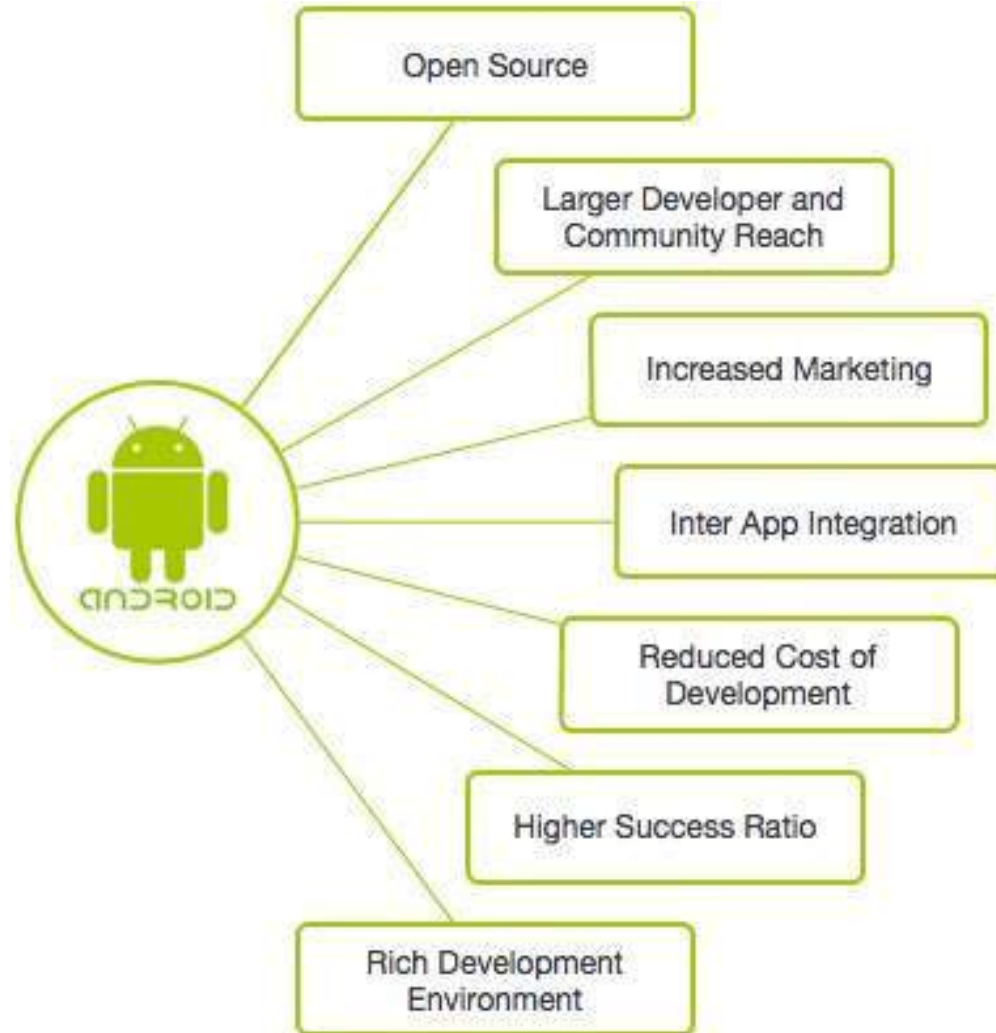


ANDROID STUDIO

INTRODUCTION

- Android is an open source and Linux-based operating system for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.
- Android programming is based on Java programming language so if you have basic understanding on Java programming then it will be a easier to learn Android application development.

Why Android ?



Features of Android

Android is a powerful operating system competing with Apple 4GS, 5Gs and supports great features. Few of them are listed below –

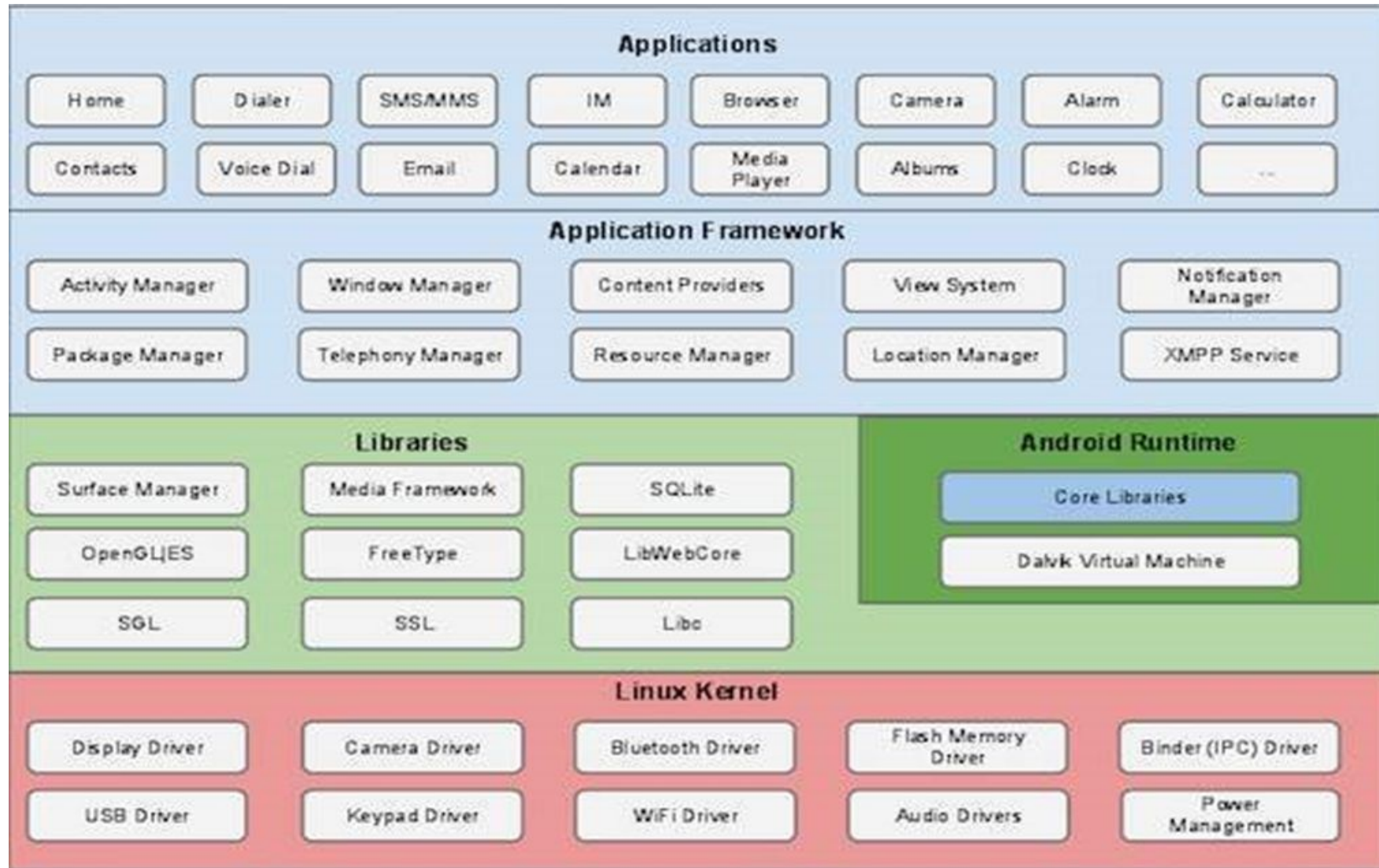
Sr. No.	Feature & Description
1	Beautiful UI Android OS basic screen provides a beautiful and intuitive user interface.
2	Connectivity GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
3	Storage SQLite, a lightweight relational database, is used for data storage purposes.
4	Media support H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
5	Messaging SMS and MMS
6	Web browser Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
7	Multi-touch Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.

Features of Android

Sr. No.	Feature & Description
8	Multi-tasking User can jump from one task to another and same time various application can run simultaneously.
9	Resizable widgets Widgets are resizable, so users can expand them to show more content or shrink them to save space.
10	Multi-Language Supports single direction and bi-directional text.
11	GCM Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.
12	Wi-Fi Direct A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.
13	Android Beam A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.

Layers of Android

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



- XMPP Service-Extensible Messaging and Presence Protocol is an open communication protocol designed for instant messaging, presence information, and contact list maintenance.
- Surface manager responsible for managing access to the display subsystem.
- SGL(Scalable Graphics Library) and OpenGL ES both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- SQLite provides database support
- FreeType provides font support.
- **Web-Kit** This open source web browser engine provides all the functionality to display web content and to simplify page loading.
- **SSL (Secure Sockets Layer)** is security technology to establish an encrypted link between a web server and a web browser.
- Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently.

Application Components

There are following four main components that can be used within an Android application –

Sr.No	Components & Description
1	Activities They dictate the UI and handle the user interaction to the smart phone screen.
2	Services They handle background processing associated with an application.
3	Broadcast Receivers They handle communication between Android OS and applications.
4	Content Providers They handle data and database management issues.

- **Activities** An activity represents a single screen with a user interface, in short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of **Activity** class as follows –

```
public class MainActivity extends Activity { }
```

- **Services** A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of **Service** class as follows –

```
public class MyService extends Service { }
```

- **Broadcast Receivers** Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and each message is broadcaster as an **Intent** object.

```
public class MyReceiver extends BroadcastReceiver { public void onReceive(context,intent){} }
```

- **Content Providers** A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider { public void onCreate(){} }
```

- **Additional Components**

S.No	Components & Description
1	Fragments Represents a portion of user interface in an Activity.
2	Views UI elements that are drawn on-screen including buttons, lists forms etc.
3	Layouts View hierarchies that control screen format and appearance of the views.
4	Intents Messages wiring components together.
5	Resources External elements, such as strings, constants and drawable pictures.
6	Manifest Configuration file for the application.

The Manifest File

Whatever component you develop as a part of your application, you must declare all its components in a *manifest.xml* which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. For example, a default manifest file will look like as following file –

```
<?xml version="1.0" encoding="utf-8"?> <manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.androidapp.myapplication"> <application android:allowBackup="true"
android:icon="@mipmap/ic_launcher" android:label="@string/app_name" android:supportsRtl="true"
android:theme="@style/AppTheme"> <activity android:name=".MainActivity"> <intent-filter> <action
android:name="android.intent.action.MAIN" /> <category android:name="android.intent.category.LAUNCHER"
/> </intent-filter> </activity> </application> </manifest>
```

Here <application>...</application> tags enclosed the components related to the application. Attribute *android:icon* will point to the application icon available under *res/drawable-hdpi*. The application uses the image named *ic_launcher.png* located in the drawable folders.

The Manifest File –contd..

The `<activity>` tag is used to specify an activity and *android:name* attribute specifies the fully qualified class name of the *Activity* subclass and the *android:label* attributes specifies a string to use as the label for the activity. You can specify multiple activities using `<activity>` tags.

The **action** for the intent filter is named *android.intent.action.MAIN* to indicate that this activity serves as the entry point for the application. The **category** for the intent-filter is named *android.intent.category.LAUNCHER* to indicate that the application can be launched from the device's launcher icon.

The *@string* refers to the *strings.xml* file explained below. Hence, *@string/app_name* refers to the *app_name* string defined in the *strings.xml* file, which is "HelloWorld". Similar way, other strings get populated in the application.e that the application can be launched from the device's launcher icon.

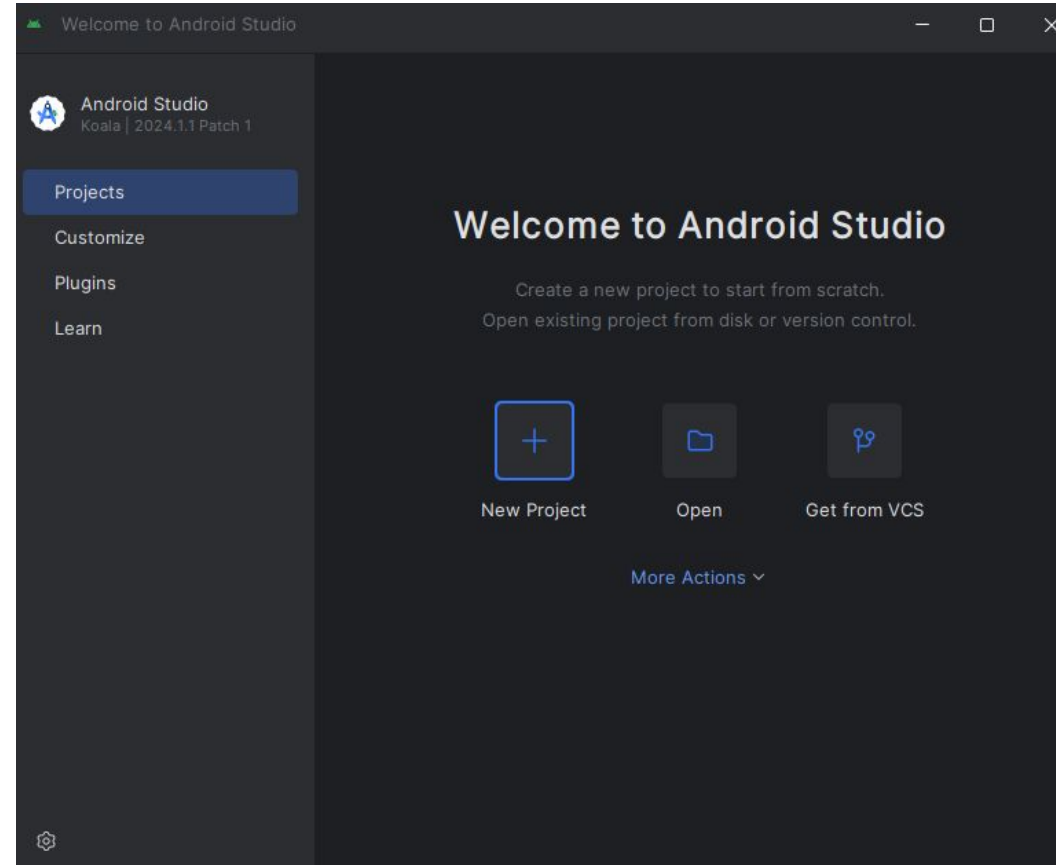
Installation of Android Studio

To install Android Studio on Windows, follow these steps:

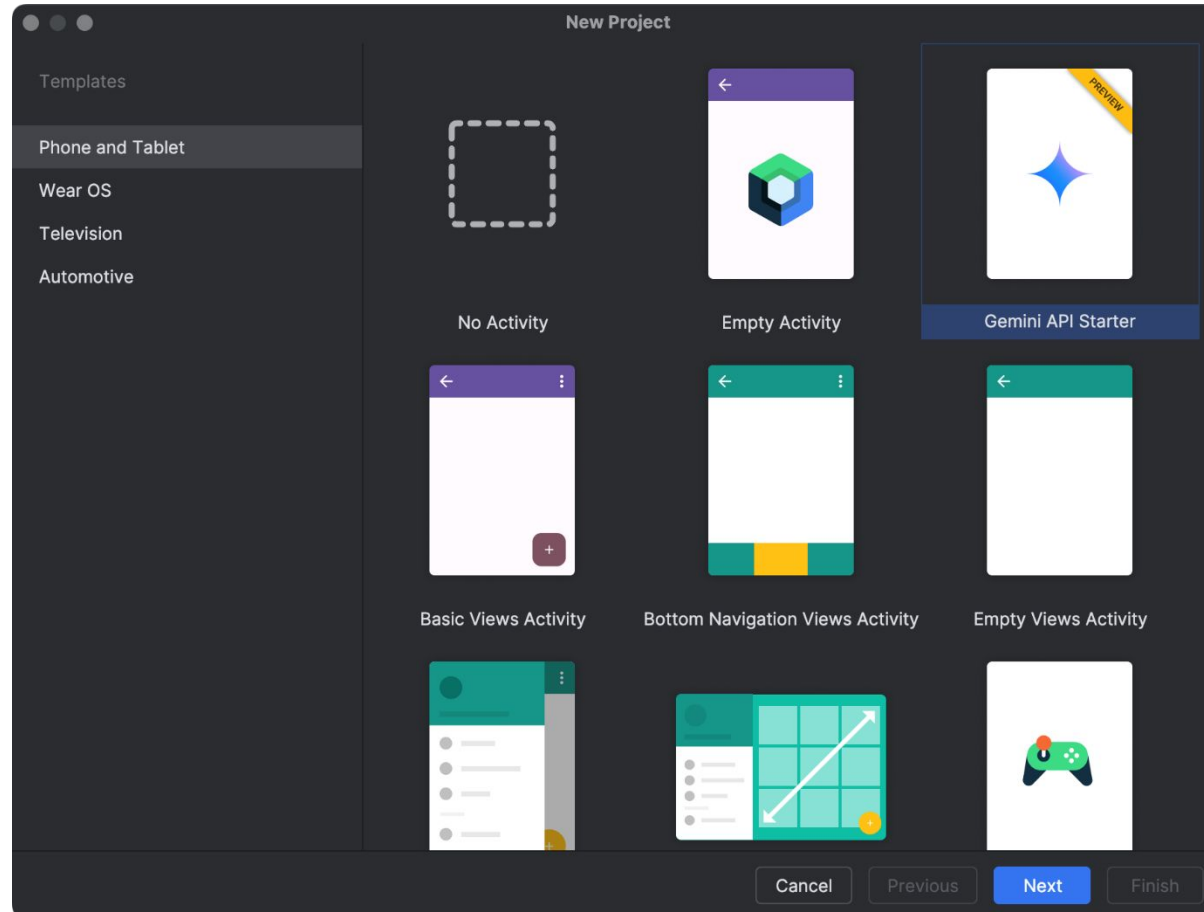
- If you downloaded an `.exe` file (recommended), double-click to launch it.
- If you downloaded a `.zip` file:
 1. Unpack the `.zip`.
 2. Copy the **android-studio** folder into your **Program Files** folder.
 3. Open the **android-studio > bin** folder.
 4. Launch `studio64.exe` (for 64-bit machines) or `studio.exe` (for 32-bit machines).
 5. Follow the **Setup Wizard** in Android Studio and install any recommended SDK packages.

Creating an android application

- To create a new project perform the following steps:
- **Step 1:** Firstly, open the Android Studio. You will see “Welcome to Android Studio” on your computer screen. After that you click on “New Project”.

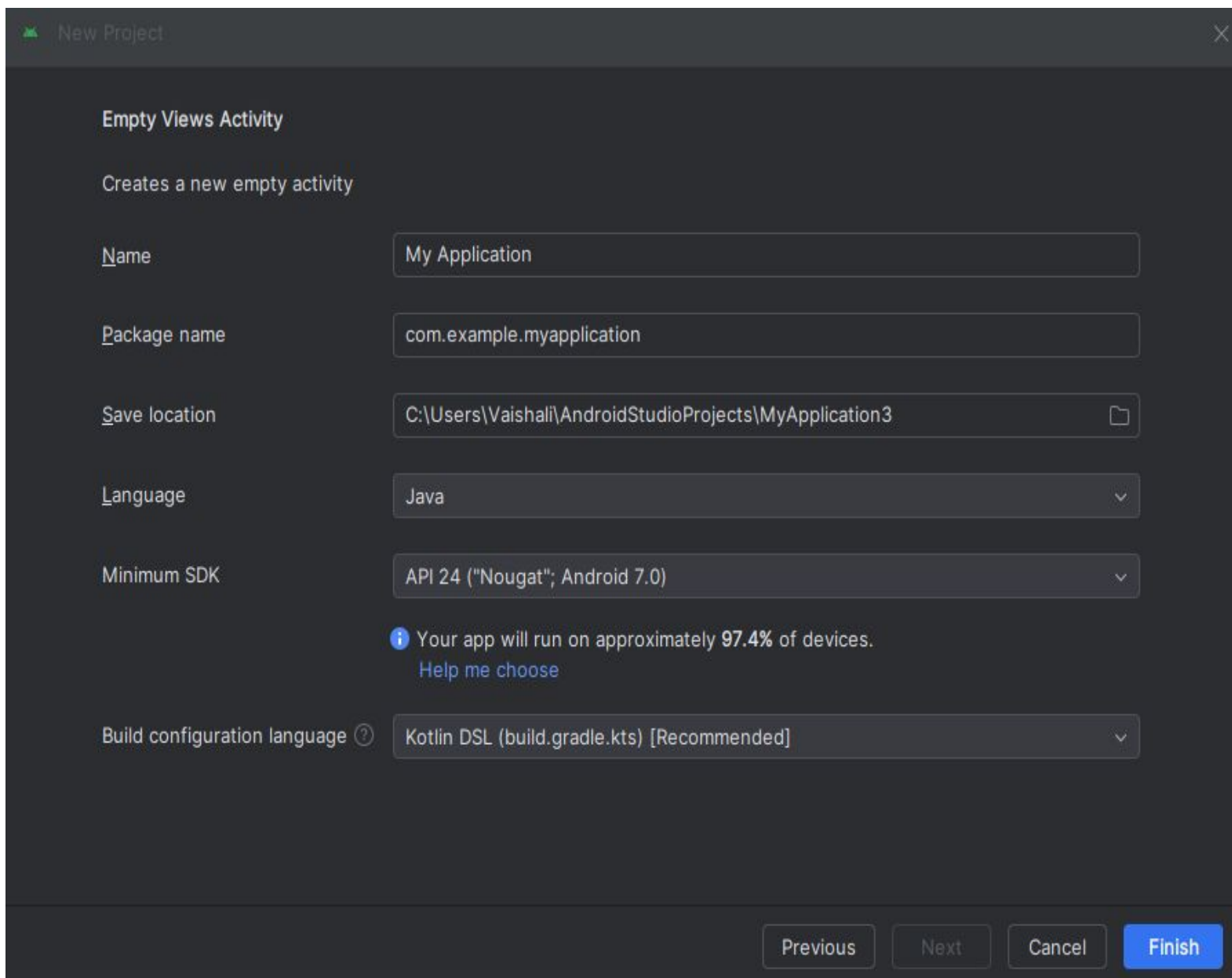


In the **New Project** screen that appears, you can select the type of project you want to create from categories of device form factors, shown in the **Templates** pane. Select Empty Views activity, click Next.



Configure your project

- The next step in creating your project is to configure some settings.
- Specify the **Name** of your project.
- Specify the **Package name**. By default, this package name becomes your project's namespace (used to access your project resources) and your project's application ID (used as the ID for publishing).
- Specify the **Save location** where you want to locally store your project.
- Select the **Language**, Kotlin or Java, you want Android Studio to use when creating sample code for your new project.
- Select the **Minimum API level** you want your app to support.



The screenshot shows the 'New Project' dialog in Android Studio. The title bar says 'New Project' with a close button. The dialog is titled 'Empty Views Activity' and has a subtitle 'Creates a new empty activity'. It contains several input fields and dropdown menus:

- Name:** My Application
- Package name:** com.example.myapplication
- Save location:** C:\Users\Vaishali\AndroidStudioProjects\MyApplication3 (with a folder icon)
- Language:** Java (dropdown menu)
- Minimum SDK:** API 24 ("Nougat"; Android 7.0) (dropdown menu)
- Build configuration language:** Kotlin DSL (build.gradle.kts) [Recommended] (dropdown menu)

Below the 'Minimum SDK' field, there is an information icon and text: 'Your app will run on approximately 97.4% of devices.' with a link 'Help me choose'.

At the bottom right, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

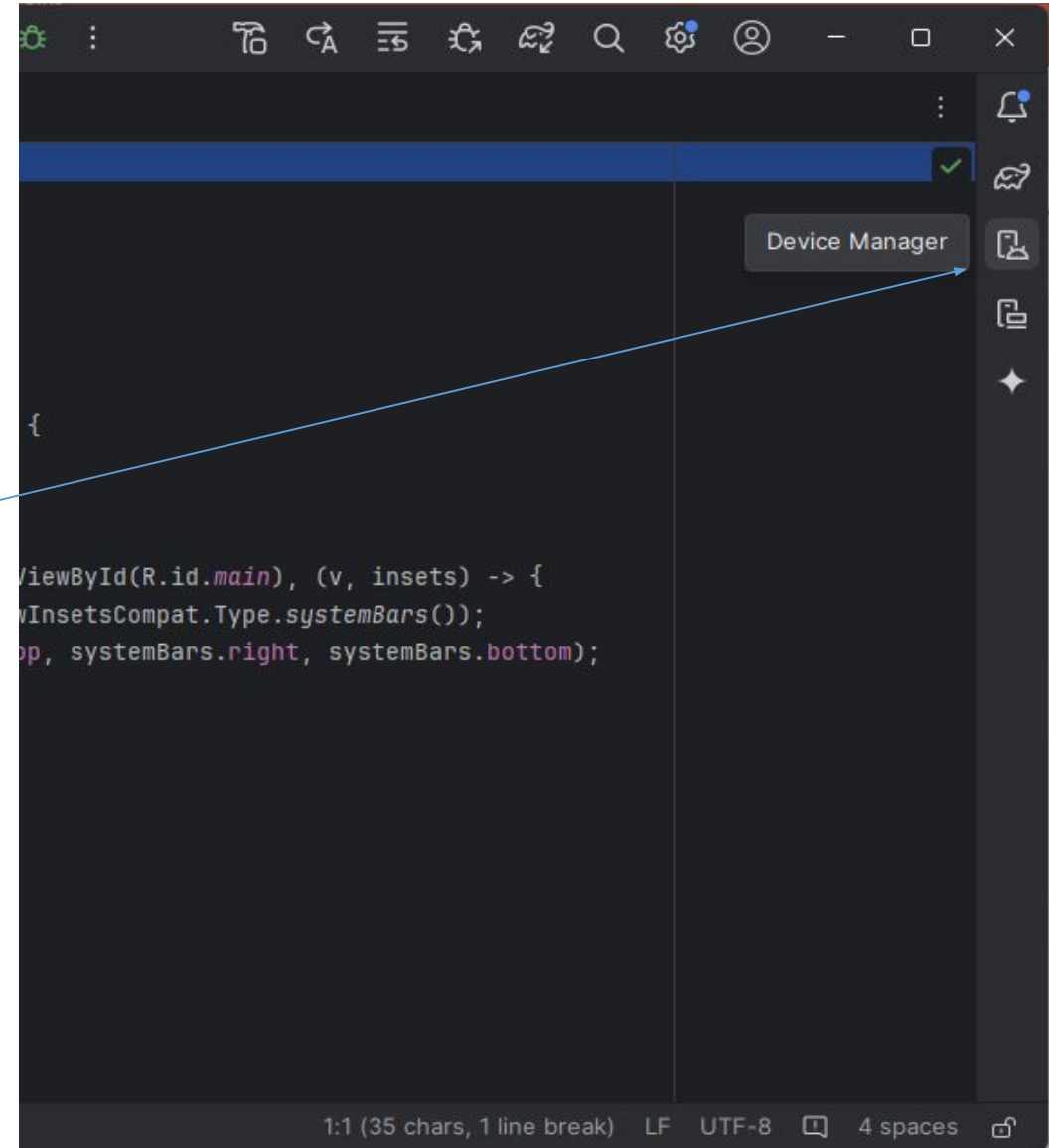
Create and manage virtual devices

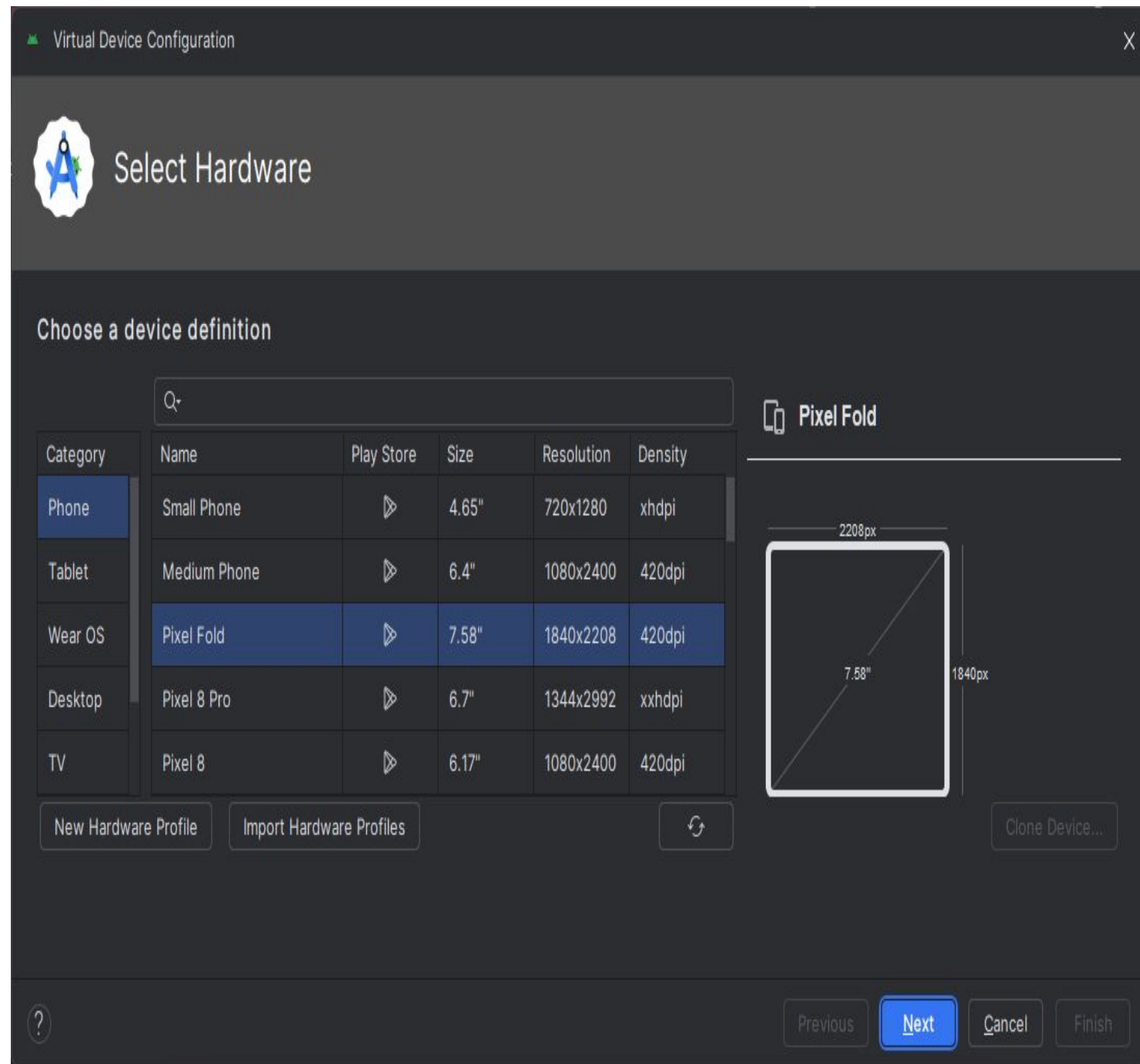
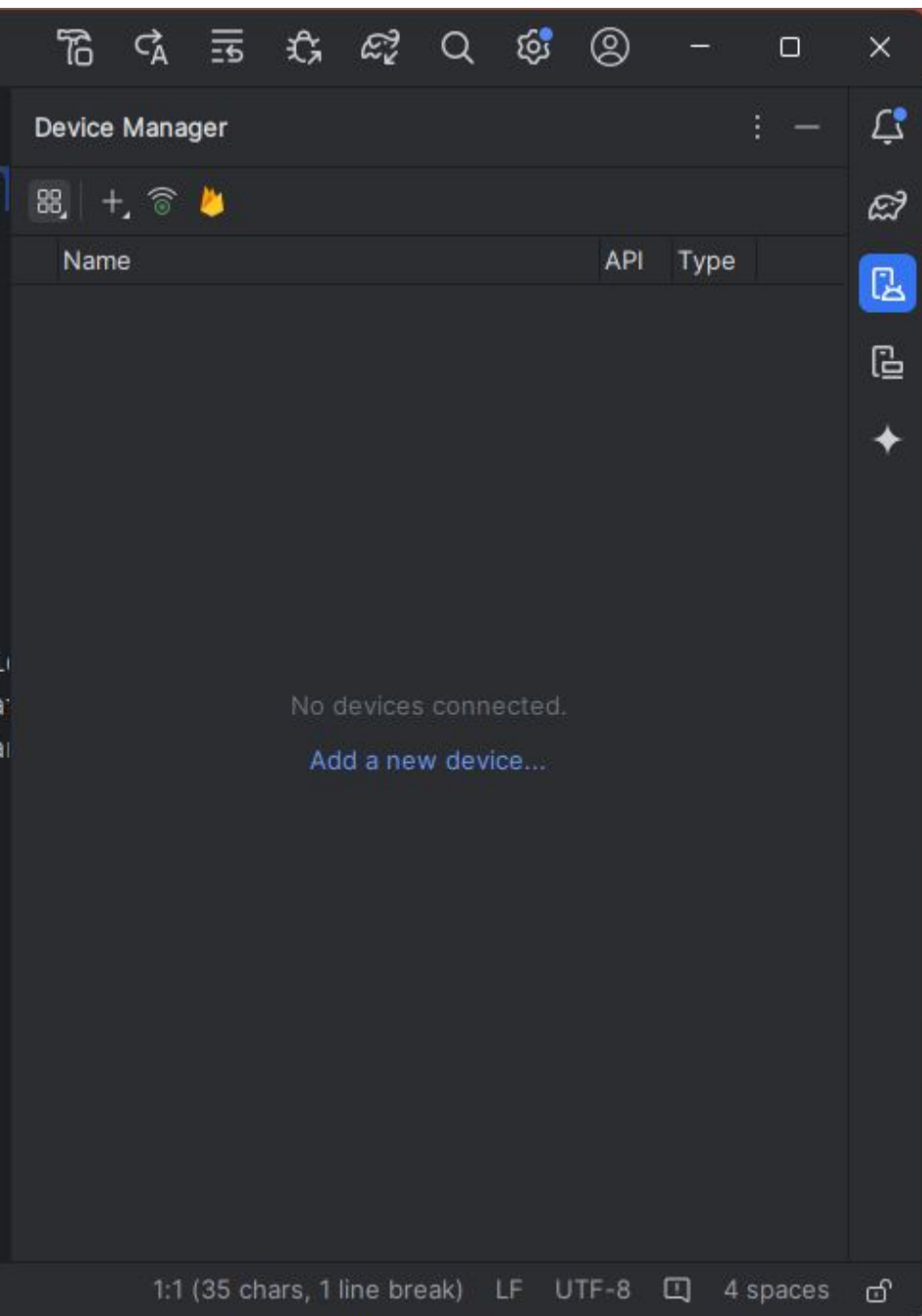
An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator.

The Device Manager is a tool you can launch from Android Studio that helps you create and manage AVDs.

To open the new **Device Manager**
Alternately

After opening a project, select View > Tool Windows > Device Manager from the main menu bar, then click the +, and then click Create Virtual Device.







System Image

Select a system image

Recommended

x86 Images

Other Images

Release Name	API ▾	ABI	xABI	Target
Oreo	27	x86		Android 8.1 (Google)
Oreo	26	x86		Android 8.0 (Google)
Nougat	25	x86		Android 7.1.1 (Google)
Nougat	24	x86		Android 7.0 (Google)
Marshmallow	23	x86		Android 6.0 (Google)
Lollipop	22	x86		Android 5.1 (Google)



Oreo



API Level

27

Type

Google APIs

Android

8.1

Google Inc.



Previous

Next

Cancel

Finish



Android Virtual Device (AVD)

Verify Configuration

AVD name: Pixel 4 XL API 27



Pixel 4 XL

6.3 1440x3040 560dpi

Change...



Oreo

Android 8.1 x86

Change...

Preferred ABI: Optimal ▾

Startup orientation:



Show Advanced Settings

AVD Name

The name of this AVD.



Previous

Next

Cancel

Finish



MA My Application ▾

Version control ▾

Pixel 4 XL API 27 ▾

app ▾



Android ▾

> app

> Gradle Scripts

</> activity_main.xml

© MainActivity.java ×

Run 'app' Shift+F10

Running Devices

Pix...



1 package com.example.myapplication; ✓

2

3 > import ...

10

11 </> public class MainActivity extends AppCompatActivity {

12

13 @Override

14 protected void onCreate(Bundle savedInstanceState) {

15

16 super.onCreate(savedInstanceState);

17 EdgeToEdge.enable(\$this\$enableEdgeToEdge: this);

18 setContentView(R.layout.activity_main);

19 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) {

20 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

21 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

22 return insets;

23

24 });

25

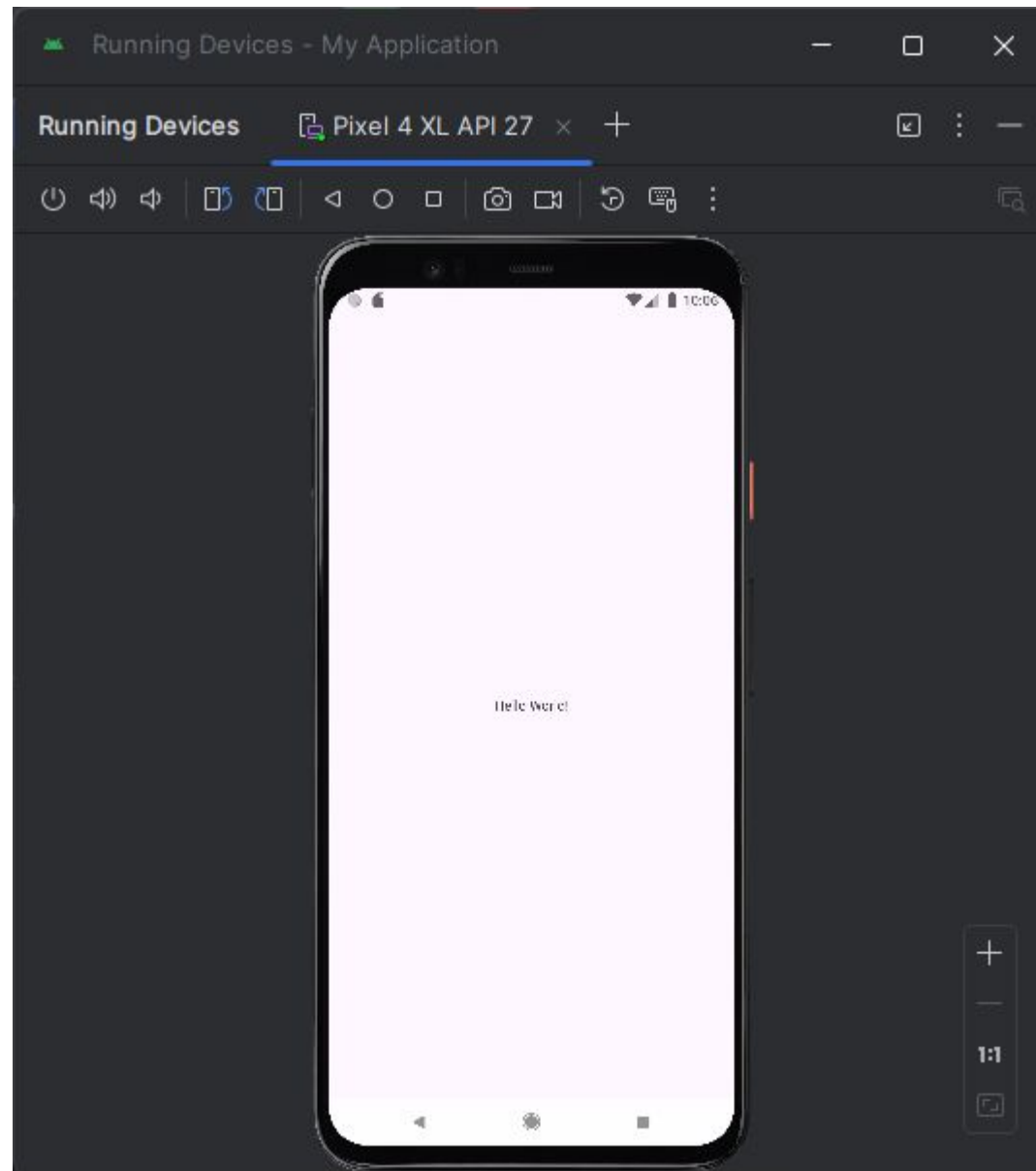
26 }

Power, Volume, Rotation, Camera, etc. icons



MyApplication3 > app > src > main > java > com > example >.myapplication > MainActivity

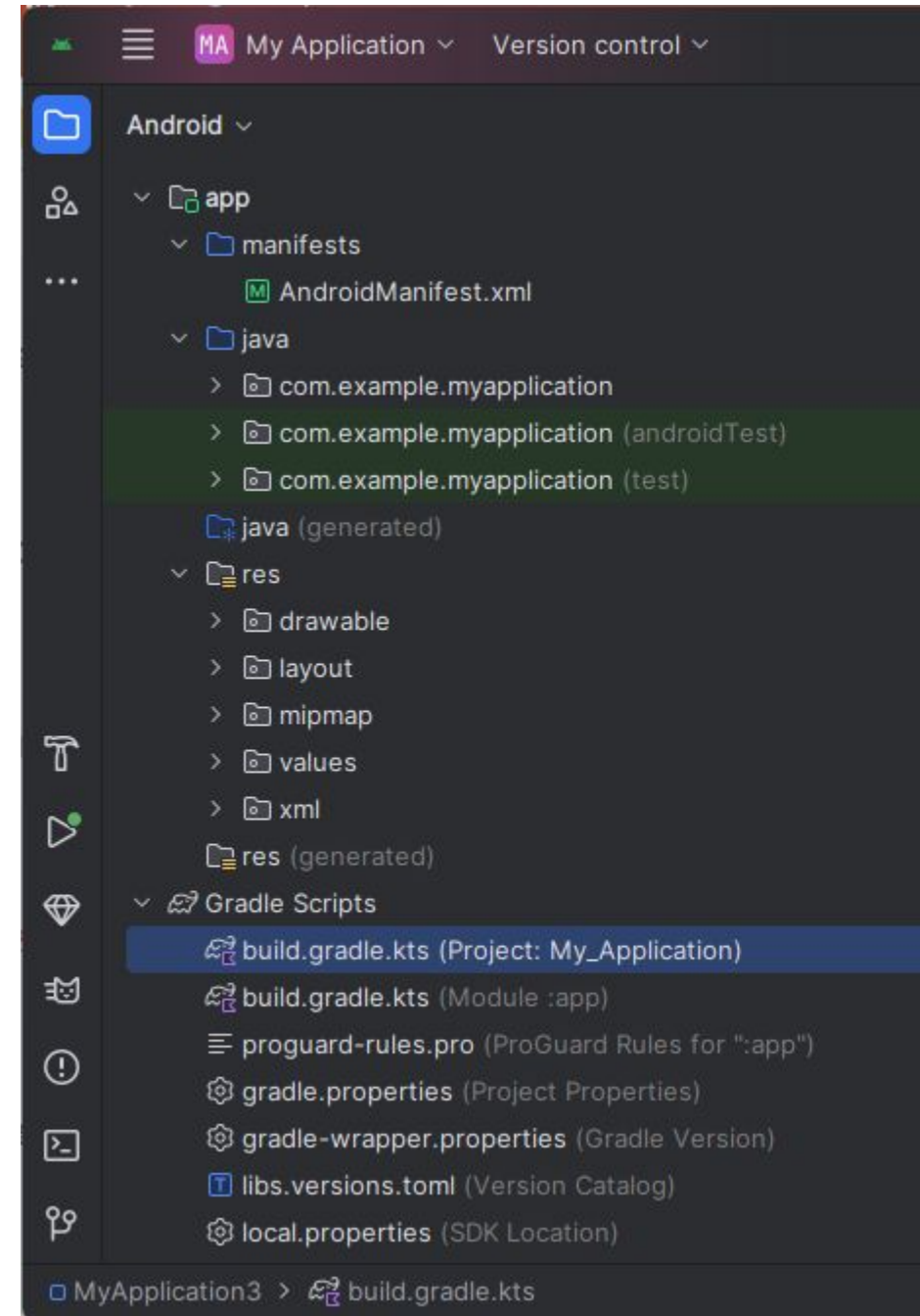
1:1 (35 chars, 1 line break) LF UTF-8 4 spaces



Android Project folder Structure

The android project contains different types of app modules, source code files, and resource files. We will explore all the folders and files in the android app.

- Manifests Folder
- Java Folder
 - Drawable Folder
 - Layout Folder
 - Mipmap Folder
 - Values Folder
- Gradle Scripts



- **Manifests Folder**

Manifests folder contains **AndroidManifest.xml** for creating our android application.

- **Java folder**

The Java folder contains all the java and Kotlin source code (.java) files that we create during the app development, including other Test files.

- **Resource (res) folder**

The resource folder is the most important folder because it contains all the non-code sources like images, XML layouts, and 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 style definitions. One of the most important files is the **strings.xml** file which contains the resources.

- **Gradle Scripts folder**

- Gradle means automated build system and it contains a number of files that are used to define a build configuration that can be applied to all modules in our application.
- In build.gradle (Project) there are build scripts and in build.gradle (Module) plugins and implementations are used to build configurations that can be applied to all our application modules.

Create your First Android Application

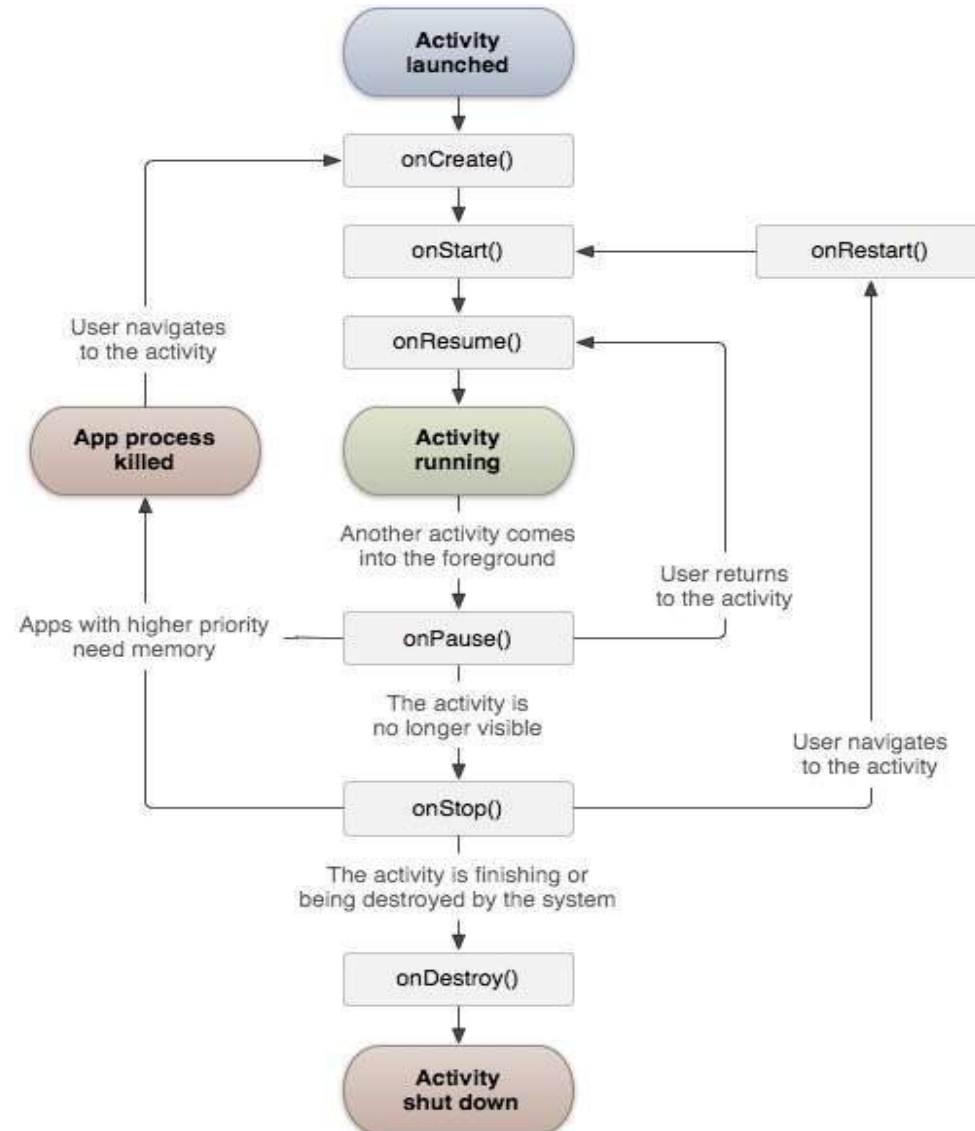
- Reference Video

<https://www.youtube.com/watch?v=uPkjgVv5loc&t=1131s>

CREATING THE ACTIVITY

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.

Activity life cycle diagram:



The Activity class defines the following call backs i.e. events. You don't need to implement all the callbacks methods.

Sr.No	Callback & Description
1	onCreate() This is the first callback and called when the activity is first created.
2	onStart() This callback is called when the activity becomes visible to the user.
3	onResume() This is called when the user starts interacting with the application.
4	onPause() The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
5	onStop() This callback is called when the activity is no longer visible.
6	onDestroy() This callback is called before the activity is destroyed by the system.
7	onRestart() This callback is called when the activity restarts after stopping it.