Program No:	**1**
Roll No :	**1545**
Title of Program :	**Android Architecture**
Objective :	**1. Layers 2. Component 3. Architecture**

Android architecture is a robust, layered stack that supports seamless app development, hardware integration, and system reliability. The architecture is organized into multiple layers and components, ensuring modularity, security, and efficiency for mobile devices.

1. Main Layers of Android Architecture

Layer	Description
Application Layer	User-installed and pre-installed (system) apps that users interact with
Application Framework	APIs and services for app developers (e.g., Activity Manager, Notification)
Android Runtime & Libraries	Core libraries and the Android Runtime (ART) or Dalvik VM
Hardware Abstraction Layer (HAL)	Interface between hardware and higher-level APIs
Linux Kernel	Core system layer, managing hardware and security

a. Application Layer

- **Topmost layer** where all Android apps operate.
- Includes apps like Phone, Contacts, SMS, Camera, and all user-installed applications.
- Directly interacts with the user.

b. Application Framework Layer

- Provides APIs and high-level services to app developers.
- Contains managers (Activity Manager, Window Manager, Resource Manager, etc.) and system services that apps use for UI, notifications, data handling, telephony, and resource management.

c. Android Runtime & Native Libraries

- **Android Runtime (ART)** is the environment in which apps run, offering optimized execution through Ahead-of-Time (AOT) and Just-in-Time (JIT) compilation, plus garbage collection for memory management.
- **Core Libraries** provide Java-like interfaces and standard functionality for data structures, networking, graphics, database access, and more.
- **Native Libraries:** C/C++ libraries such as OpenGL (graphics), SQLite (databases), SSL (security), WebKit (web browser engine).

d. Hardware Abstraction Layer (HAL)

- Serves as a bridge between device hardware (camera, sensors, Bluetooth, audio) and the upper layers.
- Ensures Android runs on different hardware by abstracting device-specific commands for universal use

e. Linux Kernel Layer

- **Foundation of the architecture** that manages drivers (display, camera, Bluetooth, audio, memory), security, process management, power usage, and resource access.

- Implements security features and multitasking, and isolates app processes (sandboxing) for reliability

2. Key Components of Android Architecture

- **Applications:** All end-user apps, running in isolated processes.
- **Application Framework:** Resource management, lifecycle management, activity stack, content providers.
- **Android OS Runtime:** ART/Dalvik VM, core and Java libraries.
- **Native Libraries:** Graphics, media, database, web rendering.
- **Hardware Abstraction (HAL):** Connects the OS with device hardware.
- **Linux Kernel:** Device drivers, memory, process, and security management

3. Modern App Architecture (Internal App Layers)

Android apps themselves often adhere to an internal layered architecture for maintainability:

- **UI Layer:** Handles rendering and user interaction; uses Activities, Fragments, and ViewModel
- **Domain Layer:** Business logic, abstracting the app's core functionalities (may be absent in simpler apps).
- **Data Layer:** Manages data access, repositories, and data synchronization from sources like local databases or remote APIs

4. Android Architecture Patterns

Common application-level architecture patterns for Android include:

- **MVC (Model-View-Controller):** Separates UI logic, data, and control flow.
- **MVP (Model-View-Presenter):** Improves testability by decoupling UI from data and logic.
- **MVVM (Model-View-ViewModel):** Promotes reactive UIs and separation of concerns.
- **Clean Architecture & MVI:** Enforce strict boundaries and predictable flows for scalable, maintainable apps

Android's layered and modular architecture ensures efficient use of hardware, secure and isolated execution of apps, and consistency across a diverse Android device ecosystem. Understanding each layer's purpose and the interactions between components is essential for developing stable, high-quality Android applications