**UNIT-III**

**Android User Interface Design Essentials:** User Interface Screen elements, Designing User Interfaces with Layouts, Drawing and Working with Animation. Layouts, Recycler View, List View, Grid View and Web view

**Input Controls**: Buttons, Checkboxes, Radio Buttons, Toggle Buttons, Spinners, Input Events, Menus, Toast, Dialogs, Styles and Themes, Creating lists, and Custom lists

* **User Interface (UI) Screen Elements**

In Android development, the **User Interface (UI) Screen Elements** are the building blocks that help design interactive layouts for apps. These elements are primarily composed of **View** and **ViewGroup** components. Below are the key UI elements:

**I. Basic UI Elements**

* **TextView** → Displays text to the user.
* **EditText** → Input field for user text input (e.g., username, password).
* **Button** → Clickable element that performs actions.
* **ImageView** → Displays images.
* **ProgressBar** → Shows loading progress.
* **CheckBox** → Allows selection of multiple options.
* **RadioButton & RadioGroup** → Allows selection of a single option from a group.
* **Switch** → A toggle button for ON/OFF states.

**II. Layouts (ViewGroups)**

These define how UI elements are arranged:

* **LinearLayout** → Arranges elements in a single row or column.
* **RelativeLayout** → Positions elements relative to each other.
* **ConstraintLayout** → A flexible layout that uses constraints for positioning.
* **FrameLayout** → Holds a single child view, useful for overlays.
* **TableLayout** → Organizes elements in rows and columns.
* **ScrollView** → Enables vertical scrolling.

**III. UI Containers**

* **Dialog** → Shows pop-up messages or input prompts.
* **PopupMenu** → Displays a small menu when a button is clicked.
* **DrawerLayout (Navigation Drawer)** → A side menu for navigation.
* **BottomSheet** → A modal or persistent UI component that slides up from the bottom.

**Basic UI Elements**

**View Components (Basic UI Elements) in Android**

View components are the essential building blocks of an Android app’s UI. They are responsible for displaying content and handling user interactions. Below are some of the most commonly used **View** components in Android development:

**1. TextView 📄**

* Used to display **static text** to the user.
* Cannot be edited by the user (unlike EditText).
* Supports text formatting, styling, and linking.

**Example:**

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello, World!"

android:textSize="18sp"

android:textColor="@android:color/black"/>

**2. EditText ✍**

* Allows the user to enter text (input field).
* Can be used for passwords, numbers, emails, etc.

**Example:**

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter your name"

android:inputType="text"/>

**3. Button 🔘**

* Used to perform an action when clicked.
* Supports different styles (e.g., default, elevated, outlined).

**Example:**

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Click Me"

android:onClick="handleClick"/>

*(In the Java/Kotlin file, define handleClick(View view) to handle button clicks.)*

**4. ImageView 🖼**

* Displays images or icons.
* Can load images from resources or URLs.

**Example:**

<ImageView

android:layout\_width="100dp"

android:layout\_height="100dp"

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

**5. CheckBox ✅**

* Allows users to select multiple options independently.

**Example:**

<CheckBox

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="I agree to the terms"/>

**6. RadioButton & RadioGroup 🎯**

* RadioButton allows users to choose only **one option** from a group.
* RadioGroup ensures that only one option is selected at a time.

**Example:**

<RadioGroup

android:layout\_width="wrap\_content"

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

<RadioButton

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Option 1"/>

<RadioButton

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Option 2"/>

</RadioGroup>

**7. ProgressBar ⏳**

* Used to indicate loading or processing progress.

**Example (Indeterminate Circular Progress):**

<ProgressBar

android:layout\_width="wrap\_content"

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

**Example (Determinate Horizontal Progress):**

<ProgressBar

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

style="@android:style/Widget.ProgressBar.Horizontal"

android:progress="50"

android:max="100"/>

**8. SeekBar 🎚**

* Allows users to **select a value** by sliding.

**Example:**

<SeekBar

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="100"/>

**Layouts (ViewGroups)**

These **layout containers** help you structure the user interface of your Android app. Here's an overview of how each one works and how to use them in **Java**:

**1. LinearLayout**

**LinearLayout** arranges child elements either in a row (horizontal) or in a column (vertical).

* **XML (LinearLayout Example)**:

<LinearLayout

android:id="@+id/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="First item" />

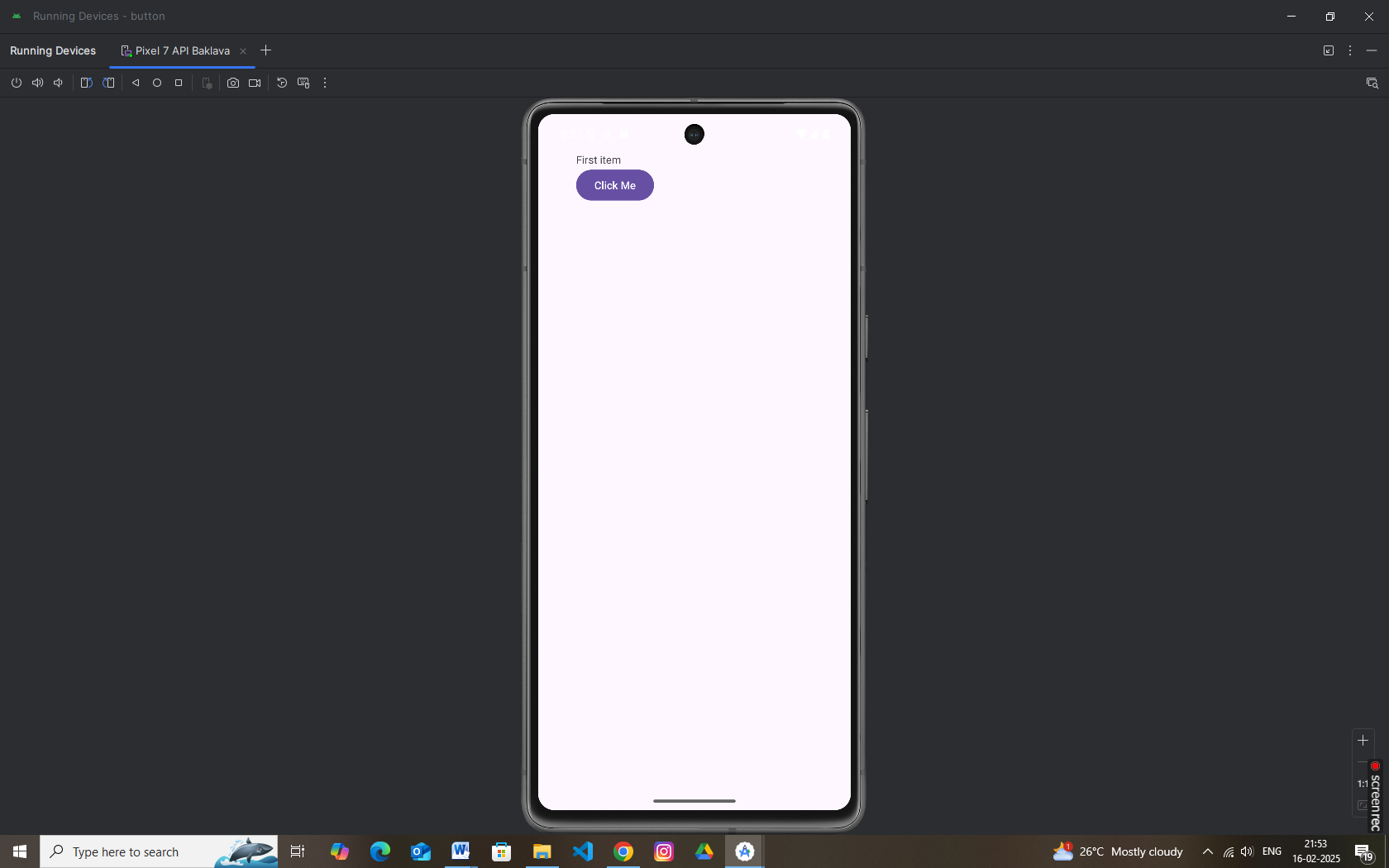
<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Click Me" />

</LinearLayout>



**2. RelativeLayout**

**RelativeLayout** positions elements relative to each other. You can position elements by specifying attributes like android:layout\_toEndOf, android:layout\_below, etc.

* **XML (RelativeLayout Example)**:

<RelativeLayout

android:id="@+id/relativeLayout"

android:layout\_width="match\_parent"

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

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Above button"

android:layout\_centerHorizontal="true" />

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

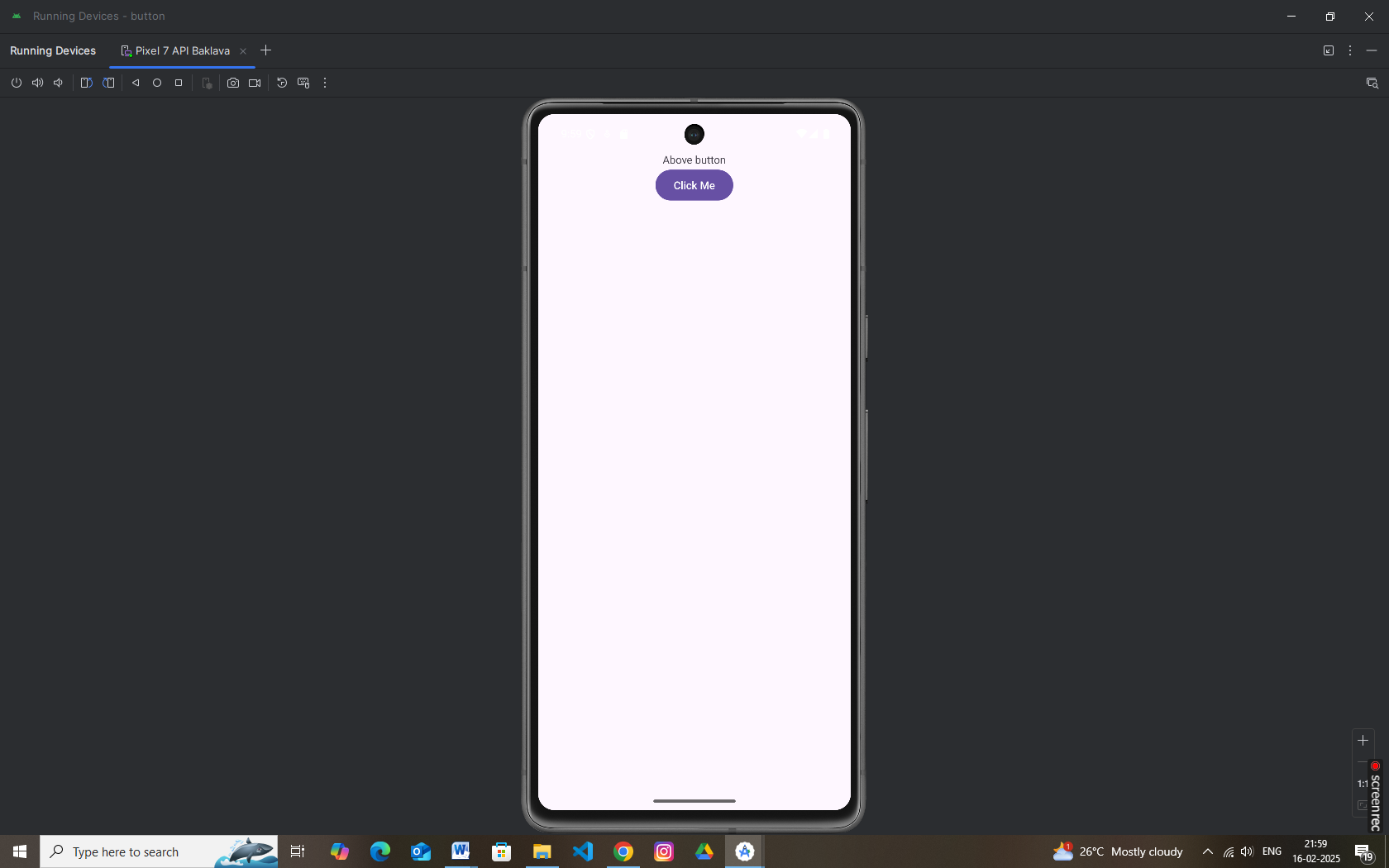
android:layout\_height="wrap\_content"

android:text="Click Me"

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

android:layout\_centerHorizontal="true" />

</RelativeLayout>



**3. ConstraintLayout**

**ConstraintLayout** is the most flexible layout container, allowing you to define complex layouts with flexible positioning and relationships between elements. It's recommended for modern UI design.

* **XML (ConstraintLayout Example)**:

<androidx.constraintlayout.widget.ConstraintLayout

android:id="@+id/constraintLayout"

android:layout\_width="match\_parent"

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

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello, world!"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintStart\_toStartOf="parent" />

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

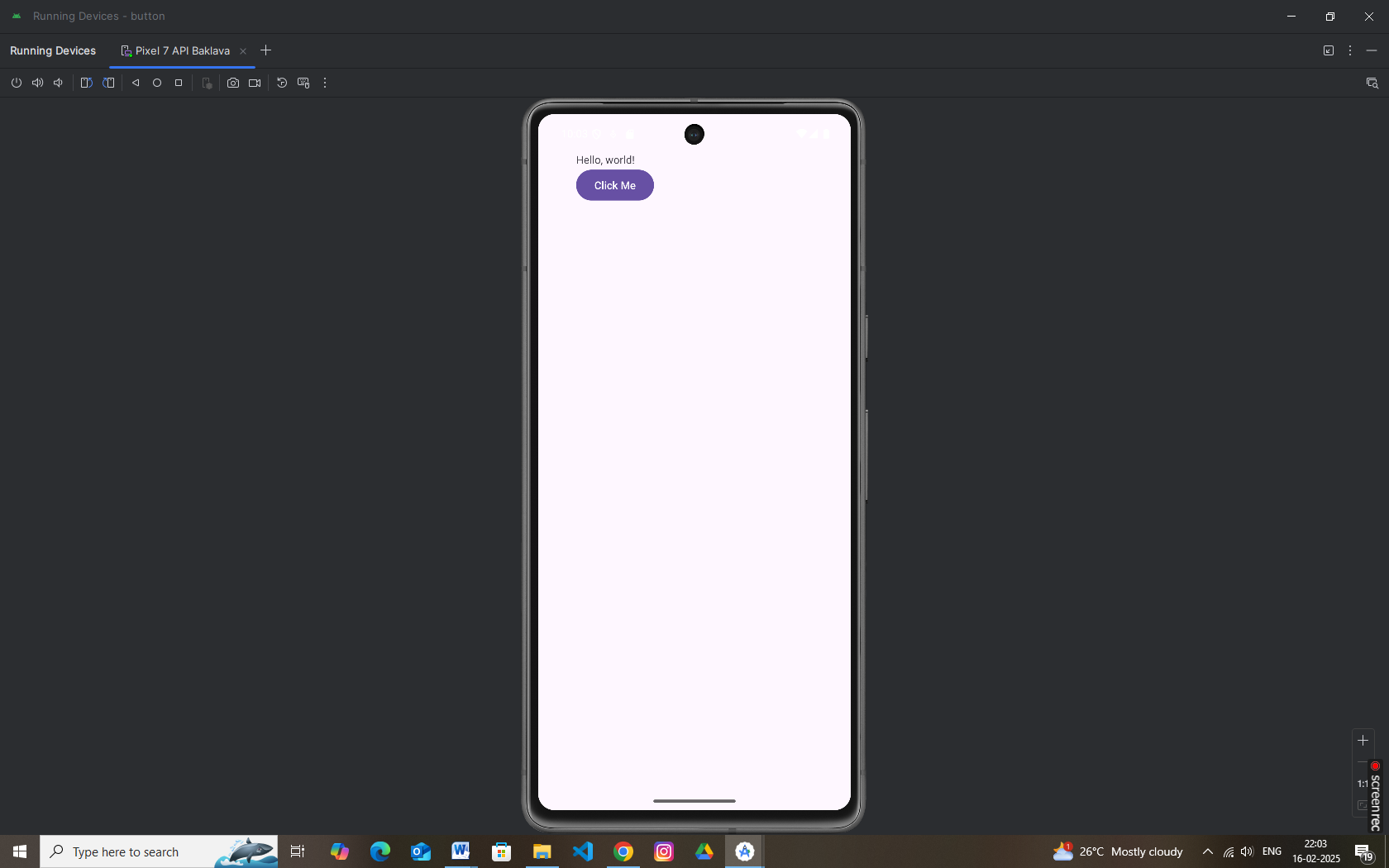
android:layout\_height="wrap\_content"

android:text="Click Me"

app:layout\_constraintTop\_toBottomOf="@id/textView"

app:layout\_constraintStart\_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>



**4. FrameLayout**

**FrameLayout** is used to stack child elements on top of each other. This layout is commonly used for displaying one element at a time (like showing a fragment or image).

* **XML (FrameLayout Example)**:

<FrameLayout

android:id="@+id/frameLayout"

android:layout\_width="match\_parent"

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

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="This is a text" />

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Click Me"

android:layout\_gravity="center" />

</FrameLayout>

