Practical 2- Create an android app with Interactive User Interface using Layouts.

An Android layout is a class that handles arranging the way its children appear on the screen. Anything that is a View (or inherits from **View**) can be a child of a layout. All of the layouts inherit from **ViewGroup** (which inherits from **View**) so you can nest layouts. You could also create your own custom layout by making a class that inherits from **ViewGroup**.

There are various in android:

1. Linear Layout :

LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.

Vertical

Horizontal

Following are the important attributes specific to LinearLayout −

- android:id

  This is the ID which uniquely identifies the layout.

- android:divider

  This is drawable to use as a vertical divider between buttons. You use a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".

- android:gravity

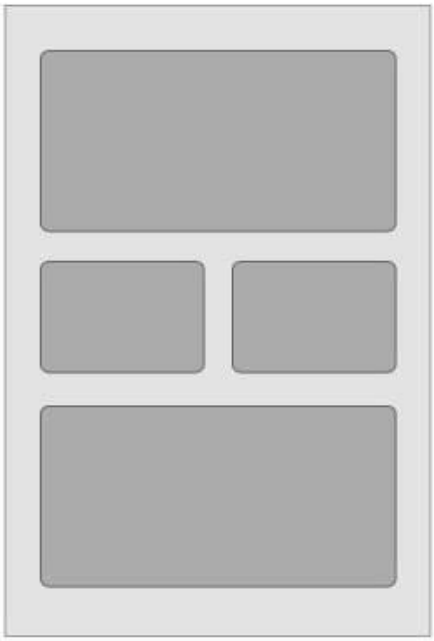
  This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.

- android:orientation

  This specifies the direction of arrangement and you will use "horizontal" for a row, "vertical" for a column. The default is horizontal.

2. Relative Layout:

Android RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.



Following are the important attributes specific to RelativeLayout −

- android:gravity

  This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.

- android:ignoreGravity

  This indicates what view should not be affected by gravity.

3. Grid Layout:

Android GridView shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a ListAdapter

An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter can be used to supply the data to like spinner, list view, grid view etc.

- android:numColumns

  Defines how many columns to show. May be an integer value, such as "100" or auto_fit which means display as many columns as possible to fill the available space.

- android:columnWidth

  This specifies the fixed width for each column. This could be in px, dp, sp, in, or mm.
- android:verticalSpacing

  Defines the default vertical spacing between rows. This could be in px, dp, sp, in, or mm.

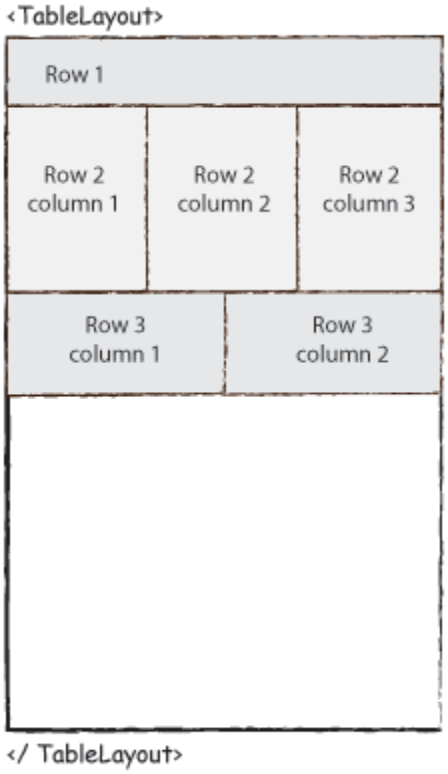
- android:horizontalSpacing

  Defines the default horizontal spacing between columns. This could be in px, dp, sp, in, or mm.

4. Table Layout:

Android TableLayout going to be arranged groups of views into rows and columns. You will use the <TableRow> element to build a row in the table. Each row has zero or more cells; each cell can hold one View object.

TableLayout containers do not display border lines for their rows, columns, or cells.



- android:collapseColumns

  This specifies the zero-based index of the columns to collapse. The column indices must be separated by a comma: 1, 2, 5.

- android:shrinkColumns

  The zero-based index of the columns to shrink. The column indices must be separated by a comma: 1, 2, 5.

- android:stretchColumns

  The zero-based index of the columns to stretch. The column indices must be separated by a comma: 1, 2, 5.