Layouts: Frame Layout

Android **Framelayout** is a ViewGroup subclass which is used to specify the position of multiple views placed on the top of each other to represent a single view screen.
Generally, we can say FrameLayout simply blocks a particular area on the screen to display a single view. Here, all the child views or elements are added in stack format means the most recently added child will be shown on the top of the screen.

But, we can add multiple children view and control their positions only by using gravity attributes in FrameLayout.

The **FrameLayout** can be defined using the code below:

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

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:orientation="vertical">



   // Add items or widgets here

</FrameLayout>
```

activity_main.xml file
In this file, we declare the FrameLayout and start adding multiple view like textView, editText, and Button etc. All the view are placed on each other, but we displace according to our requirement.

First, we add an image in the background and add other widgets on the top. On the screen, we can see the beautiful login page having an image in the background.

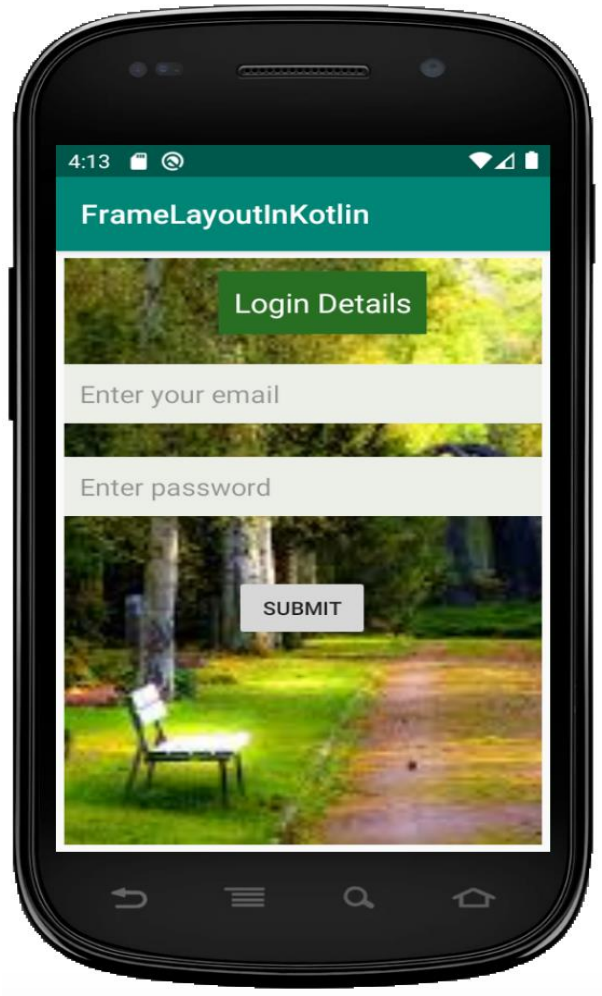
```
<?xml version="1.0" encoding="utf-8"?>


<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">

<ImageView
        android:id="@+id/imgvw1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        android:src="@drawable/img" />

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

<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:padding="10dp"
        android:hint="Enter your email" />

    <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:padding="10dp"
        android:hint="Enter password"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text= "Submit"
        android:layout_marginTop="240dp"
        android:layout_marginLeft="110dp"/>

</FrameLayout>
```

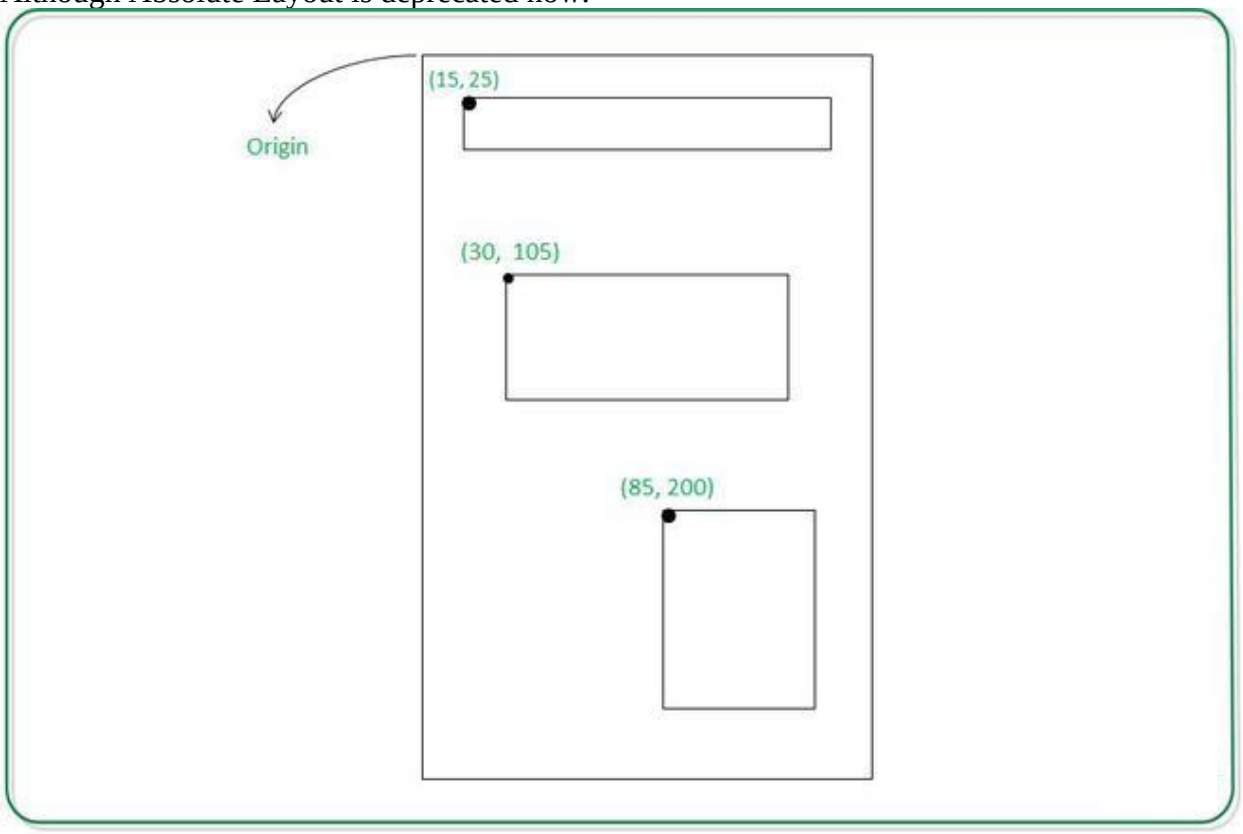
FrameLayout Output:

We need to run using Android Virtual Device(AVD) to see the output.



Absolute Layout

An **Absolute Layout** allows you to specify the exact location .i.e., X and Y coordinates of its children with respect to the origin at the top left corner of the layout. The absolute layout is less flexible and harder to maintain for varying sizes of screens that's why it is not recommended. Although Absolute Layout is deprecated now.



Some of the important Absolute Layout attributes are the following:

1. **android:id**: It uniquely specifies the absolute layout
2. **android:layout_x:** It specifies X-Coordinate of the Views (Possible values of this is in density-pixel or pixel)
3. **android:layout_y:** It specifies Y-Coordinate of the Views (Possible values of this is in dp or px)

The Syntax for Absolute Layout

```
<AbsoluteLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
<!--add child views-->
</AbsoluteLayout>
```

Example
In this example, we are going to create a basic application with Absolute Layout that is having two TextView. Note that we are going to implement this project using the **Java** language.

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
        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="fill_parent"
        android:layout_height="fill_parent"
        tools:context=".MainActivity">

        <!--Setting up TextViews-->
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_x="100px"
                android:layout_y="300px" />

        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_x="120px"
                android:layout_y="350px" />

</AbsoluteLayout>
```

Before moving further let's add some color attributes in order to enhance the app bar. Go to **app > res > values > colors.xml** and add the following color attributes.

```
<resources>
        <color name="colorPrimary">#0F9D58</color>
```

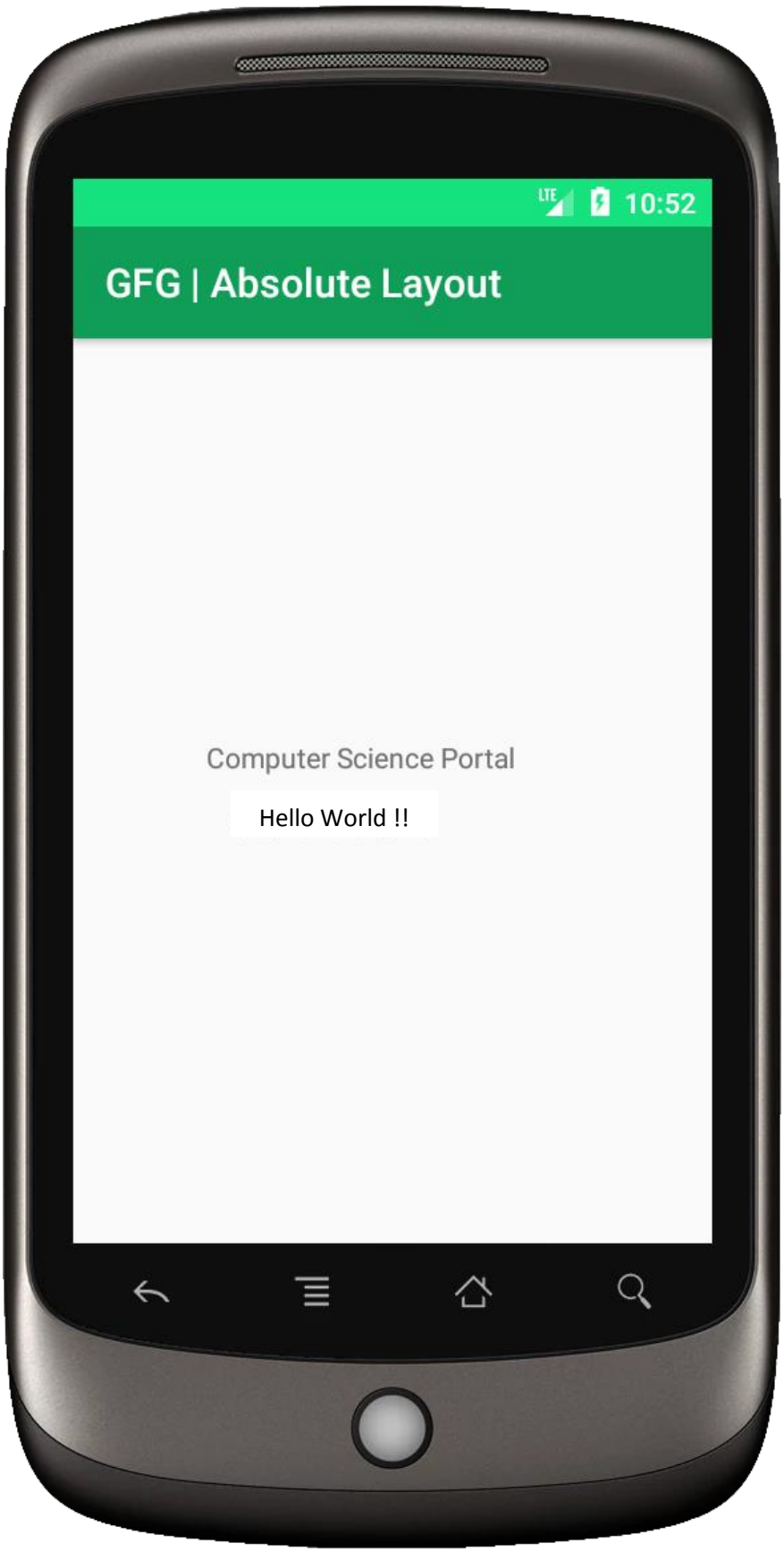
```
    <color name="colorPrimaryDark">#16E37F</color>
    <color name="colorAccent">#03DAC5</color>
</resources>
```

In this step, we will initialize the TextViews in our **MainActivity.java** file.

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView heading, subHeading;

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

        // Referencing the TextViews
        heading = (TextView) findViewById(R.id.heading);
        subHeading = (TextView) findViewById(R.id.subHeading);

        // Setting text dynamically
        heading.setText("Computer Science Portal");
```

```
                subHeading.setText("Hello World !!");

        }

}
```

You will see that TextViews are having fixed X and Y Coordinates.


10:52
GFG | Absolute Layout
Computer Science Portal
Hello World !!