Create an app that showcases different types of layouts:

* In the main activity, add three buttons.
* Button-1: on click, it should start another activity with a linear layout as shown.
* Button-2: on click, it should start another activity with a table layout as shown.
* Button-3: on click, it should start another activity with a relative layout as shown.

**Note**: I also added a back button, so we don’t have to close the application and run again for running three different layouts. I am using “OnClickListener” to use back button which when clicked will take back to the MainActivity from any of the three layouts(Linear Layout, Table layout, Relative Layout),which is a straightforward method for handing button clicks.

**MainActivity.java** : In this activity the three buttons “buttonLinear”, “buttonTable”, “buttonrelative” are initialized. Each of these buttons is associated with the method “OnClickListener”. An Intent is created to start a new activity when the button is clicked. The “LinearLayoutActivity” is started when the “buttonLinear” is clicked, same way “TableLayoutActivity”, “RelativeLayoutActivity” are started when “buttonTable”, “buttonRelative” is clicked. The “intent” calls take the user from the main activity to the selected layout activity, demonstrating basic activity switching and navigation in Android.

A screenshot of a computer

Description automatically generated

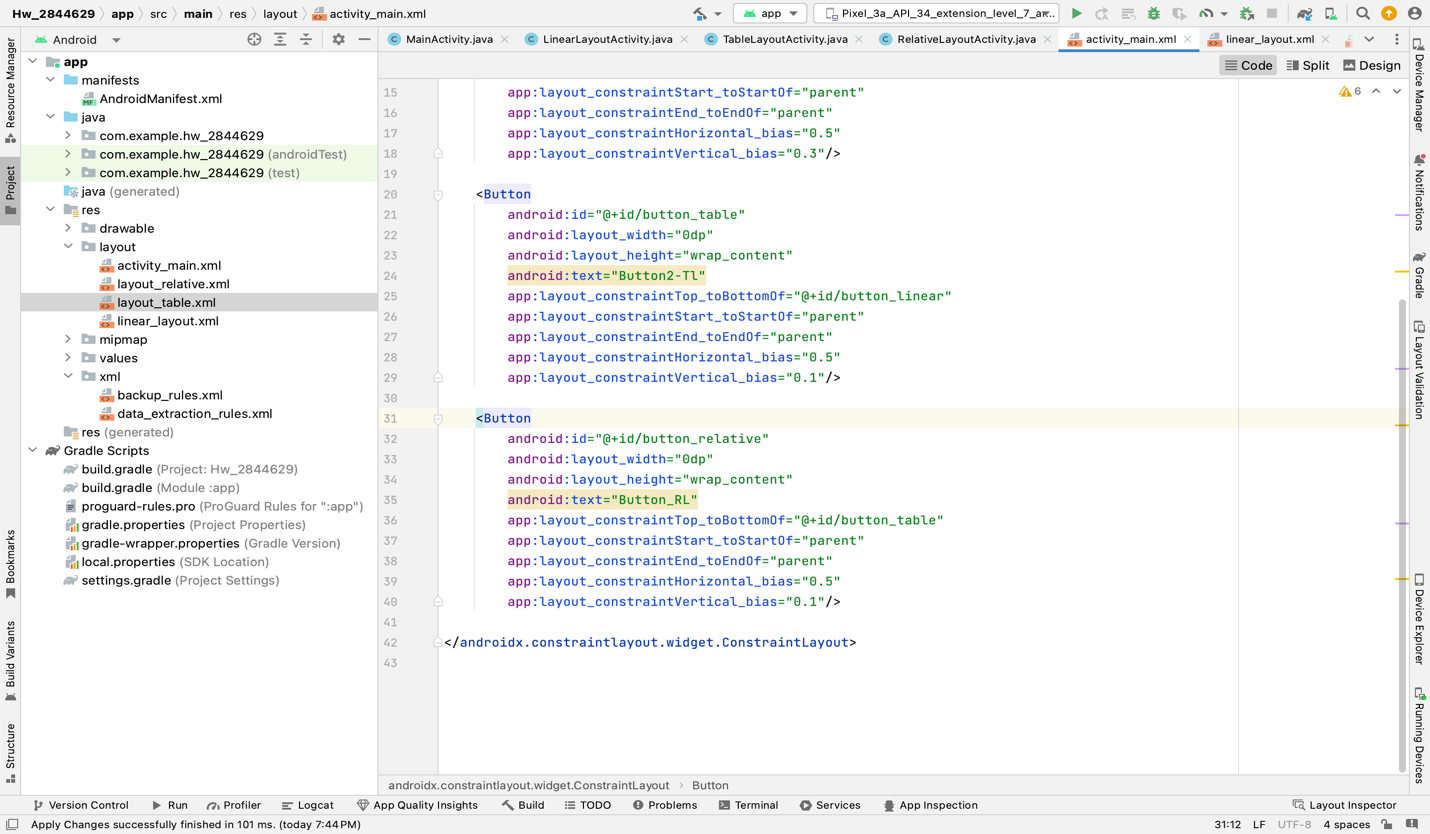
A screenshot of a computer

Description automatically generated

**MainActivity.xml** : It defines the app layout using “ConstraintLayout”. It has three buttons that are equally spaced vertically: “button\_linear”, “button\_table”, and “button\_relative”. Every button contains text describing its function and is configured to spread over the parent's width 0dp with limits to start and end of parent. The buttons are arranged in relation to one another as follows: “button\_table” is the second button from the bottom, “button\_linear” is the top button, and “button\_relative” is the third button from the bottom. A vertically aligned menu of buttons is efficiently created by this arrangement.

A screenshot of a computer

Description automatically generated



**LinearLayoutActivity.java** : To build up the activity's user interface, the “onCreate” function is overridden in the “LinearLayoutActivity” class, which extends “AppCompatActivity”. The activity's content view is initially configured to a layout specified in “R.layout.linear\_layout” upon creation. The code creates a “OnClickListener” and initializes a “Button” with the ID buttonBack. The application starts a new “Intent” to return to “MainActivity” when the “buttonBack” is clicked. This code shows how to handle button clicks in Android to switch between activities in a basic pattern.

A screenshot of a computer

Description automatically generated

**Linear\_layout.xml** : This XML layout uses a vertically oriented “LinearLayout” to define the user interface for an Android activity. After a “TextView” with the text "Linear Layout Example," two generic buttons, a “EditText” for text input, and a "Back" button with the ID “@+id/buttonBack” are included. All buttons and the EditText extend to fill the full width of the parent container, and all elements are stacked vertically with uniform margin spacing.

A screenshot of a computer

Description automatically generated

**TableLayoutActivity.java** : The activity's layout is set to "R.layout.layout\_table" by the onCreate function of the "TableLayoutActivity" class, which extends "AppCompatActivity". Initialization of a Button with the ID "buttonBack" and setting of a "OnClickListener". This button illustrates basic button interaction and activity navigation in application by creating an intent and using it to return to "MainActivity" when pressed.

A screenshot of a computer

Description automatically generated

**Layout\_table.xml** : This provides a table row with "TableRow" characteristics for every row, enabling TextView and EditText input to scale proportionately. A Checkbox and two Button elements are also included in the layout; these take up the whole width of the row because of the "layout\_span" argument. A form-like user interface is intended for this layout.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**RelativeLayoutActivity.java** : This class of an application the code sets up the user interface from “R.layout.layout\_relative” upon creation. A Button with the ID "buttonBack" is initialized, and a "OnClickListener" is set on it. A simple pattern for managing button clicks and switching between activities inside the app is shown when the buttonBack is hit, causing the application to establish an intent to return to the "MainActivity".

A screenshot of a computer

Description automatically generated

**Layout\_relative.xml** : the “RelativeLayout” is used as a root to for flexible positioning of the UI elements relative to each other. An "EditText" is positioned directly below a "TextView" that is positioned. The "EditText" is centered, and a Submit button is below it. To further demonstrate the use of both linear and relative layouts for adaptable UI design in an Android application, a LinearLayout with a Back button has been added at the bottom.

A screenshot of a computer

Description automatically generated

Androidmanifest.xml : it defines the basic configuration of the for the application. The "AndroidManifest.xml" file specifies the fundamental settings for an Android application, including four activities: TableLayoutActivity, RelativeLayoutActivity, MainActivity, and LinearLayoutActivity with MainActivity as the entry point.

A screenshot of a computer

Description automatically generated

Below are the screenshots of MainActivity and 3 layouts.

A black cell phone with a white screen

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Screenshot 1(MainActivity). Screenshot 2(LinearLayout). Screenshot 3(Table Layout).

A screenshot of a cell phone

Description automatically generated

Screenshot 4(RelativeLayout).