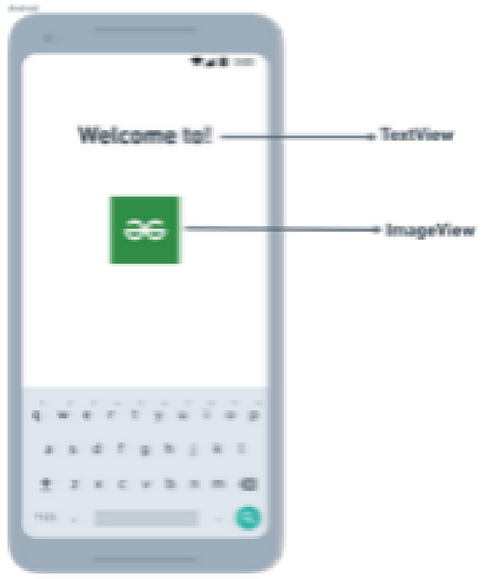
Practical No. 4
Programs related to different Layouts

Coordinate, Linear, Relative, Table, Absolute, Frame, List View, Grid View.

Android **Layout** is used to define the user interface that holds the UI controls or widgets that will appear on the screen of an android application or activity screen. Generally, every application is a combination of View and ViewGroup. As we know, an android application contains a large number of activities and we can say each activity is one page of the application. So, each activity contains multiple user interface components and those components are the instances of the View and ViewGroup. All the elements in a layout are built using a hierarchy of **View** and **ViewGroup** objects.

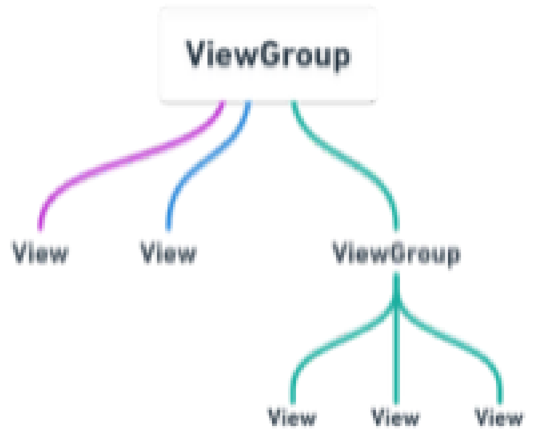
View

A **View** is defined as the user interface which is used to create interactive UI components such as TextView, ImageView, EditText, RadioButton, etc., and is responsible for event handling and drawing. They are Generally Called Widgets.



View

A **ViewGroup** acts as a base class for layouts and layouts parameters that hold other Views or ViewGroups and to define the layout properties. They are Generally Called layouts.



ViewGroup

The Android framework will allow us to use UI elements or widgets in two ways:

- Use UI elements in the XML file
- Create elements in the Kotlin file dynamically

Types of Android Layout

- **Android Linear Layout:** LinearLayout is a ViewGroup subclass, used to provide child View elements one by one either in a particular direction either horizontally or vertically based on the orientation property.
- **Android Relative Layout:** RelativeLayout is a ViewGroup subclass, used to specify the position of child View elements relative to each other like (A to the right of B) or relative to the parent (fixed to the top of the parent).
- **Android Constraint Layout:** ConstraintLayout is a ViewGroup subclass, used to specify the position of layout constraints for every child View relative to other views present. A ConstraintLayout is similar to a RelativeLayout, but having more power.
- **Android Frame Layout:** FrameLayout is a ViewGroup subclass, used to specify the position of View elements it contains on the top of each other to display only a single View inside the FrameLayout.
- **Android Table Layout:** TableLayout is a ViewGroup subclass, used to display the child View elements in rows and columns.
- **Android Web View:** WebView is a browser that is used to display the web pages in our activity layout.
- **Android ListView:** ListView is a ViewGroup, used to display scrollable lists of items in a single column.
- **Android Grid View:** GridView is a ViewGroup that is used to display a scrollable list of items in a grid view of rows and columns.

1. Linear Layout

Android **LinearLayout** is a ViewGroup subclass, used to provide child View elements one by one either in a particular direction either horizontally or vertically based on the orientation property. We can specify the linear layout orientation using **android:orientation** attribute. All the child elements are arranged one by one in multiple rows and multiple columns.

1. **Horizontal list:** One row, multiple columns.
2. **Vertical list:** One column, multiple rows.

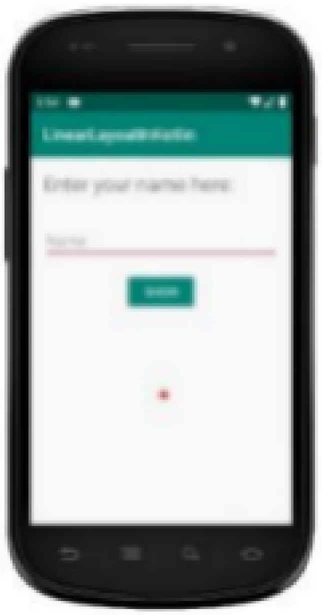
activity_main.xml

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

        <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="16dp"
                android:text="Enter your name here:"
                android:textSize="24dp"
                android:id="@+id/txtVw"/>

        <EditText
                android:id="@+id/editText"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="16dp"
                android:hint="Name"
                android:inputType="text"/>

        <Button
                android:id="@+id/showInput"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:text="show"
                android:backgroundTint="@color/colorPrimary"
                android:textColor="@android:color/white"/>
</LinearLayout>
```

MainActivity.kt

```
package com.example.linlayout

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
 import android.widget.TextView

        class MainActivity : AppCompatActivity() {

        override fun onCreate(savedInstanceState: Bundle?)
        {
                super.onCreate(savedInstanceState)
                        setContentView(R.layout.activity_main)
                        // finding the UI elements
                        val showButton
                        = findViewById<Button>(R.id.showInput)
                                val editText
                        = findViewById<EditText>(R.id.editText)
                                val textView
                        = findViewById<TextView>(R.id.txtVw)
        }
}
```

Output



2. Relative Layout

Android **RelativeLayout** is a ViewGroup subclass, used to specify the position of child View elements relative to each other like (A to the right of B) or relative to the parent (fix to the top of the parent).

Instead of using LinearLayout, we have to use RelativeLayout to design the user interface and keep our hierarchy flat because it improves the performance of the application.

XML attributes Description

It is set “true” to match the left edge of view to the left edge

layout_alignParentLeft

accepts another sibling view id and align

layout_alignParentRight

layout_alignEnd

layout_alignParentTop

layout_centerInParent

layout_alignParentBottom XML attributes Description

It accepts another sibling view id and align the view to the view to the right of the specified view id.

of parent.

It accepts another sibling view id and align the view to start of the specified view id.

It is set “true” to match the right edge of view to the right edge of parent.

layout_toRightOf

It is set “true” to match the top edge of view to the top edge of parent.

layout_toStartOf

It is set “true” to match the bottom edge of view to the bottom edge of parent.

layout_alignLeft

layout_toEndOf

layout_alignRight layout_alignStart It

layout_above

left of the specified view id

It accepts another sibling view id and align

the XML attributes Description

the view to end of specified view id.

When it is set “true”, the view will be aligned to the center of parent.

When it is set “true”, the view will be horizontally centre aligned within its parent.

When it is set “true”, the view will be vertically

layout_centerHorizontal

layout_centerVertical layout_toLeftOf

It accepts another sibling view id and places the

centre aligned within its parent.

It accepts another sibling view id and places the view left of the specified view id.

It accepts another sibling view id and places the view right of the specified view id.

It accepts another sibling view id and places the view to start of the specified view id.

It accepts another sibling view id and places the view to end of the specified view id.

It accepts another sibling view id and places the view above the specified view id.

view

below the specified view id. layout_below

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"

        android:paddingLeft="10dp"

        android:paddingRight="10dp">


        <TextView

                android:id="@+id/textView1"

                android:layout_width="wrap_content"

                android:layout_height="wrap_content"
```

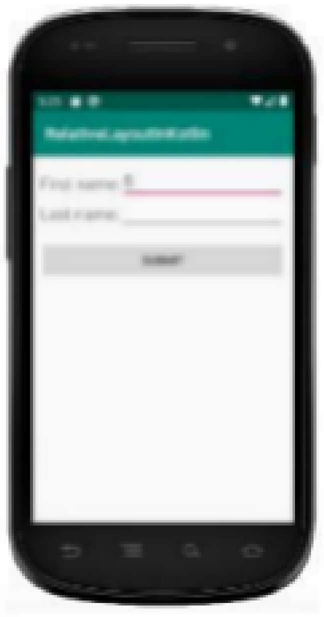
```
          android:layout_alignParentLeft="true"

          android:text="First name:"

          android:layout_marginTop="20dp"
          android:textSize="20dp"/>
<EditText

          android:id="@+id/editText1"
          android:layout_width="match_parent"

          android:layout_height="wrap_content"

          android:layout_toRightOf="@id/textView1"

          android:layout_marginTop="8dp"/>



<TextView

          android:id="@+id/textView2"

          android:layout_width="wrap_content"

          android:layout_height="wrap_content"

          android:layout_alignParentLeft="true"

          android:layout_below="@+id/textView1"

          android:layout_marginTop="10dp"

          android:text="Last name:"

          android:textSize="20dp"/>



<EditText

          android:id="@+id/editText2"

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

          android:layout_centerHorizontal="true"

          android:layout_toRightOf="@id/textView2"

          android:layout_marginTop="45dp"/>
```

```
    <Button

        android:id="@+id/btn4"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:layout_centerHorizontal="true"

        android:layout_below="@id/textView2"

        android:layout_marginTop="20dp"

        android:text="Submit" />



</RelativeLayout>
```

MainActivity.kt

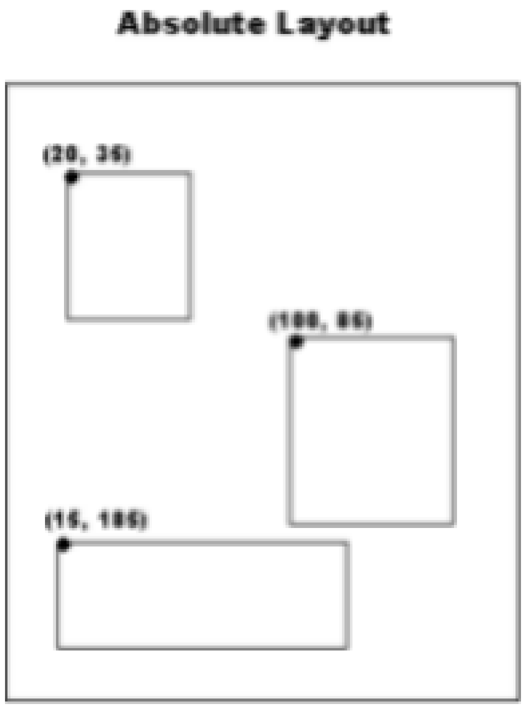
```
package com.example.relayout

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)

    }
}
```

Output



3. 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

AbsoluteLayout Attributes

Following are the important attributes specific to AbsoluteLayout –

Sr.No	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.

Public Constructors

AbsoluteLayout(Context context)
AbsoluteLayout(Context context, AttributeSet attrs)
AbsoluteLayout(Context context, AttributeSet attrs, int defStyleAttr)
AbsoluteLayout(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes)

Example

This example will take you through simple steps to show how to create your own Android application using absolute layout. Follow the following steps to modify the Android application we created in *Hello World Example* chapter −

Step	Description
1	You will use Android studio IDE to create an Android application and name it as *demo* under a package *com.example.demo* as explained in the *Hello World Example* chapter.
2	Modify the default content of *res/layout/activity_main.xml* file to include few widgets in absolute layout.
3	No need to modify string.xml, Android studio takes care of default constants
4	Run the application to launch Android emulator and verify the result of the changes done in the application.

String.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string name="app_name">demo</string> <string
name="action_settings">Settings</string>
</resources>
```

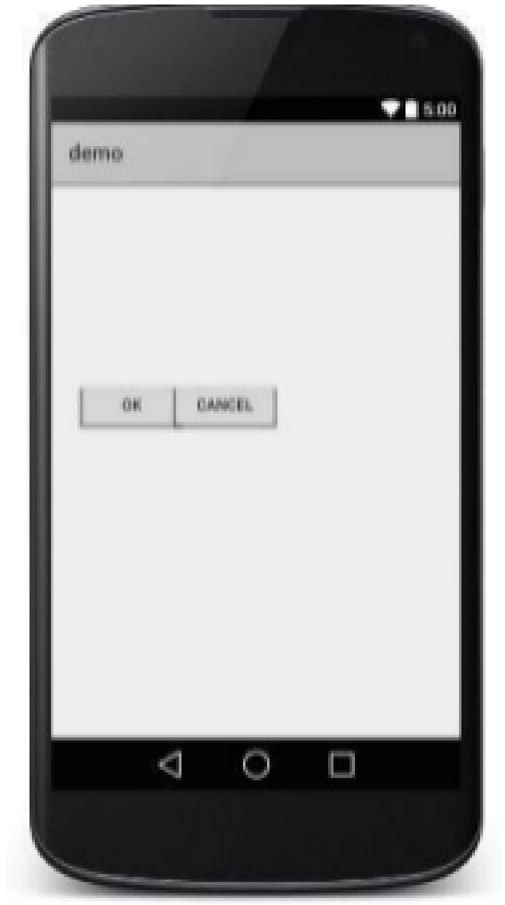
activity_main.xml

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
 android:layout_height="fill_parent">

 <Button
 android:layout_width="100dp"
 android:layout_height="wrap_content"
 android:text="OK"
 android:layout_x="50px"
 android:layout_y="361px" />
 <Button
 android:layout_width="100dp"
 android:layout_height="wrap_content"
android:text="Cancel"
 android:layout_x="225px"
 android:layout_y="361px" />

</AbsoluteLayout>
```

MainActivity.kt

```
package com.example.ablayout
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle


class MainActivity : AppCompatActivity() {
        override fun onCreate(savedInstanceState: Bundle?) {
                super.onCreate(savedInstanceState)
                setContentView(R.layout.activity_main)

                // below access the UI elements


        }
}
}
```

Output:



4.Table Layout

activity_main.xml

```
<?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"
 android:layout_marginTop="10dp"
 android:paddingLeft="5dp"
 android:paddingRight="5dp">
  <TextView
 android:id="@+id/txt"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/icc_ranking_of_players"
 android:textSize = "20sp"
 android:textStyle="bold">

 </TextView>


 <TableRow android:background="#51B435" android:padding="10dp">

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:text="@string/rank"
 android:textColor="#3E2723" />

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:text="@string/player"
 android:textColor="#3E2723" />

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:text="@string/team"
 android:textColor="#3E2723" />

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

```
android:layout_weight="1"
android:text="@string/points"
android:textColor="#3E2723" />
</TableRow>
<TableRow android:background="#F0F7F7" android:padding="5dp">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/_1" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/virat_kohli" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/nd" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/_895" />
</TableRow>
<TableRow android:background="#F0F7F7" android:padding="5dp"> <TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/_2" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/rohit_sharma" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/ind" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/_863" />
 </TableRow>
```

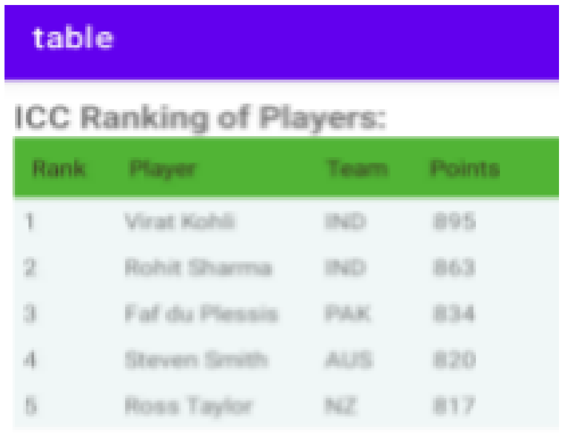
```
<TableRow android:background="#F0F7F7" android:padding="5dp">
 <TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/_3" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/faf_du_plessis" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/pak" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/_834" />

</TableRow>

<TableRow android:background="#F0F7F7" android:padding="5dp">

 <TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/_4" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/steven_smith" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="@string/aus" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
 android:text="@string/_820" />

</TableRow>

<TableRow android:background="#F0F7F7" android:padding="5dp">
```

```
<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:text="@string/_5" />
 <TextView
 android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
 android:text="@string/ross_taylor" />
<TextView
 android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
 android:text="@string/nz" />
 <TextView
 android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
 android:text="@string/_817" />

 </TableRow>

</TableLayout>
```

MainActivity.kt

```
 package com.example.table

 import androidx.appcompat.app.AppCompatActivity
 import android.os.Bundle
 class MainActivity : AppCompatActivity() {

  override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
  setContentView(R.layout.activity_main) //
finding the UI elements


  }
 }
```

Output:



table

ICC Ranking of Players:

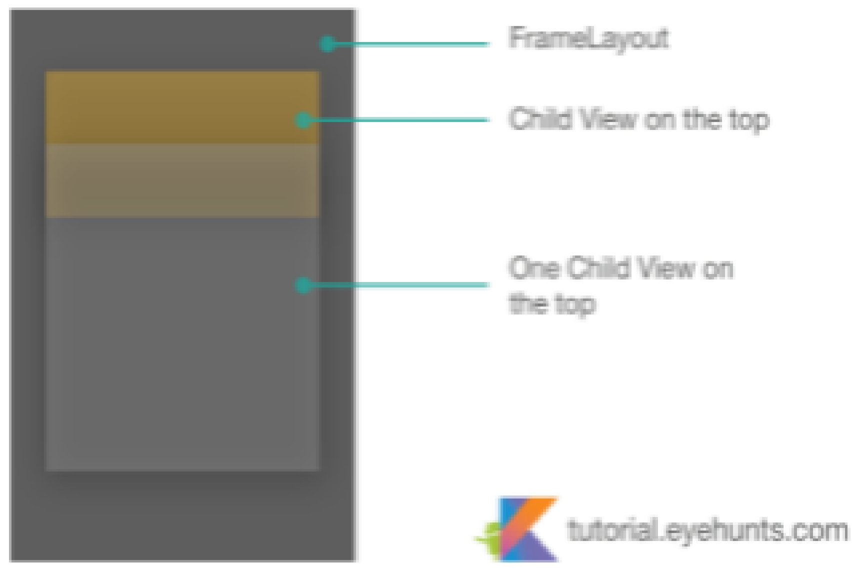
Rank	Player	Team	Points
1	Virat Kohli	IND	895
2	Rohit Sharma	IND	863
3	Faf du Plessis	PAK	834
4	Steven Smith	AUS	820
5	Ross Taylor	NZ	817

4. Frame Layout

Android FrameLayout is easy to use and very effective to control views. It shows views on top of other views. It provides a Block area on the screen to display a single item (widget). In a common scenario, FrameLayout should be used only to hold a single child view. We can also use multiple children to Android

FrameLayout but its position control by only assigning gravity (android:layout_gravityattribute) to each child.

Android



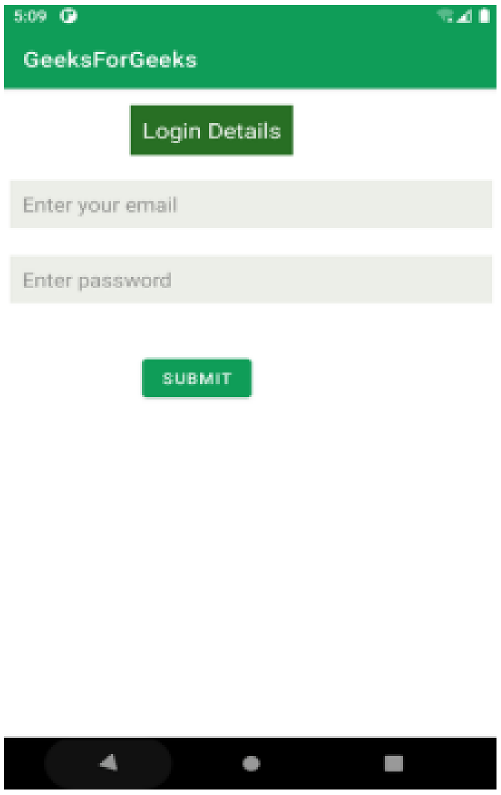
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="5dp">

        <TextView
          android:id="@+id/txtvw1"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_marginLeft="100dp"
          android:layout_marginTop="10dp"
          android:background="#286F24"
          android:padding="10dp"
          android:text="Login Details"
          android:textColor="#FFFFFF"
          android:textSize="20sp" />

        <EditText
          android:id="@+id/editText1"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:layout_centerHorizontal="true"
          android:layout_marginTop="80dp"
          android:background="#ECEEE8"
          android:hint="Enter your email"
          android:padding="10dp" />

        <EditText
          android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="150dp"
        android:background="#ECEEE8"
        android:hint="Enter password"
        android:padding="10dp" />

      <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="110dp"
        android:layout_marginTop="240dp"
        android:text="Submit" />
</FrameLayout>
```

MainActivity.kt

```
import androidx.appcompat.app.AppCompatActivity

 import android.os.Bundle
import android.widget.EditText
import android.widget.TextView

class MainActivity : AppCompatActivity() {

private lateinit var textView: TextView
private lateinit var editText1: EditText
private lateinit var editText2: EditText

override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
// finding the UI elements

textView = findViewById(R.id.txtvw1)
editText1 = findViewById(R.id.editText1);
editText2 = findViewById(R.id.editText2);
}
}
```

Output



5. List

Android ListView is a ViewGroup which is used to display the list of items in multiple rows and contains an adapter which automatically inserts the items into the list.

The main purpose of the adapter is to fetch data from an array or database and insert each item that is placed into the list for the desired result. So, it is main source to pull data from strings.xml file which contains all the required strings in Kotlin or xml files.

Android Adapter

Adapter holds the data fetched from an array and iterates through each item in data set and generates the respective views for each item of the list. So, we can say it act as an intermediate between the data sources
and adapter views such as ListView, Gridview.

Different Types of Adapter –

- **ArrayAdapter:** It always accepts an Array or List as input. We can store the list items in the strings.xml file also.
- **CursorAdapter:** It always accepts an instance of cursor as an input means ●

SimpleAdapter: It mainly accepts static data defined in the resources like array or database. ●

BaseAdapter: It is a generic implementation for all three adapter types and it can be used in the views according to our requirements.

activity_main.xml

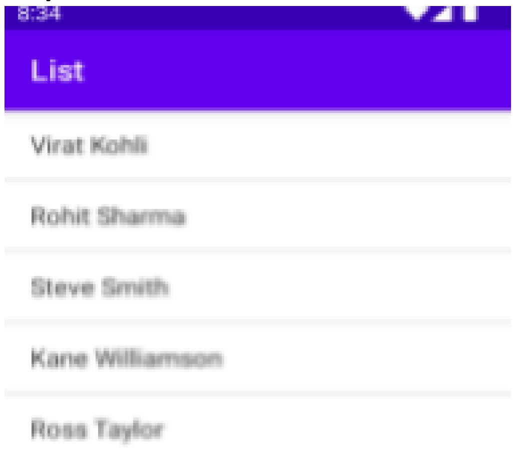
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical">
 <ListView
 android:id="@+id/userlist"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

MainActivity.kt

```
package com.example.list
import android.widget.ArrayAdapter
import android.widget.ListView
class MainActivity : AppCompatActivity() {

        override fun onCreate(savedInstanceState: Bundle?) {
                super.onCreate(savedInstanceState)
                setContentView(R.layout.activity_main)

                // use arrayadapter and define an array
                val arrayAdapter: ArrayAdapter<*>
                val users = arrayOf(
                        "Virat Kohli", "Rohit Sharma", "Steve Smith",
                        "Kane Williamson", "Ross Taylor"
                )

                // access the listView from xml file
                var mListView = findViewById<ListView>(R.id.userlist)
                arrayAdapter = ArrayAdapter(this,
                        android.R.layout.simple_list_item_1, users)
```

```
                    mListView.adapter = arrayAdapter
          }
}
```

Output:



8:34

List

Virat Kohli

Rohit Sharma

Steve Smith

Kane Williamson

Ross Taylor

6. Grid layout

A Grid View is a type of adapter view that is used to display the data in the grid layout format.

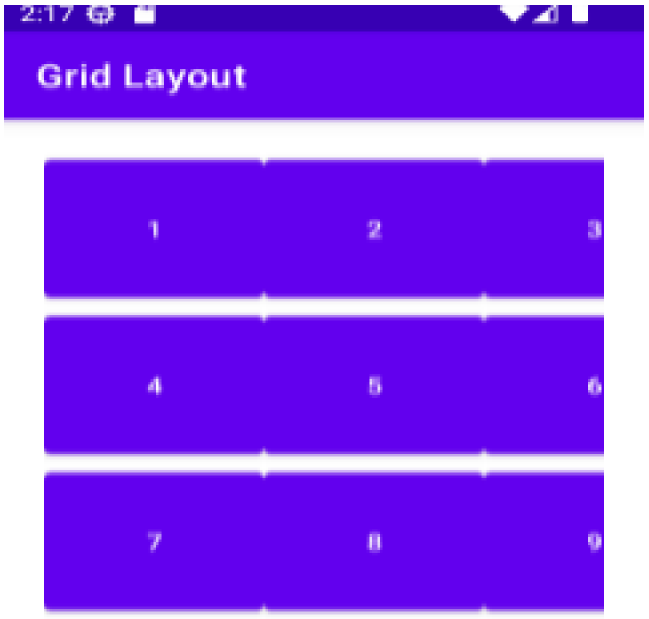
Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:rowCount="3"
    android:columnCount="3"
    android:padding="20dp">
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="1"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="2"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="3"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="4"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="5"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="6"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="7"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="8"/>
    <Button
        android:layout_width="110dp"
        android:layout_height="100dp"
        android:text="9"/>
</GridLayout>
```

MainActivtity.kt

```
package com.example.gridlayout

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
class MainActivity : AppCompatActivity() {

 override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)
    }
}
```

Output:


2:17
Grid Layout
1
2
3
4
5
6
7
8
9