UNIT I: INTRODUCTION TO ANDROID

1.1 Introduction to Android platform and Architecture

1.2 Basic components of Android, activity life cycle

1.3 Features of ART & Dalvik Virtual machine

1.4 Android Application Structure

1.5 device screen size compatibility

1.6 Android emulator

1.7 working with AndroidManifest.xml

1.1 Introduction to Android platform and Architecture

Android architecture is a software stack of components to support mobile device needs. Android software stack contains a Linux Kernel, collection of c/c++ libraries which are exposed through an application framework services, runtime, and application.

Following are main components of android architecture those are.

1. Applications
2. Android Framework
3. Android Runtime
4. Platform Libraries
5. Linux Kernel

Applications

The top layer of the android architecture is **Applications**. The native and third-party applications like contacts, email, music, gallery, clock, games, etc.

Application Framework

The **Application Framework** provides the classes used to create Android applications.
It also provides a generic abstraction for hardware access and manages the user interface and application resources

The application framework includes services like telephony service, location services, notification manager, NFC service, view system, etc.

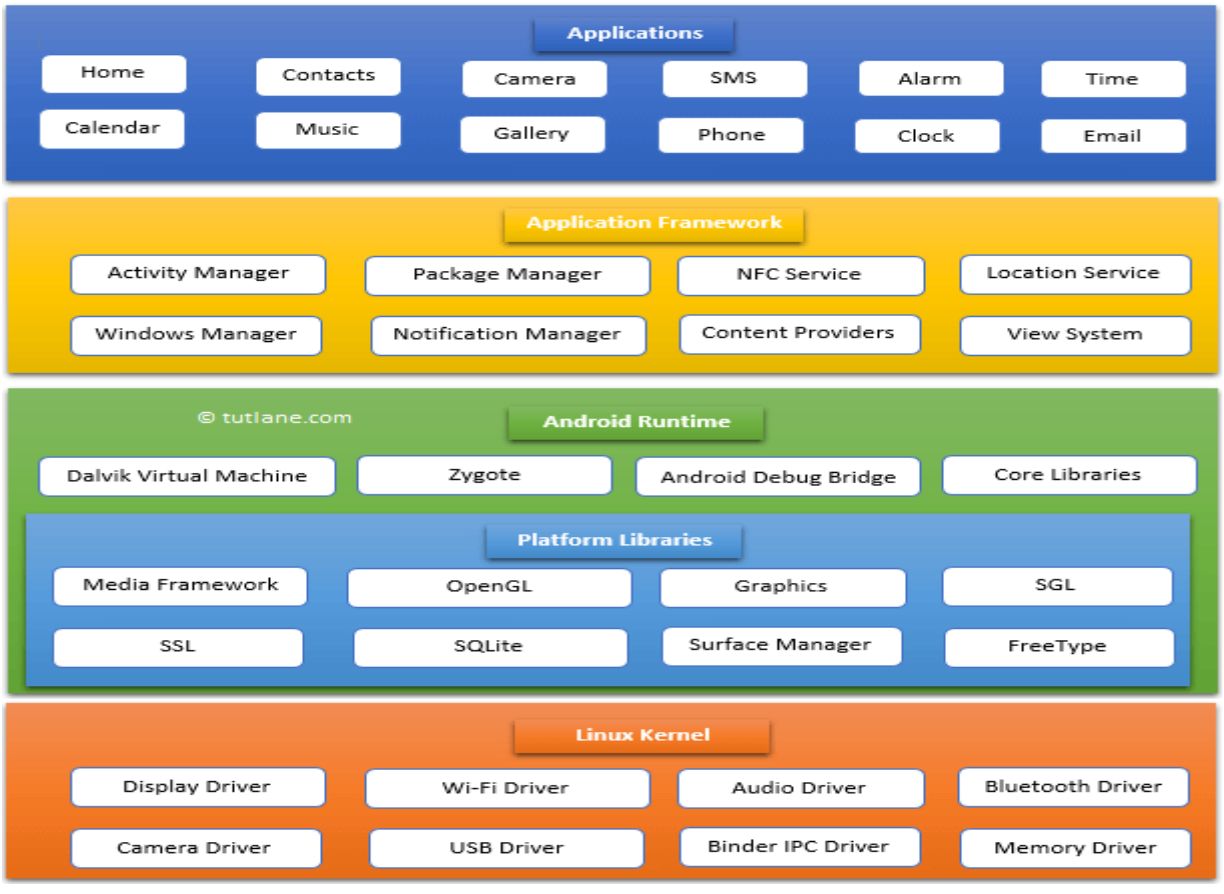
Android Runtime

Android Runtime environment is an important part of Android rather than an internal part and it contains components like **core libraries** and the **Dalvik virtual machine**.

Dalvik Virtual Machine (**DVM**) is a register-based virtual machine like Java Virtual Machine (JVM).
It is specially designed and optimized for android to ensure that a device can run multiple instances efficiently.
It relies on the Linux kernel for threading and low-level memory management.

Linux Kernel

Linux Kernel is a bottom layer and heart of the android architecture.
It manages all the drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are mainly required for the android device during the runtime.



1.2 A] Basic components of Android

Application components are the basic building blocks of an application and these components will act as an entry point to allow system or user to access our app.
The following are the basic core application components that can be used in Android application.

- Activities
- Intents
- Content Providers
- Broadcast Receivers
- Services

All these application components are defined in the android app description file

Android Activities

In android, Activity represents a single screen with a user interface (UI) and it will acts an entry point for the user's to interact with app.

Android Intents

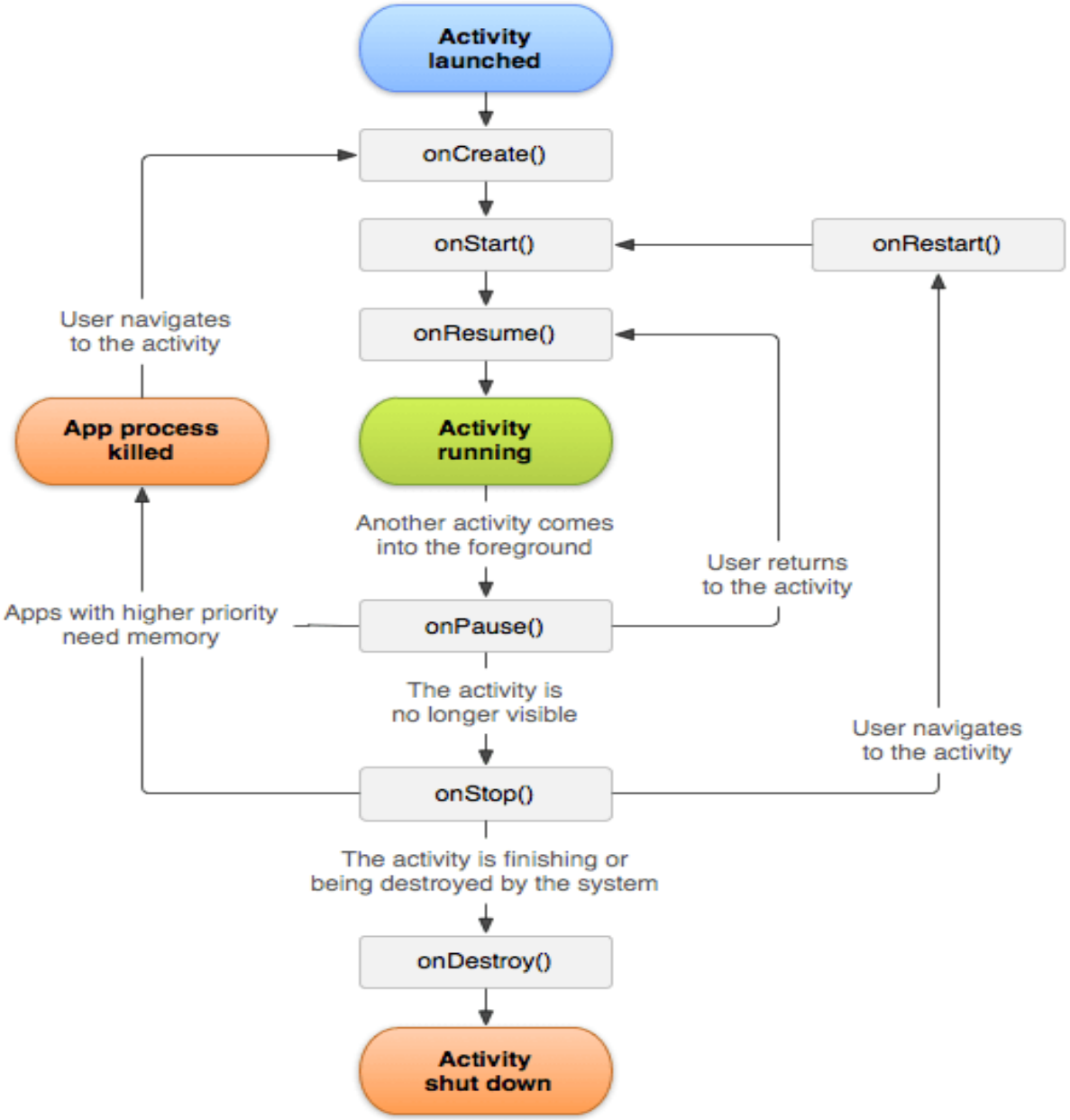
In android, Intent is a messaging object which is used to request an action from another component.
Intents are mainly used to perform the following things.

- Starting an Activity
- Starting a Service
- Delivering a Broadcast

Android Services

Service is a component that keeps an app running in the background to perform long-running operations based on our requirements. For Service, we don't have any user interface and it will run the apps in background like p
Lay music in background when the user in different app.

B] Activity life cycle



An activity represents a single screen with a user interface just like window or frame of Java. Android activity is the subclass of ContextThemeWrapper class. By the help of activity, you can place all your UI components or widgets in a single screen.

The 7 lifecycle method of Activity describes how activity will behave at different states.

1. onCreate : called when activity is first created.

```
@Override protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
}
```

2. onStart : called when activity is becoming visible to the user.

@Override **protected void** onStart() { **super**.onStart(); }
3. onResume : called when activity will start interacting with the user.

@Override **protected void** onResume() { **super**.onResume(); }
4. onPause : called when activity is not visible to the user.

@Override **protected void** onPause() { **super**.onPause(); }
5. onStop : called when activity is no longer visible to the user.

@Override **protected void** onStop() { **super**.onStop(); }
6. onRestart : called after your activity is stopped, prior to start.

7. onDestroy : called before the activity is destroyed.

1.3 Features of ART & Dalvik Virtual machine

Dalvik Virtual Machine

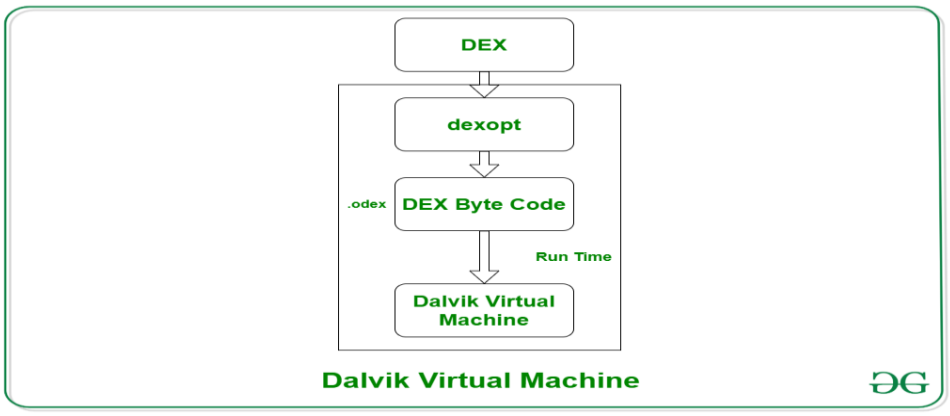
DVM is a Register-Based virtual machine that was designed and written by Dan Bornstein.

Dalvik is a discontinued process virtual machine (VM) in the Android OS that executes applications written for Android.

Dalvik bytecode format is still used as a distribution format, but no longer at runtime in newer Android versions.

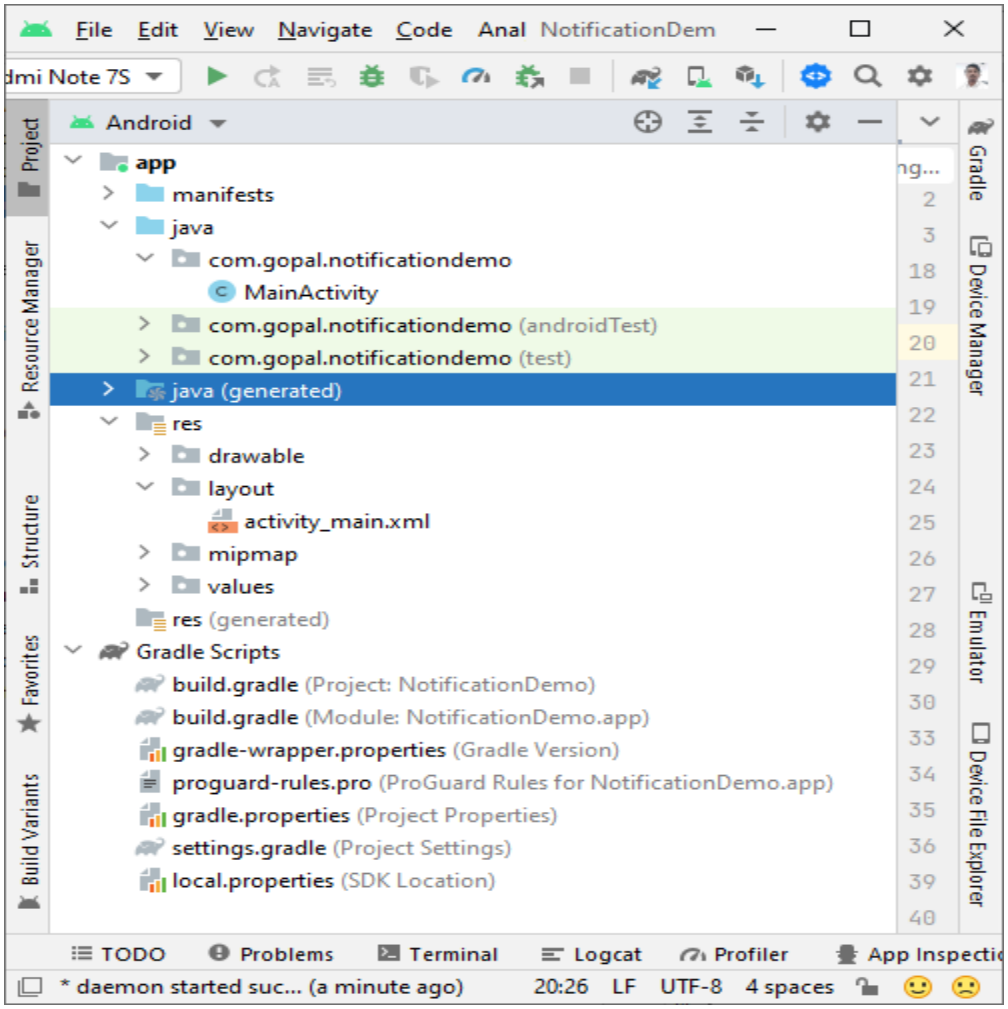
There are 2 types of files:

- **.dex(Dalvik Executable file)** file is an android's compiled code file. These .dex files are then zipped into a single .apk file.
- **.odex** file is created by the Android operating system to save space and increase the boot speed of an Android app (a .apk file).



1.4 Android Application Structure

Android Studio is the official IDE (Integrated Development Environment) developed by Jet Brains community which is freely provided by Google for android app development. After completing the setup of Android Architecture we can create android application in the studio.



The android project contains different types of app modules, source code files, and resource files.

1. Manifests Folder
2. Java Folder
3. res (Resources) Folder
   - Drawable Folder
   - Layout Folder
   - Mipmap Folder
   - Values Folder
4. Gradle Scripts

Manifests Folder

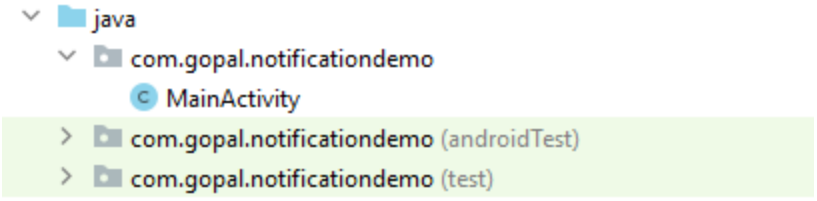
Manifests folder contains **AndroidManifest.xml** for our creating the android application. This file contains information about our application such as application components. It acts as an intermediate between android OS and our application. Following is the manifests folder structure in android application.

activity_main.xml | MainActivity.java | AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.gopal.notificationdemo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="NotificationDemo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.NotificationDemo">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Java folder

Java folder contains all the java source code (.java) files that we create during the app development, including other Test files. If we create any new project using Java, by default the class file MainActivity.java file will create automatically under the package name “com.gopal.notificationdemo”.



Resource (res) folder

The resource folder is the most important folder because it contains all the non-code sources like images, XML layouts, UI strings for our android application.

Res/drawable folder

It contains the different types of images used for the development of the application. We need to add all the images in a drawable folder for the application development.

Res/layout folder

The layout folder contains all XML layout files which we used to define the user interface of our application. It contains the **activity_main.xml** file.

Res/mipmap folder

This folder contains launcher.xml files to define icons that are used to show on the home screen. It contains different density types of icons depending upon the size of the device such as hdpi, mdpi, xhdpi.

Res/values folder

Values folder contains a number of XML files like strings, dimensions, colors and styles definitions. One of the most important file is **strings.xml** file which contains the resources.

Gradle Scripts folder

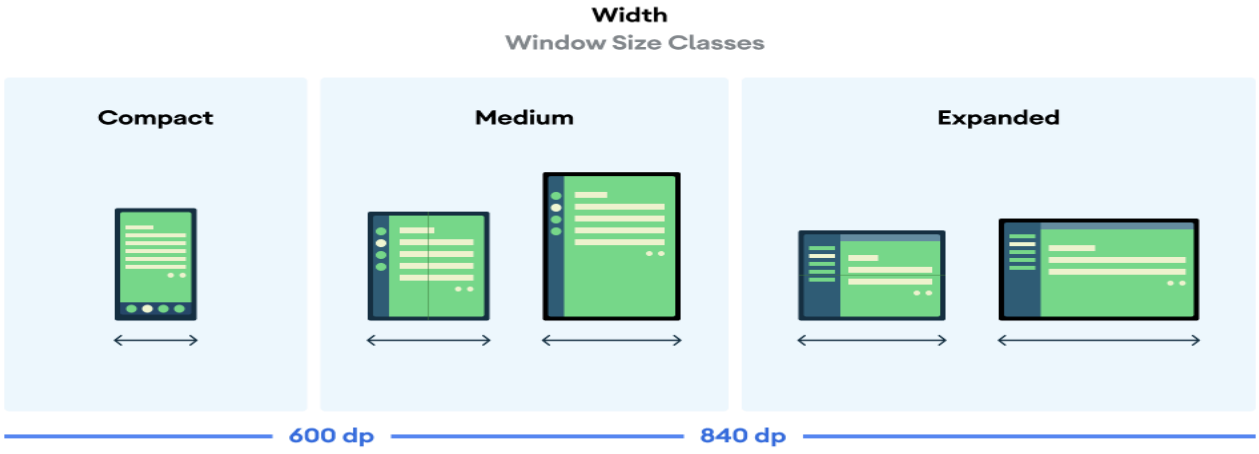
Gradle means automated build system and it contains number of files which are used to define a build configuration which can be apply to all modules in our application.

1.5 device screen size compatibility

Support for different screen sizes gives your app access to the greatest number of users and widest variety of devices.

To support as many screen sizes as possible, design your app layouts to be responsive and adaptive.

Responsive/adaptive layouts provide an optimized user experience regardless of screen size, enabling your app to accommodate phones, tablets, foldable and Chrome OS devices, portrait and landscape orientations, and resizable configurations such as multi-window mode.



1. Responsive design

The best way to create a responsive layout is to use Constraint Layout as the base layout in your UI.

Constraint Layout enables you to specify the position and size of each view according to spatial relationships with other views in the layout. All the views can then move and resize together as the screen size changes.

Responsive width and height

To ensure that your layout is responsive to different screen sizes, use wrap_content, match_parent, or 0dp (match constraint) for the width and height of most view components instead of hard-coded values:

- wrap_content — enables the view to set its size to whatever is necessary to fit the content contained in the view.
- match_parent — enables the view to expand as much as possible within the parent view.
- 0dp (match constraint) — In a ConstraintLayout, similar to match_parent. Enables the view to take all the available space within the view's constraints.

2. Responsive design

- Ensuring your layout can be adequately resized to fit the screen
- Providing appropriate UI layout according to screen configuration
- Ensuring the correct layout is applied to the correct screen
- Providing bitmaps that scale correctly
- **Screen size :** Actual physical size, measured as the screen's diagonal.For simplicity, Android groups has four generalized sizes: small, normal, large, and extra large.
- **Screen density :** The quantity of pixels within a physical area of the screen; usually referred to as dpi (dots per inch).For simplicity, Android groups has four generalized densities: low, medium, high, and extra high.
- **Orientation :** The orientation of the screen from the user's point of view.In Android, This is either landscape or portrait.
- **Resolution :** The total number of physical pixels on a screen.In Android, we do not work directly with resolution; applications should be concerned only with screen size and density.

1.6 Android emulator

1. The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.
2. The emulator provides almost all of the capabilities of a real Android device.
3. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.
4. Testing your app on the emulator is in some ways faster and easier than doing so on a physical device.
5. The emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.

1.7 working with AndroidManifest.xml

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

- 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 information's. These information's 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.

Elements of the AndroidManifest.xml file

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

1. <manifest>

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

2. <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.

3. <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.

4. <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.

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

4.2 <category>
It adds a category name to an intent-filter.

```
<manifest>
<application>
<activity android:name=".MainActivity" >
</activity>
</application>
</manifest>
```

5. <uses-permission>: This element specifies the Android Manifest permissions that are requested for the purpose of security.

The End