Ramos, Zherge R.

BSIT-3D

ITELECT

ANDROID UI BASICS

Views (Individual UI Components)

-Views are the smallest building blocks of the Android interface.

- **View -** The parent class of all UI components. Every visible element on the screen (buttons, text, images) is a View.
- **EditText -** A text input field that allows the user to type information such as usernames, passwords, notes, etc.
- **Spinner -** A dropdown UI element used for selecting one item from a list (e.g., gender, category).
- **ListView -** Displays a vertically scrollable list of items. Common in older Android apps.
- **CheckBox -** Allows users to select multiple options from a list. Useful for settings and multi-select questions.
- **RecyclerView -** A more advanced and efficient replacement for ListView. It displays large or dynamic lists while saving memory.

VIEWGROUPS & LAYOUT SYSTEM

-ViewGroups act like invisible "**containers**" that hold and arrange UI elements (Views).

ViewGroup

- **Acts as an invisible container**

  -It organizes child Views and controls how they appear on the screen.

Common Layout Types

- **LinearLayout -** Arranges child views in a single direction—either **horizontal** or **vertical**. Simple and easy to structure.
- **RelativeLayout -** Positions views relative to the parent or other views (e.g., “to the right of this button”).
- **ConstraintLayout -** A modern and very flexible layout.
  Allows you to position views using **constraints** such as “aligned to parent start,” “centered horizontally,” etc.

Layout Concepts

- **Views, ViewGroups, layout managers -** These define *what* appears on the screen and *how* it is arranged.
- **dp (density-independent pixels) -** A unit of measurement that keeps UI size consistent across different screen densities (small, medium, large, tablets).
- **Position and size views -** Done through margins, padding, constraints, and gravity. Controls how close or far elements are from each other.
- **Tailor layout to the device -** The UI should adapt to different screen sizes (phones, tablets), orientations (portrait/landscape), and densities.

RECYCLERVIEW SYSTEM (MODERN LIST HANDLING)

-RecyclerView is designed to handle long lists efficiently.

- **Recycles views -** Instead of creating a new item view every time, it **reuses old ones**, improving performance.
- **Adapter**
  The bridge between your data and your UI. It decides:
  - how many items to show
  - what each item looks like
  - how to bind data to each item

- **ViewHolder -** Holds references to the views inside each list item (e.g., TextView, ImageView).
  Improves speed by avoiding repeated **findViewById** calls.
- **Within the Adapter -** You create ViewHolders, inflate item layouts, and bind data to each item.

MENUS & THE TOOLBAR

- **Toolbar -** The top app bar where you can place titles, icons, and menu options.
- **Overflow menu -** The three-dot button that appears when there isn't enough space on the toolbar to show all actions.
- **res/menu/ -** Folder that contains XML files defining menus (toolbar menu, options menu, etc.).
- **app:showAsAction**
  Determines whether a menu item:
  - Always shows on the toolbar
  - Only shows when there is space
  - Always stays in the overflow menu

FEEDBACK: TOAST NOTIFICATIONS

- **Quick, temporary popup feedback -** A Toast gives simple messages at the bottom of the screen (e.g., "Saved successfully!").
- **Toast.makeText -** Method used to create a Toast.
- **About 3.5 seconds -** The duration of Toast.LENGTH_LONG.
- **Current context -** Required when showing a Toast (e.g., this, **MainActivity.this**).
  It tells the system *which screen* the Toast belongs to.

APPLICATION ARCHITECTURE

- **Application Layer -** Where the logic, UI code, activities, and app behavior are written. Developers build the whole user experience here.
- **onCreate() -** Called when the Activity starts.

- **Used for:**
  - Setting up the layout
  - Initializing variables
  - Linking UI elements
  - Starting listeners

TESTING & DEVICES

- **Android Virtual Device (AVD) -** A software-based emulator that simulates different Android devices.
- **Early-stage development and testing -** AVDs are easiest for testing UI, basic features, and layouts before using real devices.
- **Physical devices perform better**
  Provide:
  - Real performance
  - Accurate animations
  - Genuine gesture behavior
  - Proper hardware response
- **GPS, camera, sensors -** Only reliable on real devices; emulators cannot fully simulate them.
- **Test on different device configurations -** Ensures the app works across various screen sizes, hardware specs, and Android versions.

DATA STORAGE

Shared Preferences

- Stores small data (key-value pairs).
  Examples: dark mode ON/OFF, username, login status.

SQLite Database

- A built-in relational database in Android used for structured data (tables, columns, queries).

Important components:

- **Create helper class extending SQLiteOpenHelper**

  - Manages database creation and versioning.
- **Factory for database connections**

  -SQLiteOpenHelper provides readable and writable database instances.
- **Database version changes**

  -When the database version number increases, it triggers **onUpgrade()** to modify tables or update schema.

Internal & External Storage

- **Internal Storage** – Private to your app; safe and secure.
- **External Storage** – Public files like downloads, images, and documents.

PROGRAMMING FUNDAMENTALS

- **Compressed binary format -** Used to save data in smaller, more efficient form.
- **Code reusability -** Important for making cleaner, shorter, and more maintainable code.
- **Boolean flag -** A true/false value used for conditions (e.g., **isLoggedIn = true**).

- **Prevents typos, easier to maintain -** Using constants instead of repeating strings reduces errors.
- **Unique identifier (ID) -** Assigned to each View so it can be accessed programmatically using **findViewById()**.