CASE STUDY ON ANDROID ARCHTECTURE

GROUP -7

Our Team

BHAVIKA GONDI
AM.EN.U4AIE22013

DVSS SWAPNITH
AM.EN.U4AIE22016

LEELA NARESH
AM.EN.U4AIE22030

Table of Content

INTRODUCTION TO ANDROID

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. It is an operating system for low powered devices that run on battery and are full of hardware like Global Positioning System (GPS) receivers, cameras, light and orientation sensors, Wi-Fi and LTE (4G telephony) connectivity and a touch screen. Like all operating systems, Android enables applications to make use of the hardware features through abstraction and provide a defined environment for applications

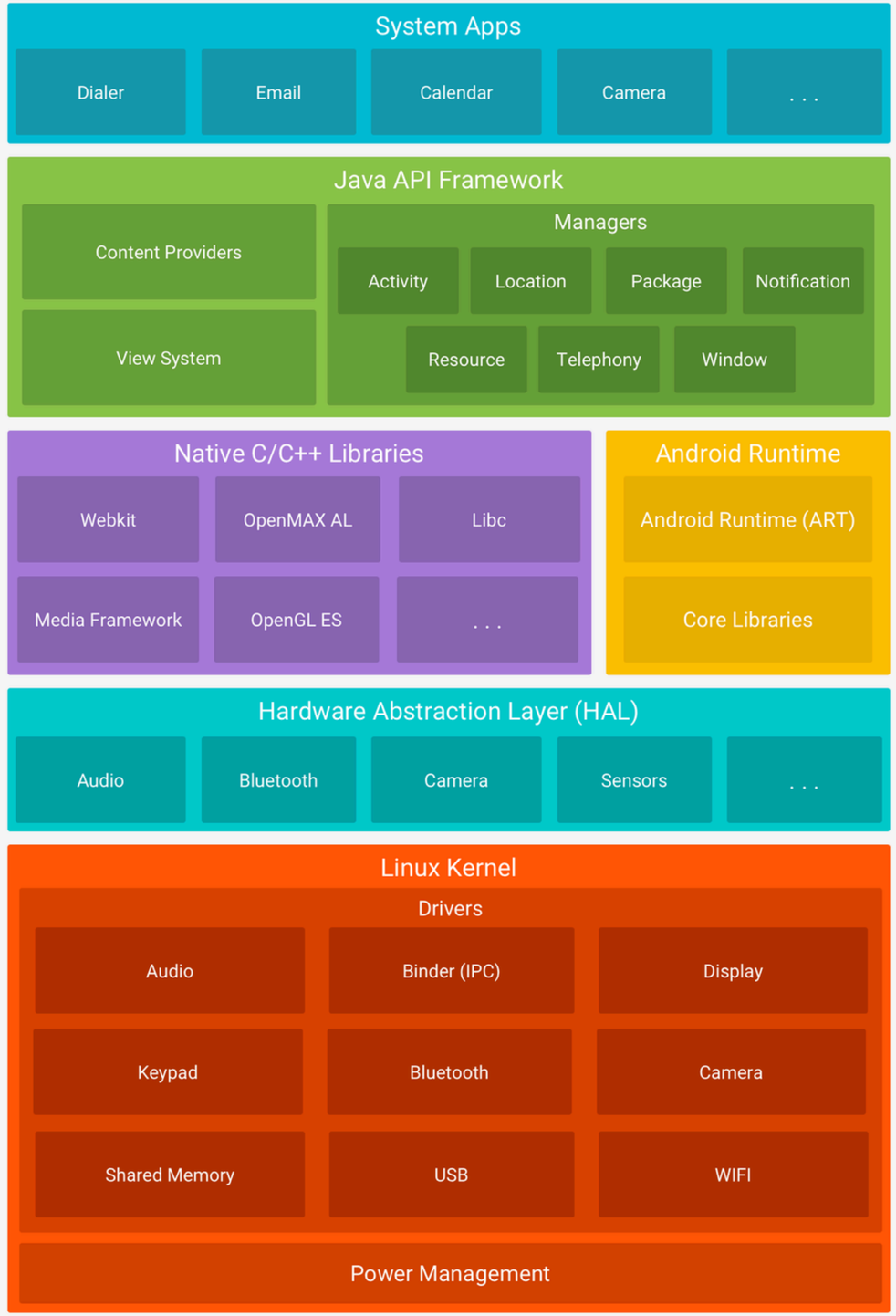
BACKGROUND & HISTORY

The platform was created by Android Inc. which was bought by Google and released as the Android Open Source Project (AOSP) in 2007. Since then, Android has seen numerous updates which have incrementally improved the operating system, adding new features and fixing bugs in previous releases. Each major release is named in alphabetical order after a dessert or sugary treat, with the first few Android versions being called "Cupcake", "Donut", "Eclair", and "Froyo", in that order. During its announcement of Android KitKat in 2013, Google explained that "Since these devices make our lives so sweet, each Android version is named after a dessert". This changed after android pie.

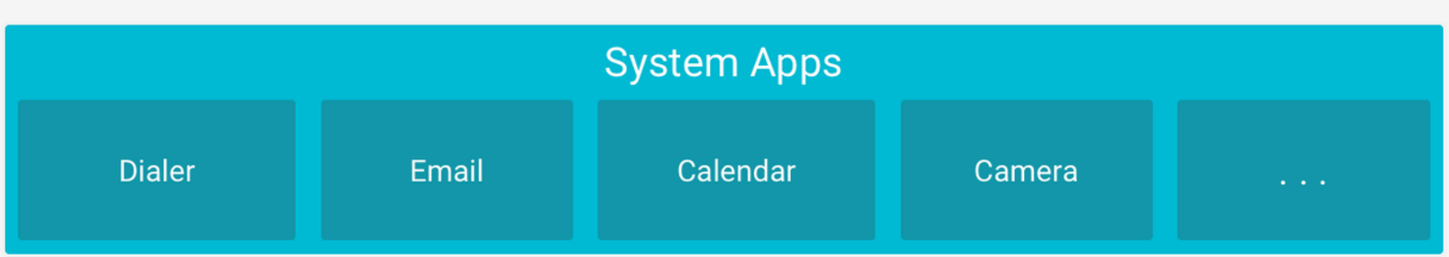
OVERVIEW OF ANDROID ARCHITECTURE

Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework services. Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application. The main components of android architecture are following: -

1. Application Layer
2. Application Framework Layer
3. Android Runtime Layer
4. Hardware Abstraction Layer
5. Linux Kernel Layer

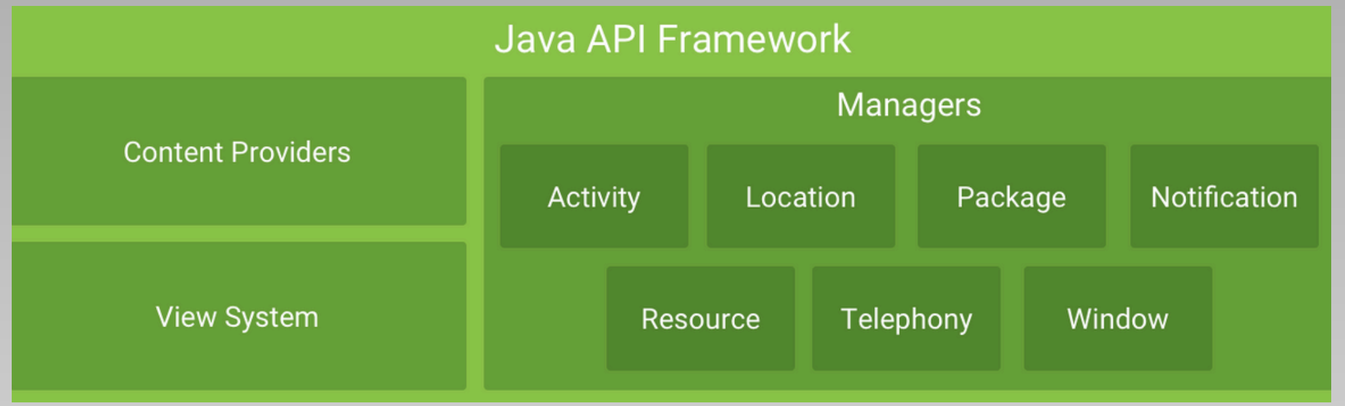


APPLICATION LAYER



- This is the layer that end-users interact with. It is on this layer where application developers publish their applications to run.
- Application Layer is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only. It runs within the Android run time with the help of the classes and services provided by the application framework.

APPLICATION FRAMEWORK LAYER



- The Application framework layer provides developers with the API's needed to develop applications this layer is written in java and it can be also called java framework.
- The android applications that can call the API's provided by this particular layer let's look at the main components such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

Content Providers:

- Content Providers act as a bridge for sharing data between applications on an Android device. They offer a secure and standardized way to access and manage data from various sources, like: SQLite Databases, Storage Mechanism, Sharing Data.

View System:

- The View System is the backbone of creating user interfaces (UIs) in Android applications. It provides a set of building blocks called Views that can be arranged and customized to create the visual elements you see on the screen, like buttons, text fields, and images.

MANAGERS

Activity Manager

Manage the life cycle of each application and the usual navigation fallback function.

Location Manager

Provide the geographic location and positioning function services.

Package Manager

Manage all applications installed in the Android system.

Notification Manager

Allows the application to display custom prompt information in the status bar.

Resource Manager

Provide various non-code resources the application uses, such as localized strings, pictures. layout files. color files, etc.

Telephony Manager

Manage all mobile device functions.

Window Manager

Manage all open windows.

ANDROID RUNTIME LAYER

ART is responsible for the execution and management of application code on Android devices.

Importance:

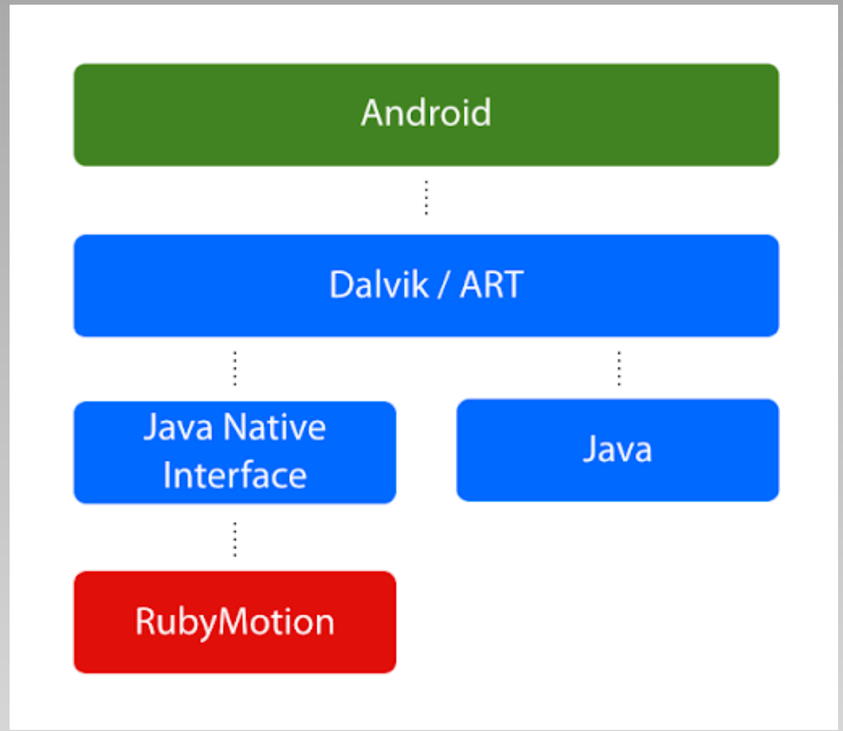
1. ART contributes to the overall efficiency of the Android system by optimizing code execution and memory management.

2. Efficient code execution and reduced runtime overhead improve battery life.

3. Developers benefit from improved runtime performance without significant modifications to their code.

4. Core Libraries ensure cross-device compatibility for Android applications.

COMPONENTS



1. ART/DVM (Dalvik Virtual Machine):

- **ART (Android Runtime):** Converts entire application bytecode into native machine code during the installation phase, reducing runtime overhead.
- **Dalvik Virtual Machine (DVM):**
  Older runtime used in Android versions before 5.0.
  Utilized Just-In-Time (JIT) compilation, converting bytecode into native code during runtime.

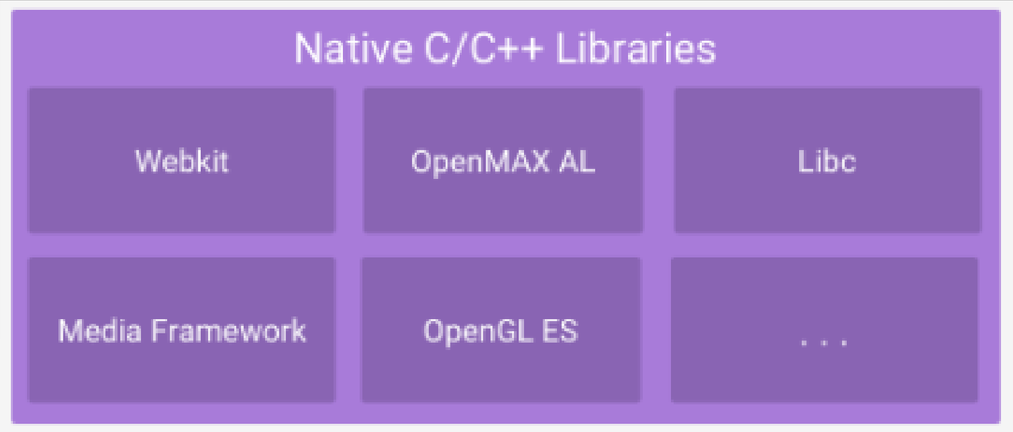
2. Core Libraries:

- **Java API Framework:**
  Provides a set of reusable and extensible software components for application development.
  Includes UI controls, data storage, connectivity, and other essential functionalities.
- **Native C/C++ Libraries:**
  Written in C/C++ and includes libraries such as libc and SQLite.
  Facilitate low-level interactions, hardware access, and performance optimization.

NATIVE C/C++ LIBRARIES

Purpose: Native C/C++ libraries are used to write performance-critical parts of an application, as they often offer better performance compared to Java or Kotlin.

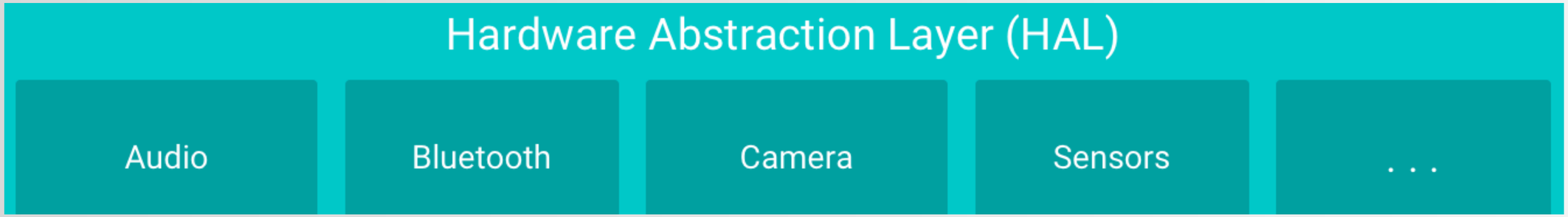
examples: libc, libm, libz, libjpeg/libpng, SQLite etc.



Used for: Graphics and Multimedia, Data Storage, System-Level Functionality, Porting Existing Code, and Performance-Critical Code.

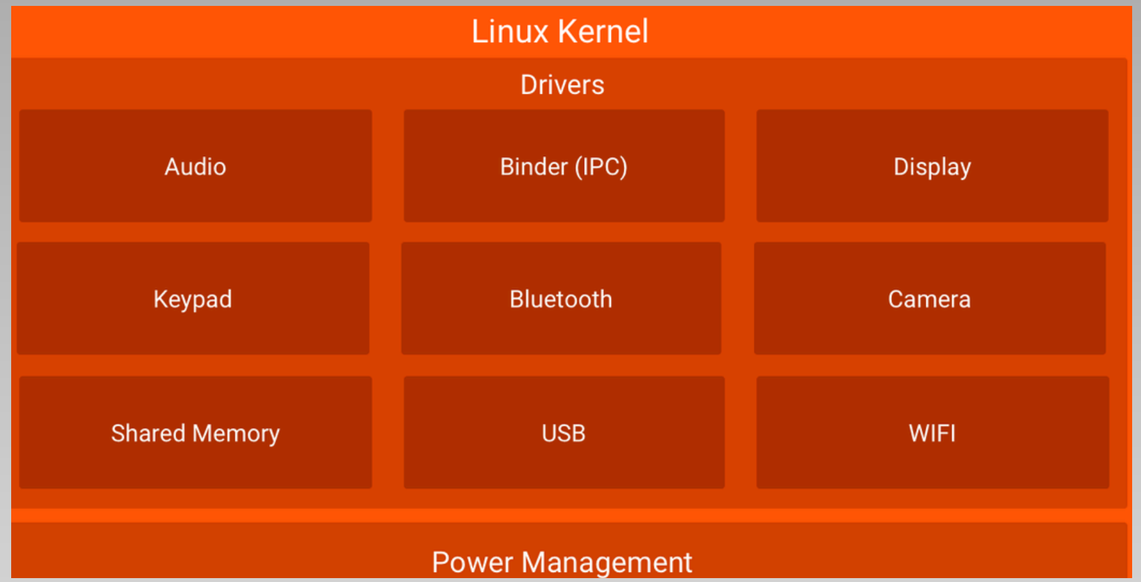
Using native C/C++ libraries in Android offers developers the flexibility to optimize performance, access hardware resources, and reuse existing codebases, but it comes with considerations related to compatibility, memory management, and security. Careful integration is essential for maximizing the benefits while addressing potential challenges.

HARDWARE ABSTRACTION LAYER (HAL)



- A hardware abstraction layer (HAL) is a logical division of code that serves as an abstraction layer between a computer's physical hardware and its software. It provides a device driver interface allowing a program to communicate with the hardware.
- The main purpose of a HAL is to conceal different hardware architectures from the OS by providing a uniform interface to the system peripherals.

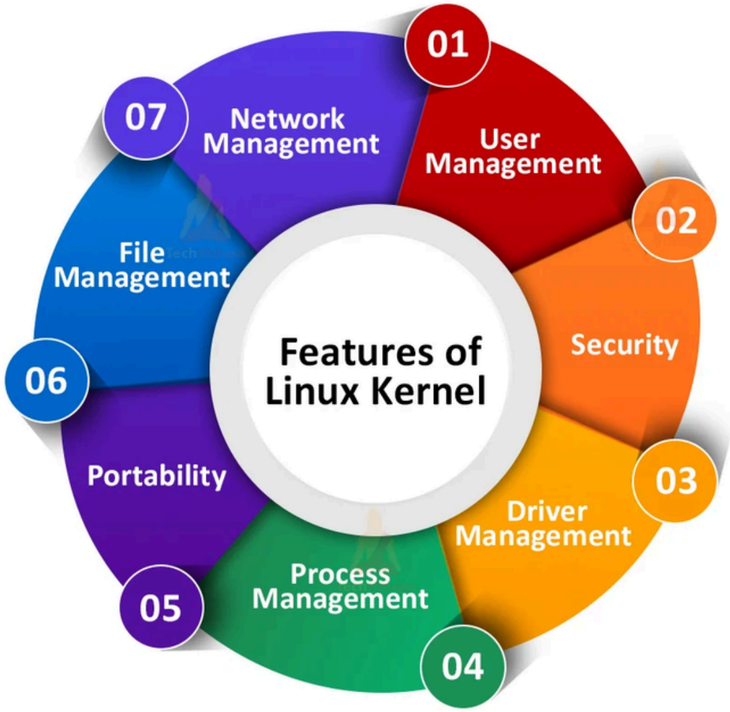
LINUX KERNAL LAYER



- The Linux kernel serves as the foundational layer in the Android architecture, providing essential functionalities such as hardware abstraction, device drivers, security features, and power management. It abstracts hardware, ensuring compatibility across diverse devices, and facilitates communication between software and hardware components. Additionally, its open-source nature allows for customization and community-driven development, ensuring ongoing updates and improvements for Android devices.
- The stability and robustness contribute to the overall reliability of the Android operating system, ensuring smooth performance and seamless user experience across a wide range of devices.
- Itsupport for multitasking and process management enables Android to run multiple applications concurrently, allowing users to seamlessly switch between tasks and multitask efficiently on their devices.

Core Components of Linux Kernel Layer in Android Architecture

- **User Management** ensures secure user interactions and access control.
- **Security** enforces system integrity and protects against unauthorized access.
- **Driver Management** facilitates communication with hardware components.
- **Process Management** handles process creation, scheduling, and resource allocation.
- **Portability** allows the kernel to run on diverse hardware platforms.
- **File Management** supports efficient storage and access control for data.
- **Network Management** ensures reliable communication between devices and networks.



CONCLUSION

The Android architecture presents a well-structured and layered framework that underlies the functionality of Android devices. The Linux kernel forms the core foundation, with the Hardware Abstraction Layer providing a standardized interface. The Android Runtime (ART) plays a pivotal role in executing and managing application code, employing both Ahead-of-Time (AOT) and Just-In-Time (JIT) compilation for optimized performance.

The integration of Java API Framework and Native C/C++ Libraries offers developers a versatile environment for building applications, balancing high-level functionalities with low-level optimizations. The system services, application framework, and user applications seamlessly interact, contributing to the overall efficiency and flexibility of the Android ecosystem.

The architecture's adaptability to diverse device types, coupled with the emphasis on performance and resource optimization, underscores its significance in powering a wide array of mobile and connected devices. As Android continues to evolve, its architecture remains a robust foundation for innovation and application development.

THANK YOU

Presented By : GROUP 7