Platform Architecture

Android is an open source, Linux-based software stack created for a wide array of devices and form factors. The following diagram shows the major components of the Android platform.

The Linux Kernel

The foundation of the Android platform is the Linux kernel. For example, the Android Runtime (ART) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management.

Using a Linux kernel allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel.

Hardware Abstraction Layer (HAL)

The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or bluetooth module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

Android Runtime

For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the Android Runtime (ART). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint. Build tools, such as d8, compile Java sources into DEX bytecode, which can run on the Android platform.

Some of the major features of ART include the following:

Ahead-of-time (AOT) and just-in-time (JIT) compilation

Optimized garbage collection (GC)

On Android 9 (API level 28) and higher, conversion of an app package's Dalvik Executable format (DEX) files to more compact machine code.

Better debugging support, including a dedicated sampling profiler, detailed diagnostic exceptions and crash reporting, and the ability to set watchpoints to monitor specific fields

Prior to Android version 5.0 (API level 21), Dalvik was the Android runtime. If your app runs well on ART, then it should work on Dalvik as well, but the reverse may not be true.

Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features, that the Java API framework uses.

Native C/C++ Libraries

Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps. For example, you can access OpenGL ES through the Android framework's Java OpenGL API to add support for drawing and manipulating 2D and 3D graphics in your app.

If you are developing an app that requires C or C++ code, you can use the Android NDK to access some of these native platform libraries directly from your native code.

Java API Framework

The entire feature-set of the Android OS is available to you through APIs written in the Java language. These APIs form the building blocks you need to create Android apps by simplifying the reuse of core, modular system components and services, which include the following:

A rich and extensible View System you can use to build an app's UI, including lists, grids, text boxes, buttons, and even an embeddable web browser

A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files

A Notification Manager that enables all apps to display custom alerts in the status bar

An Activity Manager that manages the lifecycle of apps and provides a common navigation back stack

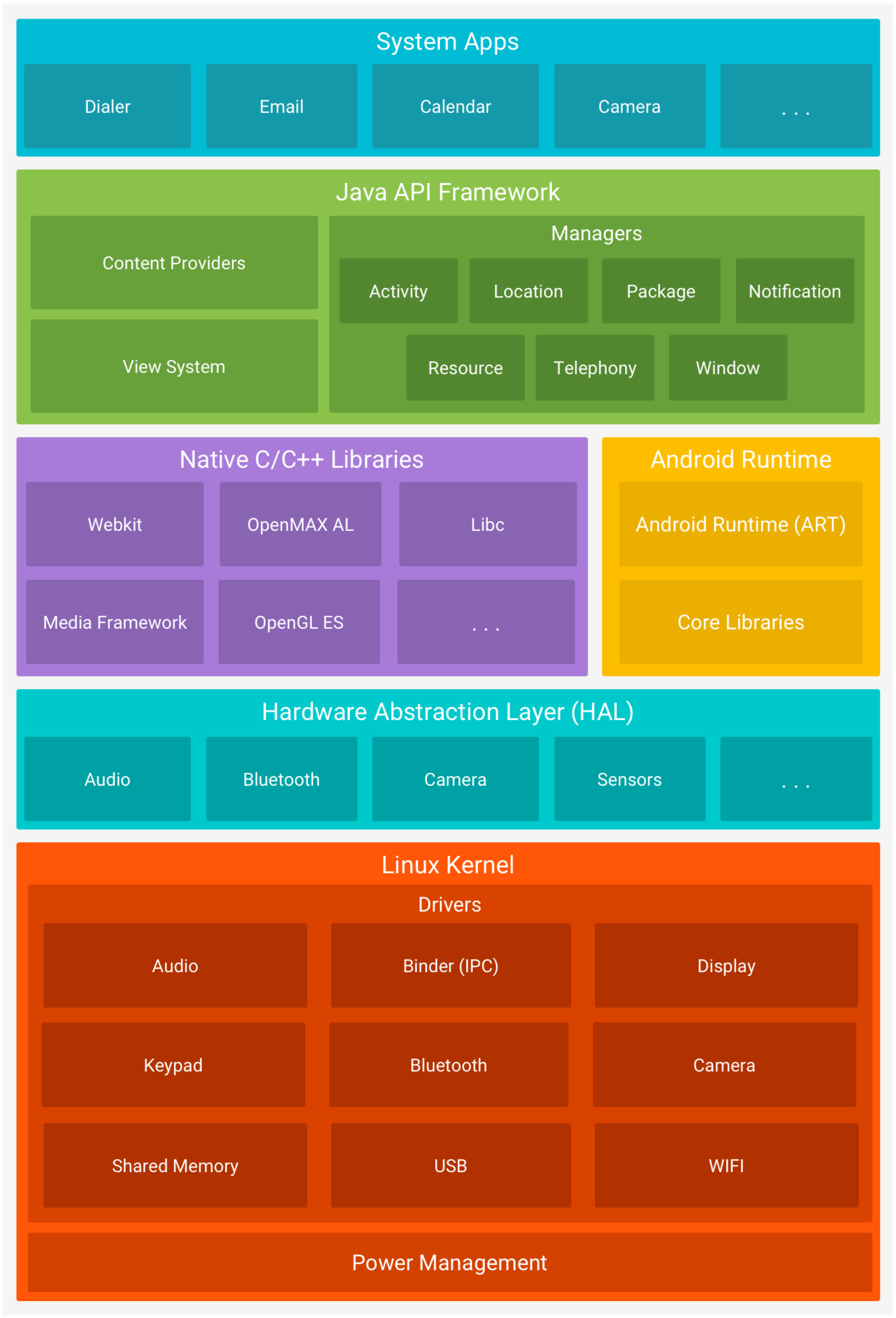
Content Providers that enable apps to access data from other apps, such as the Contacts app, or to share their own data

Developers have full access to the same framework APIs that Android system apps use.

System Apps

Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more. Apps included with the platform have no special status among the apps the user chooses to install. So a third-party app can become the user's default web browser, SMS messenger, or even the default keyboard (some exceptions apply, such as the system's Settings app).

The system apps function both as apps for users and to provide key capabilities that developers can access from their own app. For example, if your app would like to deliver an SMS message, you don't need to build that functionality yourself—you can instead invoke whichever SMS app is already installed to deliver a message to the recipient you specify.


System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management

Android Applications and Their Categories

Android is an open-source operating system, based on the Linux kernel and used in mobile devices like smartphones, tablets, etc. Further, it was developed for smartwatches and Android TV. Each of them has a specialized interface. Android has been one of the best-selling OS for smartphones. Android OS was developed by Android Inc. which Google bought in 2005. In this article, we will discuss android application types and categories as well as their advantages and disadvantages. Firstly let's see the types of applications, there are mainly 3 types of Android Applications.

Types of Android Applications

1. Native Apps

Native apps are built for particular operating systems, which are mostly Android and IOS. Also, there are more OS for mobile applications: Blackberry and Windows. This is available for download on Google Play Store and for IOS Apple App Store. Native apps are generally built to make the most of all the features and tools of the phones such as contacts, cameras, sensors, etc. Native apps ensure high performance and stylish user experience as the developers use the native device UI to build apps. WhatsApp, Spotify, Pokemon GO, etc. are examples of Natives apps. Android apps are built using Java, Kotlin, and Flutter, for the frontend, it uses the XML scripting language. And IOS apps built using Swift, Flutter/ Dart, and C#.

Advantages:

Native apps are designed for the particular operating system and it gives the best user experience.

Native apps are built with separate gestures it gives a good experience to users and it is very useful for all users.

Disadvantages:

Native apps are costly in comparison to others because they want separate maintenance.

Requires a separate codebase to add new features.

2. Web Apps

Web applications are built only the run on browsers. They are mainly the integrations of HTML, CSS, and Javascript. It runs on Chrome, Firefox, and other browsers. The responsiveness and functionality of the web apps could easily be confused with a native app since both the Native and web apps have almost the same features and responsive nature. And one of the major differences between the two is that native mobile apps can function both in the offline mode without an active internet connection and the online mode, whereas the web apps require an active internet connection for them to work. Gmail, Canva, and Google Docs are the best examples of web apps.

Advantages:

Easy to build

Web apps are used less storage than other applications.

Web Apps are preinstalled on all devices.

Web applications are easily accessible in any type of application.

Disadvantages:

Local resources are not available in web apps.

Depends on internet networks/ connections.

3. Hybrid Apps

Hybrid applications are also called Cross Platform Applications. Hybrid applications are runs on multiple platforms like Android and IOS. Also, these are made from the integration of web and native applications. Because hybrid apps use a single codebase, they can be deployed across devices. For example, when we build the android application, we can also launch it on IOS. As a cross-platform development option, developers have more freedom when designing their applications as they do not need to stick to specific design guidelines from either apple or google. Instagram, Uber, and Crypto change are examples of Hybrid apps. For Hybrid application development, we use Flutter/Dart, React Native, etc.

Advantages:

Users can use it on more than one platform.

It is integrated with browsers.

Maintained by many versions.

Shareable code makes it cheaper than a native app.

Disadvantages:

Slower compared to native apps.

There might be some user interface issues.

In hybrid apps have limitations in using all the Hardware and Operating Systems features.

Now let’s see about some categories of Android applications.

Categories of Android Applications

Some Categories of the Android Applications:

E-Commerce Apps: E-commerce apps are an example of a B2B model. It helps to people to sell and borrow different items and it saves time and money. In e-commerce applications, we can do trading of commercial goods on online marketplaces. To buy specific items and goods, you simply need to make electronic transactions like UPI, Phonepe, etc. through your smartphone or computer. Flipkart, Amazon, OLX, and, Quiker are examples of e-commerce applications.

Educational Apps: Educational apps are too much used to improve knowledge and peoples get productivity. Apps for education can make people more interactive, more engaged, and perform better. Keeping teaching methods good is integral to getting students engaged in their studies and learning apps are a fantastic way of achieving this. For example, Google Classroom, SoloLearn, edX, Duolingo, etc.

Social Media Apps: Social media apps give the opportunity to the peoples connect and communicate together. These apps are mainly used for sharing purposes and making fun. Many peoples use social media applications for influence, marketing/ business, entrepreneurship, etc. Instagram, Facebook, WhatsApp, YouTube, LinkedIn, etc. are examples of social media applications.

Productivity Apps: Productivity apps typically organize and complete complex tasks for you, anything from sending an email to figuring out a tip. The easy-to-use Google Drive app gives users access to all of the files saved to the cloud-based storage service across multiple devices. Productivity applications arise in many different forms and they often take a different approach to improving your workflow. For example, Hive, Todoist, Google Docs, etc.

Entertainment Apps: Entertainment apps are widely used apps worldwide. It contains OTT platforms and novels and other content. These platforms entertain people and give them much more knowledge about different things. Everyone is watching OTT platforms and those are trending these days, and their development is also in demand all over the world. Hotstar, Netflix, and Amazon prime video are the best examples of this entertainment applications.

Android Core Building Blocks

Android components

An android component is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.

The core building blocks or fundamental components of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

Activity: An activity is a class that represents a single screen. It is like a Frame in AWT.

View: A view is the UI element such as button, label, text field etc. Anything that you see is a view.

Intent: Intent is used to invoke components. It is mainly used to:

Start the service, Launch an activity, Display a web page, Display a list of contacts, Broadcast a message, Dial a phone call etc.

For example, you may write the following code to view the webpage.

```
Intent intent=new Intent(Intent.ACTION_VIEW);

intent.setData(Uri.parse("http://www.javatpoint.com"));

startActivity(intent);
```

Service

Service is a background process that can run for a long time.

There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

Content Provider

Content Providers are used to share data between the applications.

Fragment

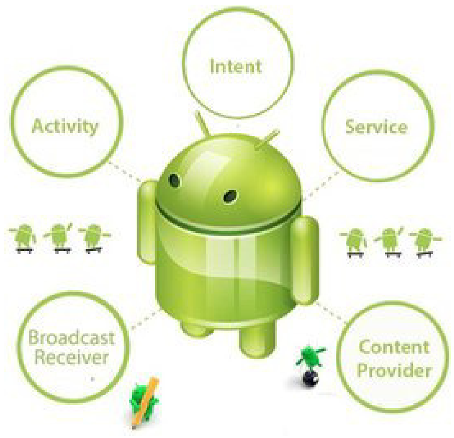
Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

AndroidManifest.xml

It contains informations about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

Android Virtual Device (AVD)

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.



AndroidManifest.xml file in android

The AndroidManifest.xml file contains information of your package, including components of the application such as activities, services, broadcast receivers, content providers etc.

It performs some other tasks also:

It is responsible to protect the application to access any protected parts by providing the permissions.

It also declares the android api that the application is going to use.

It lists the instrumentation classes. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.

This is the required xml file for all the android application and located inside the root directory.

A simple AndroidManifest.xml file looks like this:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.javatpoint.hello"

    android:versionCode="1"

    android:versionName="1.0" >

    <uses-sdk

        android:minSdkVersion="8"

        android:targetSdkVersion="15" />

    <application

        android:icon="@drawable/ic_launcher"

        android:label="@string/app_name"

        android:theme="@style/AppTheme" >

        <activity

            android:name=".MainActivity"

            android:label="@string/title_activity_main" >

            <intent-filter>

                <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />

          </intent-filter>

        </activity>

    </application>

</manifest>
```

Elements of the AndroidManifest.xml file

The elements used in the above xml file are described below.

<manifest>

manifest is the root element of the AndroidManifest.xml file. It has package attribute that describes the package name of the activity class.

<application>

application is the subelement of the manifest. It includes the namespace declaration. This element contains several subelements that declares the application component such as activity etc.

The commonly used attributes are of this element are icon, label, theme etc.android:icon represents the icon for all the android application components.

android:label works as the default label for all the application components.

android:theme represents a common theme for all the android activities.

<activity>

activity is the subelement of application and represents an activity that must be defined in the AndroidManifest.xml file. It has many attributes such as label, name, theme, launchMode etc.

android:label represents a label i.e. displayed on the screen.

android:name represents a name for the activity class. It is required attribute.

<intent-filter>

intent-filter is the sub-element of activity that describes the type of intent to which activity, service or broadcast receiver can respond to.

<action>

It adds an action for the intent-filter. The intent-filter must have at least one action element.

<category>

It adds a category name to an intent-filter