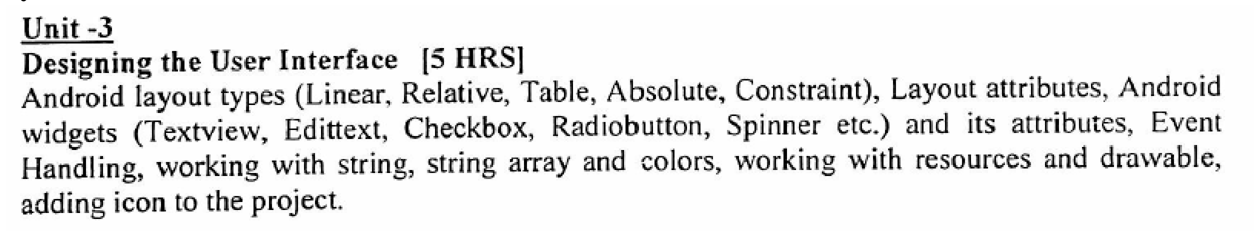
Unit-3: Designing the User Interface (UI)

Syllabus:



Unit -3

Designing the User Interface [5 HRS]

Android layout types (Linear, Relative, Table, Absolute, Constraint), Layout attributes, Android widgets (Textview, Edittext, Checkbox, Radiobutton, Spinner etc.) and its attributes, Event Handling, working with string, string array and colors, working with resources and drawable, adding icon to the project.

Types of Android Layout:

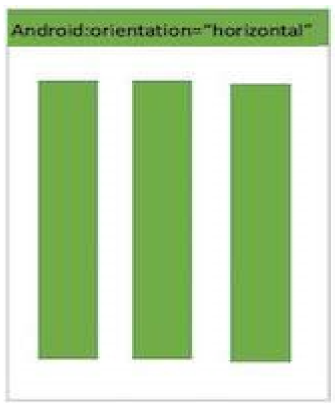
A layout defines the structure for a user interface (UI) in an Android app, such as the UI's dimensions, the placement of its components, and how those components interact with each other. Hence, it is a hierarchy of UI elements that specifies the spatial arrangement of those elements on the screen. Layouts are typically defined in XML files, but they can also be created programmatically in Java or Kotlin.

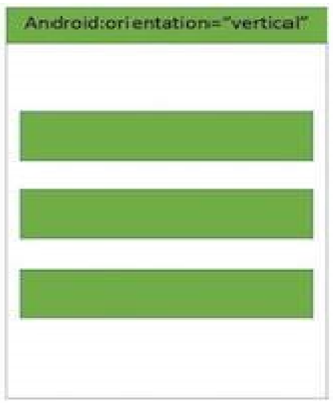
Some of the major layouts are:

1. Linear Layout
2. Relative Layout
3. Table Layout
4. Absolute Layout
5. Constraint Layout

Linear Layout:

Android LinearLayout is a view group that aligns all children in either vertically or horizontally. You can specify the layout direction with the android:orientation attribute.





Horizontal Layout Vertical Layout

All children of a LinearLayout are stocked one after another.

- In horizontal list orientation there will only be one row (Single Row) and children are aligned in multiple columns.
- In Vertical list orientation will only have one column (single row), no matter how wide they are and children are aligned in multiple rows.

LinearLayout respects the margin between children and the gravity (right, center, and left) of each child.

LinearLayout Attributes:

Following are the important attributes specific to LinearLayout:

SN	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the layout.
2	**android:baselineAligned** This must be a boolean value, either "true" or "false" and prevents the layout from aligning its children's baselines.
3	**android:baselineAlignedChildIndex** When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align.
4	**android:divider** This is drawable to use as a vertical divider between buttons. You use a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
5	**android:gravity** This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
6	**android:orientation** This specifies the direction of arrangement and you will use "horizontal" for a row, "vertical" for a column. The default is horizontal.
7	**android:weightSum** Sum up child weight. Weight is assigned to individual child widgets. Weight is the value which denotes how much space it should occupy on the screen. Large weight value allows it to expand to fill any remaining space in the parent view.

Note: (here we are doing all the demonstration in Activity_main.xml and MainActivity.java file as well as string.xml as string resource file)

Following are the required steps to create a Layout and Run in Virtual or Real Android Device.

Step 1: Create an android project with suitable name (here the project name is Android Practice)
Step 2: Create an LinearLayout in Activity_main.xml file with required attributes
Step 3: Add required widgets like button, Textview, Label etc.
Step 4: Create a strings.xml file to write string and call in android:text attribute of any widget with string name.
Step 5: Call the Activity_main.xml file in MainActivity.java file (In general it is take care by android studio itself)
Step 6: Define Activity in AndroidManifest.xml file (In general it is take care by android studio itself)
Step 7: Run the application in your Virtual Device or your Android Phone
Note: these steps are applied for all the components of android application

Example: Example to demonstrate the LinearLayout
<u>Activity_main.xml</u>

```
<?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:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/txt1"
   android:layout_gravity="center"
   android:layout_marginTop="100dp"/>

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/btnMain"
   android:layout_gravity="center"/>

<LinearLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="center">

   <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn1"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn2"/>
</LinearLayout>
</LinearLayout>
```

Strings.xml

```
<resources>
    <string name="app_name">practice2</string>
    <string name="txt1">This is outer LinearLayout</string>
    <string name="txt2">This is nested LinearLayout</string>
    <string name="btnMain">Button View</string>
    <string name="btn1">True</string>
    <string name="btn2">False</string>
</resources>
```

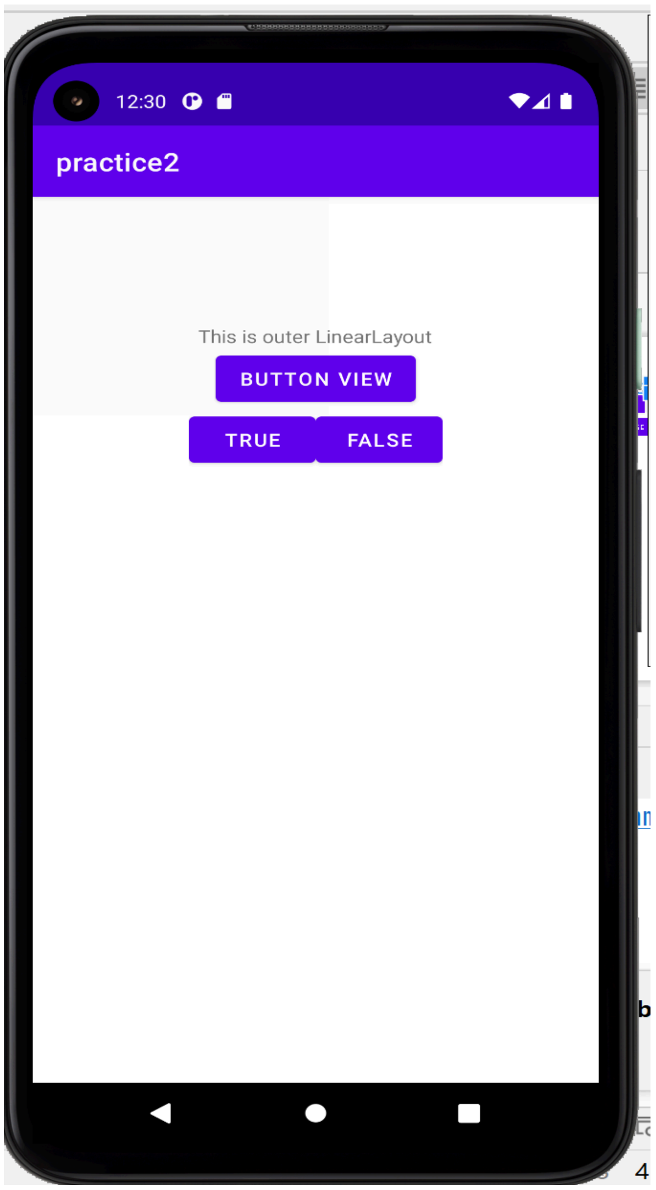
MainActivity.java

```
package com.example.androidpractice;
import androidx.appcompat.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);
    }
}
```

AndroidManifest.xml

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

    <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/Theme.AndroidPractice">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Output:



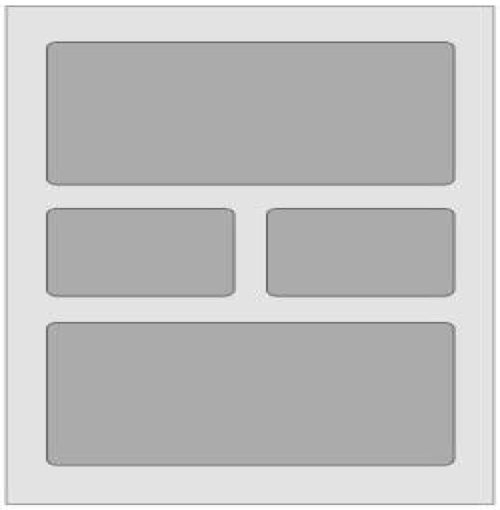
This is outer LinearLayout

BUTTON VIEW

TRUE FALSE

Relative Layout:

Android RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.



Using **RelativeLayout,** you can align two elements by right border, or make one below another, centered in the screen, centered left, and so on. By default, all child views are drawn at the top-left of the layout, so you must define the position of each view using the various layout properties available from **RelativeLayout.LayoutParams** and few of the important attributes are given below

android:layout_alignParentBottom
If true, make the bottom edge of this view match the bottom edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentEnd
If true, make the end edge of this view match the end edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentLeft
If true, make the left edge of this view match the left edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentRight
If true, make the right edge of this view match the right edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentStart
If true, make the start edge of this view match the start edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentTop
If true, make the top edge of this view match the top edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_centerHorizontal
If true, centers this child horizontally within its parent. Must be a boolean value, either "true" or "false".

android:layout_centerVertical
If true, centers this child vertically within its parent. Must be a boolean value, either "true" or "false".
android:layout_centerInParent
If true, centers this child horizontally and vertically within its parent. Must be a boolean value, either "true" or "false".

android:layout_toLeftOf
Positions the right edge of this view to the left of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_toRightOf
Positions the left edge of this view to the right of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_toStartOf
Positions the end edge of this view to the start of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_toEndOf
Positions the start edge of this view to the end of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_alignTop
Make the top edge of this view match the top edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_alignLeft
Make the left edge of this view match the left edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_alignRight
Make the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_alignBottom

Make the bottom edge of this view match the bottom edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_alignStart
Make the start edge of this view match the start edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_above
Positions the bottom edge of this view above the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name"

android:layout_below
Positions the top edge of this view below the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

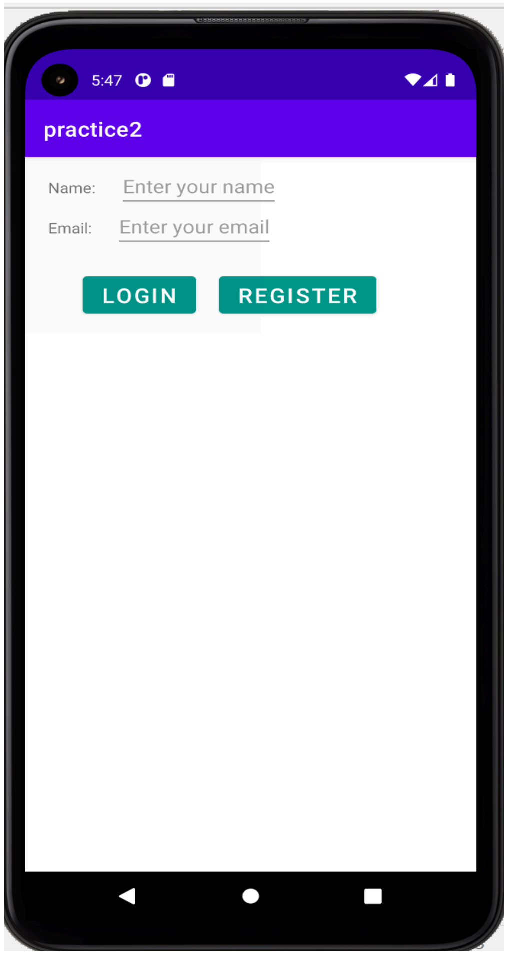
Example: Program to demonstrate the RelativeLayout

Example_1:
Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name:"
        android:layout_marginTop="20dp"
        android:layout_marginStart="20dp" />

    <EditText
```

```
        android:id="@+id/et1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:hint="Enter your name"
        android:layout_toEndOf="@id/tv1"
        android:layout_alignBaseline="@id/tv1" />

    <TextView
        android:id="@+id/tv2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email:"
        android:layout_below="@id/tv1"
        android:layout_marginTop="20dp"
        android:layout_marginStart="20dp" />

    <EditText
        android:id="@+id/et2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:hint="Enter your email"
        android:layout_toEndOf="@id/tv2"
        android:layout_alignBaseline="@id/tv2" />

    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="login"
        android:textSize="20sp"
        android:backgroundTint="#009688"
        android:layout_centerVertical="true"
        android:layout_marginStart="50dp"
        android:layout_below="@id/et2"
        android:layout_alignParentStart="true"
        android:layout_marginTop="20dp" />

    <Button
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Register"
        android:textSize="20sp"
        android:backgroundTint="#009688"
        android:layout_toEndOf="@id/btn1"
        android:layout_alignTop="@id/btn1"
        android:layout_marginStart="20dp" />

</RelativeLayout>
```

OutPut:



Example-2:

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">

   <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Enter your name here"
        android:layout_marginTop="30dp"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:layout_marginTop="80dp"
        android:layout_marginLeft="150dp"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="180dp">
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="True"
            android:layout_marginLeft="30dp"/>
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="False"
            android:layout_marginLeft="200dp"/>
    </LinearLayout>

</RelativeLayout>
```

Output:

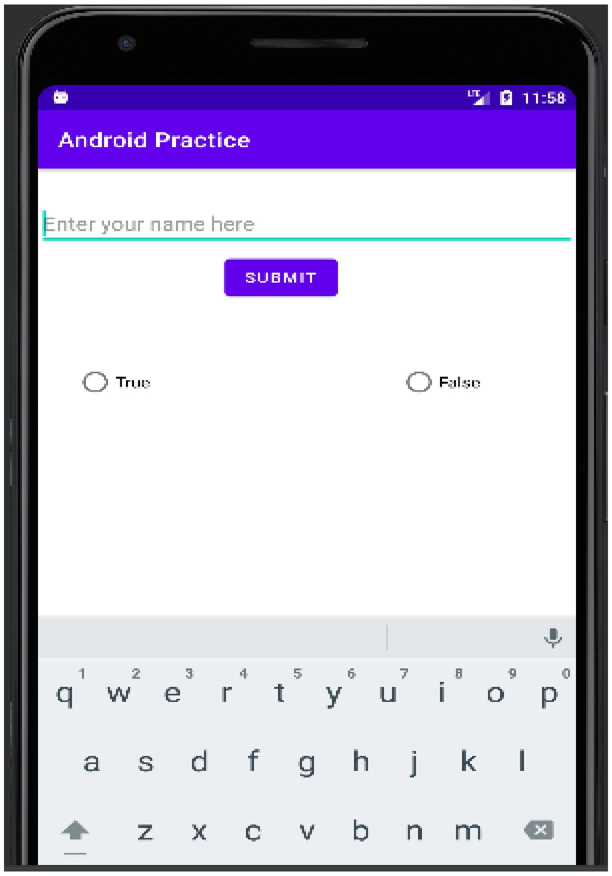

11:58
Android Practice
Enter your name here
SUBMIT
True
False

Table Layout:

Android TableLayout is going to arrange groups of views into rows and columns. We can use the <TableRow> element to build a row in the table. Each row has zero or more cells; each cell can hold one View object.

TableLayout containers do not display border lines for their rows, columns, or cells.

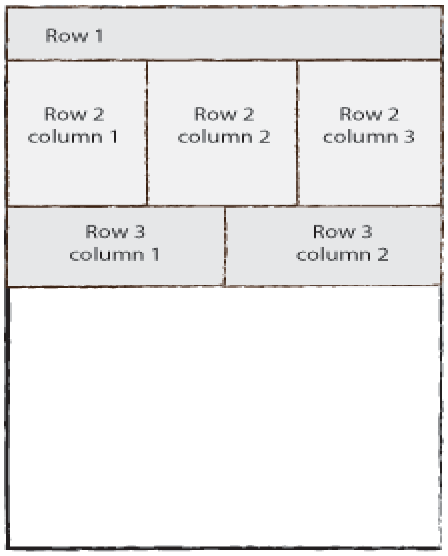


Table Layout

TableLayout Attributes
Following are the important attributes specific to TableLayout:

SN	Attribute & Description
1	android:id This is the ID which uniquely identifies the layout.
2	android:collapseColumns This specifies the zero-based index of the columns to collapse. The column indices must be separated by a comma: 1, 2, 5.
3	android:shrinkColumns The zero-based index of the columns to shrink. The column indices must be separated by a comma: 1, 2, 5.
4	android:stretchColumns

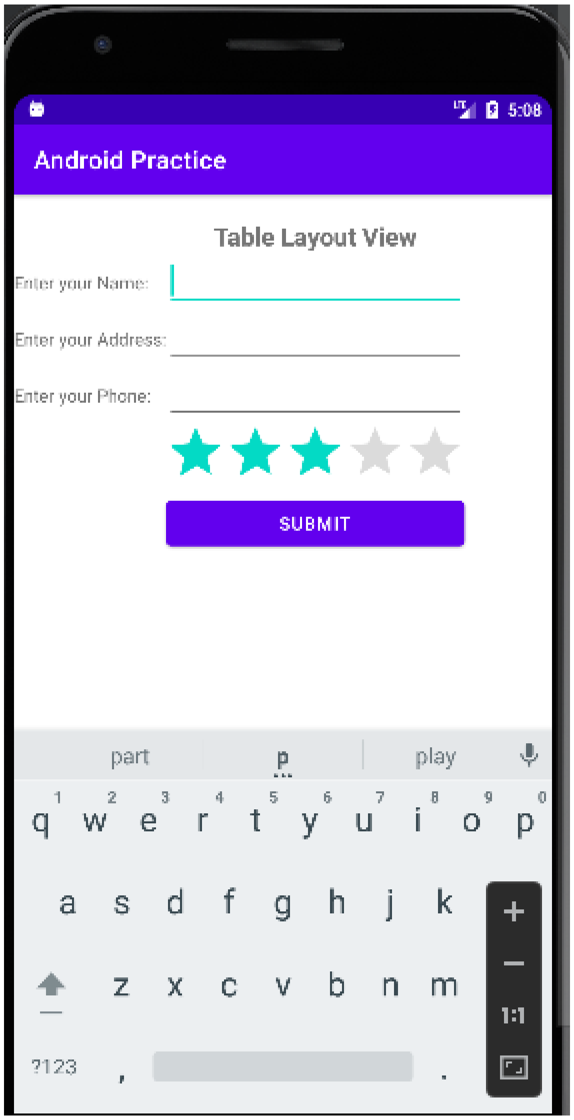
	The zero-based index of the columns to stretch. The column indices must be separated by a comma: 1, 2, 5.

Example:

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

   <TableRow
      android:layout_height="wrap_content"
      android:layout_width="match_parent">

      <TextView
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:text="Table Layout View"
         android:layout_column="2"
         android:gravity="center"
         android:textStyle="bold"
         android:textSize="20dp"
         android:layout_marginTop="20dp"/>
   </TableRow>


   <TableRow
      android:layout_width="match_parent"
      android:layout_height="wrap_content">

      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_column="1"
         android:text="Enter your Name:" />

      <EditText
         android:layout_width="80dp"
         android:layout_height="wrap_content"
         android:layout_column="2" />
   </TableRow>
   <TableRow
      android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:text="Enter your Address:" />

        <EditText
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:layout_column="2" />
    </TableRow>
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:text="Enter your Phone:" />

        <EditText
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:layout_column="2" />
    </TableRow>
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <RatingBar android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:layout_column="2"/>
    </TableRow>
    <TableRow android:layout_height="wrap_content"
        android:layout_width="match_parent">
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Submit"
            android:layout_column="2"/>
    </TableRow>

</TableLayout>
```

Output:



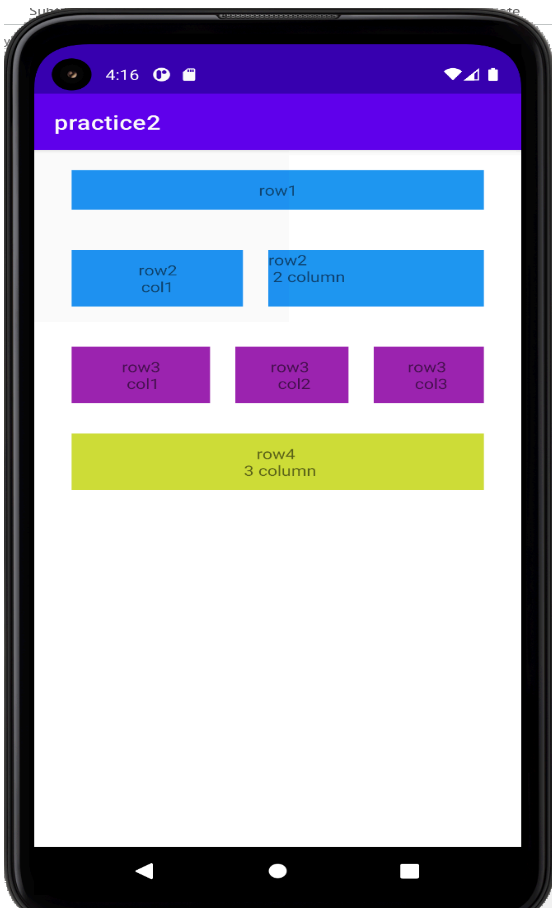
Example-2:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    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"
    android:paddingRight="20dp"
    android:paddingLeft="20dp"
```

```
tools:context=".MainActivity">

<TableRow
    android:layout_marginTop="10dp"
    >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="row1"
        android:textAlignment="center"
        android:padding="10dp"
        android:layout_margin="10dp"
        android:background="#2196F3"

        />
</TableRow>

<TableRow
    android:layout_marginTop="20dp"
    >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="row2\ncol1"
        android:textAlignment="center"
        android:padding="10dp"
        android:layout_margin="10dp"
        android:background="#2196F3"

        />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="2"
        android:text="row2 \n 2 column"
        android:layout_margin="10dp"
        android:background="#2196F3"

        />

</TableRow>

<TableRow
    android:layout_marginTop="20dp"
```

```
        >
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="row3\n col1"
            android:textAlignment="center"
            android:padding="10dp"
            android:layout_margin="10dp"
            android:background="#9C27B0"

            />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="row3 \n col2"
            android:textAlignment="center"
            android:padding="10dp"
            android:layout_margin="10dp"
            android:background="#9C27B0"

            />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="row3 \n col3"
            android:textAlignment="center"
            android:padding="10dp"
            android:layout_margin="10dp"
            android:background="#9C27B0"
            />

    </TableRow>

    <TableRow
        android:layout_marginTop="10dp"
        >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="3"
            android:text="row4 \n 3 column"
            android:textAlignment="center"
            android:padding="10dp"
            android:layout_margin="10dp"
            android:background="#CDDC39"

            />
    </TableRow>

</TableLayout>
```

Output:

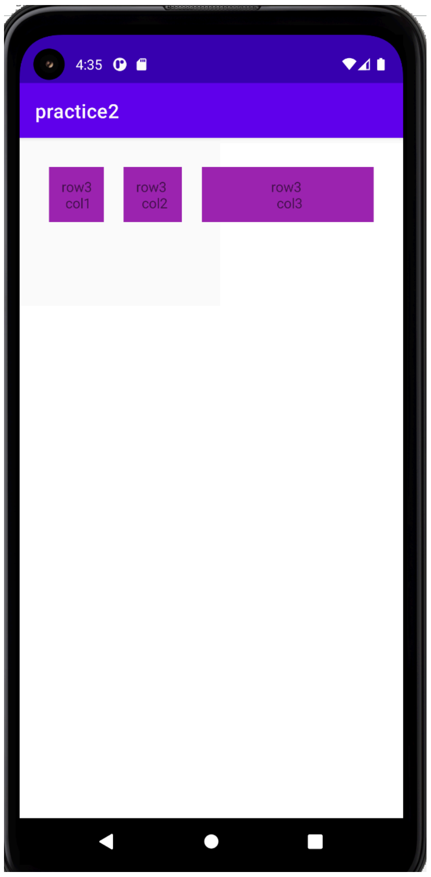


Example-3:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    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"
    android:paddingRight="20dp"
    android:paddingLeft="20dp"
    tools:context=".MainActivity"
    android:stretchColumns="2" >

    <!--
    android:stretchColumns="2"
    android:collapseColumns="2"
    -->

    <TableRow
        android:layout_marginTop="20dp"
        >
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="row3\n col1"
            android:textAlignment="center"
            android:padding="10dp"
            android:layout_margin="10dp"
            android:background="#9C27B0"

            />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="row3 \n col2"
            android:textAlignment="center"
            android:padding="10dp"
            android:layout_margin="10dp"
            android:background="#9C27B0"

            />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="row3 \n col3"
            android:textAlignment="center"
            android:padding="10dp"
            android:layout_margin="10dp"
            android:background="#9C27B0"
            />

    </TableRow>
```

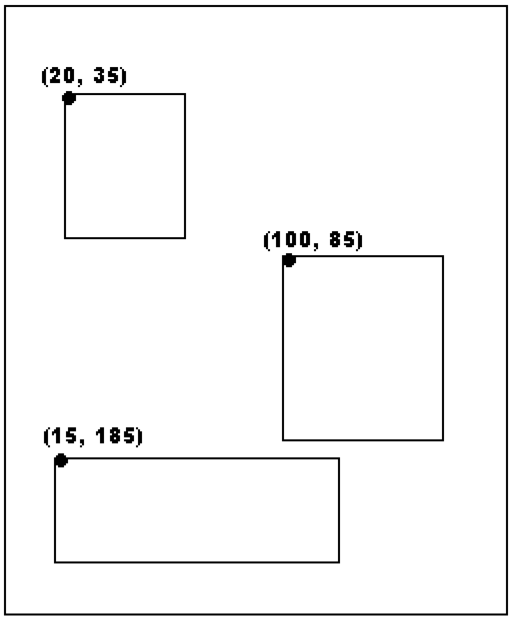
```
</TableLayout>
```

Output:



Absolute Layout:

An Absolute Layout lets you specify exact locations (x/y coordinates) of its children. Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning.



Absolute Layout

Absolute Layout Attributes
Following are the important attributes specific to AbsoluteLayout

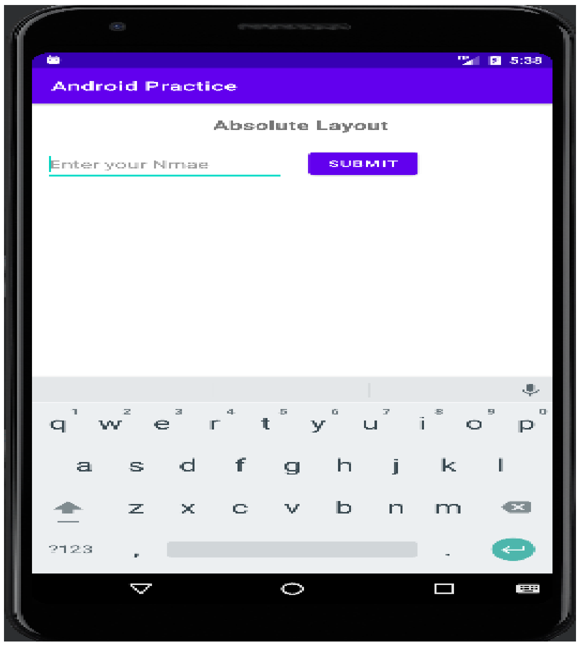
SN	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the layout.
2	**android:layout_x** This specifies the x-coordinate of the view.
3	**android:layout_y** This specifies the y-coordinate of the view.

Example:

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

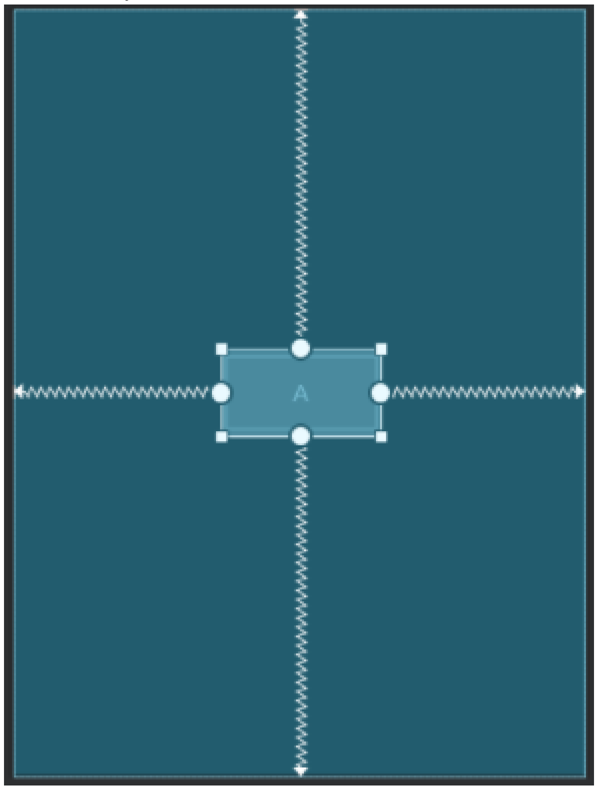
```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Absolute Layout"
        android:textStyle="bold"
        android:textSize="20dp"
        android:layout_y="20dp"
        android:layout_x="150dp"/>
    <EditText
        android:layout_width="200dp"
        android:layout_height="50dp"
        android:hint="Enter your Nmae"
        android:layout_x="10dp"
        android:layout_y="70dp"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:layout_x="230dp"
        android:layout_y="70dp"/>
</AbsoluteLayout>
```

Output:



Constraint Layout:

Constraint Layout is a ViewGroup (i.e. a view that holds other views) which allows you to create large and complex layouts with a flat view hierarchy, and also allows you to position and size widgets in a very flexible way. It was created to help reduce the nesting of views and also improve the performance of layout files.

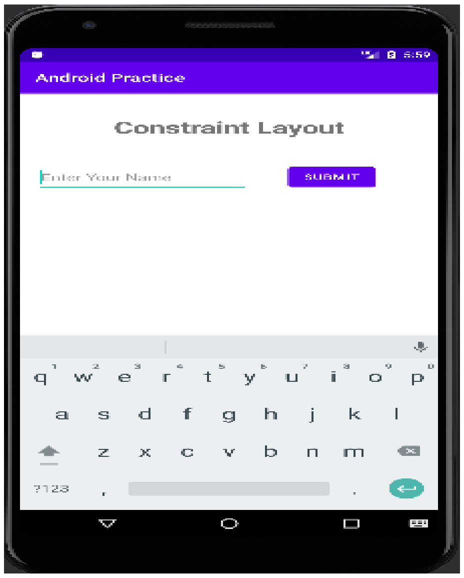


Constraint Layout

Example:

```
<?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">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Constraint Layout"
        android:textSize="30dp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.057"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="16dp" />

    <EditText
        android:layout_width="219dp"
        android:layout_height="51dp"
        android:hint="Enter Your Name"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.083"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.18" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.811"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.18" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Output:



Android Widget:

Widgets enable users to interact with an Android Studio application page. There are a lot of android widgets with simplified examples such as Button, EditText, Checkbox etc. Some of the major widgets are describe below

- TextView
- EditText
- Checkbox
- Button
- RadioButton
- Spinner
- ToggleButton
- RatingBar
- ProgressBar
- DatePicker
- TimePicker

TextView:

This is the widget used to display the text in the application.

```
<TextView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="This is text view wedget"
   android:textStyle="bold"
   android:textSize="20dp"
   android:gravity="center"/>
```

EditText:

This widget is used to input the text or data. This is a form to enter the data.

```
<EditText
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:hint="Entery your name"
   android:layout_marginLeft="10dp"/>
```

CheckBox:
Android CheckBox is a type of two state button either checked or unchecked. When the CheckBox is unchecked use can click the checkbox to check. If user clicks once again in checkbox then it becomes unchecked.

There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.

```
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Male"/>
```

Button:
Android Button is used to invoke the event or function. For example we can use a button to submit the input data from the EditText widget.

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Submit"/>
```

Radiobutton:
Android CheckBox is a type of two state button either checked or unchecked. When the radio button is unchecked, users can press or click the button to check. Once the button is checked then it can’t be unchecked by clicking once again.

```
<RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Female"/>
```

Spinner:
Android spinner is like the drop down menu with multiple values from which the end user can select only one value.
Android spinner is associated with AdapterView. So you need to use one of the adapter classes with a spinner.

```
<string-array name="spinner_item">
    <item>Item 1</item>
    <item>Item 2</item>
    <item>Item 3</item>
    <item>Item 4</item>
    <item>Item 5</item>
</string-array>

<Spinner
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_item"
    android:spinnerMode="dropdown"/>
```

ToggleButton:

Android Toggle Button can be used to display checked/unchecked (On/Off) state on the button. It is beneficial if user have to change the setting between two states. It can be used for On/Off Sound, Wifi, Bluetooth etc.
Since Android 4.0, there is another type of toggle button called switch that provides slider control. Android ToggleButton and Switch both are the subclasses of CompoundButton class.

```
<ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

RatingBar:

Android RatingBar can be used to get the rating from the user. The Rating returns a floating-point number. It may be 2.0, 3.5, 4.0 etc.
Android RatingBar displays the rating in stars. Android RatingBar is the subclass of AbsSeekBar class.
The getRating() method of android RatingBar class returns the rating number.

```
<RatingBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

DatePicker:
Android DatePicker is a widget to select dates. It allows you to select date by day, month and year. Like DatePicker, android also provides TimePicker to select time.
The android.widget.DatePicker is the subclass of FrameLayout class.

```
<DatePicker
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>
```

TimePicker:
Android TimePicker widget is used to select date. It allows you to select time by hour and minute. You cannot select time by seconds.
The android.widget.TimePicker is the subclass of FrameLayout class.

```
<TimePicker
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>
```

ProgressBar:
We can display the android progress bar dialog box to display the status of work being done e.g. downloading file, analyzing status of work etc.
In this example, we are displaying the progress dialog for dummy file download operation.
Here we are using android.app.ProgressDialog class to show the progress bar. Android ProgressDialog is the subclass of AlertDialog class.

The ProgressDialog class provides methods to work on progress bar like setProgress(), setMessage(), setProgressStyle(), setMax(), show() etc. The progress range of Progress Dialog is 0 to 10000.

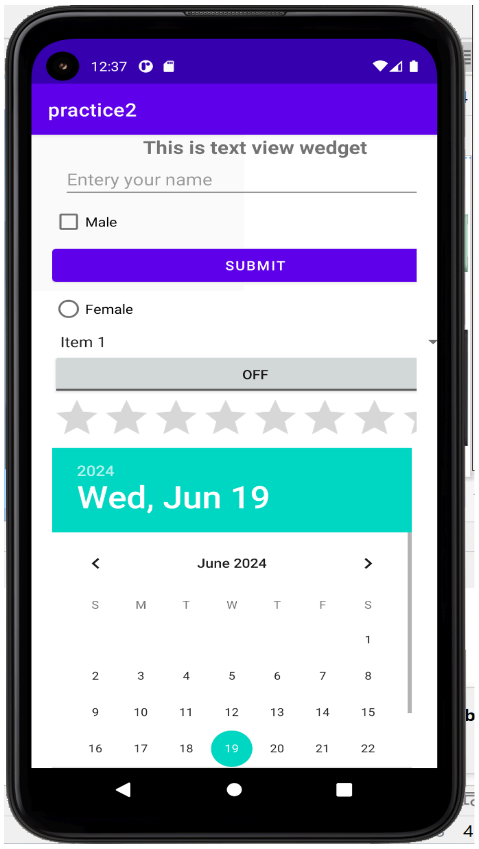
```
<ProgressBar
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>
```

Program:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
  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"
  android:paddingRight="20dp"
  android:paddingLeft="20dp"
  tools:context=".MainActivity">
  <TableRow>
    <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="This is text view wedget"
      android:textStyle="bold"
      android:textSize="20dp"
      android:gravity="center"/>

  </TableRow>
  <TableRow>
    <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Entery your name"
      android:layout_marginLeft="10dp"/>
  </TableRow>

  <TableRow>
    <CheckBox
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Male"/>

  </TableRow>

  <TableRow>

    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Submit"/>
  </TableRow>
```

```
    <TableRow>
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female"/>
    </TableRow>


    <TableRow>
        <Spinner
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:entries="@array/spinner_item"
            android:spinnerMode="dropdown"/>
    </TableRow>

    <TableRow>
        <ToggleButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </TableRow>


    <TableRow>
        <RatingBar
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </TableRow>

    <TableRow>
        <DatePicker
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </TableRow>


    <TableRow>
        <TimePicker
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </TableRow>

    <TableRow>
        <ProgressBar
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </TableRow>

</TableLayout>
```

Output:


12:37
practice2
This is text view wedget
Entery your name
Male
SUBMIT
Female
Item 1
OFF
2024
Wed, Jun 19
June 2024
S M T W T F S
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22

Event Handling:

Events are some sort of activities that a user performs to interact with the application. A simple event can be clicking a button or touching the screen. The Android framework maintains an event queue as a first-in, first-out (FIFO) basis. You can capture these events in your program and take appropriate action as per requirements.

Android Event Management consists of the following concepts:

- **Event handlers:** These are responsible for handling events and performing specific actions based on their events.
- **Event Listeners:** An event listener works as an interface in the View class. These methods are called by the Android framework when the View, with which an event listener is linked, is called by the user.
- **Event Listeners registration:** It is a process in which an Event handler is registered with an Event listener. This way the event handler is called when the Event listener is triggered.

Event Listeners & Event Handlers

Event Handler	Event Listener & Description
onClick()	**OnClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such an event.
onLongClick()	OnLongClickListener() This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use the onLongClick() event handler to handle such an event.
onFocusChange()	OnFocusChangeListener() This is called when the widget loses its focus i.e. the user goes away from the view item. You will use onFocusChange() event handler to handle such event.
onKey()	OnFocusChangeListener() This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such an event.

onTouch()	OnTouchListener() This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such an event.
onMenuItemClick()	OnMenuItemClickListener() This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such an event.
onCreateContextMenu()	onCreateContextMenuItemListener() This is called when the context menu is being built(as the result of a sustained "long click)

There are many more event listeners available as a part of View class like OnHoverListener(), OnDragListener() etc which may be needed for your application.

Touch Mode:

Users can interact with their devices by using hardware keys or buttons or touching the screen. Touching the screen puts the device into touch mode. The user can then interact with it by touching the on-screen virtual buttons, images, etc. You can check if the device is in touch mode by calling the View class's isInTouchMode() method.

Focus:

A view or widget is usually highlighted or displays a flashing cursor when it's in focus. This indicates that it's ready to accept input from the user.

- **isFocusable():** it returns true or false
- **isFocusableInTouchMode():** checks to see if the view is focusable in touch mode. (A view may be focusable when using a hardware key but not when the device is in touch mode)

Program: Onclick() event handling

Activity_mail.xml

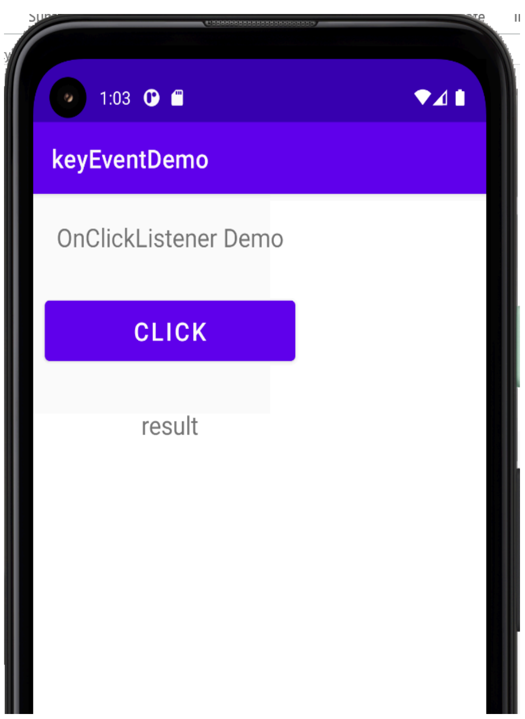
```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

<TableRow>
    <TextView
```

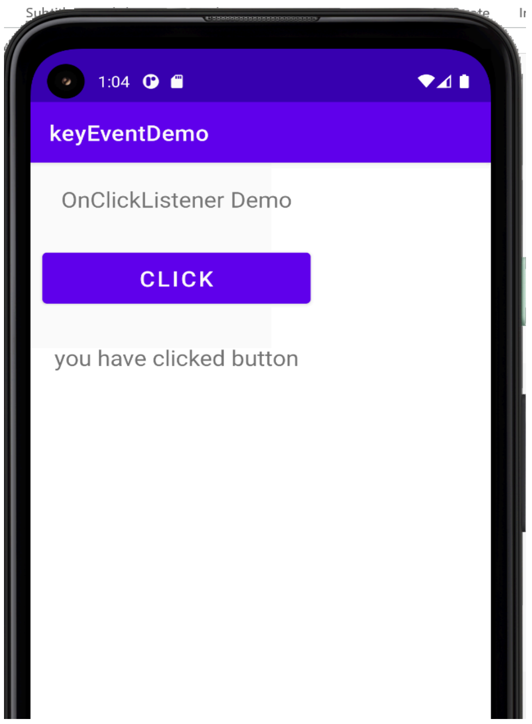
```
        android:id="@+id/tv1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="OnClickListener Demo"
        android:textSize="20sp"
        android:padding="10dp"
        android:gravity="center"
        android:layout_margin="10dp"
        android:layout_column="2"

        />
</TableRow>
    <TableRow>
        <Button
            android:id="@+id/btn1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="click"
            android:textSize="20sp"
            android:padding="10dp"
            android:gravity="center"
            android:layout_margin="10dp"
            android:layout_column="2"
            >

        </Button>

    </TableRow>
    <TableRow>
        <TextView
            android:id="@+id/tv2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="result"
            android:textSize="20sp"
            android:padding="10dp"
            android:gravity="center"
            android:layout_margin="10dp"
            android:layout_column="2"

            />

    </TableRow>

</TableLayout>
```

MainActivity.java

```
package com.example.keyeventdemo;

import androidx.appcompat.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    Button b;
    TextView t;

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

        b=(Button)findViewById(R.id.btn1);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                t=(TextView) findViewById(R.id.tv2);
                t.setText("you have clicked button");
            }
        });
    }
}
```

Output:





Activity_mail.xml

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

<TableRow>
    <TextView
        android:id="@+id/tv1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="OnLongClickListener Demo"
        android:textSize="20sp"
        android:padding="10dp"
        android:gravity="center"
        android:layout_margin="10dp"
        android:layout_column="2"
```

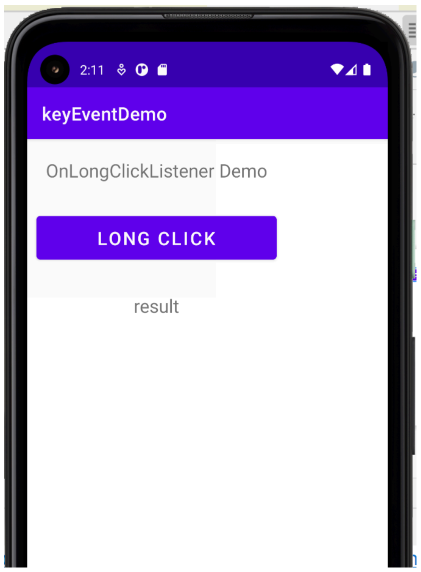
```xml
        />
</TableRow>
    <TableRow>
        <Button
            android:id="@+id/btn1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Long Click"
            android:textSize="20sp"
            android:padding="10dp"
            android:gravity="center"
            android:layout_margin="10dp"
            android:layout_column="2"
            >

        </Button>

    </TableRow>
    <TableRow>
        <TextView
            android:id="@+id/tv2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="result"
            android:textSize="20sp"
            android:padding="10dp"
            android:gravity="center"
            android:layout_margin="10dp"
            android:layout_column="2"

            />

    </TableRow>

</TableLayout>
```

MainActivity.java

```java
package com.example.keyeventdemo;

import androidx.appcompat.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import android.os.Bundle;
```

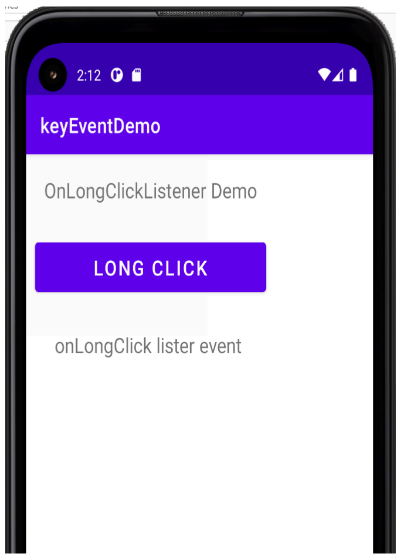
```
public class MainActivity extends AppCompatActivity {
    Button b;
    TextView t;

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

        b=(Button)findViewById(R.id.btn1);
        b.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {
                t=(TextView) findViewById(R.id.tv2);
                t.setText("onLongClick lister event ");
                return false;
            }
        });
    }
}
```

Output:



OnLongClickListener Demo

result



OnLongClickListener Demo

onLongClick lister event

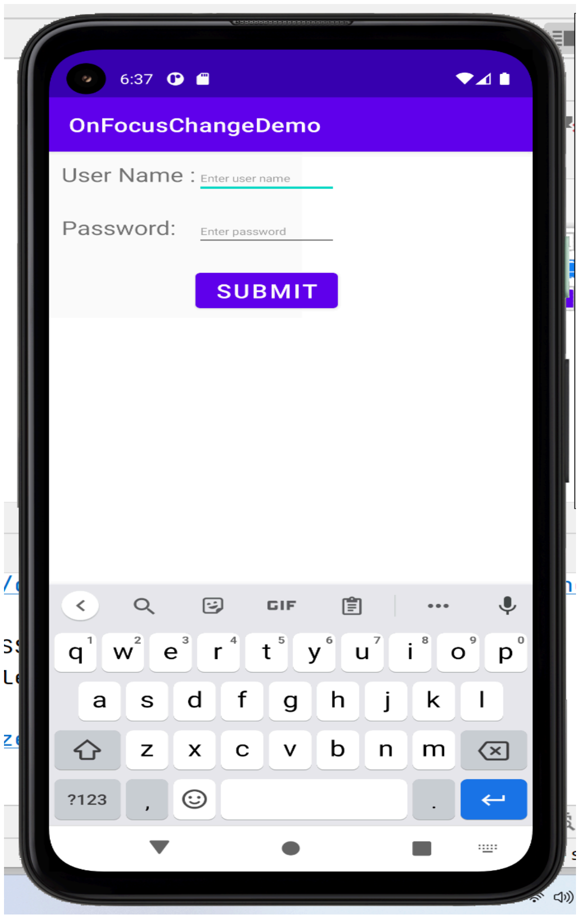
Program: OnFocusChange() event handler

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

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="User Name :"
            android:textSize="20sp"
            android:layout_column="1"
            />
        <EditText
            android:id="@+id/e1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:hint="Enter user name "
            android:textSize="10sp"
            android:layout_column="2"


            />

    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Password:"
            android:textSize="20sp"
            android:layout_column="1"
            />
        <EditText
            android:id="@+id/e2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:hint="Enter password"
            android:textSize="10sp"
            android:layout_column="2"


            />

    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        >

        <Button

            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="submit"
            android:textSize="20sp"
            android:layout_column="2"




            />

    </TableRow>



</TableLayout>
```

Activity_main.java

```
package com.example.onfocuschangedemo;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.EditText;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity {

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

        EditText ed1,ed2;
        ed1=(EditText) findViewById(R.id.e1);
        ed2=(EditText) findViewById(R.id.e2);

        ed1.setOnFocusChangeListener(new View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                Toast.makeText(getApplicationContext(), "Focus changed edittext1 change",
Toast.LENGTH_SHORT).show();
            }
        });

        ed2.setOnFocusChangeListener(new View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                Toast.makeText(getApplicationContext(), "Focus changed edittext1 change",
Toast.LENGTH_SHORT).show();
            }
        });


    }
}
```

Output:



Working with String:

String resource provides the text string for your application with optional text styling and formatting.
The xml file that contains string resource is store in res/values/string.xml directive

```
<resources>
    <string name="app_name">Android Practice</string>
    <string name="rtltext">This is outer LinearLayout</string>
```

```
</resources>
```

Now we can use the string written in this resource by the string name.

```
<TextView
    android:id="@+id/txt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/rltxt"/>
```

String Array:

We can make an array of string in string resource file as following

```
<resources>
  <string-array name="faculties">
    <item>Select Your Faculty</item>
    <item>BCA</item>
    <item>BBA</item>
    <item>BIM</item>
    <item>BBM</item>
  </string-array>
</resources>
```

This array of string is used in spinner widget as following

```
<Spinner
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:entries="@array/faculties"
    android:spinnerMode="dropdown"/>
```

Working with Colors:

A color resource provides a set of colors that you can use all over the application. The color resource is stored in res/values/colors.xml directives as shown bellow

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

We can call color resource in any widget in the application as follows

```
<TextView
   android:id="@+id/txt"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/rltxt"
   android:textColor="@color/black"
```

Working with Drawable:

A drawable resource is a general concept for a graphic that can be drawn to the screen and which you can retrieve with APIs such as getDrawable(int) or apply to another XML resource with attributes such as android:drawable and android:icon.
Android supports bitmap files in the following formats:

- .png (preferred)
- .webp (preferred, requires API level 17 or higher)
- .jpg (acceptable)
- .gif (discouraged)

Image files are saved in drawable resource have in res/drawable/image_name.png
Image can call in ImageView widget as following

```
<ImageView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/ic_launcher_background"/>
```

Adding icon to the project:

With an image saved at res/drawable/image_name.png, we can add this image as icon to the project in android manifest file as following

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

End of Unit - 3