**Lab Manual: Layouts in Android**

**Objective:**

To understand and implement different types of layouts in Android: LinearLayout, TableLayout, RelativeLayout, FrameLayout, GridLayout, and ScrollView.

**1. LinearLayout**

**Description:**A LinearLayout arranges its children in a single row or column.

**Properties:**

* orientation: Defines the direction of the layout (horizontal or vertical).
* gravity: Specifies how children are positioned.
* weight: Allows children to expand to fill space.

**Example: Vertical LinearLayout**

<LinearLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="TextView 1" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button 1" />

</LinearLayout>

**2. TableLayout**

**Description:**A TableLayout arranges its children into rows and columns.

**Properties:**

* android:stretchColumns: Defines which columns should stretch to fill space.
* TableRow: Used to define a row within the TableLayout.

**Example: Simple TableLayout**

<TableLayout

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

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<TableRow>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Row 1, Column 1" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Row 1, Column 2" />

</TableRow>

<TableRow>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Row 2, Column 1" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Row 2, Column 2" />

</TableRow>

</TableLayout>

**3. RelativeLayout**

**Description:**A RelativeLayout allows positioning of child views relative to each other or to the parent layout.

**Properties:**

* layout\_alignParentTop, layout\_centerInParent, etc.: Used for positioning child views relative to the parent.
* layout\_toLeftOf, layout\_toRightOf, etc.: Used for positioning relative to other views.

**Example: Simple RelativeLayout**

<RelativeLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="TextView 1"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button 1"

android:layout\_below="@id/textView1"

android:layout\_centerHorizontal="true" />

</RelativeLayout>

**4. FrameLayout**

**Description:**A FrameLayout is designed to block out an area on the screen to display a single item.

**Properties:**

* Child views are drawn in a stack; the most recent child added is drawn on top.

**Example: Simple FrameLayout**

<FrameLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<ImageView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:src="@drawable/sample\_image" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Overlay Text"

android:layout\_gravity="center" />

</FrameLayout>

**5. GridLayout**

**Description:**A GridLayout places its children in a grid.

**Properties:**

* android:rowCount, android:columnCount: Define the number of rows and columns.
* layout\_row, layout\_column: Define the row and column of each child view.

**Example: Simple GridLayout**

<GridLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:rowCount="2"

android:columnCount="2">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Row 1, Col 1"

android:layout\_row="0"

android:layout\_column="0" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Row 1, Col 2"

android:layout\_row="0"

android:layout\_column="1" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Row 2, Col 1"

android:layout\_row="1"

android:layout\_column="0" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Row 2, Col 2"

android:layout\_row="1"

android:layout\_column="1" />

</GridLayout>

**6. ScrollView**

**Description:**A ScrollView is used to display a vertically scrollable area of the screen.

**Properties:**

* Can contain only one direct child.
* Commonly used with other layouts like LinearLayout for scrolling content.

**Example: Simple ScrollView with LinearLayout**

<ScrollView

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Scrollable TextView 1" />

<!-- Add more views here -->

</LinearLayout>

</ScrollView>

**Conclusion:**

Each layout has its unique properties and use cases. Practice with different combinations to understand which layout is suitable for different UI requirements.

**Assignment:**

1. Create a layout using RelativeLayout with a TextView centered on the screen and a Button below it.
2. Design a form using TableLayout with labels and input fields.
3. Implement a complex screen using FrameLayout and ScrollView to understand layering and scrolling.