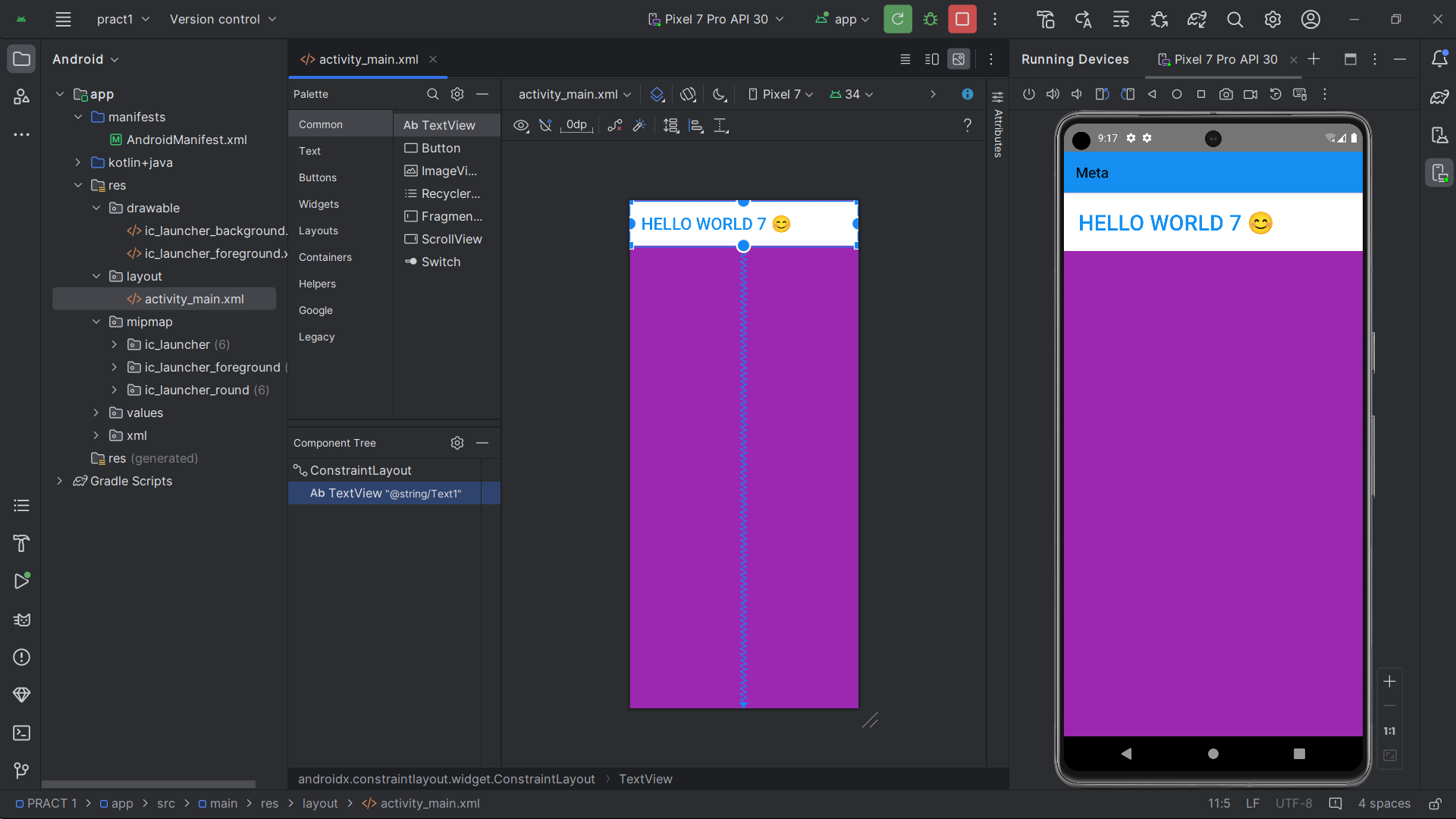
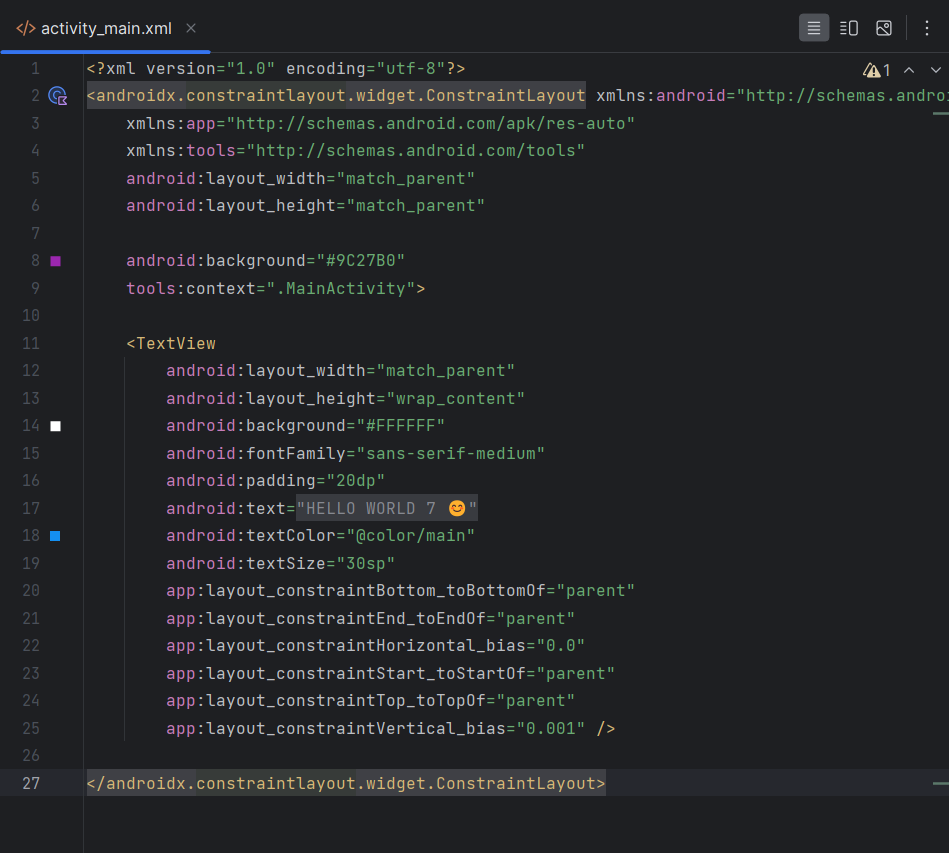
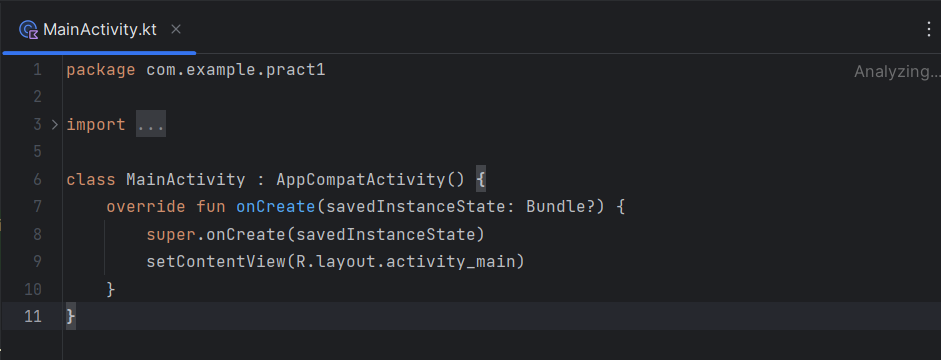
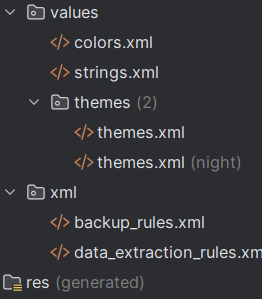
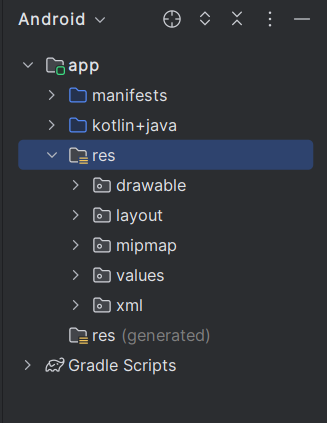
**PRACTICAL1 & 2**

**Creating a Project, Android Components, Activities, Services, Content Providers, Broadcast Receivers, Interface overview, Creating Android Virtual device, USB debugging mode, Android Application Overview. Simple “Hello World” program.**

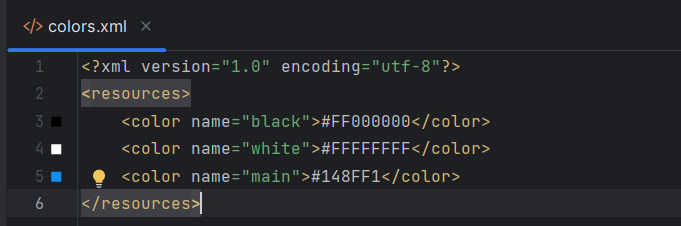
**Android Resources: (Color, Theme, String, Drawable, Dimension, Image),** 

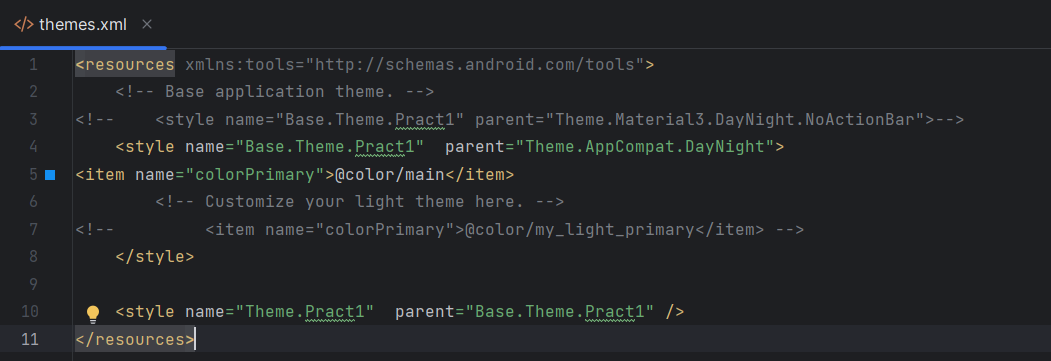


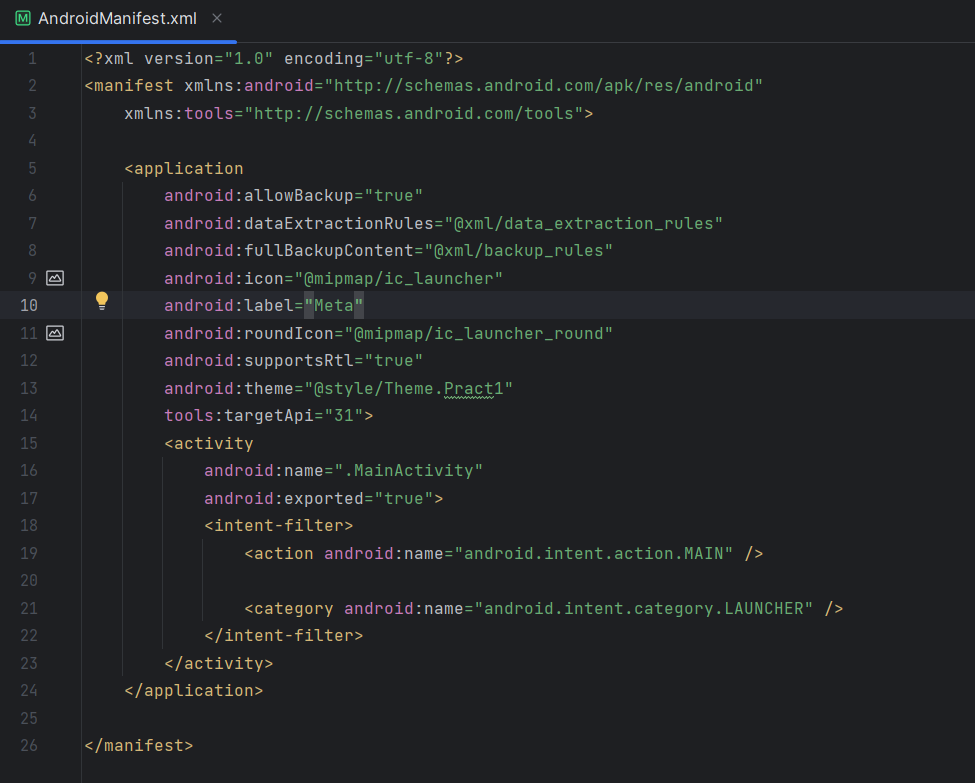


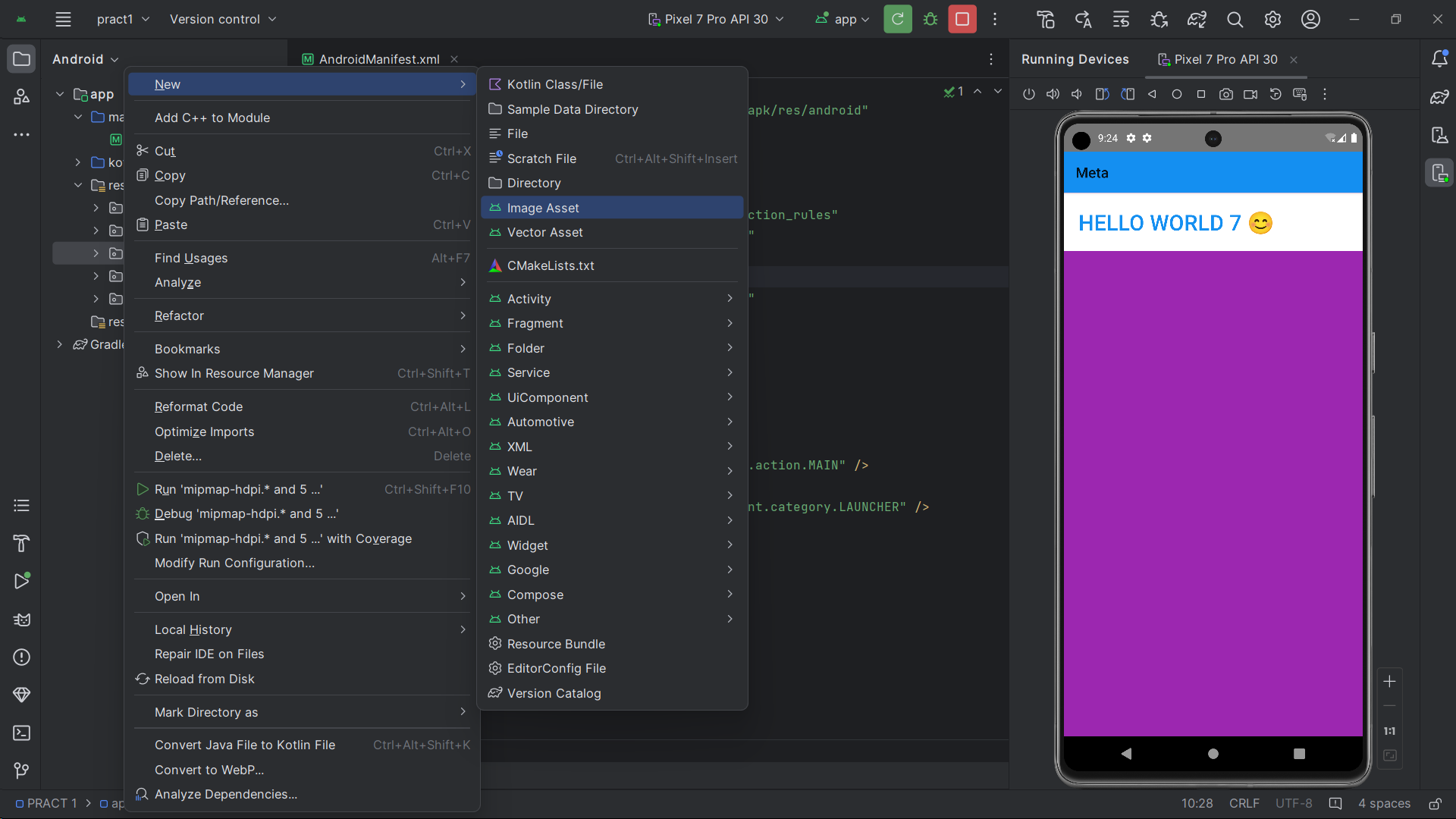


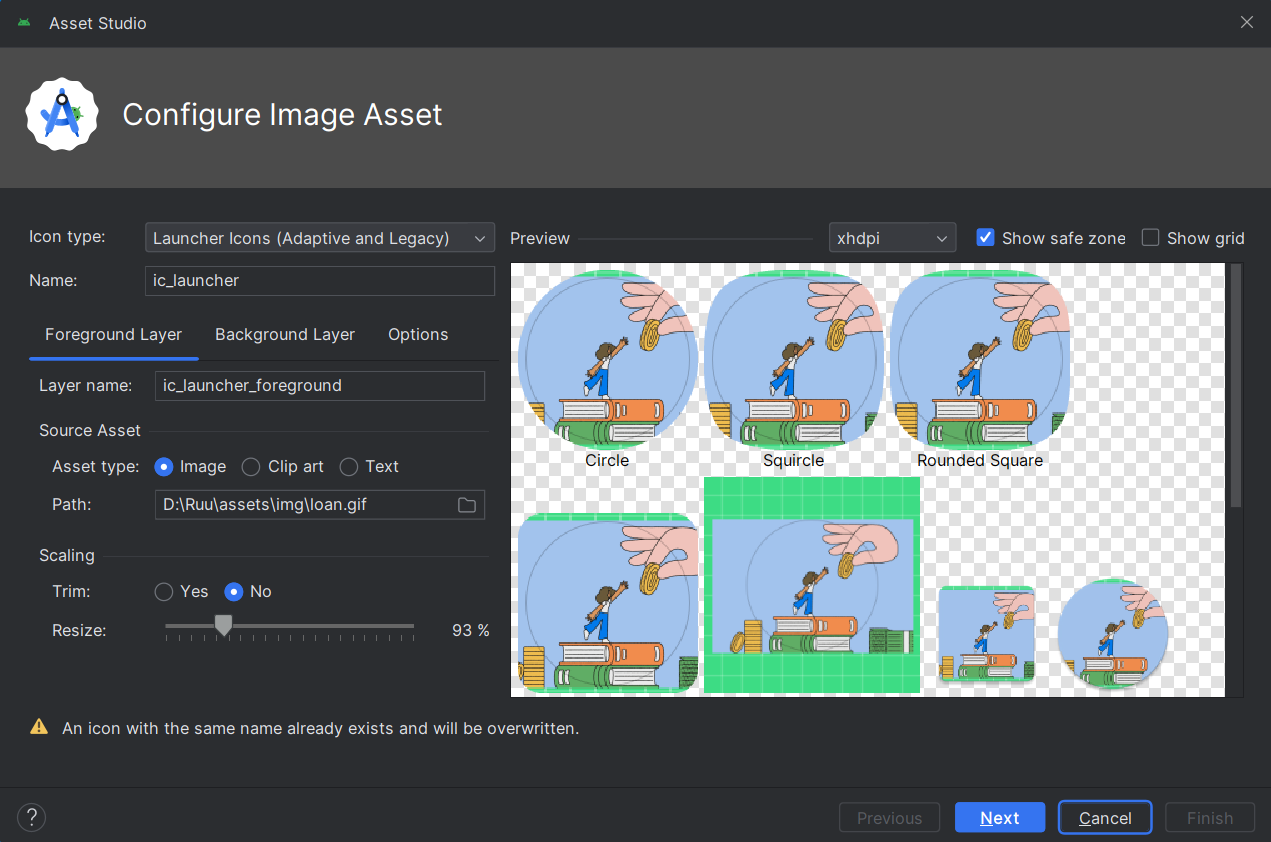


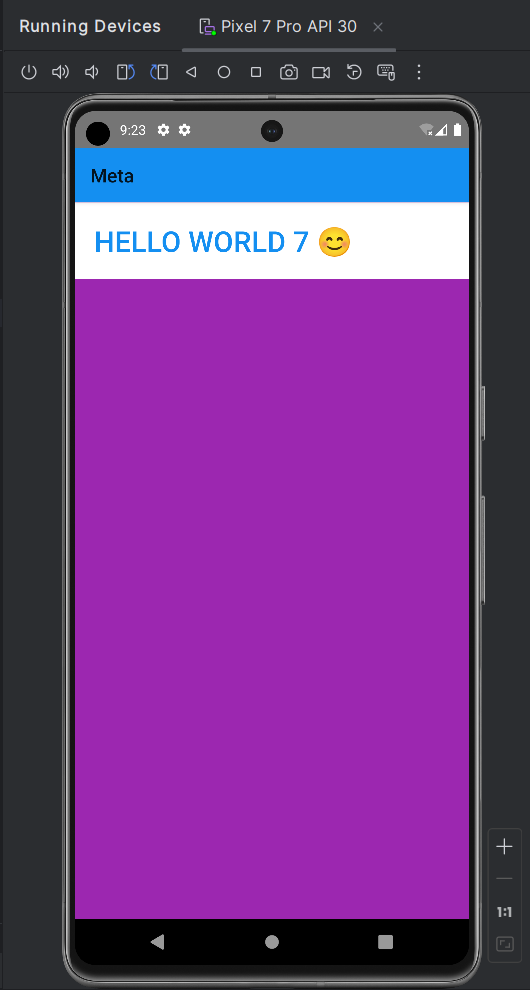




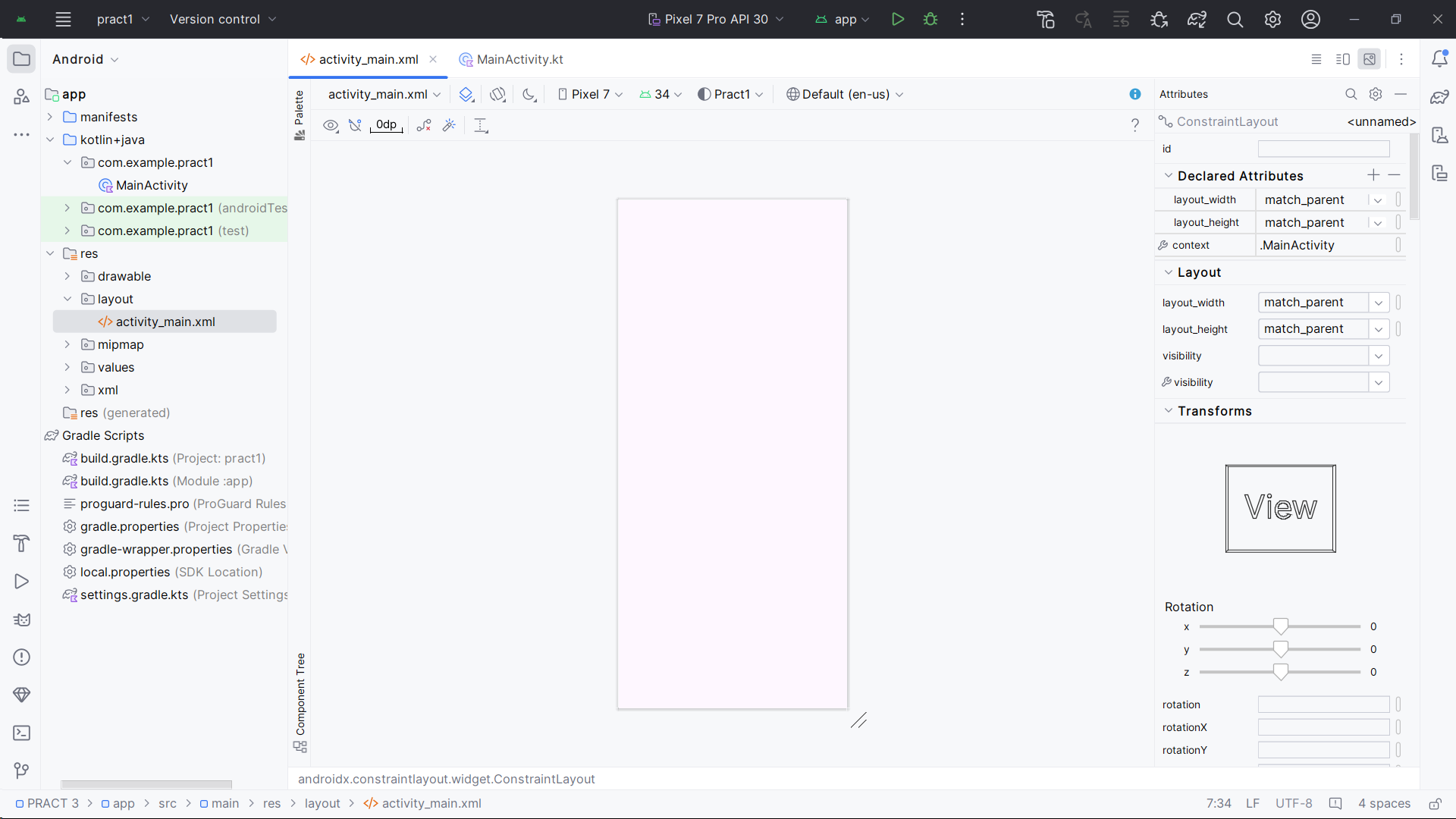


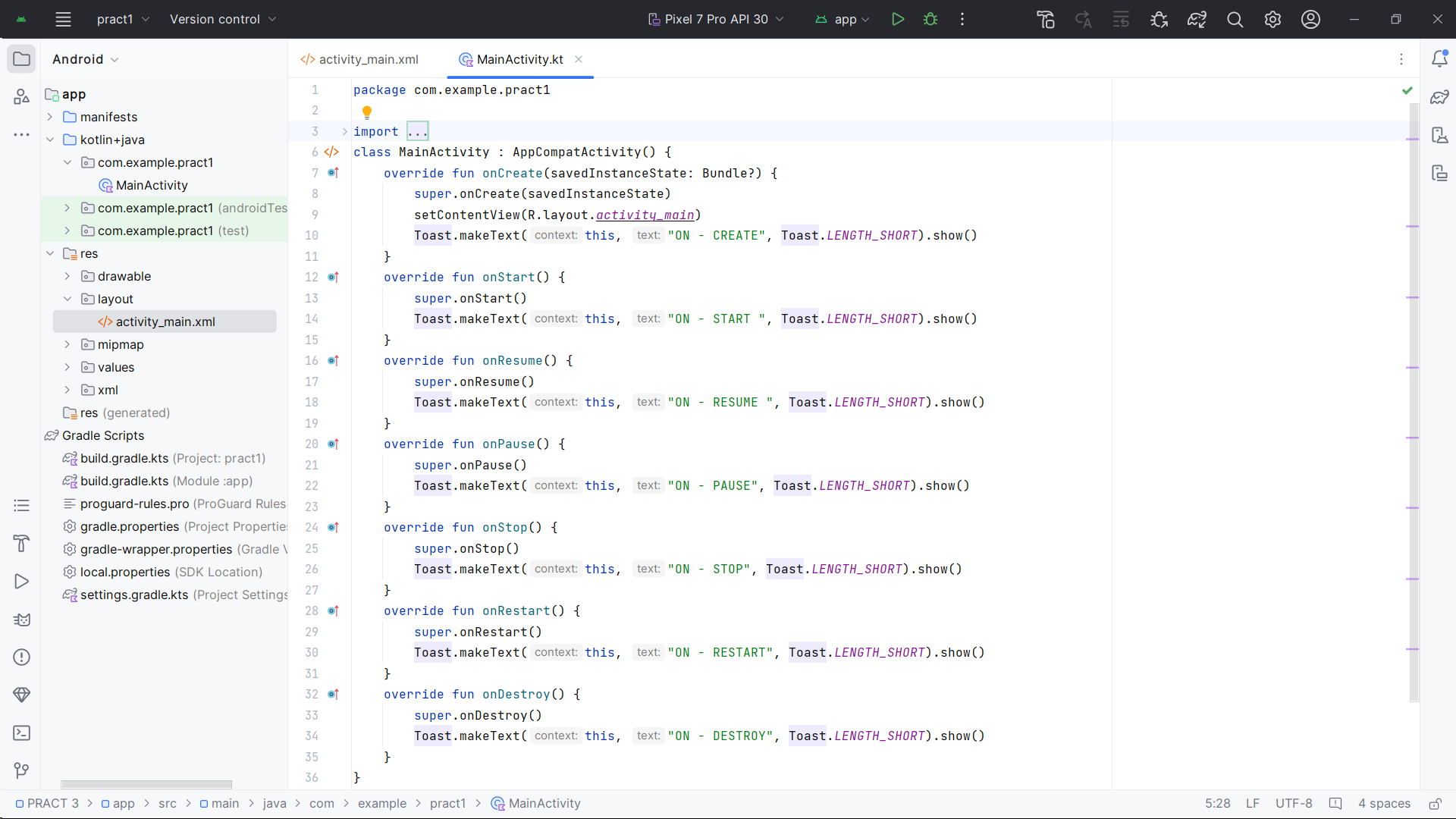


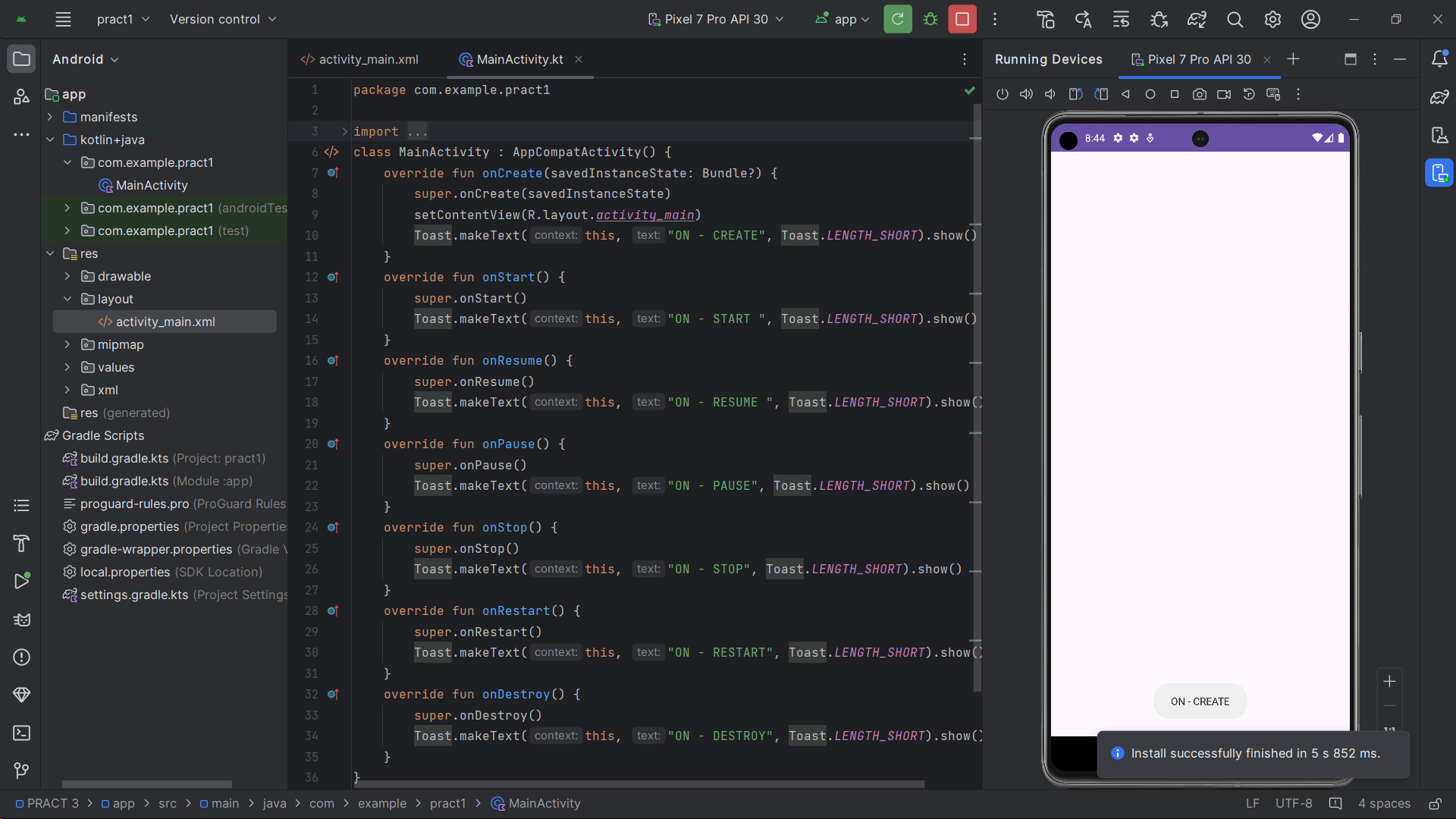




**PRACTICAL 3**

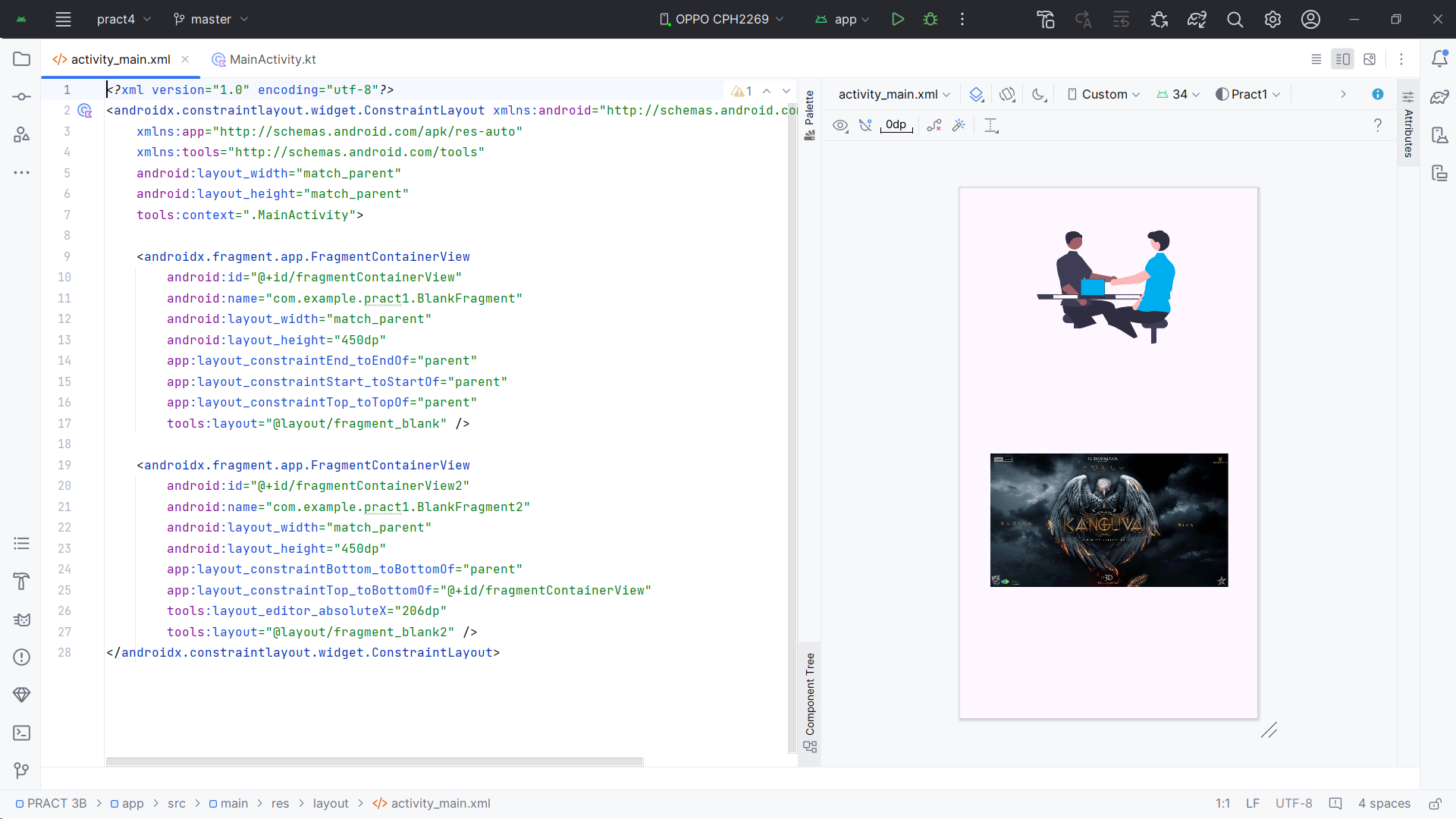


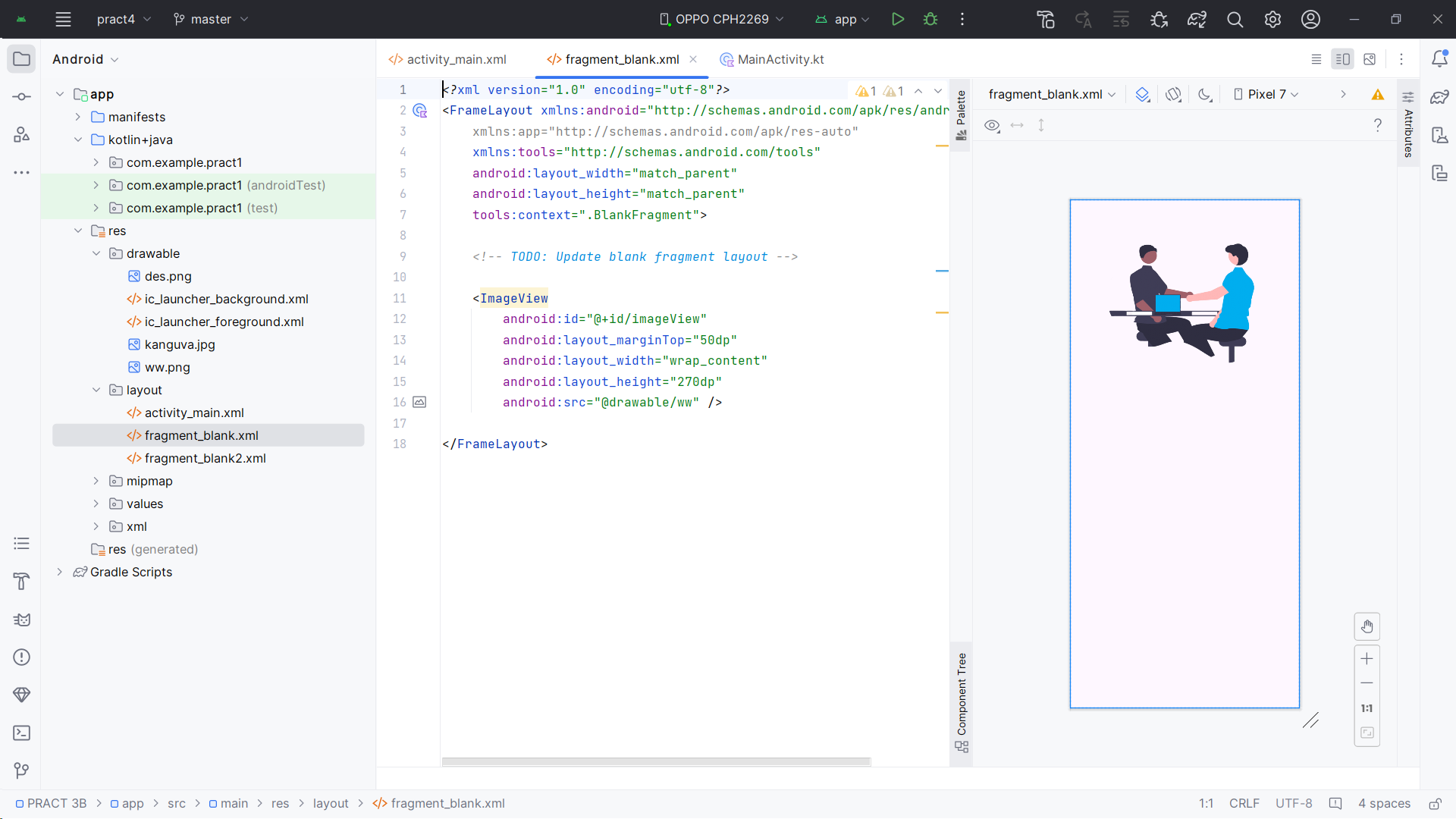


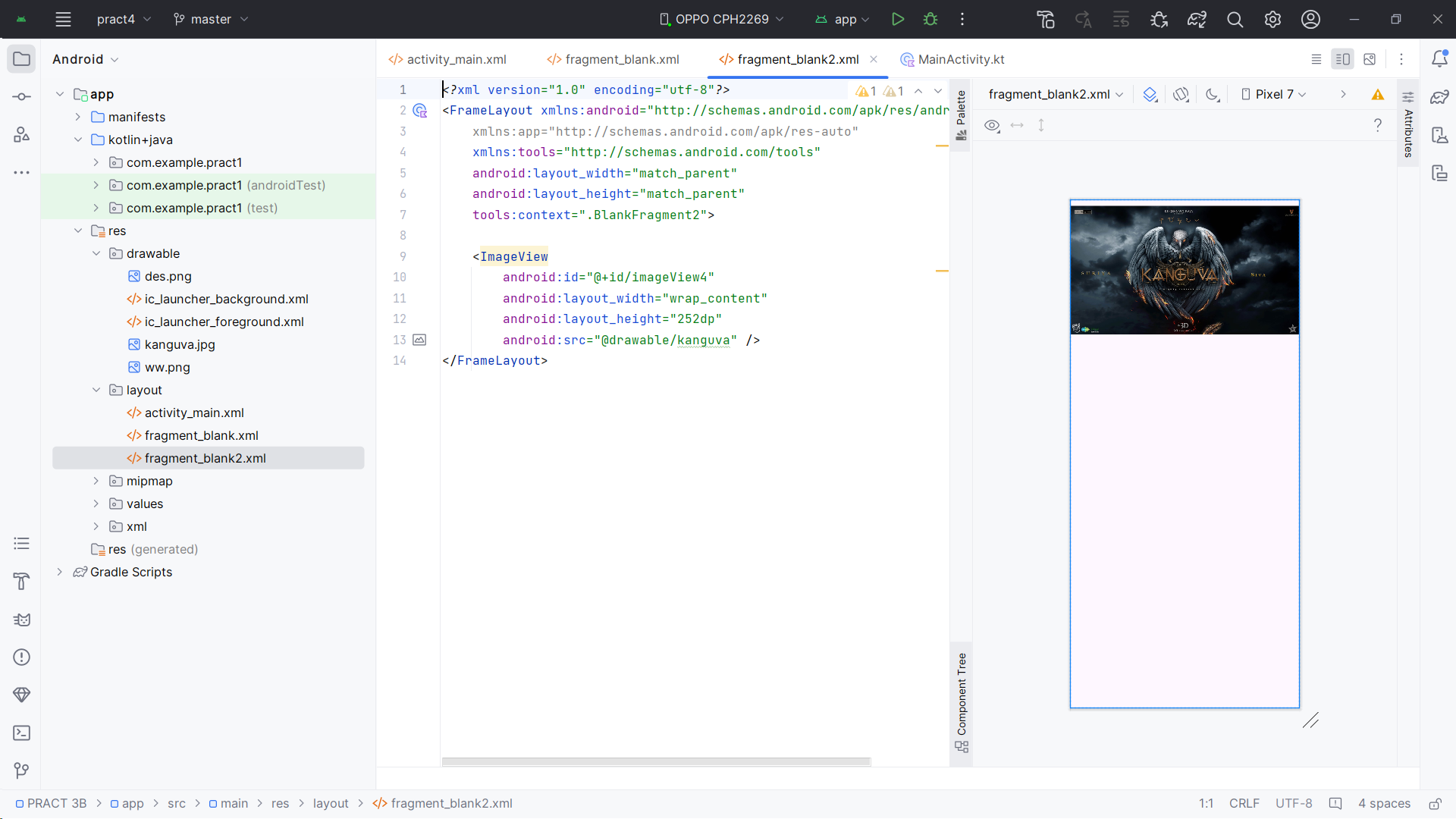
**** 

**PRACTICAL 3B Fragments**

**Life Cycle of fragments and multiple fragments.**









**PRACTICAL 4**

**Programs related to different Layouts (**Coordinate, Linear, Relative, Table, Absolute, Frame, List View, Grid View**)**

**Aim: Programs related to different Layouts**

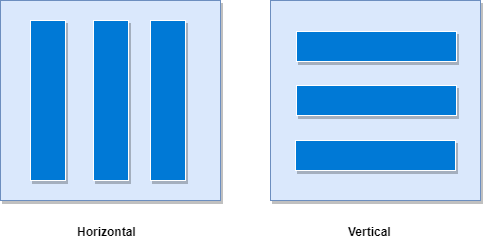
**[Linear, Relative, Table, Grid Layout.]**

**Linear Layout**

In android, Linear Layout is a View Group subclass which is used to render all child View instances one by one either in Horizontal direction or Vertical direction based on the orientation property.

In android, we can specify the linear layout orientation using android: orientation attribute.

Following is the pictorial representation of linear layout in android applications.



In **LinearLayout**, the child **View** instances arranged one by one, so the horizontal list will have only one row of multiple columns and vertical list will have one column of multiple rows.

**Activity\_main.xml**

<?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:paddingLeft="20dp"

    android:paddingRight="20dp"

    android:orientation="vertical" >

    <EditText

        android:id="@+id/txtTo"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:hint="To"/>

    <EditText

        android:id="@+id/txtSub"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

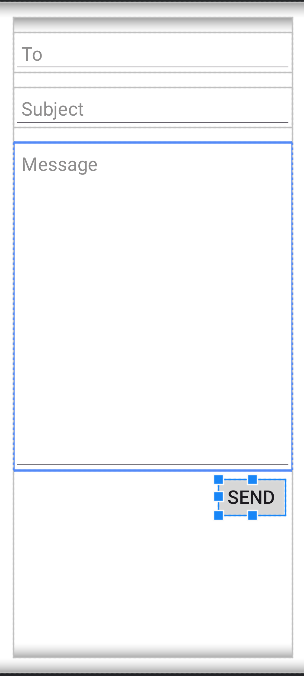
        android:hint="Subject"/>

    <EditText

        android:id="@+id/txtMsg"

        android:layout\_width="match\_parent"

        android:layout\_height="0dp"

        android:layout\_weight="1"

        android:gravity="top"

        android:hint="Message"/>

    <Button

        android:layout\_width="100dp"

        android:layout\_height="wrap\_content"

        android:layout\_gravity="right"

        android:text="Send"/>

</LinearLayout>

MainActivity.java

public class MainActivity extends AppCompatActivity {

    @Override

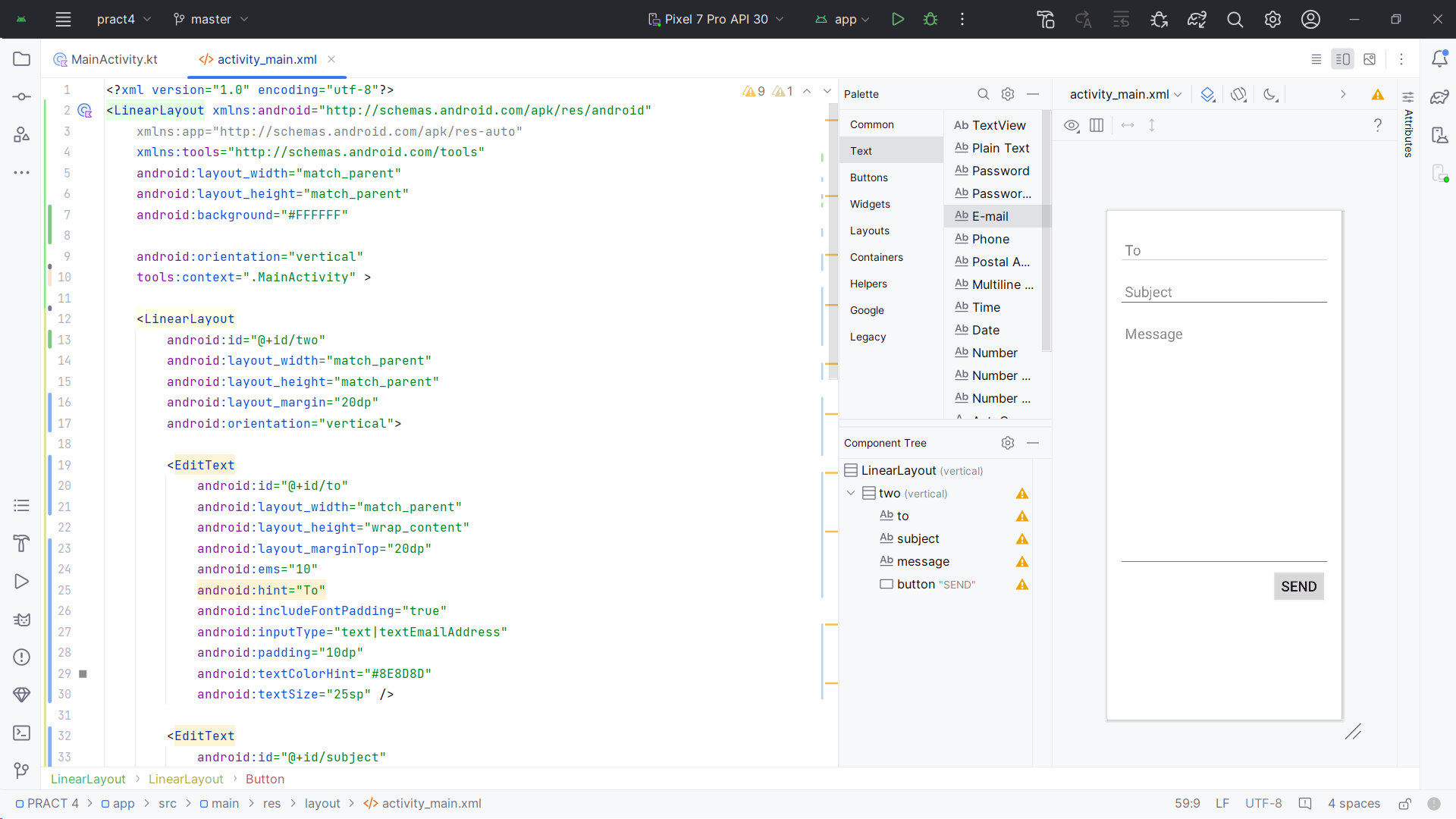
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

    }

}



The relative layout is used to arrange the child views in a proper order which means arranging the child objects relative to each other. Generally, if we create an application using a linear layout that consists of 5 buttons. Even if we specify weight and gravity properties they will not be relatively arranged. To arrange them in a proper order we need to use the relative layout. To arrange them we need some advanced properties. Basically, we use layout\_width, layout\_height, layout\_text properties while creating an application using the linear layout. But we need some more advanced properties which are supported by relative layout. There are so many properties that are supported by relative layout. some of the most used properties are listed below

layout\_alignParentTop

layout\_alignParentBottom

layout\_alignParentRight

layout\_alignParentLeft

layout\_centerHorozontal

layout\_centerVertical

layout\_above

layout\_below

