Unit 1: Basics of Android Operating System

1

Unit Structure

1.1 LEARNING OBJECTIVE

After studying this unit, a student should be able to Understand:

- The history of the Android OS and various versions.
- The fundamentals Android application terminologies.
- The Minimum Privilege model implemented in Android and its relation with the programming aspects.

1.2 INTRODUCTION

Android is designed by Google for the touch screen devices. Android is the world's most popular mobile operating system, used in the billions of devices ranging from phones to watches, tablets, TVs, and more. The Open Handset Alliance (OHA) has played an important role in the overall development of the Android OS. The members of OHA are the mobile handset, chip, Network operators, Software developers, Component manufacturers, and some other mobile carriers. The OS contains a Linux-based Operating System (OS), middleware, and critical mobile applications. There are drivers like display, camera, flash memory, keypad, other sensors like accelerometer, gyroscope (and few other), Wi-Fi, GPS, Bluetooth, and audio. As like other kernels, here this kernel also serves as an abstraction between the hardware and the rest of the software on the phone. There are core services like security, memory management, process management, and the network stack.

Depending on the capability of the mobile handset Android OS version can be chosen. The Android OS provides the Application Programming Interfaces (APIs) in terms of Java and Kotlin. To utilize these functionalities, the one should have some knowledge of event handling (also called as the callbacks), and Extensible Mark-up Language (XML). Nowadays, Android Studio is available for Android App development. The Android Studio provides the fastest tools for building apps on every type of Android device. Android studio supports the installation on Windows, Linux, and Mac OS, in 32 bit and 64-bit versions. The https://developer.android.com is the web URL for the recent development updates in the Android OS.

1.2.1 IMPORTANT FEATURES OF ANDROID OS

Following are the essential key features that make the Android OS as more popular.

- The user of the Android OS gets informed timely through the Notification features. The user can take the appropriate action due to the notifications in the android. Every notification opens a suitable app for further action. Furthermore, the bundled notifications can be expanded.
- During the Android OS updates, the user can snooze the notifications.
- The multiple applications can be closed altogether.
- Android OS allows the customization of the android phone such as transferring the contacts, installing widgets, downloading the wallpapers, installing the customized keyboard, setting the default apps, the lock screen can be customized, and hiding apps.
- Majority of the Android users do not use the root users; however, it gives more flexibility to remove the apps. Through the process of rooting an Android OS, it is possible to access all settings and sub-settings deeply. It is similar to the administrative access to the Personal Computer. The rooting process allows removing the unnecessary apps provided by phone's carrier or its manufacturer, sponsored apps, unwanted apps called bootware. The rooted Android OS allows to block the spam and improve the performance. The bootloader should be unlocked to enable the rooting.
- Android OS provides features like multi-user PC, where one can lock or hide apps.
- Android OS allows capturing gestures performed on a device's fingerprint sensor. The fingerprint sensor is useful for authentication purpose.
- Google assistant provides lots of help through the media control commands given through the voice.
- The Latest innovations on the hardware fronts, like Near Field Communication (NFC), Telecommunications, and libraries of the other sensors are available.
- Android is available with free ads using Google Admob to monetize the apps developed for the android device and loaded with advertisements.
- Due to the flexibility and features of Android OS, there are a variety of handsets available with the different price ranges.

1.3 HISTORY OF ANDROID OS

In case of the PC OS, where the significant responsibilities to handle the x86 architectures, memory management, process management, process scheduling, peripheral management, multi-user management, and optimizations are the significant tasks. Also, there are network communication tasks like in the Local Area Network (LAN) and the Internet-connected environment. Usually, there are wired and wireless network connection to the PC and set of communication protocols, that the OS handles.

Similar to the PC OS, the mobile device OS handles the necessary access to the applications and process, to manage the cellular, network connectivity, and phone access. The mobile OS is tied to specific hardware. The mobile OS is different from the normal PC's OS where the mobile handsets have the ARM architecture microprocessor, and handsets are smaller in size. The mobile works on battery power; there is no separate keyboard or mouse. The user interface is a touch screen and no expandable RAM. There is no Ethernet cable; hence, the mobile must be able to communicate with other wireless protocols. Similar to the different mobile OSs like iOS, Symbian, Blackberry, HP's WebOS, Microsoft's Phone OS the Android is another mobile OS with better functionalities.

The co-founder and former CEO of Android Inc. Mr. Andrew E Rubin has a nickname as Android. His co-workers gave the nickname Android due to his love for the robots. In 17th August 2005, Google acquired Android Incorporation. In 2007, Google announced the development of the Android OS.

1.4 VERSIONS OF ANDROID OS

At the time of writing this course material, the Marshmallow was available in most of the android devices. Table-1.1 shows the android versions, their code name and API levels. The important features got added into the newer version of the Android with Application Level Interfaces (APIs) levels. The developer has to take important consideration about the target devices where the app is going to be deployed. It will perform well with better functionalities, provided that the handset hardware capabilities are available with the same potential.

Codename	**Version**	**API Level**
Android Q	10	29
Pie	9.0	28
Oreo	8.0 & 8.1	26 & 27
Nougat	7.0 & 7.1	24 & 25
Marshmallow	6.0	23
Lollipop	5.0 & 5.1	21 & 22
KitKat	4.4	19
Jelly Bean	4.1 to 4.4	16 to 18
Ice Cream Sandwich	4.0.3, 4.0.4	15
Honeycomb	3.0 – 3.2.6	11to 13
Gingerbread	2.3.3 to 2.3.7	9-10
Froyo	2.2 – 2.2.3	8
Eclair	2.0 – 2.1	5 – 7
Donut	1.6	4
Cupcake	1.5	3
Petit Four	1.1	2
No Name	1.0	1

Table-7: The code name, Android version and API levels

The recent version of Android Pie having the following critical features:

- Multi-camera support and camera updates
- Indoor positioning with Wi-Fi Round-Trip-Time (RTT) which is based on IEEE 802.11mc Wi-Fi protocol
- Machine Learning & Neural Networks API Support

1.5 ARCHITECTURE OF THE ANDROID OS

As shown in Figure-50, the Android stack is consisting of Power Management features, Linux kernel, hardware abstraction layer, open source libraries, android runtime system, application framework, and at the top the Applications.

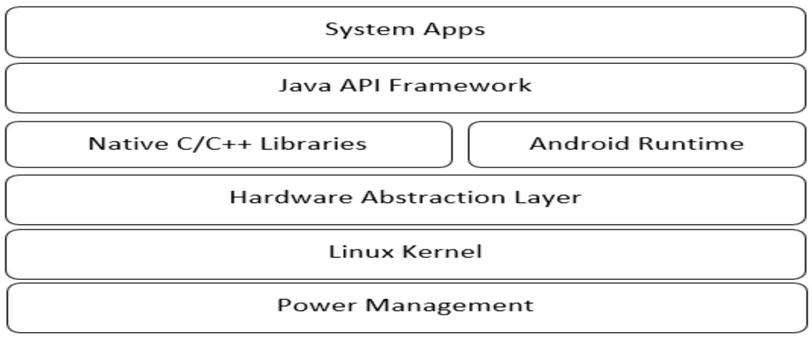


Figure-50 Layers in the Android Stack (Architecture)

1.5.1 POWER MANAGEMENT FEATURES

Figure-51 shows the power management strategies in the Android OS. One of the ways for power management is done through the App restrictions such as used with the help of the buckets.The app can be restricted in such a way that it should not consume the resources such as memory, CPU or other air interfaces. The app running in the background, accessing hardware or the processor highly consumes the energy.

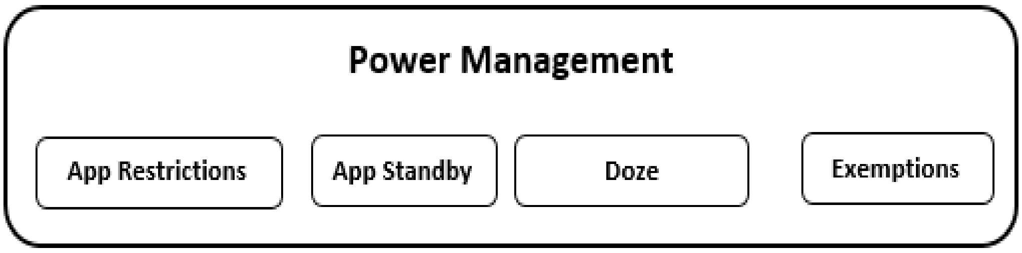


Figure-51 Power Management Strategies

The apps sending it in the standby mode so that, the app should not access the network activity and other jobs those not actively used by the user. The inactive applications are detected based on the longer clock time of the idle time. The sync activities are stopped for the app either for a day. The standby mode can be exited once the device is plugged in for the charging. The adb tool can be useful for application testing in case of the standby and active modes.

The Doze mode keeps the device CPU, and network activity completely stopped. The Doze mode also keeps the pending sync and job works. It can complete the work by entering into the maintenance window on the regular intervals.

1.5.2 LINUX KERNEL

Android was created on the open source kernel of Linux. It provides security, memory management, process management, network stack, and driver model. Linux kernel contains code for all the different chip architectures and hardware drivers it supports.

The Linux kernel having several drivers for display, camera, Bluetooth, shared memory, binder like Inter-Process Communication (IPC), USB, keypad, Wi-Fi, Audio, and power management purpose.

There are stable, and long term supported kernels for android. The latest features included in the Linux kernels are energy-aware scheduling, networking, SDCard, USB, System on Chip (SoC) for ARM64, and x86, etc. Figure-52 shows the Linux kernel drivers for the various hardware like camera, audio, display, USB, Shared memory, Wi-FI, Keypad, and IPC.

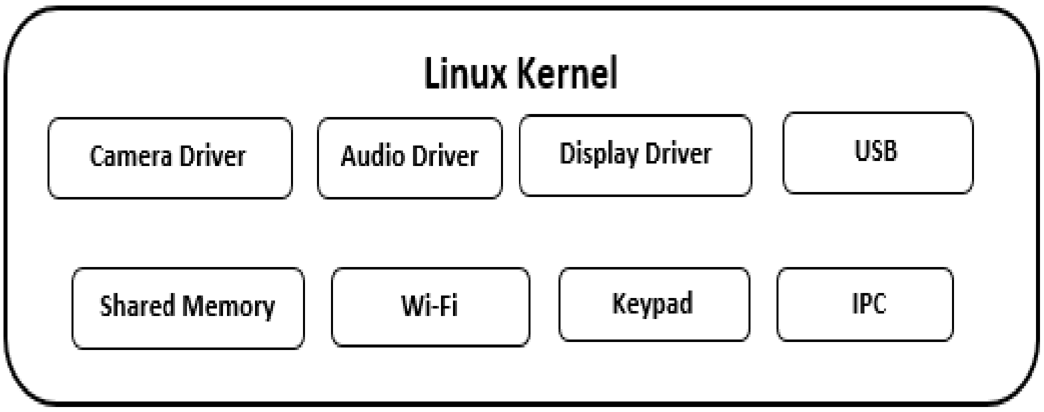


Figure-52 Linux Kernel and its sub-components

1.5.3 HARDWARE ABSTRACTION LAYER (HAL)

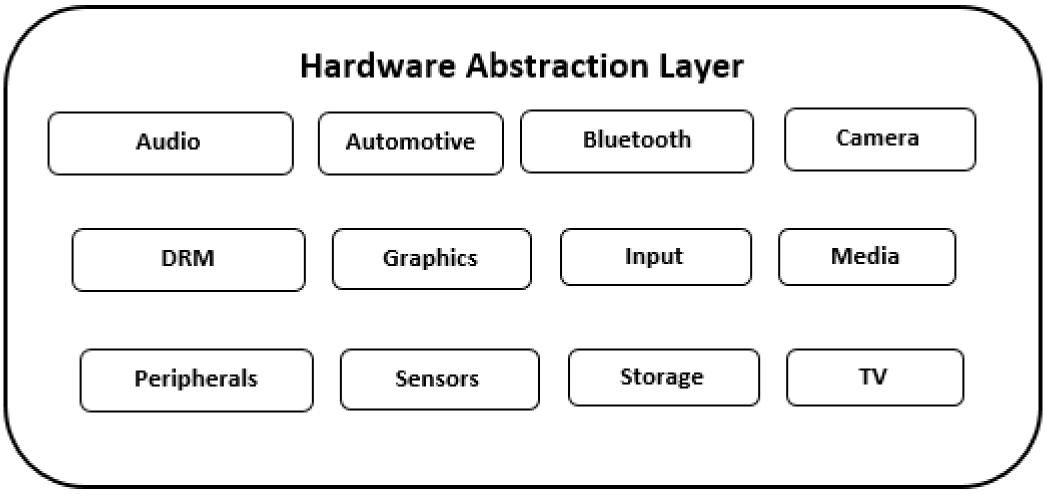


Figure-53: Hardware Abstraction Layer

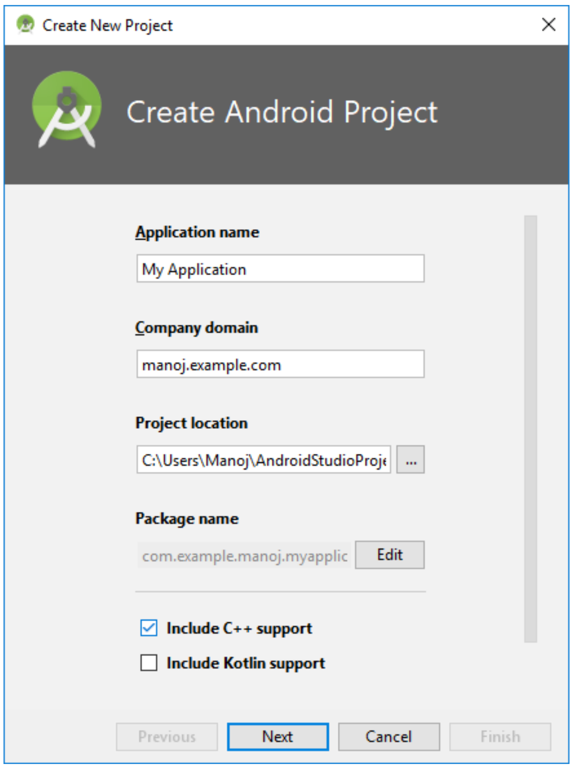
Figure-53 shows the HAL and its sub-components. A HAL defines a standard interface for hardware vendors to implement, which enables Android to be agnostic about lower-level driver implementations. Using a HAL allows you to implement functionality without affecting or modifying the higher level system. You must implement the corresponding HAL (and driver) for the specific hardware your product provides.

The hardware abstraction layer is audio, Automotive, Camera, Bluetooth, GPS, radio, Wi-Fi, Sensors, storage, TV, peripherals, media, and graphics. The core libraries and Dalvik VM are part of Android run time.

1.5.4 NATIVE C/ C++ LIBRARIES

Native Development Kit (NDK) is a companion tool to write the critical performance code and makes it possible to write apps using C/C++ code. It is possible to mix existing C/C++ native libraries into an Android Java project. Without changing the fundamental Android application model, the C or C++ code can build activities, handle user input, use hardware sensors, access application resources, etc. It can be packed into the APK bundle. The direct low-level operations such as at hardware access, can be achieved through the native libraries, as well as the cross-platform functionalities such as iOS and Android, can be achieved. To utilize the native C++ libraries, the Run Time Type information (RTTI) is useful to find the run time information about the class such as finding the variables, methods, and other information about the state of the method and objects. This is similar to the Reflection APIs in Java.

Java Native Interface (JNI) allows accessing the native code from Java and vice versa. The native code written in C/C++ can be directly converted into the code understandable to the CPU. The functionality implementation is possible in the using C and C++ because it has a mechanism like Java Virtual Machine (JVM). To develop the Java project for the native support, it is required to click on the checkbox of C++ support. Figure-54 shows the option for NDK support.


Create New Project

Create Android Project

Application name: My Application

Company domain: manoj.example.com

Project location: C:\Users\Manoj\AndroidStudioProje

Package name: com.example.manoj.myapplic

[✓] Include C++ support

[ ] Include Kotlin support

Figure-54: Starting a new NDK supported the project –including the C++

There are several applications to utilize the C/C++ native libraries such as to achieve the low latency applications in case of the games and the physics simulations etc. It is also an intention to reuse the existing libraries written in C/C++. CMake and LLDB debugger are useful tools for such application development.

Table-8 shows a list of C/C++ libraries provided from the recent to the basic version of the Android OS.

C/C++ Native Library	**Description**
Surface Texture & Surface Manager	Surface connection with the OpenGL Responsible for rendering windows and drawing surfaces of various apps on the screen.
Shared Memory JNI	Upon creation of shared memory & a file descriptor used in the inter-process communication.
Neural Networks	Hardware acceleration for on-device machine learning operations. The API supports on-device model creation, compilation, and execution.
Hardware Buffer	pipelines for cross-process buffer management.
AAudio,	Read and Write the audio stream with low latency.

Native media	The image read and write operations.
OpenGL	Cross-Language, Cross-Platform Application Programming Interface (API) For Rendering 2D And 3D Vector Graphics.
Native Multi-network	Multiple networks operations.
Choreographer	Timing of Animations, Input, And Drawing in Native Code.
camera	Photo Capturing and other operations.
Vulkan	API for high-performance 3D graphics rendering.
Tracing	Write trace events to the system buffer.
OpenMAX AL	Multimedia Handling Is based on Khronos Group.
OpenSL	Native audio handling is based on the Khronos Group.
EGL	Native Platform Graphics Interface
Bitmap	Pixel Buffer
Dynamic Linker	Access to Dynamic Linker
Zlib	The zlib compression library provides in-memory compression and decompression functions, including integrity checks of the uncompressed data.
Logging	Log messages to Logcat from native code.
C/C++ library	<Stdlib.h> and <stdio.h>
FreeType	The library is used for rendering fonts.
WebKit	Library mainly provides Web Browsing engine and a lot more related features.
SQLite Library	Apps local database for lightweight operations.

Table-8: Summary of the Native Library in C/C++

1.5.5 ANDROID RUNTIME (ART)

ART translates the bytecode into the native instructions. ART is also called as the Application Runtime Environment. Previously, there was a Dalvik virtual machine which is not replaced by the ART. It uses the Ahead-of-Time (AOT) compilation process which compiles the entire application into the machine code upon the installation of the app. It reduces the Dalvik's procedure of interpretation and trace-based Just-in-Time (JIT) Compilation, which enhances the overall execution efficiency and saves the battery power. The Android 5.0 Lollipop onwards the previous Dalvik is replaced by the ART. ART improves memory management and protects battery power. During the installation of the software, it takes time as it compiles the code from the APK bundle. However, it improves the overall performance of the applications, at the runtime.

ART adds the sampling profiler to supports the number of debugging options for monitoring the garbage collection process, threads, locks, stack traces, breakpoints, and watchpoints. Some techniques of Dalvik do not work on ART.

1.5.6 JAVA API FRAMEWORK

The application framework is consisting of a set of managers including activity, location, package, notification, resource, Telephony, and window manager. There are content providers and view system. Figure-55 shows the Java Application Programming Interface Framework.

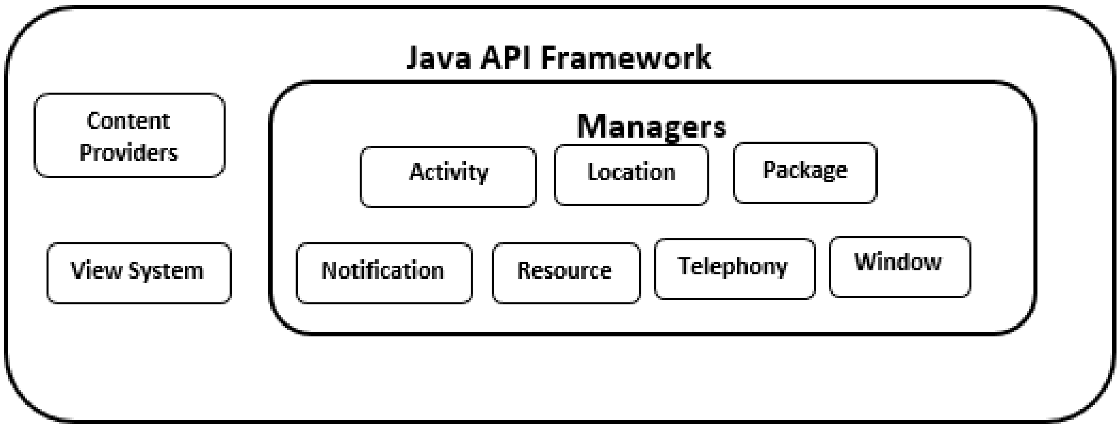


Figure-55 Java API Framework

1.5.6.1 VIEW SYSTEM

All the Android features can be utilized through the Java API Framework. The views, i.e. graphical user interface components can be used along with their events through the view system. The view also called as the widgets. Figure-56 shows the hierarchy of the View Group and Views.

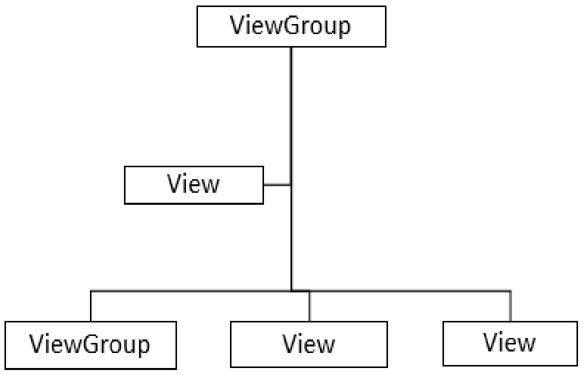


Figure-56 View Components and widgets Hierarchy

Here ViewGroup is the layouts such as Linear Layout, Constraint Layout, Frame Layout, or Table Layout. Sometimes it is a combination of more than one layouts. In the Android Studio, the design view shows the layouts. Figure-57 shows the various layouts; Figure-58 shows the various widgets.

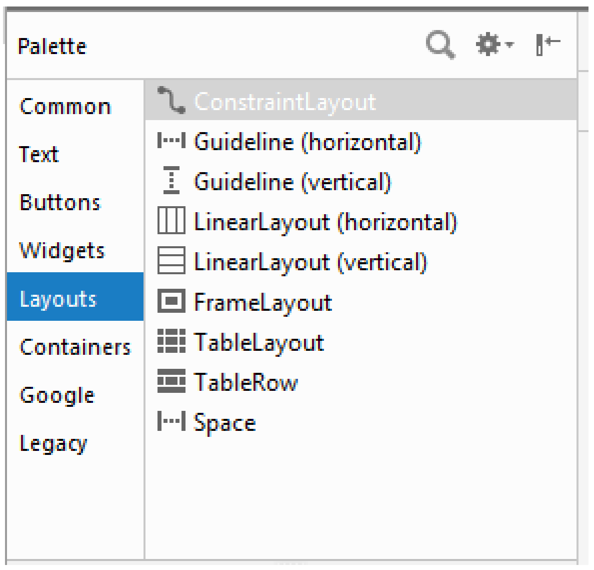


Figure-57 View Group in Android Studio

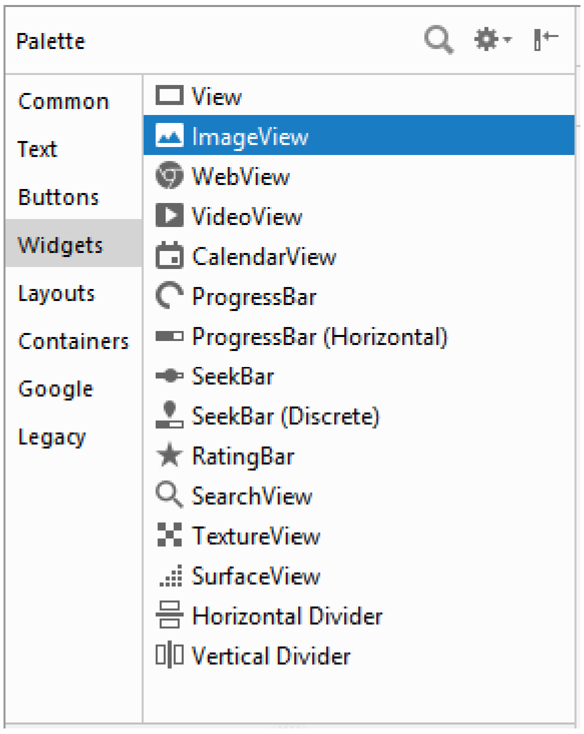


Figure-58 views, i.e. the widgets in android studio GUI

There are several widgets such as Image View, web view, video, calendar, progress bar, seek bar, Rating Bar, search view, Texture View, and Surface view. There are several text-based widgets similar to the readymade patterns. There are buttons like a toggle button, image, checkbox, radio button, switch, and floating action button.

1.5.6.2 CONTENT PROVIDERS

Content providers provide the mechanism to share the data from one widget to another, one data store to another, or between the different applications. The content provider can also work to supply data to the threads and their synchronizations and cursor.

1.5.6.3 ACTIVITY MANAGER

The Activity Manager is a class android.app.ActivityManager which Android System manages a stack of activities which are in different states (starting, running, paused, stopped, destroyed). An activity has one or more windows (e.g., dialogs). An activity becomes roughly a token. Activity Manager gives information about and interacts with, activities, services, and the containing process.

A window has one or more surfaces (e.g., surface views). However, in a window manager, a window is called a session. A surface is called a window.

1.5.6.4 WINDOW MANAGER

Window Manager is an interface responsible for organizing the screen. The window manager creates surfaces for the application. It allocates surface and decides where they(Applications) go and how they are layered. At create, a surface for all the applications and they draw directly into the surface without going through the Window Manager,Some basic things regarding activity, windows, and surfaces.

1.5.6.5 PACKAGE MANAGER

The abstract class provides essential information about package installation and its availability. All Java classes are part of a package. The package may consist of other packages and classes or interfaces. At the runtime, it is required to find out presence or absence of the package. The package manager plays an important role in package access related operations.

1.5.6.6 NOTIFICATION MANAGER

It is the class useful to handle the notifications to the device user. The happening of some event related to the app, sometimes in the background are notified to the user. These notifications are either in the form of the alerts, or the toast, blinking of the light, a sound, or vibrating the phone.

1.5.6.7 LOCATION MANAGER

The device location identifier where it takes the support of the GPS and proximity of the geographical location. The API such as addProximityAlert() takes the longitude, latitude and the radius to create the alert when the device enters into the proximity of the specified location.

1.5.6.8 TELEPHONY MANAGER

The android.telephony.TelephonyManager class provides information about the telephony services such as subscriber id, sim serial number, phone network type, etc. Moreover, you can determine the phone state, etc. The information related to International Mobile Equipment Identity (IMEI) number, subscriber ID, Sim card information, network information, type of the network such as Global System for Mobile (GSM) or Code Division Multiple Access (CDMA) can be easily accessed through the Telephony Manager Class.

1.5.6.9 RESOURCE MANAGER

Resource Manager manages resources such as icons, styles, colors, strings, layout, and drawable.

Following code shows the string resources separately maintained in the XML file.

```
<resources>
   <string name="app_name">Interchange</string>
   <string name="title_activity_second">SecondActivity</string>
</resources>
```

The style can be applied similar to the cascaded style sheet for HTML code. A style resource defines the format and looks for a UI. A style can be applied to an individual View (from within a layout file) or an entire Activity or application (from within the manifest file). Android studio creates the style.xml file to maintain the style.

The colors can be maintained in the separate file with the name color.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="colorPrimary">#3F51B5</color>
   <color name="colorPrimaryDark">#303F9F</color>
   <color name="colorAccent">#FF4081</color>
</resources>
```

1.5.7 APPLICATIONS

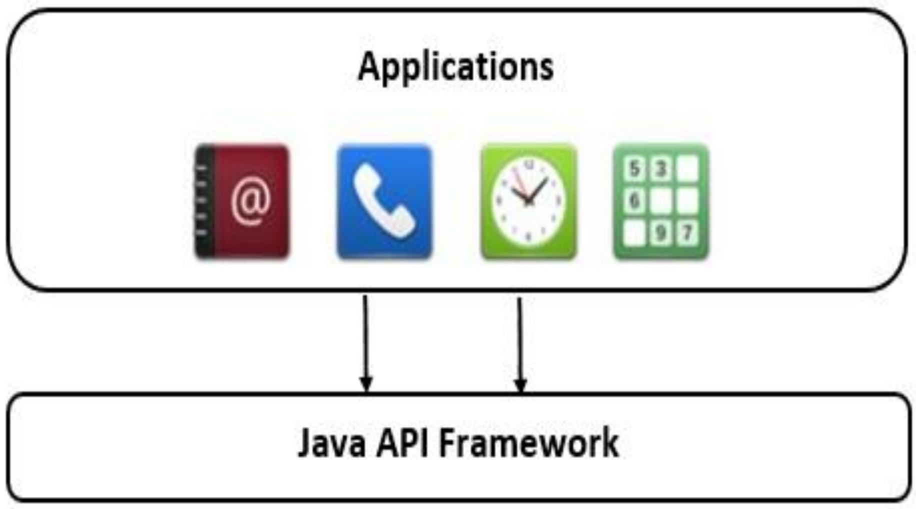


Figure-59 The applications are at the top layer

The top layer is of applications such as email, alarm, calculator, camera application, contacts, media player, SMS, MMS manager, and photo album. Figure-59 shows the application layer interacting with the Java API framework layer.

Android studio utilizes the Software Development Kit (SDK) with API libraries and developer tool necessary to build, test and debug apps for Android. You can download Android Development Tools (ADT) bundle, ADT plug-in, Android SDK tools, Android platform tools, Latest Android platform, latest Android system image for the emulator.

All the code in a single APK file is considered to be one application and is the file that Android-powered devices use to install the app. Once an application is installed, on a device, each Android application lives in its security sandbox. An Android OS is a multi-user operating system in which each application is a different user. By default, the system assigns each application a unique Linux user ID (the ID is used by the system and is unknown to the application). The system sets permissions for all the files in an application so that only the user ID assigned to the application can access them.

1.6 APPLICATION & TASK MANAGEMENT IN ANDROID OS

1.6.1 APPLICATION TERMINOLOGY

An Android application consists of one or more activities, services, listeners, and intent receivers. Each process has its virtual machine (VM), so an application's code runs in isolation from other applications. By default, every application runs in its own Linux process. Android starts the process when any of the application components need to be executed, then shuts down the process when it’s no longer needed, or when the system must recover memory for other applications. It implements the principals of least privileges. Each application by default has access only to the components that it requires to do its work and no more. It creates a very secure environment in which an application cannot access parts of the system for which it is not given permission. User can share data with other application and for an application to access system services. All the application permissions must be given at install time.

View: In Android, the View is the foundation of everything that shows up on the screen. Buttons, toolbars, and inputs, everything is a View.

Activity: It is a single focused thing that a user can do. An activity is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. To create an activity, the Activity class should be extended. To add an activity in the Android Studio right click on Project Window -> Select activity. It shows several readymade options like Gallery with options like Basic activity, Empty Activity, Login Activity, Master Detail activity, and other navigation and scrolling activity. Figure-60 shows the various activity types.

Difference between Basic and Empty Activity

Empty Activity is the same as Blank Activity is the simplest activity which you can create. Empty activity can use fragment. Android Studio creates the separate two layout files for it. Basic Activity is more advanced than Empty/Blank Activity because have the toolbar and one button (Floating Action Button - FAB).

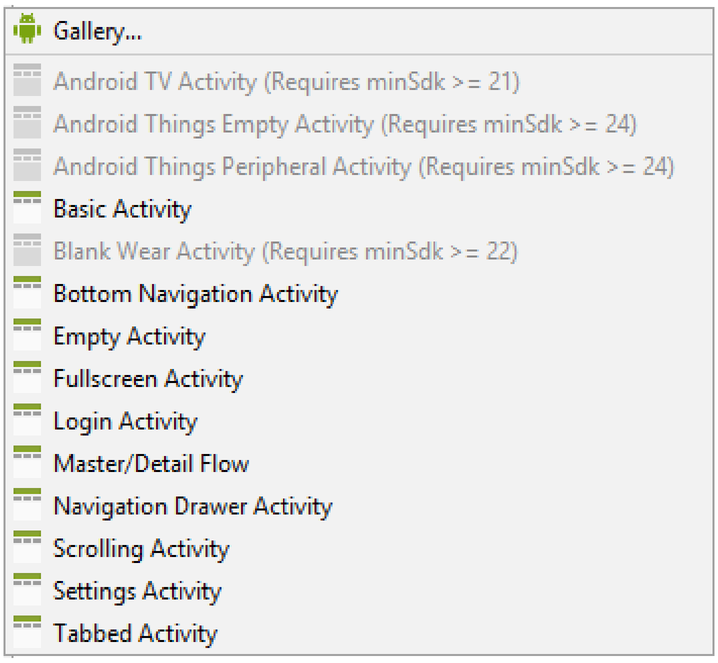

Gallery...
Android TV Activity (Requires minSdk >= 21)
Android Things Empty Activity (Requires minSdk >= 24)
Android Things Peripheral Activity (Requires minSdk >= 24)
Basic Activity
Blank Wear Activity (Requires minSdk >= 22)
Bottom Navigation Activity
Empty Activity
Fullscreen Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity
Settings Activity
Tabbed Activity

Figure-60 Different activities for the app development

Fragment: It is the portion of the user interface (subsection or module section) used to design the multi-pane meaningful user interface. The fragments can be reused in multiple activities. A fragment is having its own lifecycle and receives the events.

Service: A service is a general-purpose entry point for keeping an app running in the background for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity.

Listener: Listeners are generally the interfaces in Java programming. They are the abstract methods with an empty body, and final and static variables (usually called as constants). It is the responsibility of the class which implements the Listener interface. There are several listeners associated with the Graphical User Interface (GUI) Components such as ActionListeners and an ItemListener. There are widgets in the Android such as Button, CheckBox, RatingBar, ProgressBar, ImageView, TextView, EditText, etc. To attached the events with the widgets, the programmer should register them with each other.

Broadcast Receiver: A broadcast receiver is a component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements. Because broadcast receivers are another well-defined entry into the app, the system can deliver broadcasts even to apps that aren't currently running.

Resources: There are resources in the Android animation, color state list, layout, menu, string, style, and font. There are static resources like bool, color, dimension, ID, Integer, Integer Array and Typed Array.

1.6.2 ANDROIDMANIFEST.XML FILE

Manifest File AndoridManifest.xml is the important file which maintains the minimum privileges model of the Android OS. ".apk" is the Android application package file, which is compiled and packaged in a single file that includes all of the application's code (.dex files), resources, assets, and manifest file. Android programs are compiled into ".dex," i.e., Dalvik Executable files. The. dex files are zipped into a single .apk file on the device.

1.6.3 SECURITY FEATURES OF ANDROID APP

Each app lives in the security sandbox. Android is the multi-user Linux system in which each app is a different user. Each process has its own VM to run in isolation. The system assigns each app a unique Linux user ID. The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.

It provides the encrypted file storage, to avoid loss in case of the stolen or lost devices. The Java security features implemented on cryptographic algorithms can be used while creating the apps. The permissions can also be given at the run time.

1.6.4 PRINCIPLE OF LEAST PRIVILEGE

Each app by default has access only to the components that it requires to do its work and no more. For example, if the developer wants to develop an app where the Bluetooth is going to be accessed to share the data then it has to be mentioned in the AndroidManifest.xml file.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

Similarly, if the app is going to access the Internet following line should be added to AndroidManifest.xml.

```
<uses-permission android:name="android.permission.INTERNET" />
```

To add the Wi-Fi Access permission following lines can be used.

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

The examples given are the static permissions assigned in the androidmanifest.xml file. However, at runtime also the permissions can be given and revoked using the APIs checkSelfPermission(), requestPermissions() from the ActivityCompact and ContextCompact class.

1.7 APPLICATION LIFE CYCLE

By assuming Java as a programming language and automatic garbage collection feature, the application and the processes can be killed. The processes or application not required for the further usage need to be killed and the memory and other resources have to be reclaimed. The automatic garbage collection system decides the inclusion of the activity, service, and broadcast receiver role while continue running an application working or to stop. Hence the application and process life cycle is not in the control of the user and developer. However, there is an Activity life cycle, which includes the various stages and corresponding events. The stages include the activity launch, activity running, and activity shut down.

There is core set of callback methods like onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). Whenever new activity comes in the foreground then onPause() method gets called. Whenever the user returns to the activity back then onResume() method is called. If the application is in the pause mode and if the higher priority method requires memory, then the app process gets killed. When the activity is no longer visible then onStop() method is invoked. If the activity is finishing or is being destroyed by the system then onDestroy() method is invoked. Whenever the user navigates, then onRestart() method is gets invoked. Figure-61 shows the activity life cycle stages.

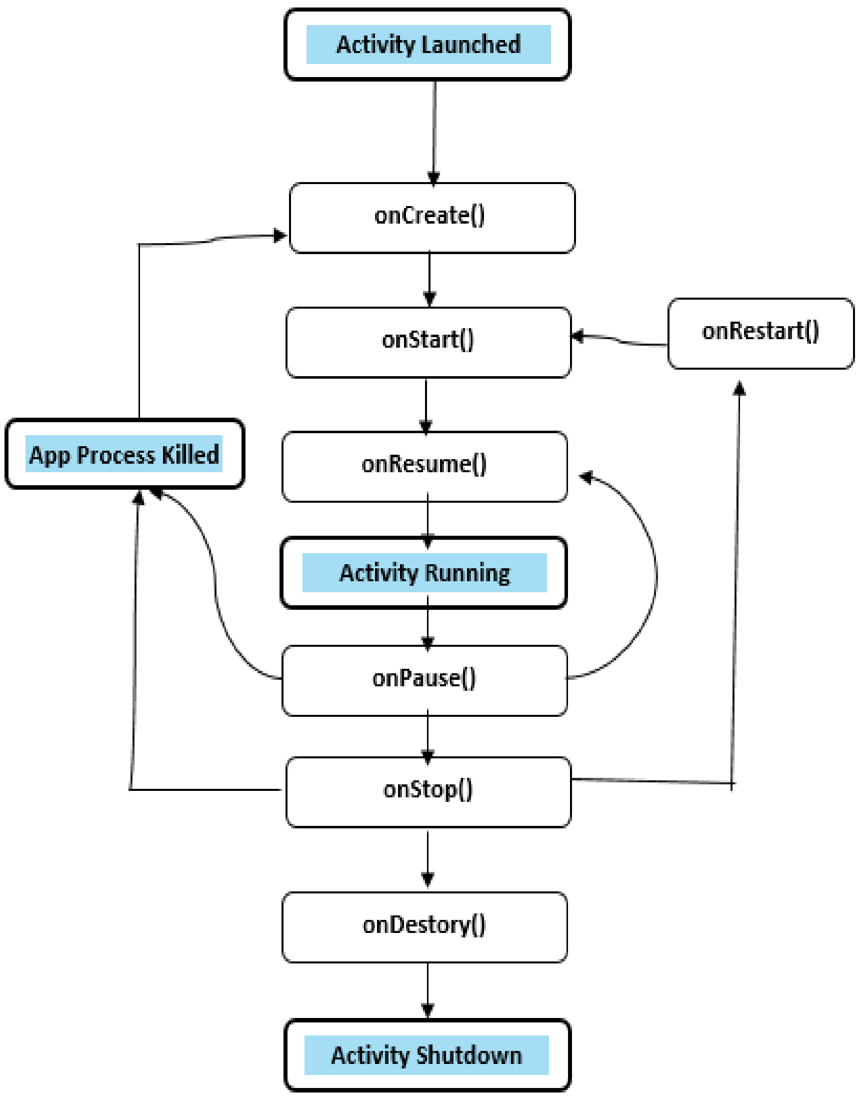


Figure-61 Activity Life-cycle methods

1.10 LET US SUM UP

- The Android architecture and its different layers are studies in the detailed.
- The discussion on the Linux Kernel functionalities and its subcomponents are discussed.
- The application terminologies like Activity, Service, Intent, and Broadcast receivers have been discussed.
- The comprehensive discussion on the power management techniques (e.g. bucket creation) in android architecture is done.
- The discussion on the Least Privileges model is done.

1.8 CHECK YOUR PROGRESS

➢ **Answer the following in True / False statements.**

1. Android is a Linux kernel based OS.
2. Androidmanifest file is written in HTML.
3. Flutter can be useful for UI development in Android app development only.
4. The permissions can be added to the Androidmanifest.xml only at the design time.
5. The functionality achieved using the NDK similar to Java API framework from the latency point of view.
6. @overrides is just a useless token in Java.
7. Constraint Layout is a Widget.
8. The more restrictions are imposed on the android apps when a mobile device is in battery mode.

➢ **Match the columns correctly**

Group A		**Group B**
Andrew Rubin		Version 6.0
Pie		Android Nickname
Marshmallow		Version 9.0
FreeType		Python
Statically Typed		Rendering fonts
Dynamically Typed		Java
Zlib		Logcat Messages
Logging		Compression Library

➢ **Fill in the blanks.**

1. The code name of the Android 8.0 is _______________.
2. The method useful to add the runtime permissions is _______________.
3. The ____________ in C++ is similar to the Reflection mechanism in Java.
4. The five different buckets to categorize the apps to save battery power are __________, ___________, ____________, ________ and __________.

- **Choose the correct option(s).**

1. Which of the following is the statically typed programming language?

   a) Kotlin b) Python c) Javascript d) Java

2. Semicolons are optional in

   a) Kotlin b) Java c) C++ d) Python

- **Answer the following Questions in one or two sentences.**

1. What is APK?
2. Name the languages supported for Android development.
3. Names of the dynamically typed languages?
4. What is the meaning of the dynamically typed languages?
5. Name of the statically typed languages?
6. What is the meaning of the statically typed languages?
7. What is material Design?
8. What is Flutter?
9. Write the permission line to perform the read and write operation through the android app.

1.9 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

- **True/False Answers**

1. True
2. False
3. False (Flutter can be useful for UI development in Android and iOS both).
4. False (The permissions can be added at runtime also.)
5. False (As it is native to the CPU, it is faster.)
6. False (Causes a compilation error if the method is not found in one of the parent classes or implemented interfaces).
7. False (Constraint Layout is a View Group.)
8. True.

- **Fill in the blanks**

1. Oreo
2. requestPermissions()
3. RTTI (Run Time Type Information).

➢ **Multiple choice**

1. Kotlin, Java
2. Kotlin, Python

➢ **Answer of the short Questions**

1. Android Package Kit
2. Kotlin, Java, C++ & C
3. Groovy, JavaScript, Lisp, Lua, Objective-C, PHP, Prolog, Python, Ruby, Smalltalk, and Tcl.
4. Dynamic type checking is the process of verifying the type safety of a program at runtime.
5. Ada, C, C++, C#, Java, Fortran, Haskell, Pascal, and Scala.
6. A language is statically-typed if the type of a variable is known at compile time instead of at runtime.
7. Material Design is a visual language that synthesizes the classic principles of good design with the innovation of technology and science.
8. Flutter allows you to build beautiful native apps on iOS and Android from a single codebase.
9. <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

1.11 FURTHER READING

1. https://material.io/design/introduction/
2. https://flutter.dev/docs/get-started/flutter-for/android-devs
3. https://www.kernel.org/category/releases.html

1.12 ASSIGNMENTS

1. What are the battery saving techniques you apply for your mobile handset, discuss in details.
2. What are the different sensors available in your mobile handset?
3. What are the advantages and disadvantages of the rooted Android OS?

1.13 ACTIVITIES

1. If you have an Android phone then find its OS version number and corresponding code name?

2. Enable the developer option in your android phone.

3. Install recent Android Studio on your personal Computer.

1.14 CASE STUDIES

Case Study 1:

Install any e-Wallet app and enlist the functional and non-functional requirements for developing it using Android Studio.

Case Study 2:

Calculate the cost required to create an imaginary android app for a news agency which shows the text, image, and videos. Evaluate the technical feasibility and difficulties to achieve the functionalities for the visually impaired person. Is the recent Android version providing the text to speech conversion APIs?

Case Study 3:

Find what are the major thrust areas in the testing of mobile app? How the testing of the mobile app differs from the normal website testing?