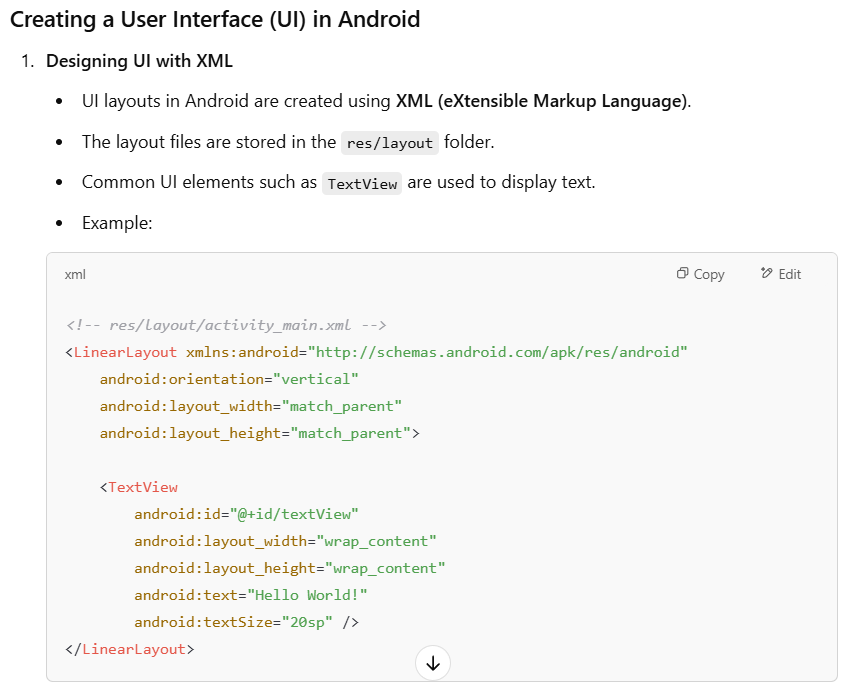
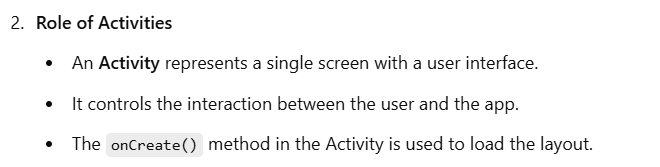
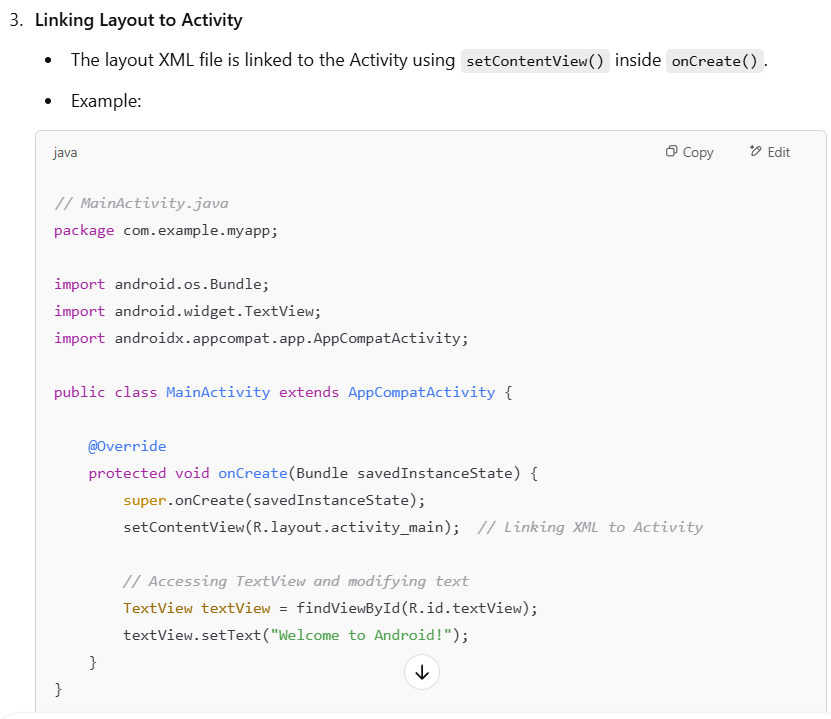
1. **Describe the process of creating a user interface in Android. Include details on the use of XML, the role of Activities, and how you can link your layout files to your activity code.**

Ans:-

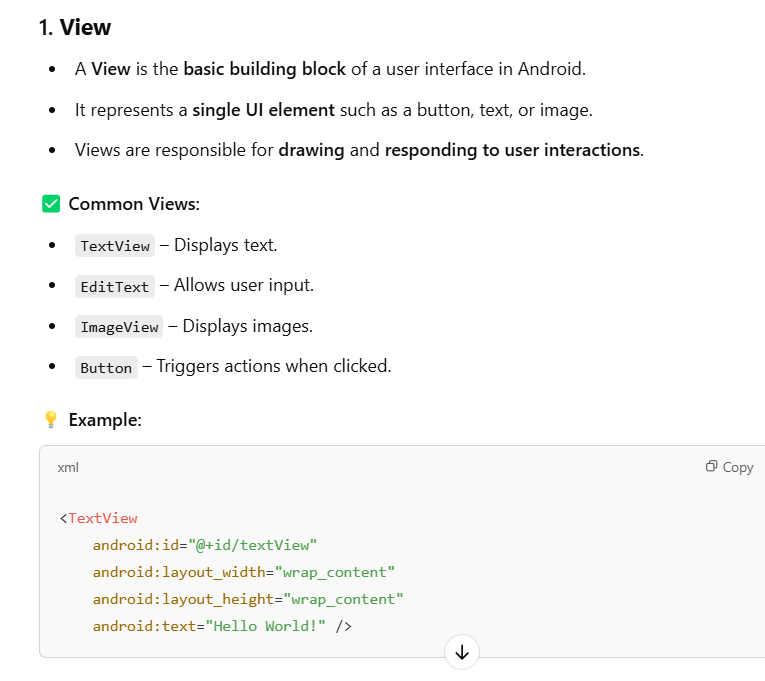


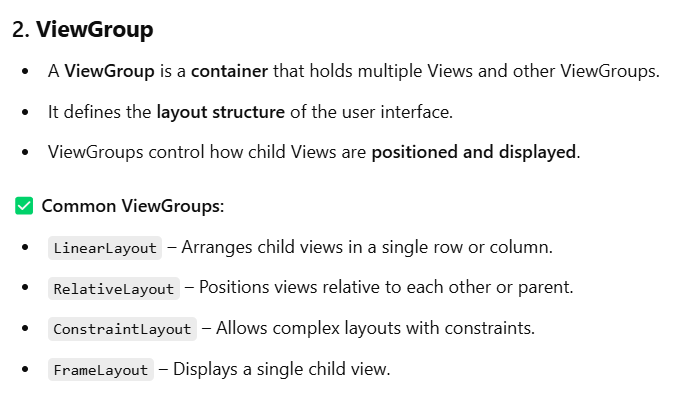


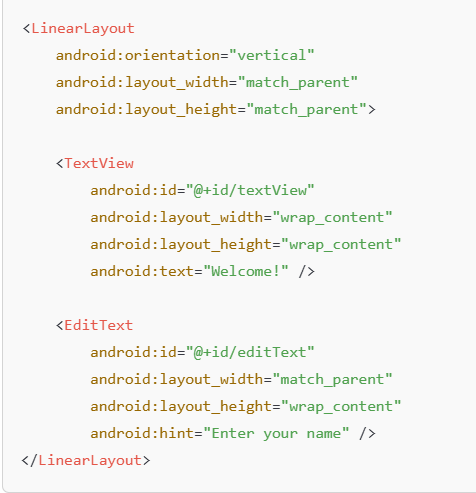


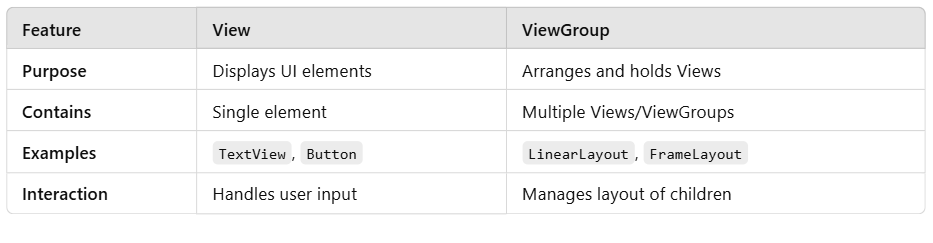
1. **Explain the differences between a View and a ViewGroup in Android. How does each contribute to the layout of a user interface? Provide examples of common Views and ViewGroups used in Android development.**

Ans : -









1. **Discuss the purposes and differences of FrameLayout, LinearLayout, RelativeLayout, and TableLayout in Android. Provide examples of scenarios where each layout type would be most appropriate.**

Ans : -

**1. FrameLayout**

-> A FrameLayout is a simple layout that places one child view on top of another.

-> It stacks views on top of each other, with the most recent child displayed on top.

✅ Best Use Case:

Used when you want to display a single view or overlap multiple views, such as adding a button on an image

**Example:**

<FrameLayout

android:layout\_width="match\_parent"

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

<ImageView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:src="@drawable/image" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Overlay Text"

android:layout\_gravity="center" />

</FrameLayout>

**2. LinearLayout:**

-> A LinearLayout arranges child views in a single row or column.

-> android:orientation can be set to either horizontal or vertical.

✅ Best Use Case:

Used when you want to align elements sequentially.

**Example:**

<LinearLayout

android:orientation="vertical"

android:layout\_width="match\_parent"

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

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Name" />

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter Name" />

</LinearLayout>

**3. RelativeLayout**

-> A RelativeLayout positions child views relative to each other or parent.

-> It uses attributes like layout\_alignParentTop, layout\_below, and layout\_toRightOf to define the position.

✅ Best Use Case:

Used when you want a dynamic and flexible layout with views placed relative to each other.

**Example:**

<RelativeLayout

android:layout\_width="match\_parent"

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

<TextView

android:id="@+id/label"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Username" />

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

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

android:hint="Enter Username" />

</RelativeLayout>

**4. TableLayout**

-> A TableLayout arranges views in rows and columns similar to a table.

->Each row is defined using a TableRow.

✅ Best Use Case:

Used when you want to display data in a tabular format.

**Example:**

<TableLayout

android:layout\_width="match\_parent"

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

<TableRow>

<TextView

android:text="Name" />

<TextView

android:text="Age" />

</TableRow>

<TableRow>

<TextView

android:text="John" />

<TextView

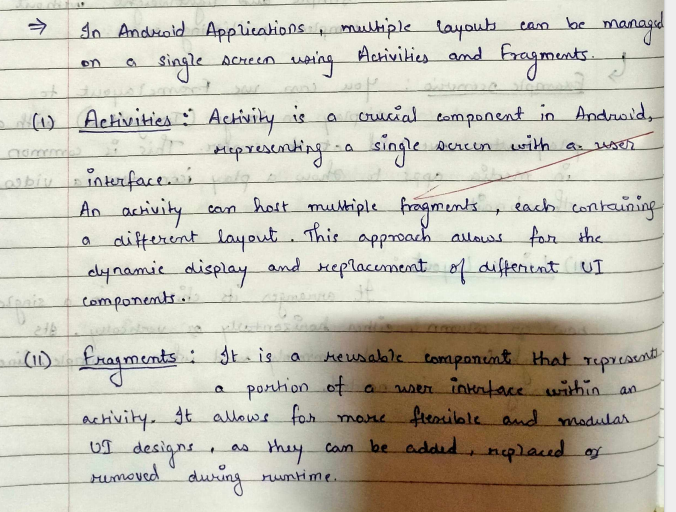
android:text="25" />

</TableRow>

</TableLayout>

1. **How can multiple layouts be managed on a single screen in an Android application? Discuss the use of Fragment and Activity, and provide an example where multiple layouts are used effectively.( also refer chatgpt ans)**

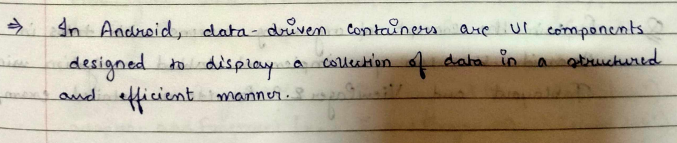
Ans :-



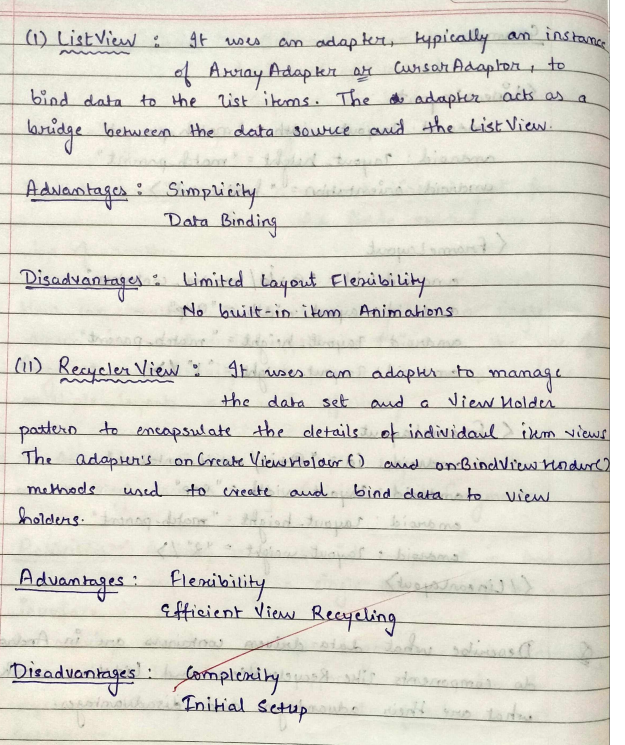


1. **Describe what data-driven containers are in Android. How do components like RecyclerView and ListView work, and what are their advantages and disadvantages?**

Ans :-

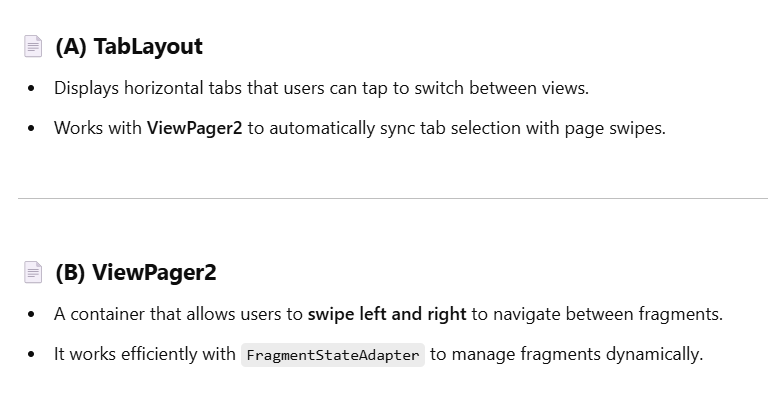


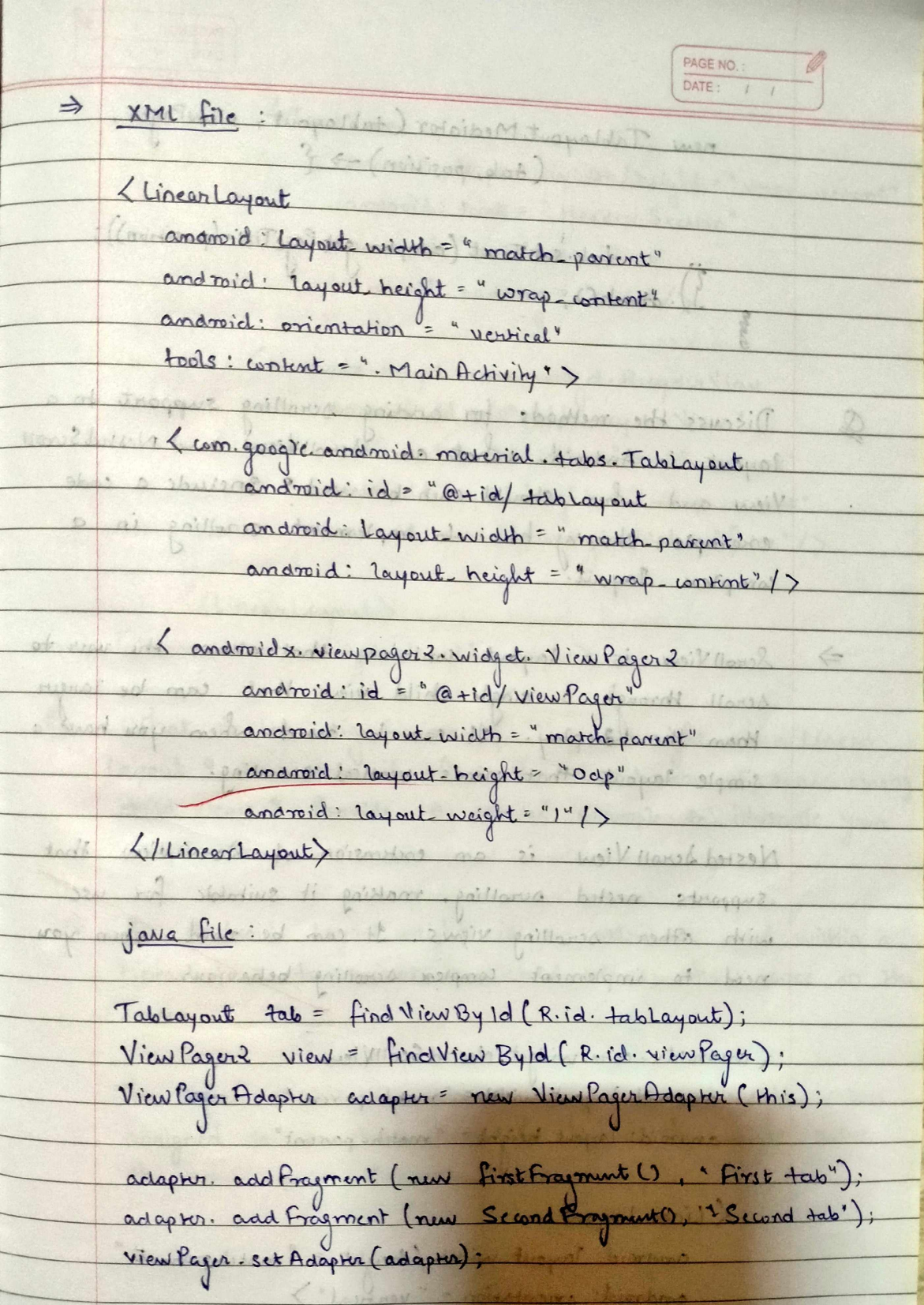


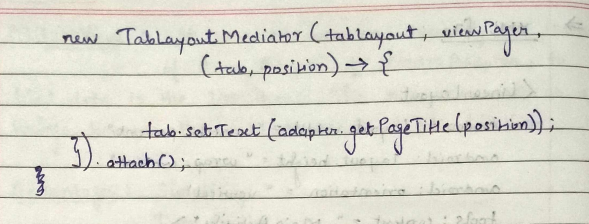


1. **Explain the concept of organizing screens with tabs in Android. How can you implement tabbed navigation using TabLayout and ViewPager2? Provide a detailed example.**

Ans :-





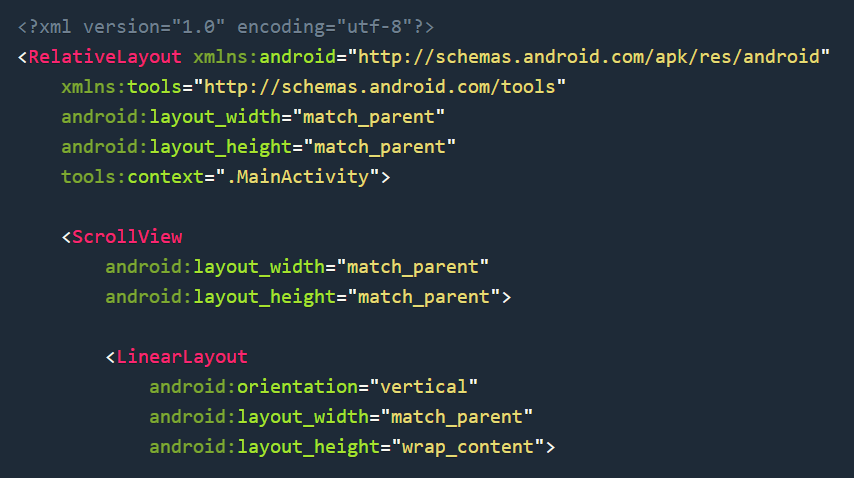


1. **Discuss the methods for adding scrolling support to a layout in Android. What are ScrollView and NestedScrollView, and when should each be used? Include a code example showing how to implement scrolling in a complex layout.**

Ans:-



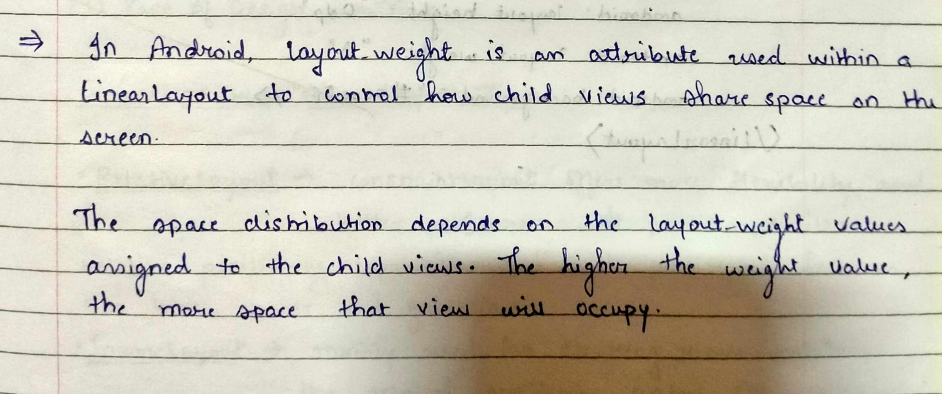
**Example:**



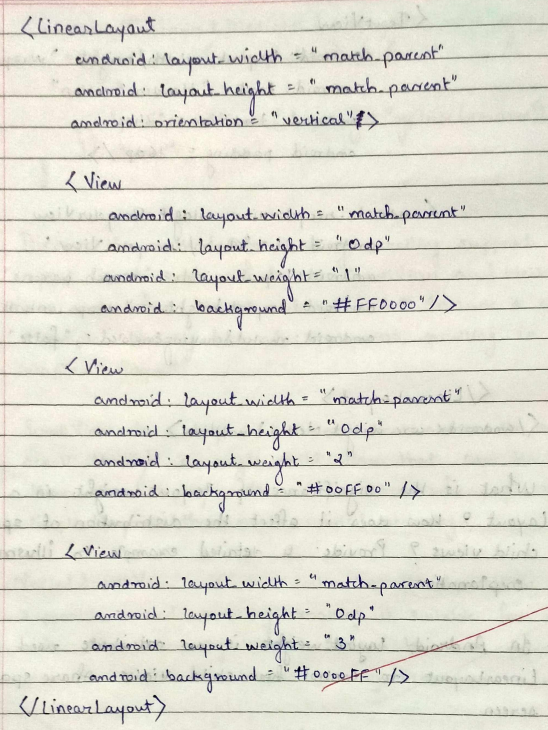


1. **What is the significance of layout\_weight in a LinearLayout? How does it affect the distribution of space among child views? Provide a detailed example to illustrate your explanation.**

Ans :-

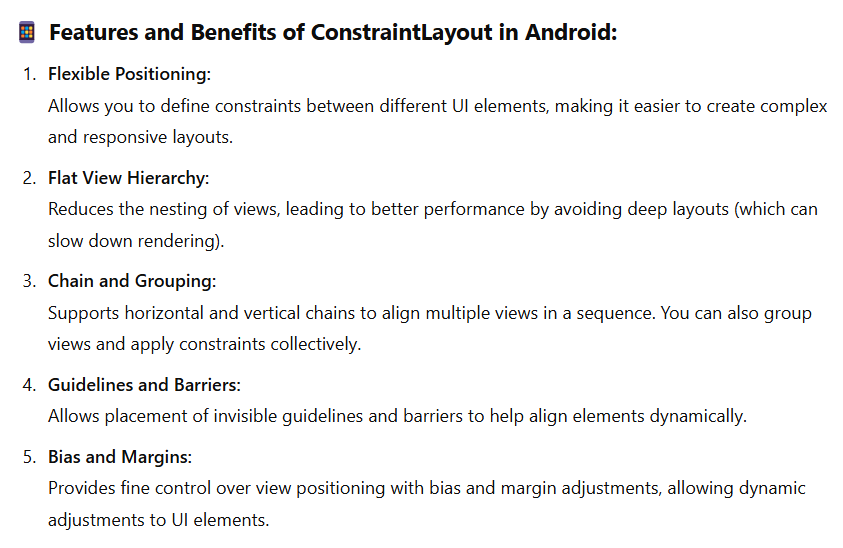


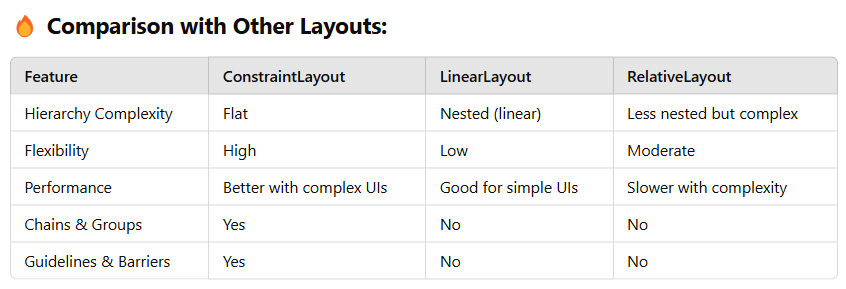
**Example:**



1. **Describe the features and benefits of using ConstraintLayout in Android. How does it compare to other layout types in terms of flexibility and performance? Provide an example of a complex UI layout that would be best suited for ConstraintLayout.**

Ans:-





**Example:**

<androidx.constraintlayout.widget.ConstraintLayout

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

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

android:layout\_width="match\_parent"

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

<ImageView

android:id="@+id/profile\_pic"

android:layout\_width="100dp"

android:layout\_height="100dp"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

android:src="@drawable/profile" />

<TextView

android:id="@+id/user\_name"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:layout\_constraintStart\_toEndOf="@id/profile\_pic"

app:layout\_constraintTop\_toTopOf="@id/profile\_pic"

android:text="User Name"

android:textSize="18sp" />

<Button

android:id="@+id/btn\_follow"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@id/profile\_pic"

app:layout\_constraintStart\_toStartOf="@id/profile\_pic"

android:text="Follow" />

</androidx.constraintlayout.widget.ConstraintLayout>

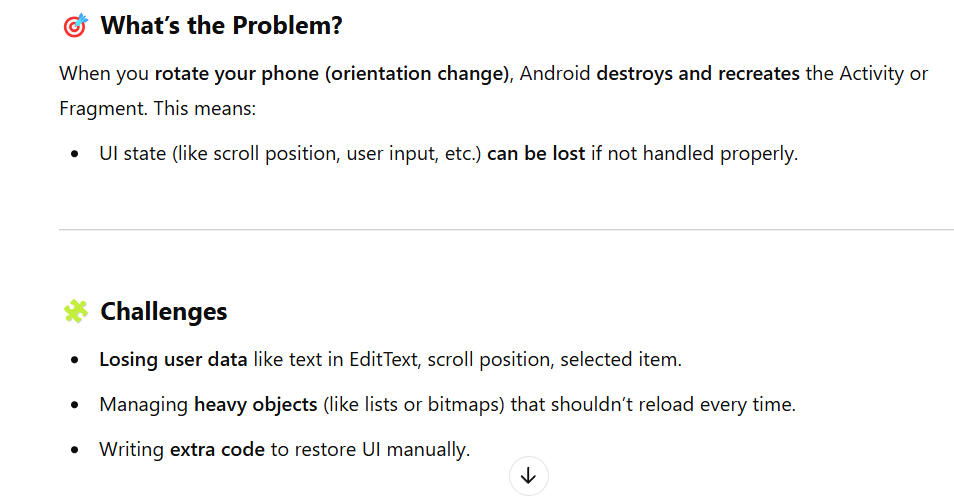
**Best for:**

-> Dynamic layouts with multiple UI elements.

-> Adapting UI for different screen sizes with ease.

1. **How can you manage and maintain UI state across orientation changes in Android? Discuss the challenges involved and provide solutions, including the use of onSaveInstanceState, ViewModel, and other techniques.**

Ans:-



Other challenges: 1) loss UI state, 2) loss data, 3)perfrom asynchronous task

