Layout in Android:-

[Linear layout](https://abhiandroid.com/ui/linear-layout/) is a simple layout used in android for layout designing. In the [Linear layout](https://abhiandroid.com/ui/linear-layout/) all the elements are displayed in linear fashion means all the childs/elements of a [linear layout](https://abhiandroid.com/ui/linear-layout/) are displayed according to its orientation. The value for orientation property can be either horizontal or vertical.

**Types Of Linear Layout Orientation**

There are two types of linear layout orientation:

1. Vertical
2. Horizontal

As the name specified these two orientations are used to arrange there child one after the other, in a line, either vertically or horizontally. Let’s we describe these in detail.

**1.Vertical:**

In this all the child are arranged vertically in a line one after the other. In below code snippets we have specified orientation “vertical” so the childs/views of this layout are displayed vertically.

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

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"> <!-- Vertical Orientation set -->

<!-- Child Views(In this case 2 Button) are here -->

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button1"

android:id="@+id/button"

android:background="#358a32" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button2"

android:id="@+id/button2"

android:background="#0058b6" />

</LinearLayout>

**2. Horizontal:**

In this all the child are arranged horizontally in a line one after the other. In below code snippets we have specified orientation “horizontal” so the childs/views of this layout are displayed horizontally.

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

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"> <!-- Horizontal Orientation set -->

<!-- Child Views(In this case 2 Button) are here -->

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button1"

android:id="@+id/button"

android:background="#358a32" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button2"

android:id="@+id/button2"

android:background="#0058b6" />

</LinearLayout>

## Relative Layout In Android With Example

The [Relative Layout](https://abhiandroid.com/ui/relative-layout/) is very flexible layout used in android for custom layout designing. It gives us the flexibility to position our component/view based on the relative or sibling component’s position. Just because it allows us to position the component anywhere we want so it is considered as most flexible layout. For the same reason [Relative layout](https://abhiandroid.com/ui/relative-layout/) is the most used layout after the [Linear Layout](https://abhiandroid.com/ui/linear-layout/) in Android. It allow its child view to position relative to each other or relative to the container or another container.

Lets see different properties of Relative Layout which will be used while designing Android App UI:

**1.above:** Position the bottom edge of the view above the given anchor view ID and must be a reference of the another resource in the form of id. Example, android:layout\_above=”@+id/[textView](https://abhiandroid.com/ui/textview/)” .

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge"

android:text="Text2"

android:id="@+id/textView2"

android:layout\_above="@+id/textView"

android:layout\_marginBottom="100dp"

android:layout\_centerHorizontal="true"/>

**2. alignBottom:** alignBottom is used to makes the bottom edge of the view match the bottom edge of the given anchor view ID and it must be a reference to another resource, in the form of id. Example: android:layout\_ alignBottom =”@+id/button1″

In the below example we have aligned a view with id textView2 Bottom of another view with id [textView](https://abhiandroid.com/ui/textview/). Below is the coded and layout image.

**<!-- textView2 alignBottom of textView -->**

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge"

    android:layout\_centerHorizontal="true"

android:id="@+id/textView2"

android:layout\_alignBottom="@+id/textView"

android:text="Text2 alignBottom of Text1"

android:layout\_marginBottom="90dp"

/>

**3. alignLeft:** alignLeft is used to make the left edge of the view match the left edge of the given anchor view ID and must be a reference to another resource, in the form of Example: android:layout\_ alignLeft =”@+id/button1″.

Below is the code and layout image in which we have aligned a view with id textView2 left of another view with id textView.

**<!-- textView2 alignLeft of textView -->**

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge"

android:id="@+id/textView2"

android:layout\_alignLeft="@+id/textView"

android:text="Text2 alignLeft of Text1"

android:layout\_below="@+id/textView"

android:layout\_marginTop="20dp"/>

**4. alignRight:** alignRight property is used to 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 like this example: android:layout\_alignRight=”@+id/button1″

Below is the code and layout image in which we have aligned a view with id textView2 right of another view with id textView.

**<!-- textView2 alignRight of textView-->**

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge"

android:id="@+id/textView2"

android:layout\_alignRight="@+id/textView"

android:text="Text2 alignRight of Text1"

android:layout\_below="@+id/textView"

android:layout\_marginTop="20dp"/>

Relative Layout Example:-

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

**<!--Text View for Displaying SIGN IN Text At Top of UI-->**

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge"

android:text="SIGN IN"

android:id="@+id/textView3"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true" />

**<!--Text View for Displaying Username-->**

<TextView

android:id="@+id/userName"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="@dimen/activity\_horizontal\_margin"

android:layout\_marginTop="110dp"

android:text="UserName:"

android:textColor="#000000"

android:textSize="20sp" />

**<!--Text View for Displaying Password-->**

<TextView

android:id="@+id/password"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/userName"

android:layout\_margin="@dimen/activity\_horizontal\_margin"

android:text="Password:"

android:textColor="#000000"

android:textSize="20sp" />

**<!--Edit Text for Filling Username-->**

<EditText

android:id="@+id/edt\_userName"

android:layout\_width="fill\_parent"

android:layout\_height="40dp"

android:layout\_marginLeft="@dimen/activity\_horizontal\_margin"

android:layout\_marginTop="100dp"

android:layout\_toRightOf="@+id/userName"

android:hint="User Name" />

**<!--Edit Text for Filling Password-->**

<EditText

android:layout\_width="fill\_parent"

android:layout\_height="40dp"

android:layout\_below="@+id/edt\_userName"

android:layout\_centerVertical="true"

android:layout\_toRightOf="@+id/password"

android:hint="Password" />

**<!--Button for Clicking after filling details-->**

<Button

android:id="@+id/btnLogin"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/password"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="20dp"

android:background="#03B424"

android:text="Login"

android:textColor="#ffffff"

android:textStyle="bold" />

</RelativeLayout>

## Table Layout Tutorial With Example In Android

In Android, [Table Layout](https://abhiandroid.com/ui/tablelayout/) is used to arrange the group of views into rows and columns. [Table Layout](https://abhiandroid.com/ui/tablelayout/) containers do not display a border line for their columns, rows or cells. A Table will have as many columns as the row with the most cells.

**Important Points About Table Layout In Android:**

* For building a row in a table we will use the <TableRow> element. Table row objects are the child views of a table layout.
* Each row of the table has zero or more cells and each cell can hold only one view object like [ImageView](https://abhiandroid.com/ui/imageview/), [TextView](https://abhiandroid.com/ui/textview/) or any other view.
* Total width of a table is defined by its parent container
* Column can be both stretchable and shrinkable. If shrinkable then the width of column can be shrunk to fit the table into its parent object and if stretchable then it can expand in width to fit any extra space available.
* <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
* android:layout\_width="match\_parent"
* android:layout\_height="match\_parent"
* android:collapseColumns="0"> <!-- collapse the first column of the table row-->
* <!-- first row of the table layout-->
* <TableRow
* android:id="@+id/row1"
* android:layout\_width="fill\_parent"
* android:layout\_height="wrap\_content">
* <!-- Add elements/columns in the first row-->
* </TableRow>
* </TableLayout>

Simple Examle of TableLayout

<TableLayout 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"

android:stretchColumns="1">

<TableRow android:padding="5dip">

<TextView

android:layout\_height="wrap\_content"

android:layout\_marginBottom="20dp"

android:layout\_span="2"

android:gravity="center\_horizontal"

android:text="@string/loginForm"

android:textColor="#0ff"

android:textSize="25sp"

android:textStyle="bold" />

</TableRow>

<TableRow>

<TextView

android:layout\_height="wrap\_content"

android:layout\_column="0"

android:layout\_marginLeft="10dp"

android:text="@string/userName"

android:textColor="#fff"

android:textSize="16sp" />

<EditText

android:id="@+id/userName"

android:layout\_height="wrap\_content"

android:layout\_column="1"

android:layout\_marginLeft="10dp"

android:background="#fff"

android:hint="@string/userName"

android:padding="5dp"

android:textColor="#000" />

</TableRow>

<TableRow>

<TextView

android:layout\_height="wrap\_content"

android:layout\_column="0"

android:layout\_marginLeft="10dp"

android:layout\_marginTop="20dp"

android:text="@string/password"

android:textColor="#fff"

android:textSize="16sp" />

<EditText

android:id="@+id/password"

android:layout\_height="wrap\_content"

android:layout\_column="1"

android:layout\_marginLeft="10dp"

android:layout\_marginTop="20dp"

android:background="#fff"

android:hint="@string/password"

android:padding="5dp"

android:textColor="#000" />

</TableRow>

<TableRow android:layout\_marginTop="20dp">

<Button

android:id="@+id/loginBtn"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:layout\_span="2"

android:background="#0ff"

android:text="@string/login"

android:textColor="#000"

android:textSize="20sp"

android:textStyle="bold" />

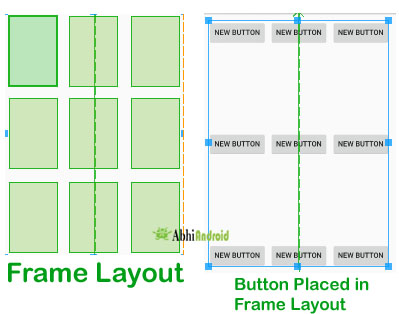
</TableRow>

</TableLayout>

## Frame Layout Tutorial With Example In Android Studio

[Frame Layout](https://abhiandroid.com/ui/framelayout/) is one of the simplest layout to organize view controls. They are designed to block an area on the screen. [Frame Layout](https://abhiandroid.com/ui/framelayout/) should be used to hold child view, because it can be difficult to display single views at a specific area on the screen without overlapping each other.

We can add multiple children to a [FrameLayout](https://abhiandroid.com/ui/framelayout/) and control their position by assigning gravity to each child, using the **android:layout\_gravity**attribute.



#### ****Attributes of Frame Layout:****

Lets see different properties of [Frame Layout](https://abhiandroid.com/ui/framelayout/) which will be used while designing Android App UI:

**1. android:id**

This is the unique id which identifies the layout in the R.[java](https://abhiandroid.com/java/) file.

Below is the id attribute’s example with explanation included in which we define the id for Frame Layout.

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

android:id="@+id/frameLayout"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"/>

**2. android:foreground**

Foreground defines the drawable to draw over the content and this may be a color value. Possible color values can be in the form of “#rgb”, “#argb”, “#rrggbb”, or “#aarrggbb”. This all are different color code model used.

Example: In Below example of foreground we set the green color for foreground of [frameLayout](https://abhiandroid.com/ui/framelayout/) so the [ImageView](https://abhiandroid.com/ui/imageview/) and other child views of this layout will not be shown.

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

android:id="@+id/framelayout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_gravity="center"

android:foregroundGravity="fill"

android:foreground="#0f0"><!--foreground color for a FrameLayout-->

<LinearLayout

android:orientation="vertical"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

>

<!-- Imageview will not be shown because of foreground color which is drawn over it-->

<ImageView

android:layout\_width="200dp"

android:layout\_height="200dp"

android:layout\_marginBottom="10dp"

android:src="@mipmap/ic\_launcher"

android:scaleType="centerCrop"

/>

<!--Textview will not be shown because of foreground color is drawn over it-->

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:gravity="center\_horizontal"

android:text="abhiAndroid"/>

</LinearLayout>

</FrameLayout>

## Absolute Layout With Example In Android Studio

In Android, an Absolute Layout is a layout used to design the custom layouts. In this layout you can specify the exact location of its children by using x and y coordinates.

**Android Absolute Layout Syntax Code:**

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

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<!-- add child view’s here -->

</AbsoluteLayout>

**Attributes of Absolute Layout:**

**1.id:** In android id is a attribute used to uniquely identify a Absolute Layout.  
Below is id attribute’s example code with explanation included.

<AbsoluteLayout

android:id="@+id/absoluteLayout"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"/>

**2.layout\_x:** In Absolute Layout layout\_x attribute is used to specify the x- coordinate of the view([TextView](https://abhiandroid.com/ui/textview/) or any other view). The possible value for this is in dp or px.

Grid Layout in Android:-

We can gird pattern design using this layout, grid provide collection of rows and columns to implement operation.