**ANDROID BOOTING HISTORY AND INTRODUCTIONS**

**Android:**

**The Android code as "open-source under the Apache License" was released by Google which is why most of the manufacturers customize the code for their products, and this is one of the reasons Android has had a considerable growth compared to any of the platforms in history of computers.**

**"the Operating System provides a software platform on top of which other application programs can run"**

**The Operating Systems provide an API which ensures that the third-party applications run and operate on device.**

**Application:**

**An application is a set of instructions for the computer to perform a useful task for the user, and the term applies to the Smartphone apps as well.**

**There are many Smartphone applications in the stores today, which can be installed on the system and perform a specified task, e.g. voice recorder, audio player, map, document viewer, etc.**

**Smartphone applications are divided into three categories (Native, Mobile Web, and Hybrid)**

**Applications: Native Native applications are those that can be installed on the device throughout an OS-specific online application store, and the applications that are already installed on the device by manufacturer.**

**Native applications are developed specifically for one platform using a specific programming language which can take full advantages of all the device features.**

**Native Apps are: • The fastest • Most reliable • Most responsive • May use all the available features of the device; such as camera, compass, notifications, etc.**

**Applications: Mobile Web Web apps are actually just a version of a website that has been resized to fit a Smartphone screen, which is the reason they can run on any mobile platform including Featured-Phone and Smartphone.**

**"The concept unveiled with the emergence of HTML5”, a Markup Language used to develop websites and recently mobile version websites.**

**Web applications are actually the websites, not real apps, which in many ways look and feel like Native applications.**

**Web apps are accessed using mobile web browser and they will offer an install option which would add a bookmark to the Smartphone's home screen.**

**Applications: Hybrid**

**The word Hybrid means something that is powered by more than one source of power.**

**The meaning applies to the Smartphone apps as well, a Hybrid is an applications that is composed of both Native and Web apps, in which the Native part of the app acts as a wrapper to the Web part.**

**In Hybrid concept the "Web apps are built into a Native mobile container or framework", thus in this approach the majority of the code is written in HTML5 and the little Native part is written in OS-specific programming language. In this approach the features of the phone, including: camera, GPS, compass, notifications, etc.,**

**will be accessible throughout the Native part of the application, while the contents of the website will be retrieved by the Web part of the application.**

**Android Architecture**

**• Kernel: The kernel provides the fundamental software needed for booting, memory management, process management, network stack, and driver model.**

**Android Runtime: Android Runtime consists of Dalvik Virtual Machine and Core Java libraries.**

**• Dalvik Virtual Machine: is a type of JVM used in Android devices to run applications and is Optimized for low processing power and low memory environments**

**• Core Java Libraries: "These are different from Java SE and Java ME libraries.**

**However these libraries provides most of the functionalities defined in the Java SE libraries."**

**Android Architecture**

**• Libraries:"These libraries are written in C or C++" programming languages and they enable the device to handle different types of data.**

**Some of those libraries are:**

**• Surface Manager: The Surface Manager is used for display management and it is responsible for composing different drawing surfaces on the screen.**

**"It manages the access for different processes to compose 2D and 3D graphic layers."**

**• Media framework: Media framework provides different media codecs allowing the recording and playback of different media formats.**

**• SQLite: SQLite is the database engine used in Android for data storage purposes.**

**• WebKit: It is an Open Source browser engine. Android Architecture**

**Application development requires software engineering knowledge and skills which includes understanding of at least one programming language.**

**Smartphone applications are developed using JAVA, C, Objective-C and other programming languages, based on what language is supported by the OS.**

**For example Kotlin, JAVA, C, and C++ for Android, Objective-C and Swift for iOS, and C, C#, C++, and Visual Basic are used to develop applications for Windows Phone.**

**Application Development**

**Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.**

**It is available for download on Windows, macOS and Linux based operating systems.**

**IDE**

**• Operating System: Windows/Mac/Linux**

**• CPU: Intel Core i5 or higher with Virtualization Support**

**• RAM: 8GB**

**• Storage: SSD Preferred (with 8GB of free space)**

**ANDROID BOOTING SEQUENCES**

Bootloader
Kernel
init
Zygote DVM
SystemServer
Boot ROM
ManagersManagersManagersManagers
Stage 2
Stage 3
Stage 4
Stage 5
S...

**Stage 1**

**• On pressing POWER button, the Boot ROM code starts executing from a predefined location which is hardwired in ROM.**

**• It loads the Bootloader into RAM.**

**• Bootloader is code that is executed before any Operating System starts to run.**

**• Bootloaders is a low-level code contains the instructions that tell a device how to start up and find the system kernel.**

**• The bootloader usually lives on the system board in non-volatile memory and is often specific to a device.**

**• The boot loader is usually split into stages. Stage 1 will load stage 2 from a specific sector on the disk, then stage 2 will initialize the system and load the kernel from 'boot' flash partition into RAM.**

**Stage 2**

**• Once kernel boots, it start setup cache, protected memory, scheduling, loads drivers, starts kernel daemons, mounts root file system, initializing I/O , starts interrupts, initializes process table.**

**• the 'kernel' is the central component of most operating systems; it is a bridge between applications and the actual data processing done at the hardware level.**

**• A kernel is the lowest level of easily replaceable software that interfaces with the hardware in your device.**

**• It is responsible for interfacing all of your applications that are running in “user mode” down to the physical hardware, and allowing processes, known as servers, to get information from each other using inter-process communication (IPC).**

**• When kernel finish system setup first thing it look for “init” in system files and launch root process or first process of system.**

**Stage 3**

**• During booting of an Android device, init is the first user space process start\_kernel() spawns.**

**• The init process has two responsibilities. – Mounts directories like /sys , /dev or /proc and – Runs /init.rc script**

**• Init.rc is responsible for the initial set up of the system.**

**• It imports /init.${ro.hardware}.rc which is the primary vendor supplied .rc file.**

**• The 'init.rc' file is intended to provide the generic initialization instructions, while the ‘init.${ro.hardware}.rc ' file is intended to provide the machinespecific initialization instructions.**

**• The init process is what will set up all native services and this is similar to a regular Linux system boot.**

**• The init process first creates a shared memory region and stores a fd to the region. Then init process maps the region into its virtual space with mmap with MAP\_SHARED flag, as a result, any updates to this area can be seen by all processes.**

**• After that, init process will load properties from following files:  /default.prop /system/build.prop /system/default.prop /data/local.prop**

**• At this stage, you can finally see the Android logo in your screen.**

**• init runs the C++ program /system/bin/app\_process, and gives the resulting process the name "zygote“.**

**Stage 4**

**• Zygote is a daemon started as a system service whose goal is to launch applications.**

**• When app\_process launches Zygote, it creates the first Dalvik VM and calls Zygote’s main () method.**

**• Once Zygote starts, it preloads all necessary Java classes and resources, starts System Server and opens a socket to listen for requests for starting applications.**

**• Zygote receives a request to launch an App through /dev/socket/zygote. Once it happens it trigger a fork() call.**

**• When a process forks, it creates a clone of itself. It replicates itself in another memory space. This is done pretty efficiently. When this happens to Zygote, it creates an exact and clean new Dalvik VM, preloaded with all necessary classes and resources that any App will need. This makes the process of creating a VM and load resources pretty efficiently.**

**• As we know, Android runs on Linux. The Linux Kernel implements a strategy call Copy On Write (COW).**

**• What this means is that during the fork process, no memory is actually copy to another space. It is shared and marked as copy-on-write. Which means that when a process attempt to modify that memory, the kernel will intercept the call and do the copy of that piece of memory.**

**• In the case of Android those libraries are not writable. This means that all process forked from Zygote are using the exact same copy of the system classes and resources.**

**• The Zygote enables code sharing across the Dalvik VM, achieving a lower memory footprint and minimal startup time.**

**Stage 5**

**• After zygote preloads all necessary Java classes and resources, it starts System Server.**

**• The system server is the core of the Android system and as described in the boot sequence post it is started as soon as Dalvik is initialized and running.**

**• The other system services will be running in the context of the System Server process.**

**• The first thing that happens is that the server will load a native library called android\_servers that provides interfaces to native functionality.**

**• Then the native init method that will setup native services is called.**

**• After setting up the native services it create the server thread. This thread will start the remaining services in the system according to the necessary start order.**

**• Each service is running in a separate Dalvik thread in the SystemServer process.**

**• Once System Services up and running in memory, Android has completed booting process, At this time “ACTION\_BOOT\_COMPLETED” standard broadcast action will fire.**