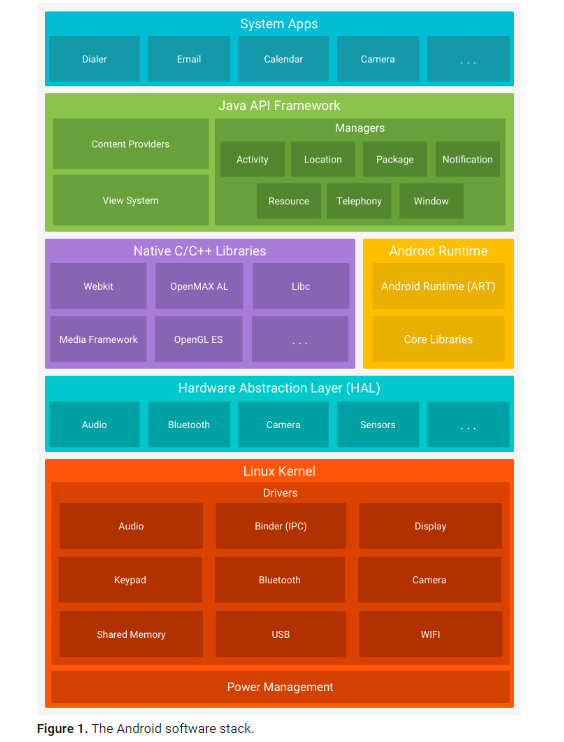
**Android Architecture**



## Applications

* Applications is the top layer of android architecture.
* The pre-installed applications like home, email, 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

* The entire feature-set of the Android OS is available to you through APIs written in the Java language.
* Application Framework provides several important classes which are used to create an Android application.
* It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources.

**Activity Manager :**

* + This class gives information about and interacts with activities, services, and the containing process.
  + An [Activity Manager](https://developer.android.com/guide/components/activities/intro-activities) that manages the lifecycle of apps and provides a common [navigation back stack](https://developer.android.com/guide/components/tasks-and-back-stack).
  + it decides when to start processes, and manages all of the top-level components (activities, receivers, services, content providers) running in them.
  + Activity Manager belongs to android.app Manage screens, it's state, position in stack and process.

**Window Manager** :-

* + It is responsible for organizing the screen. It allocates surface and decide where the Applications go and how they are layered.
  + it automatically performs window transitions and animations when opening or closing an app or rotating the screen.

**Resource Manager**

* It is a tool window for importing, creating, managing, and using resources in your app.
* providing access to non-code resources such as localized strings, graphics, and layout files

**Content Provider :**

* Content providers can help an application manage access to data stored by itself, stored by other apps, and provide a way to share data with other apps.
* It help to encapsulate the data, and provide mechanisms for defining data security.

[View System](https://developer.android.com/guide/topics/ui/overview)

* To build an app’s UI, including lists, grids, text boxes, buttons, and even an embeddable web browser

[Notification Manager](https://developer.android.com/guide/topics/ui/notifiers/notifications)

* that enables all apps to display custom alerts in the status bar.

## Native 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++.

* Surface Manager : responsible for managing access to display subsystem.
* WebKit : it is Open source web browser engine that responsible for Browser support
* SQLite : It is database support and used to manipulate data.
* FreeType : Font related Support.
* Media : Playing and recording audio and video formats.
* SSL (Secure Socket Layer) : responsible for internet security and allows user for authentication encryption and decryption over internet.
* SGL(Scalable Graphics Lib) / OpenGL : both are cross-language and cross-platform. API are used for 2D and 3D graphics. It is low level graphics library.

**Android Runtime**

* Android version 5.0 or higher, each app runs in its own process and with its own instance of the [Android Runtime (ART)](https://source.android.com/devices/tech/dalvik/index.html).
* ART is written to run multiple virtual machines on low-memory devices by executing DEX files.
* [**d8**](https://developer.android.com/studio/command-line/d8) is a command-line tool that Android Studio and the Android Gradle plugin use to compile your project's Java bytecode into DEX bytecode that runs on Android devices
* the Android runtime (ART) ahead-of-time compiler further optimizes compressed Dalvik Executable format (DEX) files by converting the DEX files in an app package into a more compact representation. This change allows your app to start faster and consume less disk space and RAM.

ART include the following major features :

* Ahead-of-time (AOT) and just-in-time (JIT) compilation
* Optimized garbage collection (GC)
* Better debugging support and the ability to set watchpoints to monitor specific fields

## The Linux Kernel

The foundation of the Android platform is the Linux kernel. For example, [the Android Runtime (ART)](https://developer.android.com/guide/platform#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](https://source.android.com/security/overview/kernel-security.html) and allows device manufacturers to develop hardware drivers for a well-known kernel.

## Hardware Abstraction Layer (HAL)

The [hardware abstraction layer (HAL)](https://source.android.com/devices/architecture/hal-types) provides standard interfaces that expose device hardware capabilities to the higher-level [Java API framework](https://developer.android.com/guide/platform#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](https://source.android.com/devices/camera/index.html) or [bluetooth](https://source.android.com/devices/bluetooth.html) module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.