.
<item>	It is used to create a menu item and it represents a single item on the menu. This element may contain a nested <menu> element in order to create a submenu.
<group>	It's an optional and invisible for <item> elements. It is used to categorize the menu items so they share properties such as active state and visibility.

Following is the example of defining a menu in an XML file (**menu_example.xml**).

<?xml

<?xml

Attribute	Description
android: id	It is used to uniquely identify an element in the application.
android:icon	It is used to set the item's icon from drawable folder.
android: title	It is used to set the item's title
android:showAsAction	It is used to specify how the item should appear as an action item in the app bar.

```
<?xmlversion="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/mail"


 android:icon="@drawable/ic_mail"


         android:title="@string/mail”>
   <item android:id="@+id/upload"
         android:icon="@drawable/ic_upload"
        android:title="@string/upload"
        android:showAsAction="ifRoom" />
    <item android:id="@+id/share"
        android:icon="@drawable/ic_share"
android:icon="@drawable/ic_share"
        android:title="@string/share" />
</menu>
```

The **<item>** element in **menu** supports different type of attributes to define item’s behaviour and appearance. Following are the some of commonly used **<item>** attributes in android applications.

In case if we want to add **submenu** in **menu** item, then we need to add a **<menu>** element as the child of an **<item>**. Following is the example of defining a submenu in menu item.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/file"
        android:title="@string/file" >
        <!-- "file" submenu -->
        <menu>
            <item android:id="@+id/create_new"
                android:title="@string/create_new" />
            <item android:id="@+id/open"
                android:title="@string/open" />
        </menu>
    </item>
</menu>
```

Load Android Menu from an Activity

	Exp.No. **Date:**

Once we are done with creation of menu, we need to load the menu resource from our activity using **MenuInflater.inflate()** like as shown below.

```
 @Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {


super.onCreateContextMenu(menu, v, menuInfo);



MenuInflater inflater = getMenuInflater();
 inflater.inflate(R.menu.menu_example, menu);
}
```

If you observe above code we are calling our menu using MenuInflater.inflate() method in the form of **R.menu.menu_file_name**. Here our xml file name is **menu_example.xml** so we used file name **menu_example**.

Handle Android Menu Click Events

In android, we can handle a menu item click events using **ItemSelected()** event based on the menu type. Following is the example of handling a context menu item click event using **onContextItemSelected()**.

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.mail:
            // do something
            return true;
        case R.id.share:
            // do something
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

If you observe above code, the **getItemId()** method will get the id of selected menu item based on that we can perform our actions.

	Exp.No. **Date:**

Android Different Types of Menus

In android, we have a three fundamental type of Menus available to define a set of options and actions in our android applications.

The following are the commonly used Menus in android applications.

- Options Menu
- Context Menu
- Popup Menu

Android Options Menu

In android, **Options Menu** is a primary collection of menu items for an activity and it is useful to implement actions that have a global impact on the app, such as Settings, Search, etc.

To know more about Options Menu, check this Android Options Menu with Examples.

Android Context Menu

In android, **Context Menu** is a floating menu that appears when the user performs a long click on an element and it is useful to implement actions that affect the selected content or context frame.

To know more about Context Menu, check this Android Context Menu with Examples.

Android Popup Menu

In android, **Popup Menu** displays a list of items in a vertical list that's anchored to the view that invoked the menu and it's useful for providing an overflow of actions that related to specific content.

To know more about Popup Menu, check this Android Popup Menu with Examples.

Android Phone Calls with Examples

In android, we can easily make a phone call from our android applications by invoking built-in phone calls app using Intents action (**ACTION_CALL**).

Generally, the Intent object in android with proper action (**ACTION_CALL**) and data will help us to launch a built-in phone calls app to make a phone calls in our application.

In android, Intent is a messaging object which is used to request an action from another app component such as activities, services, broadcast receivers, and content providers. To know more about an Intent object in android check this Android Intents with Examples.

To make a phone call using Intent object in android application, we need to write the code like as shown below.

```
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
startActivity(callIntent);
```

If you observe above code, we are using Intent object **ACTION_CALL** action to make a phone call based on our requirements.

Now we will see how to make a phone call in android application using Intent action (**ACTION_CALL**) with examples.

Android Phone Call Example

Following is the example of making a phone call by invoking the default phone calls app using an Intent object in the android application.

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

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="Mobile No"
        />
    <EditText
        android:id="@+id/mblTxt"
        android:layout_width="wrap_content"


  android:layout_height="wrap_content"
  android:layout_marginLeft="100dp"
        android:ems="10">
    </EditText>
    <Button
        android:id="@+id/btnCall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:text="Call" />
</LinearLayout>
```

Now open our main activity file **MainActivity.java** from **\src\main\java\com.tutlane.phonecallexample** path and write the code like as shown below

MainActivity.java

	Exp.No. Date:

```
package com.tutlane.phonecallexample;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;


public class MainActivity extends AppCompatActivity {
    private EditText txtPhone;
    private Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


       txtPhone = (EditText)findViewById(R.id.mblTxt);
        btn = (Button)findViewById(R.id.btnCall);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                callPhoneNumber();
            }
        });
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults)
    {
        if(requestCode == 101)
        {
            if(grantResults[0] == PackageManager.PERMISSION_GRANTED)
            {
                callPhoneNumber();
            }
        }
    }

    public void callPhoneNumber()
    {
        try
        {
            if(Build.VERSION.SDK_INT > 22)
            {
                if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE) !=

PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(MainActivity.this, new String[]{Manifest.permission.CALL_PHO
NE}, 101);
                    return;
                }

                Intent callIntent = new Intent(Intent.ACTION_CALL);
                callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
                startActivity(callIntent);



}

else {
                Intent callIntent = new Intent(Intent.ACTION_CALL);
                callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
                startActivity(callIntent);
            }
        }
        catch (Exception ex)
        {
           ex.printStackTrace();
        }
    }
}
```

If you observe the above code we are adding **Runtime** permissions to make sure our application work in both old / latest android OS versions and we used Intent action (**ACTION_CALL**) to make a phone call on button click using default phone calls app. As discussed, we need to add **CALL_PHONE** permission in our android manifest.

Now open android manifest file (**AndroidManifest.xml**) and write the code like as shown below

AndroidManifest.xml

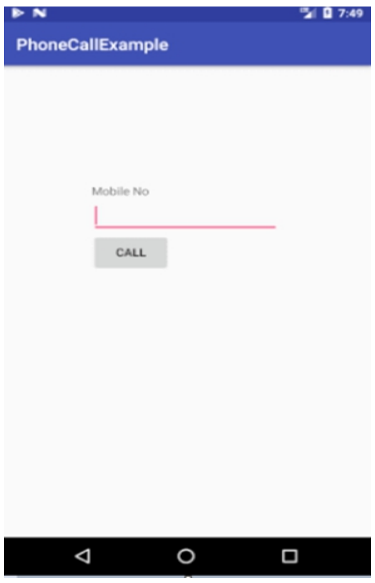
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.tutlane.phonecallexample">
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <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>
 </application>
</manifest>
```

If you observe above **AndroidManifest.xml** file we added a **CALL_PHONE** permission in manifest file.

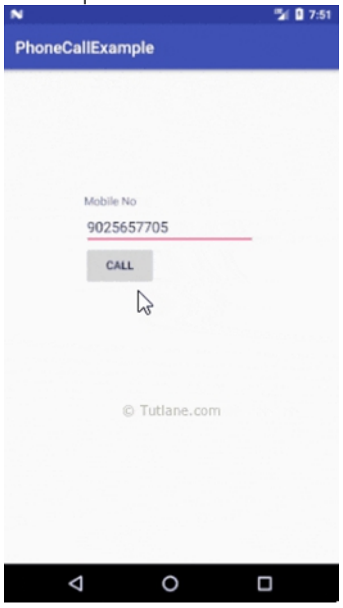
	Exp.No. **Date:**

Output of Android Phone Call Example

When we run the above program in the android studio we will get the result as shown below.



Once we enter the phone number and click on the **Call** button, it will invoke built-in phone calls app and it will make a call to respective phone numbers like as shown below.



This is how we can make phone calls using Intents in android applications based on our requirements.

	Exp.No. **Date:**

Android Send SMS with Examples

In android, we can send SMS from our android application in two ways either by using **SMSManager** API or Intents based on our requirements.

If we use **SMSManager** API, it will directly send SMS from our application. In case if we use Intent with proper action (**ACTION_VIEW**), it will invoke a built-in SMS app to send SMS from our application.

Android Send SMS using SMSManager API

In android, to send SMS using SMSManager API we need to write the code like as shown below.

```
SmsManager smgr = SmsManager.getDefault();
smgr.sendTextMessage(MobileNumber,null,Message,null,null);
```

SMSManager API required **SEND_SMS** permission in our android manifest to send SMS. Following is the code snippet to set **SEND_SMS** permissions in manifest file.

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

Android Send SMS using Intent

In android, Intent is a messaging object which is used to request an action from another app component such as activities, services, broadcast receivers, and content providers. To know more about an Intent object in android check this Android Intents with Examples.

To send SMS using the Intent object, we need to write the code like as shown below.

```
Intent sInt = new Intent(Intent.ACTION_VIEW);
sInt.putExtra("address", new String[]{txtMobile.getText().toString()});
sInt.putExtra("sms_body",txtMessage.getText().toString());
sInt.setType("vnd.android-dir/mms-sms");
```

Even for Intent, it required a **SEND_SMS** permission in our android manifest to send SMS. Following is the code snippet to set **SEND_SMS** permissions in manifest file.

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

Now we will see how to send SMS in android application using **SMSManager** API with examples.

Android Send SMS Example

Following is the example to send SMS using **SMSManager** API in the android application.

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

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="Mobile No" />

<EditText
        android:id="@+id/mblTxt"
        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="Message"
        android:layout_marginLeft="100dp" />
    <EditText
        android:id="@+id/msgTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10" />
    <Button
        android:id="@+id/btnSend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:text="Send SMS" />
</LinearLayout>
```

Now open our main activity file **MainActivity.java** from **\src\main\java\com.tutlane.sendsmsexample** path and write the code like as shown below

MainActivity.java

```
package com.tutlane.sendsmsexample;
import android.content.Intent;
import android.net.Uri;
import android.provider.Telephony;
import android.support.v7.app.AppCompatActivity;


import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private EditText txtMobile;
    private EditText txtMessage;
    private Button btnSms;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtMobile = (EditText)findViewById(R.id.mblTxt);
        txtMessage = (EditText)findViewById(R.id.msgTxt);
        btnSms = (Button)findViewById(R.id.btnSend);
        btnSms.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try{
                    SmsManager smgr = SmsManager.getDefault();
                    smgr.sendTextMessage(txtMobile.getText().toString(),null,txtMessage.getText().toString(),
null,null);
                    Toast.makeText(MainActivity.this, "SMS Sent Successfully",
Toast.LENGTH_SHORT).show();
                }
                catch (Exception e){
                    Toast.makeText(MainActivity.this, "SMS Failed to Send, Please try again",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

If you observe above code, we are sending SMS using **SMSManager** api on button click. As discussed, we need to add a **SEND_SMS** permission in our android manifest.

Now open android manifest file (**AndroidManifest.xml**) and write the code 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.tutlane.sendsmsexample">
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
```

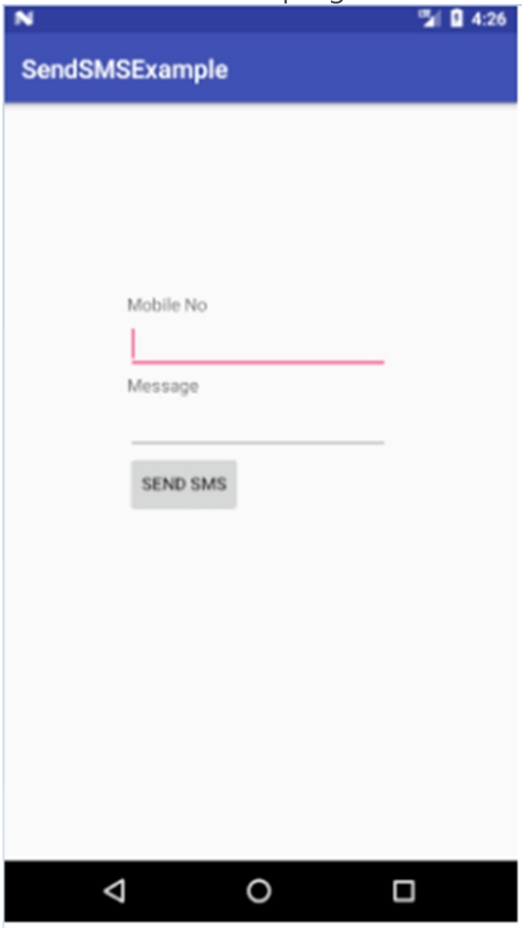
	Exp.No. Date:

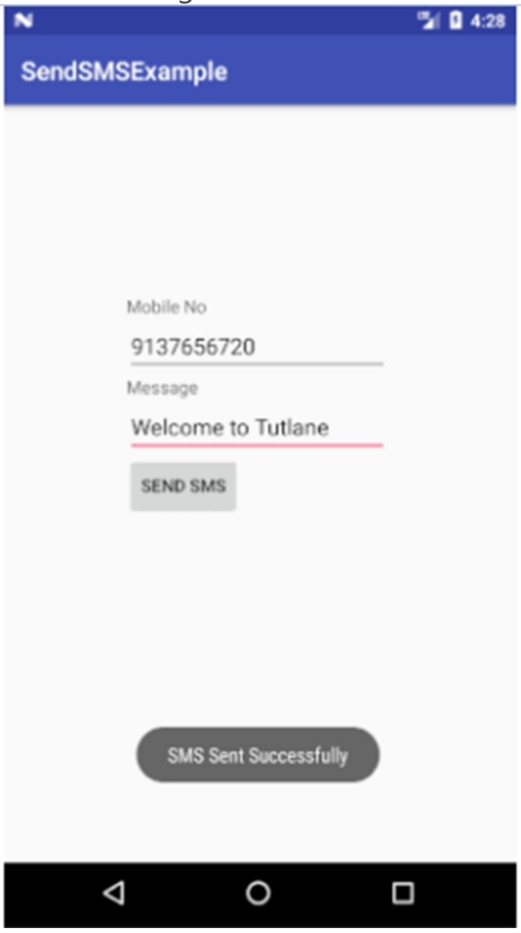
```
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

If you observe above **AndroidManifest.xml** file, we added a **SEND_SMS** permissions in manifest file.

Output of Android Send SMS Example

When we run above program in android studio we will get the result like as shown below.





Once you enter all details and click on **Send SMS** button it will send SMS and show the alert message like as mentioned in above image. The above example we implemented using **SMSManager** API. In case if we want to use Intents to send SMS to replace button click code like as shown below.

```
btnSms.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      try{
         Intent i = new Intent(Intent.ACTION_VIEW);
         i.setData(Uri.parse("smsto:"));
         i.setType("vnd.android-dir/mms-sms");
         i.putExtra("address", new String(txtMobile.getText().toString()));
         i.putExtra("sms_body",txtMessage.getText().toString());
         startActivity(Intent.createChooser(i, "Send sms via:"));
      }
      catch(Exception e){
         Toast.makeText(MainActivity.this, "SMS Failed to Send, Please try again",

Toast.LENGTH_SHORT).show();
      }
   }
});
```

This is how we can send SMS using either SMSManager API or Intent objects in android applications based on our requirements.

EXPERIMENT 6

Develop an application that inserts some notifications into Notification area and whenever a notification is inserted, it should show a toast with details of the notification.

MainActivity.java

```
package com.programmerworld.mynotificationapp;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NotificationCompat;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    private String CHANNEL_ID = "My Notification";

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

    public void buttonSetNotification(View view){
        Intent intentNotification = new Intent(this, MainActivity.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
intentNotification, PendingIntent.FLAG_UPDATE_CURRENT);

        NotificationCompat.Builder builderNotificationCompat = new
NotificationCompat.Builder(this, CHANNEL_ID)
                .setContentIntent(pendingIntent)
                .setContentTitle("My Notification")
                .setContentText("My Notification text here.")
                .setSmallIcon(android.R.drawable.btn_star_big_on);
        NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        NotificationChannel notificationChannel = new
NotificationChannel(CHANNEL_ID, "This is my first Notification",
NotificationManager.IMPORTANCE_DEFAULT);


notificationManager.createNotificationChannel(notificationChannel);
notificationManager.notify(0,builderNotificationCompat.build());
    }
}
```

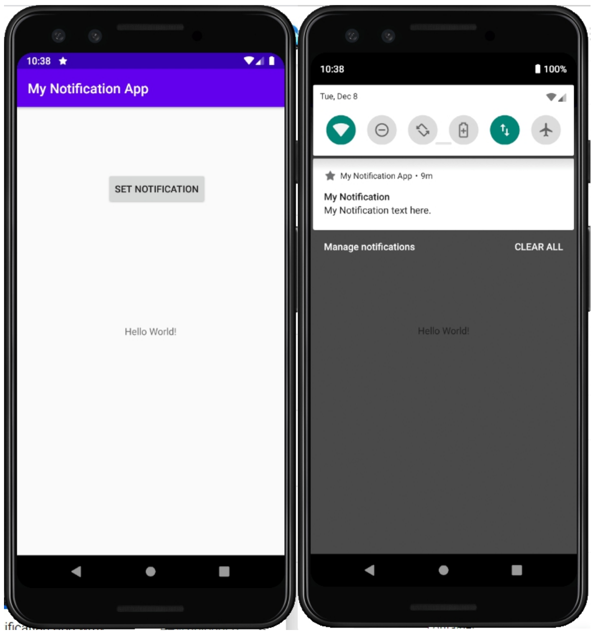
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="132dp"
        android:layout_marginTop="96dp"
        android:onClick="buttonSetNotification"
        android:text="@string/set_notification"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>



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

```
    package="com.programmerworld.mynotificationapp">

    <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>
    </application>

</manifest>
// Top-level build file where you can add configuration options common to
all sub-projects/modules.
buildscript {
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.0.1"

        // NOTE: Do not place your application dependencies here; they
belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}
```

```
task clean(type: Delete) {
    delete rootProject.buildDir
}
apply plugin: 'com.android.application'



android {

    compileSdkVersion 30
    buildToolsVersion "29.0.3"

    defaultConfig {
        applicationId "com.programmerworld.mynotificationapp"
        minSdkVersion 26
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

}
```

	Exp.No. **Date:**


10:38
My Notification App
SET NOTIFICATION
Hello World!
10:38
100%
Tue, Dec 8
My Notification App • 9m
My Notification
My Notification text here.
Manage notifications
CLEAR ALL
Hello World!

EXPERIMENT- 7 USING INTENTS

Activity_a.xml

```
<?xml version="1.0" encoding="utf-8"?>
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    tools:context=".ActivityA">
    <TextView
        android:layout_width="wrap_content"
android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Large Text"
        android:id="@+id/textView1"
        android:layout_centerVertical="true"
android:layout_centerHorizontal="true" />

    <EditText  android:layout_width="200dp"
        android:layout_height="wrap_content"  android:id="@+id/editText1"
        android:layout_above="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="77dp" />
    <Button
        android:layout_width="wrap_content"  android:layout_height="wrap_content"
        android:text="Ask Question"  android:id="@+id/button1"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:onClick="onClick"  android:layout_marginTop="56dp" />
</RelativeLayout>
```

Activity_b.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"  tools:context=".ActivityB">
<Button android:id="@+id/button1"  android:layout_width="wrap_content"

    android:layout_height="wrap_content"
    android:layout_below="@+id/editText1"
    android:layout_centerHorizontal="true"

  android:layout_marginTop="86dp"    android:text="@string/answer_text" />
<TextView  android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="35dp"  android:text="Large Text"
    android:textAppearance="?android:attr/textAppearanceLarge" />
<EditText  android:id="@+id/editText1"   android:layout_width="300dp"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="66dp" android:ems="10"  android:inputType="text" />
    </RelativeLayout>
```

ActivityA.java

```
package com.example.inentsapp1;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
public class ActivityA extends Activity {
    private static final int request_code = 5;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_a);
    }
    public void onClick(View view) {
        Intent i = new Intent(this, ActivityB.class);
        final EditText editText1 = (EditText) findViewById(R.id.editText1);
        String myString = editText1.getText().toString();
        i.putExtra("qString", myString);
      // startActivity(i);
        startActivityForResult(i, request_code);
    }
    protected void onActivityResult(int requestCode, int resultCode, Intent data)

    {  if ((requestCode == request_code) &&
```

Exp.No.
Date:

```
 (resultCode == RESULT_OK)) {
        TextView textView1 =
                (TextView) findViewById(R.id.textView1);


         String returnString =  data.getExtras().getString("returnData");

        textView1.setText(returnString);
    }
    }
}
```

ActivityB.java

```
package com.example.inentsapp1;
import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.view.View;
import android.widget.TextView;
import android.widget.EditText;
public class ActivityB extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_b);
        Bundle extras = getIntent().getExtras();
        if (extras == null) {  return;       }
        String qString = extras.getString("qString");
        final TextView textView = (TextView) findViewById(R.id.textView1);
        textView.setText(qString);
    }
        public void onClick (View view){
            finish();
        }
        @Override
        public void finish () {
            Intent data = new Intent();
            EditText editText1 = (EditText) findViewById(R.id.editText1);
            String returnString = editText1.getText().toString();
            data.putExtra("returnData", returnString);
            setResult(RESULT_OK, data);
            super.finish();
        }   }
```

Experiment 8–USING SHARED PREFERENCES

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
     >
    <Button
        android:id="@+id/btnSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:onClick="Save"
        android:text="Save" />

    <Button
        android:id="@+id/btnRetr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:onClick="Get"
        android:text="Retrieve" />

    <Button
        android:id="@+id/btnClear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/etEmail"
        android:layout_centerVertical="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:onClick="clear"
        android:text="Clear" />

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

       android:layout_marginTop="20dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <EditText
        android:id="@+id/etName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Name"
        android:inputType="text"
        android:layout_alignParentTop="true"
        android:layout_alignLeft="@+id/etEmail"
        android:layout_alignStart="@+id/etEmail" />

</RelativeLayout>
```

MainActivity.java

```
package com.example.sharedpreferencesapp;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends Activity {
    SharedPreferences sharedpreferences;
    TextView name;
    TextView email;
    public static final String mypreference = "mypref";
    public static final String Name = "nameKey";
    public static final String Email = "emailKey";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        name = (TextView) findViewById(R.id.etName);
        email = (TextView) findViewById(R.id.etEmail);
        sharedpreferences = getSharedPreferences(mypreference,
                Context.MODE_PRIVATE);
        if (sharedpreferences.contains(Name)) {
            name.setText(sharedpreferences.getString(Name, ""));

}
        if (sharedpreferences.contains(Email)) {
            email.setText(sharedpreferences.getString(Email, ""));

        }

    }

    public void Save(View view) {
        String n = name.getText().toString();
        String e = email.getText().toString();
        SharedPreferences.Editor editor = sharedpreferences.edit();
        editor.putString(Name, n);
        editor.putString(Email, e);
        editor.commit();
    }

    public void clear(View view) {
        name = (TextView) findViewById(R.id.etName);
        email = (TextView) findViewById(R.id.etEmail);
        name.setText("");
        email.setText("");

    }

    public void Get(View view) {
        name = (TextView) findViewById(R.id.etName);
        email = (TextView) findViewById(R.id.etEmail);
        sharedpreferences = getSharedPreferences(mypreference,
                Context.MODE_PRIVATE);

        if (sharedpreferences.contains(Name)) {
            name.setText(sharedpreferences.getString(Name, ""));
        }
        if (sharedpreferences.contains(Email)) {
            email.setText(sharedpreferences.getString(Email, ""));

        }
    }
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

Experiment-9 –USING FILES

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:paddingLeft="100dp"
        android:text="Student Details:" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="20dp"
        android:layout_marginTop="24dp"
        android:ems="10" >
       <requestFocus />
    </EditText>
    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/editText1"
        android:layout_below="@+id/editText1"
        android:layout_marginTop="24dp"
        android:ems="10" />
    <EditText
        android:id="@+id/editText3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/editText2"
        android:layout_below="@+id/editText2"
        android:layout_marginTop="24dp"
        android:ems="10" />
```

```
<EditText
   android:id="@+id/editText4"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignRight="@+id/editText3"
   android:layout_below="@+id/editText3"
   android:layout_marginTop="24dp"
   android:ems="10" />
   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/editText1"
      android:layout_alignBottom="@+id/editText1"
      android:layout_alignParentLeft="true"
      android:paddingLeft="10dp"
      android:text="Student Name:" />
   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/editText2"
      android:layout_alignBottom="@+id/editText2"
      android:layout_alignParentLeft="true"
      android:paddingLeft="10dp"
      android:text="HTNO" />
   <TextView
      android:id="@+id/textView3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/editText3"
      android:layout_alignBottom="@+id/editText3"
      android:layout_alignParentLeft="true"
      android:paddingLeft="10dp"
      android:text="Marks:" />
   <TextView
      android:id="@+id/textView4"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/editText4"
      android:layout_alignBottom="@+id/editText4"
      android:layout_alignParentLeft="true"
      android:paddingLeft="10dp"
      android:text="Data:" />
   <Button
```

```
        android:id="@+id/button1"

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText4"
        android:layout_below="@+id/editText4"
        android:layout_marginLeft="70dp"
        android:layout_marginTop="16dp"
        android:text="save" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/button1"
        android:layout_alignBottom="@+id/button1"
        android:layout_toRightOf="@+id/button1"
        android:text="read" />

</RelativeLayout>
```

MainActivity.java

```
package com.example.fileapp;

import android.content.Context;
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 java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;

public class MainActivity extends AppCompatActivity {
    EditText editTextStudentName,editTextHTNo,editTextMarks,editTextData;
    Button saveButton,readButton;
    @Override
```

	Exp.No. **Date:**

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

    editTextStudentName=findViewById(R.id.editText1);
    editTextHTNo=findViewById(R.id.editText2);
    editTextMarks=findViewById(R.id.editText3);
    editTextData=findViewById(R.id.editText4);
    saveButton=findViewById(R.id.button1);
    readButton=findViewById(R.id.button2);

    //Performing Action on Read Button
    saveButton.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View arg0) {
            String filename="IT32";
            String studentName=editTextStudentName.getText().toString();
            String HTNo=editTextHTNo.getText().toString();
            String Marks=editTextMarks.getText().toString();
            String Data=editTextData.getText().toString();
            FileOutputStream fos;
            try {
                fos = openFileOutput(filename, Context.MODE_PRIVATE);
                //default mode is PRIVATE, can be APPEND etc.
                fos.write(studentName.getBytes());
                fos.write("\n".getBytes());
                fos.write(HTNo.getBytes());
                fos.write("\n".getBytes());
                fos.write(Marks.getBytes());
                fos.write("\n".getBytes());
               fos.write(Data.getBytes());
                fos.write("\n".getBytes());
                fos.close();
                Toast.makeText(getApplicationContext(),"Student Data saved",
                        Toast.LENGTH_LONG).show();
            } catch (FileNotFoundException e) {e.printStackTrace();}
            catch (IOException e) {e.printStackTrace();}
        }
    });

    //Performing Action on Read Button
    readButton.setOnClickListener(new View.OnClickListener(){

        @Override
        public void onClick(View arg0) {
```

	Exp.No. Date:

```
              String filename="IT32";
                 StringBuffer stringBuffer = new StringBuffer();
              try {
  //Attaching BufferedReader to the FileInputStream by the help of InputStreamReader
    BufferedReader inputReader = new BufferedReader(new InputStreamReader(

                openFileInput(filename)));
                 String inputString;
                 //Reading data line by line and storing it into the stringbuffer
                 while ((inputString = inputReader.readLine()) != null) {
                    stringBuffer.append(inputString + "\n");
                 }

              } catch (IOException e) {
                 e.printStackTrace();
              }
              //Displaying data on the toast

Toast.makeText(getApplicationContext(),stringBuffer.toString(),Toast.LENGTH_LONG).show();
         }
      });
   }
}
```

	Exp.No. **Date:**

Experimnet 10
Saving Data to Internal Storage as files

Aim
To create an android application to save data in a text file.

Objective
Create an android application to save data in a text file. Load file from memory and show in the view.

Apparatus/Tools/Equipments/Components
Android Studio

Principle/Theory/Flow-chart/Algorithm

Procedure

FileActivity.java

```
import android.app.Activity; import android.os.Bundle; import android.view.View; import
android.widget.Button; import android.widget.EditText; import android.widget.Toast;
import java.io.FileInputStream; import java.io.FileOutputStream;
import java.io.InputStreamReader; import java.io.OutputStreamWriter;

public class FileActivity extends Activity {


EditText mEtText;
Button mBtnSave, mBtnLoad;

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


mEtText = (EditText) findViewById(R.id.et_message); mBtnSave = (Button)
findViewById(R.id.btn_save); mBtnLoad = (Button) findViewById(R.id.btn_load);


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

```
String message = mEtText.getText().toString(); try {

FileOutputStream fout = openFileOutput("textfile.txt", MODE_PRIVATE);
OutputStreamWriter osw = new OutputStreamWriter(fout);


osw.write(message); osw.flush();
osw.close();
Toast.makeText(getApplicationContext(), "File saved successfully",
Toast.LENGTH_LONG).show();
mEtText.setText("");

} catch (Exception e) { e.printStackTrace();
}


}
});

mBtnLoad.setOnClickListener(new View.OnClickListener() { @Override
public void onClick(View v) { try {

FileInputStream fin = openFileInput("textfile.txt"); InputStreamReader isr = new
InputStreamReader(fin);

char[] inputBuffer = new char[100]; String s = "";
int charRead;

while ((charRead = isr.read(inputBuffer)) > 0) {
String readString = String.copyValueOf(inputBuffer, 0, charRead); s += readString;
inputBuffer = new char[100];
}

mEtText.setText(s);
Toast.makeText(getApplicationContext(), "File loaded successfully",
Toast.LENGTH_LONG).show();


} catch (Exception e) { e.printStackTrace();
}

}

});

}
}
```

activity_file.xml

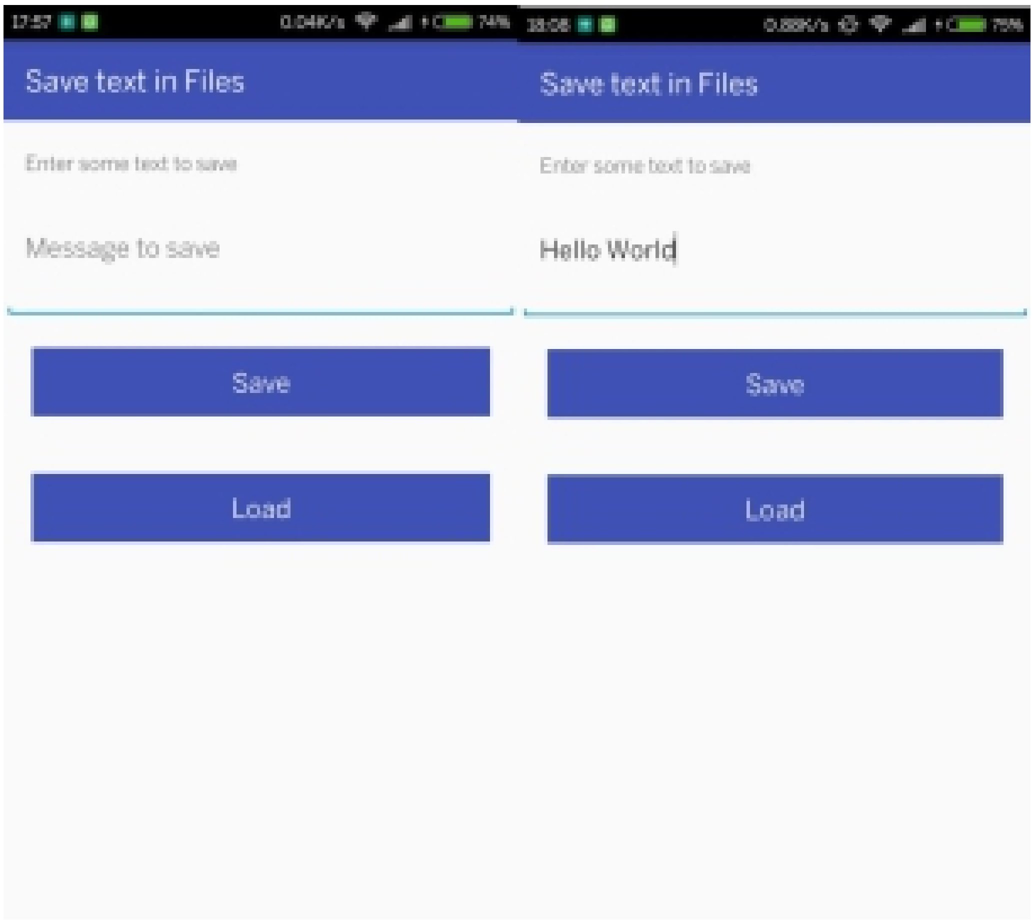
```
<?xml  version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent" android:layout_height="match_parent"
android:orientation="vertical">

<TextView
android:id="@+id/textView2" android:layout_width="match_parent"
android:layout_height="wrap_content" android:layout_gravity="center_horizontal"
android:background="@color/colorPrimary" android:gravity="center_vertical"
android:paddingBottom="15dp" android:paddingLeft="15dp" android:paddingTop="15dp"
android:text="Save text in Files" android:textColor="#ffffff" android:textSize="20sp" />

<TextView
android:layout_width="match_parent" android:layout_height="wrap_content"
android:layout_gravity="center_horizontal" android:layout_marginTop="20dp"
android:paddingLeft="15dp" android:text="Enter some text to save" />

<EditText
android:id="@+id/et_message" android:layout_width="match_parent"
android:layout_height="100dp" android:ems="10" android:hint="Message to save"
android:paddingLeft="15dp" />

<Button
android:id="@+id/btn_save" android:layout_width="match_parent"
android:layout_height="wrap_content" android:layout_gravity="center_horizontal"
android:layout_margin="20dp" android:background="@color/colorPrimary"
android:text="Save" android:textColor="#ffffff" />

<Button
```

```
android:id="@+id/btn_load" android:layout_width="match_parent"
android:layout_height="wrap_content" android:layout_gravity="center_horizontal"
android:layout_margin="20dp" android:background="@color/colorPrimary"
android:text="Load" android:textColor="#ffffff" />

</LinearLayout>
```

Observations

Input-Output

Screenshots



EXPERIMENT 11

Create an application that shows the given URL (from a text field) in a browser.

Android - WebView

WebView is a view that display web pages inside your application. You can also specify HTML string and can show it inside your application using WebView. WebView makes turns your application to a web application.

In order to add WebView to your application, you have to add **<WebView>** element to your xml layout file. Its syntax is as follows −

```
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/webview"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
/>
```

In order to use it, you have to get a reference of this view in Java file.

To get a reference, create an object of the class WebView. Its syntax is −

WebView browser = (WebView) findViewById(R.id.webview);

In order to load a web url into the WebView, you need to call a method **loadUrl(String url)** of the WebView class, specifying the required url. Its syntax is:

browser.loadUrl("http://www.tutorialspoint.com");

Apart from just loading url, you can have more control over your WebView by using the methods defined in WebView class. They are listed as follows −

Sr.No	Method & Description
1	**canGoBack()** This method specifies the WebView has a back history item.
2	**canGoForward()** This method specifies the WebView has a forward history item.

3	**clearHistory()** This method will clear the WebView forward and backward history.
4	**destroy()** This method destroy the internal state of WebView.
5	**findAllAsync(String find)** This method find all instances of string and highlight them.
6	**getProgress()** This method gets the progress of the current page.
7	**getTitle()** This method return the title of the current page.
8	**getUrl()** This method return the url of the current page.

If you click on any link inside the webpage of the WebView, that page will not be loaded inside your WebView. In order to do that you need to extend your class from **WebViewClient** and override its method. Its syntax is −

```
private class MyBrowser extends WebViewClient {
   @Override
   public boolean shouldOverrideUrlLoading(WebView view, String url) {
      view.loadUrl(url);
      return true;
   }
}
```

Here is an example demonstrating the use of WebView Layout. It creates a basic web application that will ask you to specify a url and will load this url website in the WebView.

To experiment with this example, you need to run this on an actual device on which internet is running.

Steps	Description
1	You will use Android studio to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to add WebView code.
3	Modify the res/layout/activity_main to add respective XML components
4	Modify the AndroidManifest.xml to add the necessary permissions
5	Run the application and choose a running android device and install the application on it and verify the results.

Following is the content of the modified main activity file **src/MainActivity.java**.

```
package com.example.myapplication;
import android.app.Activity;
import android.os.Bundle;

import android.view.View;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import android.widget.EditText;


public class MainActivity extends Activity  {
   Button b1;

EditText ed1;


 private WebView wv1;

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

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

      wv1=(WebView)findViewById(R.id.webView);
      wv1.setWebViewClient(new MyBrowser());

      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            String url = ed1.getText().toString();

            wv1.getSettings().setLoadsImagesAutomatically(true);
            wv1.getSettings().setJavaScriptEnabled(true);
wv1.setScrollBarStyle(View.SCROLLBARS_INSIDE_OVERLAY);
            wv1.loadUrl(url);
         }
      });
   }

   private class MyBrowser extends WebViewClient {
      @Override
      public boolean shouldOverrideUrlLoading(WebView view, String url) {
         view.loadUrl(url);
         return true;
      }
   }
}
```

Following is the modified content of the xml **res/layout/activity_main.xml**.

In the following code **abc** indicates the logo of tutorialspoint.com

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"



 xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
   android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"

  android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"

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

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

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

   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/editText"
      android:hint="Enter Text"
      android:focusable="true"
      android:textColorHighlight="#ff7eff15"
      android:textColorHint="#ffff25e6"
      android:layout_marginTop="46dp"
      android:layout_below="@+id/imageView"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_alignRight="@+id/imageView"
      android:layout_alignEnd="@+id/imageView" />

   <ImageView

android:layout_width="wrap_content"
      android:layout_height="wrap_content"

 android:id="@+id/imageView"
      android:src="@drawable/abc"
      android:layout_below="@+id/textView"
      android:layout_centerHorizontal="true" />

   <Button
      android:layout_width="wrap_content"

      android:layout_height="wrap_content"
      android:text="Enter"
      android:id="@+id/button"
      android:layout_alignTop="@+id/editText"
      android:layout_toRightOf="@+id/imageView"
      android:layout_toEndOf="@+id/imageView" />

   <WebView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/webView"
      android:layout_below="@+id/button"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true"
      android:layout_alignParentBottom="true" />

</RelativeLayout>
```

Following is the content of the **res/values/string.xml**.

```
<resources>
   <string name="app_name">My Application</string>
</resources>
```

	Exp.No. **Date:**

Following is the content of **AndroidManifest.xml** file.

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


 <activity
      android:name=".MainActivity"
      android:label="@string/app_name" >

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

    </activity>


</application>
</manifest>
```

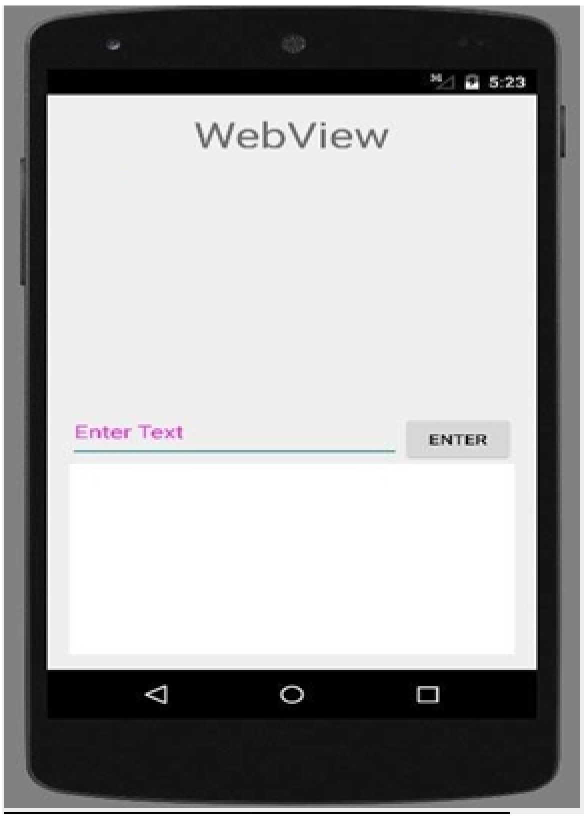
Let's try to run your WebView application. To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. Android studio will display as shown below

Now just specify a url on the url field and press the browse button that appears, to launch the website. But before that please make sure that you are connected to the internet. After pressing the button, the following screen would appear –

Note. By just changing the url in the url field, your WebView will open your desired website.

Above image shows webview of tutorialspoint.com

	Exp.No. **Date:**



EXPERIMENT 13

12. Create an alarm that rings every Sunday at 8:00 AM. Modify it to use a time picker to set alarm time.

```
package com.example.myalarm;

import android.media.Ringtone;

import android.media.RingtoneManager;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.widget.TextClock;

import android.widget.TimePicker;



import java.util.Timer;

import java.util.TimerTask;

public class MainActivity extends AppCompatActivity {

TimePicker alarmTime;

TextClock currentTime;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

alarmTime = findViewById(R.id.timePicker);

currentTime = findViewById(R.id.textClock);
```

	Exp.No. **Date:**

```
final Ringtone r = RingtoneManager.getRingtone(getApplicationContext(),
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_RINGTONE));



Timer t = new Timer();

t.scheduleAtFixedRate(new TimerTask() {

@Override

public void run() {

if (currentTime.getText().toString().equals(AlarmTime())){

r.play();

}else{

r.stop();



}

}

}, 0, 1000);

}

public String AlarmTime(){

Integer alarmHours = alarmTime.getCurrentHour();

Integer alarmMinutes = alarmTime.getCurrentMinute();

String stringAlarmMinutes;

if (alarmMinutes<10){
```

```
stringAlarmMinutes = “0”;


stringAlarmMinutes = stringAlarmMinutes.concat(alarmMinutes.toString());

}else{

stringAlarmMinutes = alarmMinutes.toString();


}String stringAlarmTime;

if(alarmHours>12){

alarmHours = alarmHours – 12;

stringAlarmTime = alarmHours.toString().concat(“:”).concat(stringAlarmMinutes).concat(” PM”);

}else{

stringAlarmTime = alarmHours.toString().concat(“:”).concat(stringAlarmMinutes).concat(” AM”);

}


return stringAlarmTime;

}

}

<?xml version=”1.0″ encoding=”utf-8″?>

<manifest xmlns:android=”http://schemas.android.com/apk/res/android&#8221;

package=”com.example.myalarm”>

<application
```

	Exp.No. **Date:**

```
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>

</application>



</manifest>

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android&#8221;

xmlns:app="http://schemas.android.com/apk/res-auto&#8221;

xmlns:tools="http://schemas.android.com/tools&#8221;
```

```
android:layout_width=”match_parent”

android:layout_height=”match_parent”

tools:context=”com.example.myalarm.MainActivity”>



<TimePicker

android:id=”@+id/timePicker”

android:layout_width=”300dp”

android:layout_height=”wrap_content”

android:layout_alignParentBottom=”true”

android:layout_centerHorizontal=”true” />

<TextClock



android:id=”@+id/textClock”

android:layout_width=”150dp”

android:layout_height=”80dp”

android:layout_alignParentTop=”true”

android:layout_centerHorizontal=”true”
  android:layout_marginTop=”37dp” />

</RelativeLayout>
```

Exp.No.
Date:

EXPERIMENT 14

Develop an application that shows all contacts of the phone along with details like name, phone number, mobile number etc.

Android Contacts : Get Contact Details Phone Number Programmatically

In this **get contact list details programmatically** we will learn how to retrieve contact **name, phone number, email**, etc. information in Android app programmatically.

Two examples are there in this as per the below.

1. Android Get Contact Details Programmatically

2. Android Get Contact List And Display in ListView

1. Android Get Contact Details Programmatically

Contact based application will need to access contact details, this tutorial will guide you for this purpose.

First, check the output of get contact list details Android Studio programmatically then we will implement it.

Step 2: Updating AndroidManifest.xml file

add required permissions between <manifest>….</manifest> tag.

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

Final code for **AndroidManifest.xml** file

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

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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>
    </application>

</manifest>
```

Step 3: Updating activity_main.xml file

We will make user interface for getting name, phone number and email address.

Three textviews are used for this purpose.

activity_main.xml file

```
<?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:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.getcontactdetailsdemonuts.MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/tvname"
        android:textColor="#000"
        android:text="Name:"/>

    <TextView

    android:layout_width="match_parent"
        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/tvphone"
        android:textColor="#000"
        android:text="Phone:"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/tvmail"
        android:textColor="#000"
        android:text="Email:"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:id="@+id/btn"
        android:text="Select contact" />
</LinearLayout>
```

Step 4: Preparing MainActivity.java class

MainActivity.java

```
import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
```

	Exp.No. **Date:**

```
    private Button btn;
    private TextView tvname, tvphone,tvmail;


@Override
    protected void onCreate(Bundle savedInstanceState) {

 super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        btn = (Button) findViewById(R.id.btn);
        tvname = (TextView) findViewById(R.id.tvname);
        tvphone = (TextView) findViewById(R.id.tvphone);
        tvmail = (TextView) findViewById(R.id.tvmail);


        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_PICK,
ContactsContract.Contacts.CONTENT_URI);
                startActivityForResult(intent, 1);
            }
        });

    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == Activity.RESULT_OK) {
            Uri contactData = data.getData();
            Cursor c =  getContentResolver().query(contactData, null, null, null, null);
            if (c.moveToFirst()) {

                String phoneNumber="",emailAddress="";
                String name =
c.getString(c.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
                String contactId = c.getString(c.getColumnIndex(ContactsContract.Contacts._ID));
                //http://stackoverflow.com/questions/866769/how-to-call-android-contacts-
list   our upvoted answer

                String hasPhone =
c.getString(c.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER));

                if ( hasPhone.equalsIgnoreCase("1"))
                    hasPhone = "true";
                else
```

```
                    hasPhone = "false" ;

                if (Boolean.parseBoolean(hasPhone))
                {



Cursor phones =
getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
null,ContactsContract.CommonDataKinds.Phone.CONTACT_ID +" = "+ contactId,null, null);
                    while (phones.moveToNext())

 {


phoneNumber=phones.getString(phones.getColumnIndex(ContactsContract.CommonDataK
inds.Phone.NUMBER));
                    }
                    phones.close();
                }

                // Find Email Addresses
                Cursor emails =
getContentResolver().query(ContactsContract.CommonDataKinds.Email.CONTENT_URI,nul
l,ContactsContract.CommonDataKinds.Email.CONTACT_ID + " = " + contactId,null, null);
                while (emails.moveToNext())
                {
                    emailAddress =
emails.getString(emails.getColumnIndex(ContactsContract.CommonDataKinds.Email.DATA)
);
                }
                emails.close();

                //mainActivity.onBackPressed();
                // Toast.makeText(mainactivity, "go go go", Toast.LENGTH_SHORT).show();

                tvname.setText("Name: "+name);
                tvphone.setText("Phone: "+phoneNumber);
                tvmail.setText("Email: "+emailAddress);
                Log.d("curs", name + " num" + phoneNumber + " " + "mail" + emailAddress);
              }
            c.close();
        }
    }

}
```

2. Android Get Contact List And Display in ListView

Welcome to android get contact list with phone numbers tutorial.

Get contact list in Android Studio example guides you to get a contact list and show in the custom listview.

Get contact list in Android example will show how to show all contacts alphabetically.

First, go through the output, then we will develop with Android Studio.

Step 2: Updating AndroidManifest.xml file

add required permissions between <manifest>….</manifest> tag.

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

Final code for **AndroidManifest.xml** file

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

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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>
    </application>

</manifest>
```

Step 3: Adding ListView Item file

Make a new layout resource file named **"lv_item.xml"**

```
 <?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#fff"
            android:textSize="25sp"

             android:gravity="center_vertical"
            android:paddingLeft="10dp"
            android:layout_marginTop="10dp"
            android:text="Name" />

        <TextView
            android:id="@+id/number"
            android:text="ds"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#fff"
            android:textSize="25sp"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="10dp" />

    </LinearLayout>

    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:background="@color/colorAccent"/>
```

```
</LinearLayout>
```

Step 4: Creating a model class

```
public class ContactModel {

    private String name, number;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;

}
}
```

Step 5: Making a custom adapter class

CustomAdapter.java

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;
import java.util.ArrayList;

public class CustomAdapter extends BaseAdapter {

    private Context context;
    private ArrayList<ContactModel> contactModelArrayList;
    public CustomAdapter(Context context, ArrayList<ContactModel> contactModelArrayList)
{

        this.context = context;
        this.contactModelArrayList = contactModelArrayList;
    }


 @Override
    public int getViewTypeCount() {
        return getCount();
    }
    @Override
    public int getItemViewType(int position) {

        return position;
    }

    @Override
    public int getCount() {
        return contactModelArrayList.size();
    }


  @Override
    public Object getItem(int position) {
        return contactModelArrayList.get(position);
    }

    @Override
    public long getItemId(int position) {

return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder;

        if (convertView == null) {
            holder = new ViewHolder();
            LayoutInflater inflater = (LayoutInflater) context
                    .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            convertView = inflater.inflate(R.layout.lv_item, null, true);

            holder.tvname = (TextView) convertView.findViewById(R.id.name);
            holder.tvnumber = (TextView) convertView.findViewById(R.id.number);
            convertView.setTag(holder);
        }else {
            // the getTag returns the viewHolder object set as a tag to the view
            holder = (ViewHolder)convertView.getTag();
        }

        holder.tvname.setText(contactModelArrayList.get(position).getName());
        holder.tvnumber.setText(contactModelArrayList.get(position).getNumber());


 return convertView;
    }

    private class ViewHolder {

        protected TextView tvname, tvnumber;

    }
}
```

Step 6: Updating MainActivity

Update **activity_main.xml** as below source code

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.contactlistviewdemonuts.MainActivity">

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

         android:id="@+id/listView"
        android:divider="@null"
        />

</RelativeLayout>
```

MainActivity.java

```
import android.database.Cursor;
import android.provider.ContactsContract;
```

	Exp.No. Date:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.ListView;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private ListView listView;
    private CustomAdapter customAdapter;
    private ArrayList<ContactModel> contactModelArrayList;

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

        listView = (ListView) findViewById(R.id.listView);

        contactModelArrayList = new ArrayList<>();

        Cursor phones = getContentResolver().query(ContactsContract.CommonDataKinds.
                 Phone.CONTENT_URI, null,null,null,
          ContactsContract.CommonDataKinds.
        Phone.DISPLAY_NAME+" ASC");
        while (phones.moveToNext())
        {
            String name=phones.getString(phones.getColumnIndex(ContactsContract.

                                 CommonDataKinds.Phone.DISPLAY_NAME));
            String phoneNumber = phones.getString(phones.getColumnIndex(ContactsContract.
                                 CommonDataKinds.Phone.NUMBER));

            ContactModel contactModel = new ContactModel();
            contactModel.setName(name);
            contactModel.setNumber(phoneNumber);
            contactModelArrayList.add(contactModel);

 Log.d("name>>",name+"  "+phoneNumber);
        }
        phones.close();

        customAdapter = new CustomAdapter(this,contactModelArrayList);
        listView.setAdapter(customAdapter);
    }

}
```

EXPERIMENT 15

Design an android application for menu.

AIM: To design an application options menu.

MainActivity.java

```
package com.javatpoint.optionmenu;import android.os.Bundle;
import android.app.Activity; import android.view.Menu; import
android.view.MenuItem;import android.widget.Toast;
public class MainActivity extends Activity
{
@Override
protected void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.main, menu);//Menu Resource, Menureturn true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
switch (item.getItemId())
{
case R.id.item1:
Toast.makeText(getApplicationContext(),"Item 1
Selected",Toast.LENGTH_LONG).show();return true;
```

```
  case R.id.item2:
Toast.makeText(getApplicationContext(),"Item 2
  Selected",Toast.LENGTH_LONG).show();
  return true; case R.id.item3:
  Toast.makeText(getApplicationContext(),"Item 3
  Selected",Toast.LENGTH_LONG).show();return true;
  default:
  return super.onOptionsItemSelected(item);
  }
  }
  }
```

MainActivity.xml

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

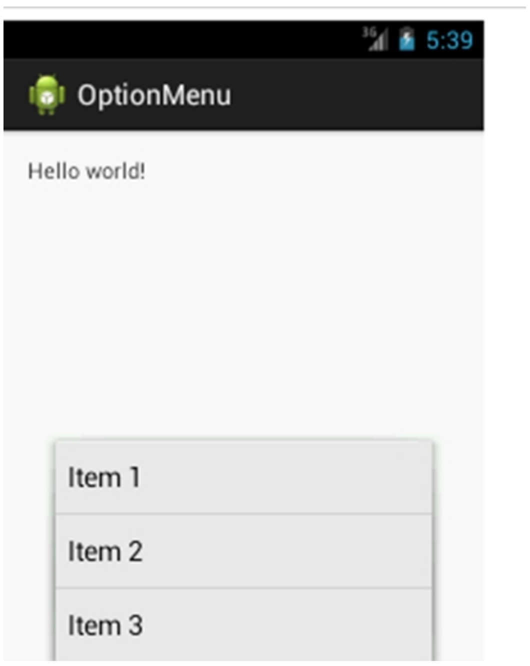
SecondActivity.xml

```
<menu xmlns:androclass="http://schemas.android.com/apk/res/android" >
<item  android:id="@+id/item1"android:title="Item 1"/>
<item  android:id="@+id/item2"android:title="Item 2"/>
```

	Exp.No. **Date:**

```
<item  android:id="@+id/item3"android:title="Item 3"/>
</menu>
```

OUTPUT:



	Exp.No. Date:

EXPERIMENT 16

Create a user registration application that stores the user details in a database table.

DbHandler.java

```
package com.example.sqliteexample; import
android.content.ContentValues;import android.content.Context; 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_TAB
LE);
}
```

```
@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 againonCreate(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.valueOf(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 listview from SQLite Database for that right click on your layout folder à Go to New à select LayoutResource 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 belowdetails.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:layout_height="wrap_content"
android:layout_gravity="center" android:layout_marginTop="20dp" android:text="Back"
/>
</LinearLayout>
```

Create an another layout file (list_row.xml) in /res/layout folder to show the data in listview, for that rightclick on layout folder à add new Layout resource file à Give name as list_row.xml and write the code likeas 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"
```

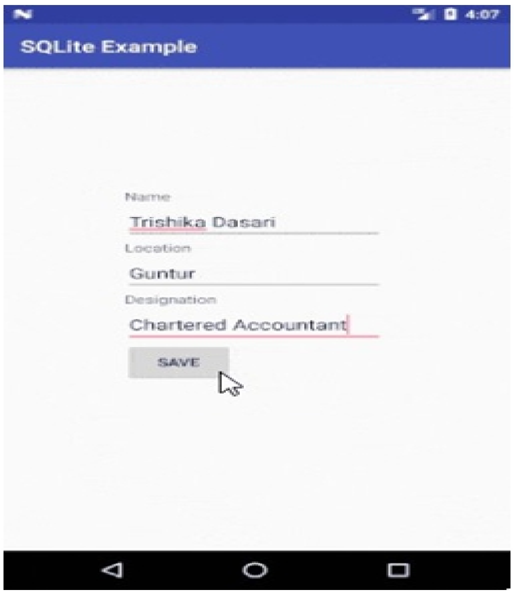
```
android:textStyle="bold" android:textSize="17dp" />
<TextView android:id="@+id/designation"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/name" android:layout_marginTop="7dp"
android:textColor="#343434" android:textSize="14dp" />
<TextView android:id="@+id/location"
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.tutlane.sqliteexample path and writethe 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(),"DetailsInserted
Successfully",Toast.LENGTH_SHORT).show();
}});}
}
```

OUTPUT:



	Exp.No. **Date:**