Designing Android UIs with XML and Jetpack Compose

Learn to craft beautiful, responsive Android app interfaces using both traditional XML layouts and modern Jetpack Compose.

This 4-hour course covers everything from basic UI elements to advanced layouts and event handling techniques.

by Australian Teachers

Basics of XML Layout: Views, Layouts, and Attributes

Views

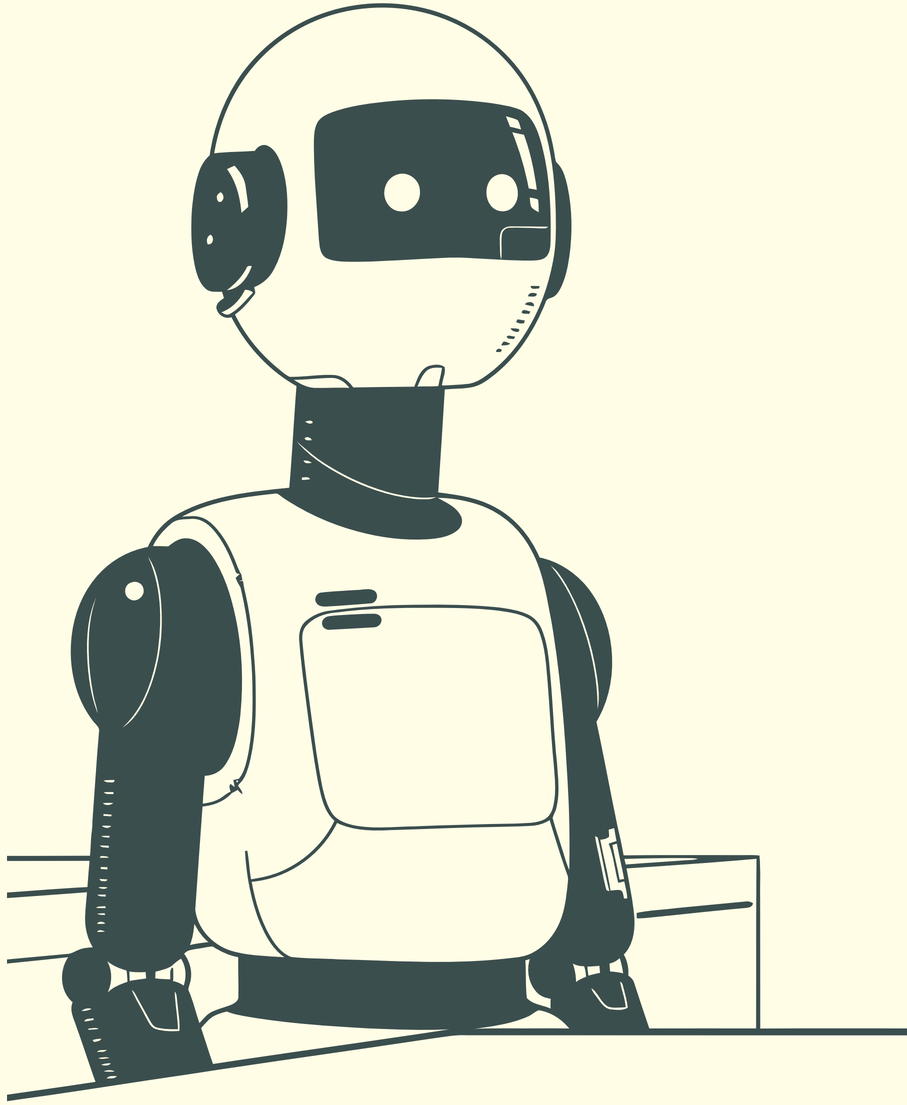
Basic building blocks like TextView, Button, and ImageView that display content and accept user interaction.

Layouts

Container elements such as RelativeLayout, LinearLayout, and GridLayout that organize views.

Attributes

Properties like layout_width, layout_height, and id that control appearance and behavior.

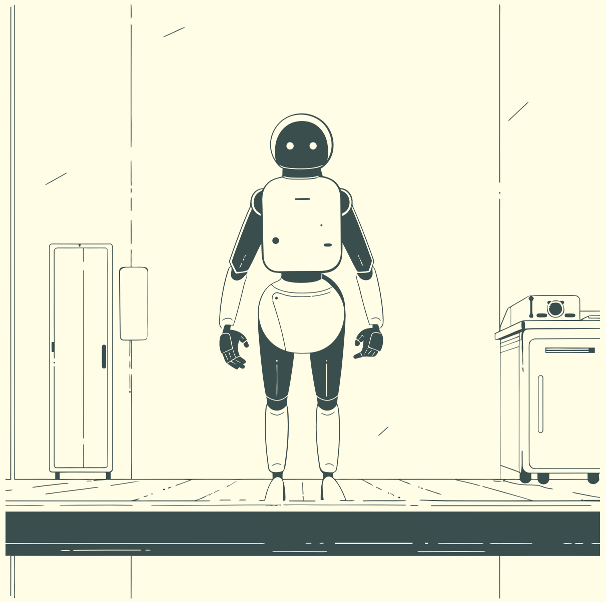


Australian Teachers Corporation

Common UI Elements: Button, TextView, ImageView, EditText

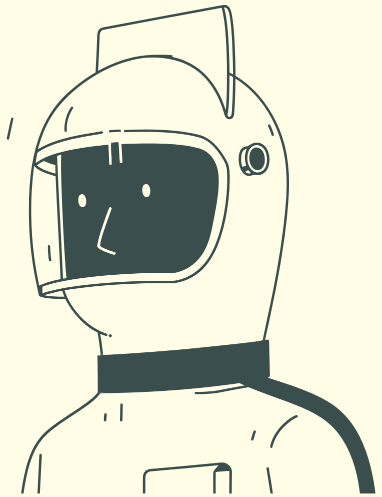
TextView

Displays text to users. Customize with different fonts, sizes, and styles.



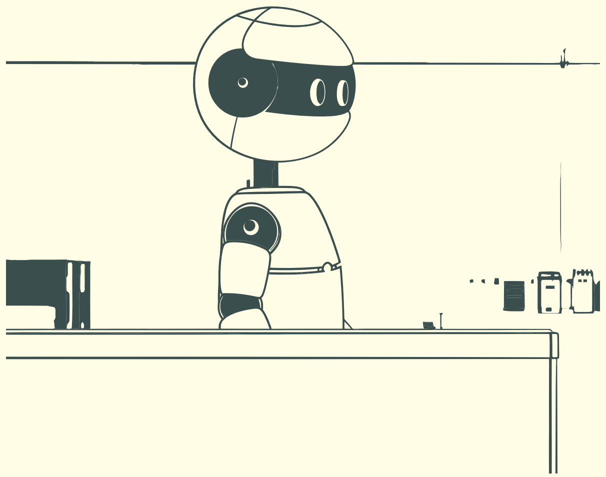
Button

Enables user interaction through taps. Can be styled with colors, shapes, and icons.

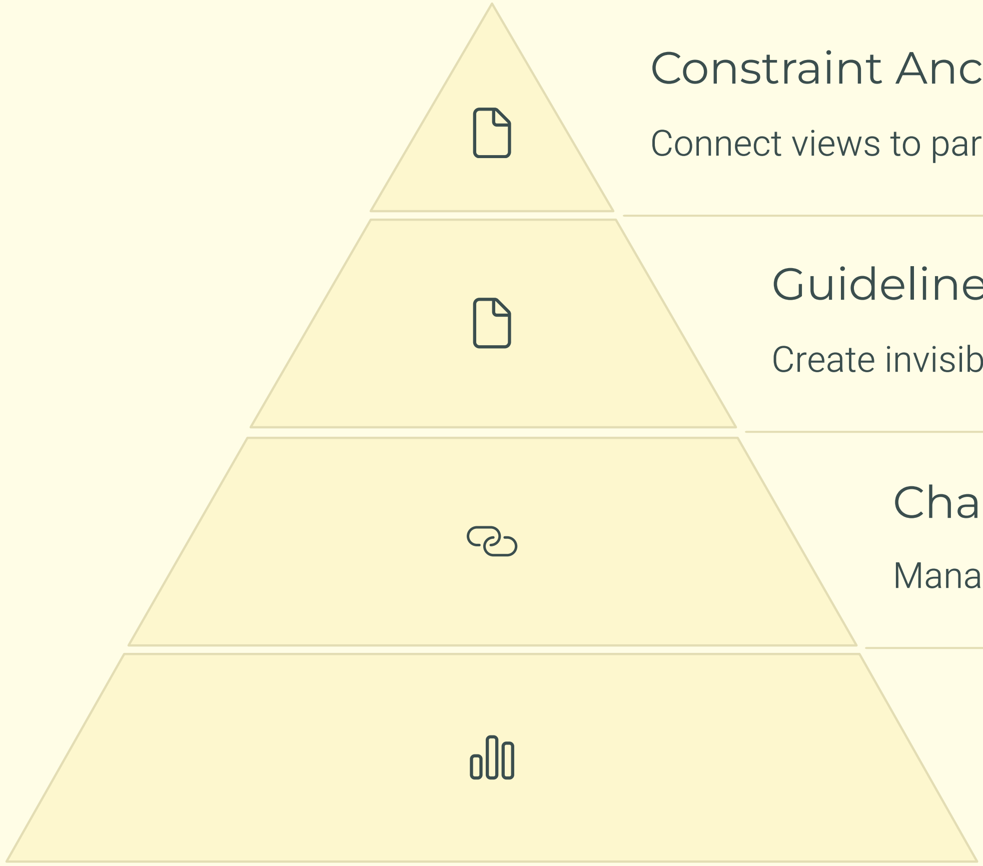


EditText

Accepts user input with various input types. Add validation for proper data entry.



Working with ConstraintLayout and Nested Layouts



Constraint Anchors

Connect views to parent or siblings

Guidelines & Barriers

Create invisible alignment helpers

Chains & Groups

Manage collections of elements

Optimization Techniques

Reduce nesting depth for better performance

Introduction to Jetpack Compose

Declarative UI

Define what your UI should look like, not how to build it.

Composable Functions

Build UIs with reusable function components.

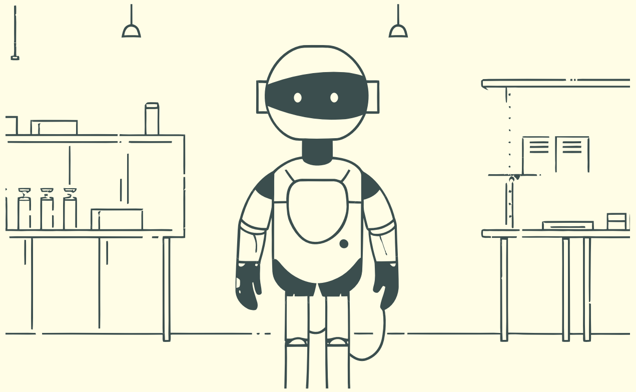
State Management

UI automatically updates when state changes.

Live Preview

See changes without running the app.

Implementing View Binding and Event Handling

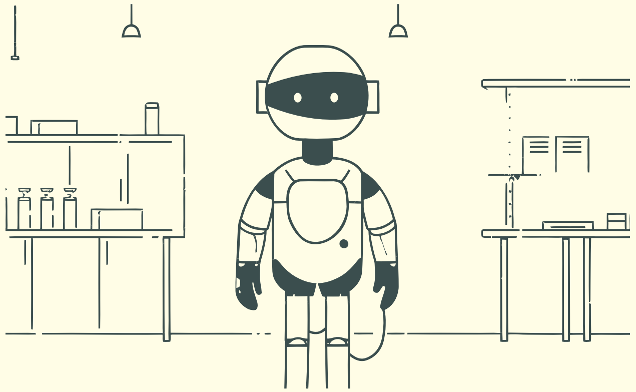


View Binding

Type-safe way to access views without findViewById(). Eliminates null pointer exceptions and improves code readability.

Click Listeners

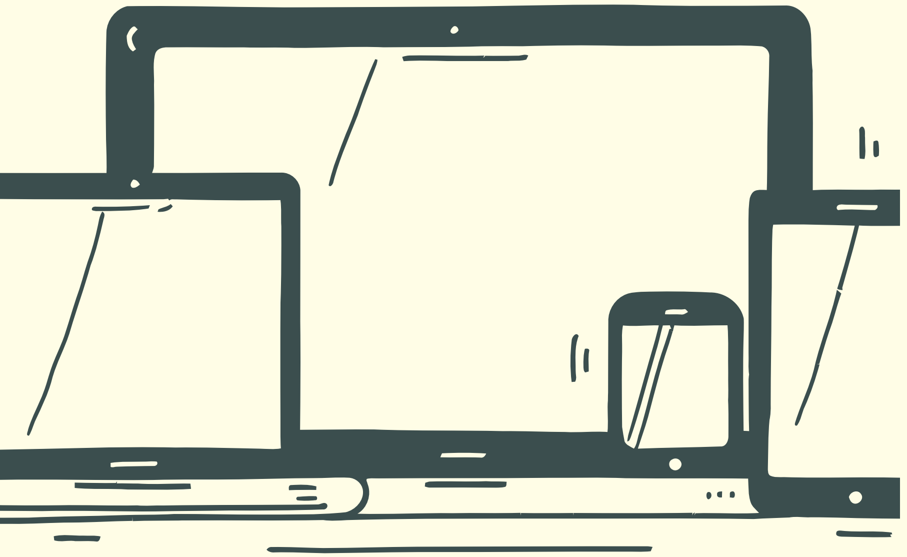
Respond to user interactions like taps and long presses. Implement as lambda functions for cleaner code.



State Observers

Track and respond to changes in your app's data. Use LiveData or State in Compose for reactive UIs.

Challenges and Solutions in Android UI Design



Challenge	Solution
Screen Size Diversity	Use ConstraintLayout and density-independent pixels (dp)
Orientation Changes	Create separate layouts or use responsive constraints
Dark/Light Themes	Implement theme