Responsive Design

1. Introduction to Responsive Design in Android (slides 2)

Apps run on a wide variety of devices—from smart watches to phone to tablet. To provide a great user experience on all types of devices, app's UI needs to different display sizes and configurations. App's should be able to occupy available screen spaces, handle changes as runtime, such as change in orientation, resizing in split screen.

Jetpack compose as a modern toolkit for UI development with native support for responsive design.

2. Jetpack Compose Layouts: Adapting to Different Screen Sizes (slide 3-4)

Key Terminology:

Before diving into how Jetpack Compose adapts to different screen sizes, it's essential to understand the structure of composables in an app:

- **App-level composable**: The root composable that takes up the entire space allocated to the app. It serves as the main container for all other composables. This composable is responsible for the overall structure and responsiveness of the app's UI.
- **Screen-level composable**: A composable that resides within the app-level composable. It occupies the full space available to the app, generally representing specific sections or screens within the app (e.g., Home screen, Settings screen). Each screen-level composable typically corresponds to a destination when navigating through the app.
- **Individual composables**: These include all other composables inside the screen-level composables. They can represent smaller UI elements (like buttons or text), reusable content groups, or other composables designed to form the detailed structure of each screen.

Applying These Concepts to Responsive Design:

- The **app-level composable** manages the overall layout structure and ensures the app adapts across all devices. For instance, it might contain a `Scaffold` composable to manage a flexible layout structure.
- **Screen-level composables** adjust their layouts based on the screen size and orientation. On a phone, a screen-level composable may show a single-column layout, while on a tablet, it may switch to a multi-column layout.
- **Individual composables** like `Text`, `Button`, and `Image` components automatically resize and re-arrange themselves to fit the available space, ensuring the UI remains functional and visually appealing on any screen size.

3. Adapting Layouts with Window Size Classes in Jetpack Compose (slide 5-7)

As we look at how Jetpack Compose can handle different screen sizes, one powerful tool is **Window Size Classes**, which provide an efficient way to categorize devices and adjust layouts accordingly. This allows us to build UIs that feel natural across a wide range of devices, from phones to tablets and even foldables.

What are Window Size Classes?

Window Size Classes are predefined categories that Jetpack Compose uses to group screens into three main types:

- **Compact**: Screens with limited horizontal or vertical space, typically seen on mobile phones.
- **Medium**: Devices like small tablets or foldables in a semi-expanded state.
- **Expanded**: Large screens such as tablets or desktops, which provide a significant amount of space for multi-column layouts or more complex UI arrangements.

Why Use Window Size Classes?

- They simplify the process of creating adaptive layouts.

- Instead of manually checking the screen dimensions, you can rely on these predefined categories to handle layout shifts.
- This ensures your app is flexible and looks great on all device types without requiring specific layout adjustments for each screen size.