LAB#03

INTRODUCING BASIC LAYOUT IN ANDROID ACTIVITY s

Layout

Using layouts to construct your screens in XML is best practice in Android. Each layout definition is stored in a separate file, each containing a single layout, in the res/layout folder. The filename then becomes the resource identifier.
The following list includes some of the most commonly used layout classes available in the Android SDK

- LinearLayout
- RelativeLayout.
- GridLayout.

Linear Layout:

The Linear Layout is one of the simplest layout classes. It allows you to create simple UIs (or UI elements) that align a sequence of child Views in either a vertical or a horizontal line.

Let's try this:

```
<?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"
   >

   <TextView
      android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Name:"
      />

   <EditText
      android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Name"
```

```
    />

  <TextView
    android:id="@+id/textView"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Email"
    tools:text="Email" />

  <EditText
    android:id="@+id/editText"
android:layout_width="match_parent"
android:layout_height="wrap_content"       android:ems="10"
    android:inputType="textEmailAddress"
    android:text="Email"
    tools:text="Email" />

  <Button
    android:id="@+id/button"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Submit" />

</LinearLayout>
```

Relative Layout

The Relative Layout lets you define the positions of each child View relative to the others and to the screen boundaries.

Let's try this:

```
<?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"
 >
<LinearLayout android:id="@+id/buttonbar"
android:layout_alignParentTop="true"
android:layout_width="fill_parent"
android:layout_height="match_parent"
android:orientation="vertical"
```

```
   android:padding="5dp"
   >


   <Button
      android:id="@+id/button2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
      android:text="Cancel" />

   <Button
      android:id="@+id/button3"
android:layout_width="match_parent"
android:layout_height="wrap_content"
      android:text="Ok" />

</LinearLayout>

   <ListView
      android:id="@+id/lstvw"
android:layout_marginTop="100dp"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_alignParentLeft="true"
      android:entries="@array/player">

   </ListView>

</RelativeLayout>
```

Grid Layout:

The Grid Layout is incredibly flexible and can be used to greatly simplify layouts and reduce or eliminate the complex nesting often required to construct UIs using the layouts described above.

The Grid Layout uses an arbitrary grid to position Views. By using row and column spanning, the Space View, and Gravity attributes, you can create complex without resorting to the often complex nesting required to construct UIs using the Relative Layout described previously.

Let's try this:

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

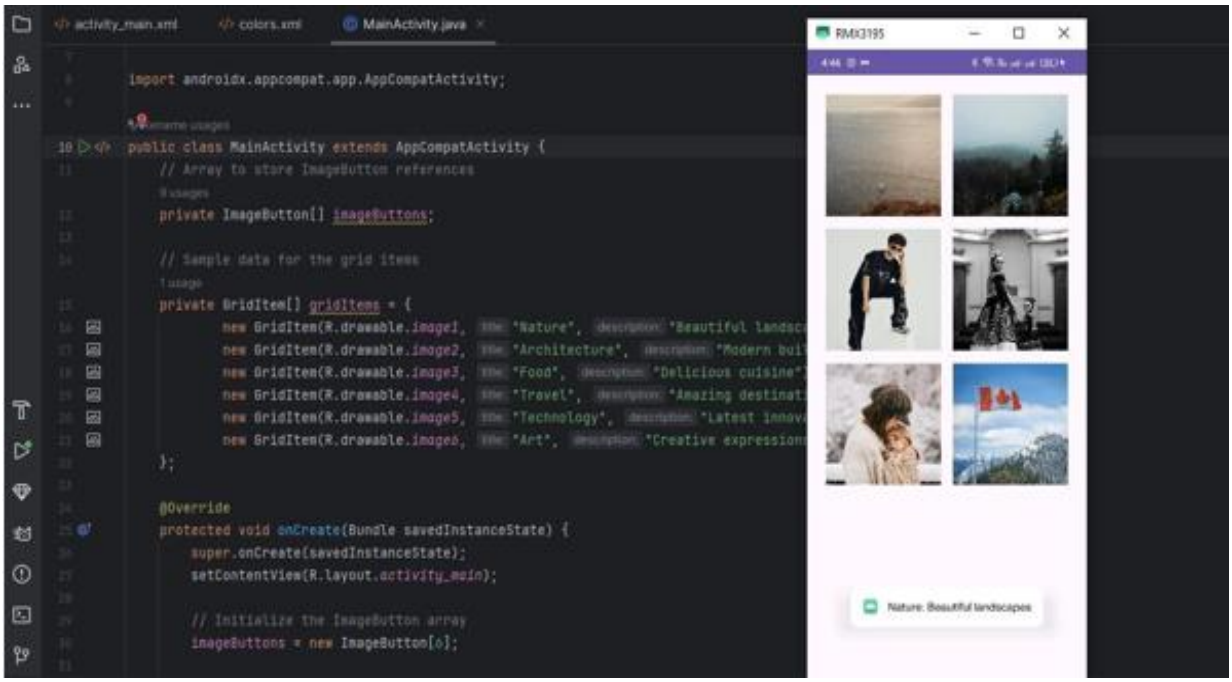
```
   android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
  >
   <ListView

   android:layout_gravity="fill"
      android:entries="@array/player"
      >
</ListView>
<LinearLayout
      android:layout_gravity="fill_horizontal"
android:orientation="horizontal"
android:padding="5dp">
   <Button
      android:text="Cancel"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_weight="1"/>
   <Button
      android:text="OK"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_weight="1"/>
   </LinearLayout>

</GridLayout>
```

Lab Task

1. Create a Grid Layout based on tiles using Image Button group


activity_main.xml
colors.xml
MainActivity.java
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
// Array to store ImageButton references
private ImageButton[] imageButtons;
// Sample data for the grid items
private GridItem[] gridItems = {
new GridItem(R.drawable.image1, "Nature", "Beautiful landsc
new GridItem(R.drawable.image2, "Architecture", "Modern bui
new GridItem(R.drawable.image3, "Food", "Delicious cuisine"
new GridItem(R.drawable.image4, "Travel", "Amazing destinat
new GridItem(R.drawable.image5, "Technology", "Latest innov
new GridItem(R.drawable.image6, "Art", "Creative expression
};
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
// Initialize the ImageButton array
imageButtons = new ImageButton[6];
RMX3195
Nature: Beautiful landscapes