



Installation of Android Studio

Prerequisite : Java

Steps:

- Check java is installed or not: java -version on cmd
- If not install the Java
- Download android studio : <https://developer.android.com/studio>
- Install it.
- **How to Install Android Studio on Windows 10:**
https://www.youtube.com/watch?v=0zx_eFyHRU0

With Screenshot :

<https://www.c-sharpcorner.com/article/how-to-download-and-install-android-studio-in-windows-10/>



What is HAXM....

HAXM stands for "**Hardware Accelerated Execution Manager**". It is used for launching Emulators and must be installed and in usable status. Please note that Emulator launching means Virtualisation. So, we need to ensure that Intel Hardware to launch Virtualisation Technology(VT) is enabled in our machine.

Error 1 : Not installed

<https://stackoverflow.com/questions/32011025/cant-install-intel-haxm-for-android-studio-error-x86-emulation-currently-requi>

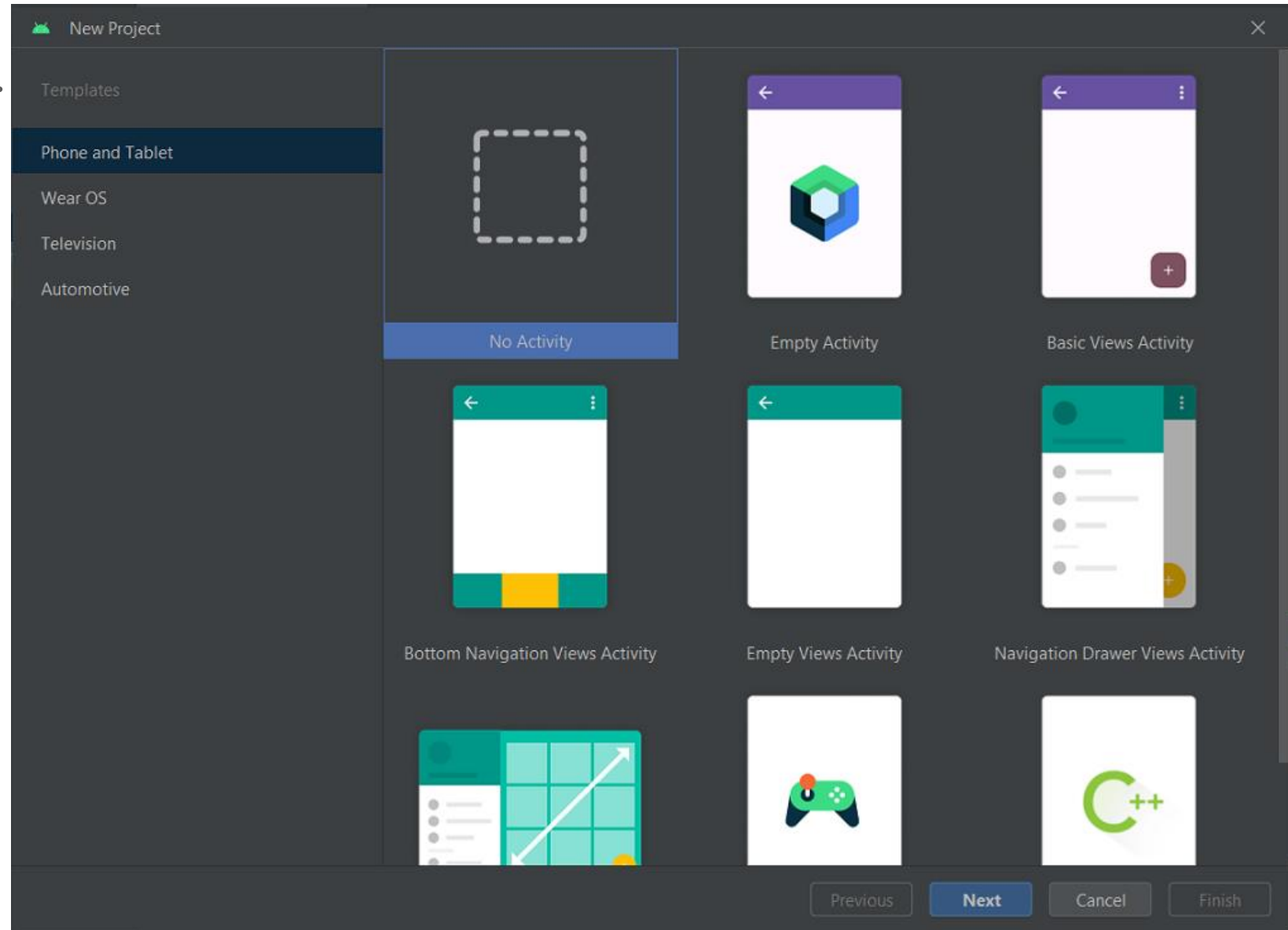


Creating an android application



Step 1: Create a new project

1. Open Android Studio.
2. File->New->New Project.
3. Select No Activity.
Click Next.





New Project

No Activity

Creates a new empty project

Name: My Application

Package name: com.example.myapplication

Save location: C:\Users\admin\AndroidStudioProjects\MyApplication4

Language: Java

Minimum SDK: API 24 ("Nougat"; Android 7.0)

i Your app will run on approximately **95.4%** of devices.
[Help me choose](#)

Build configuration language *?*: Kotlin DSL (build.gradle.kts) [Recommended]

Previous Next Cancel Finish

4. Give your application a name such as My First App.

5. Make sure the Language is set to Java

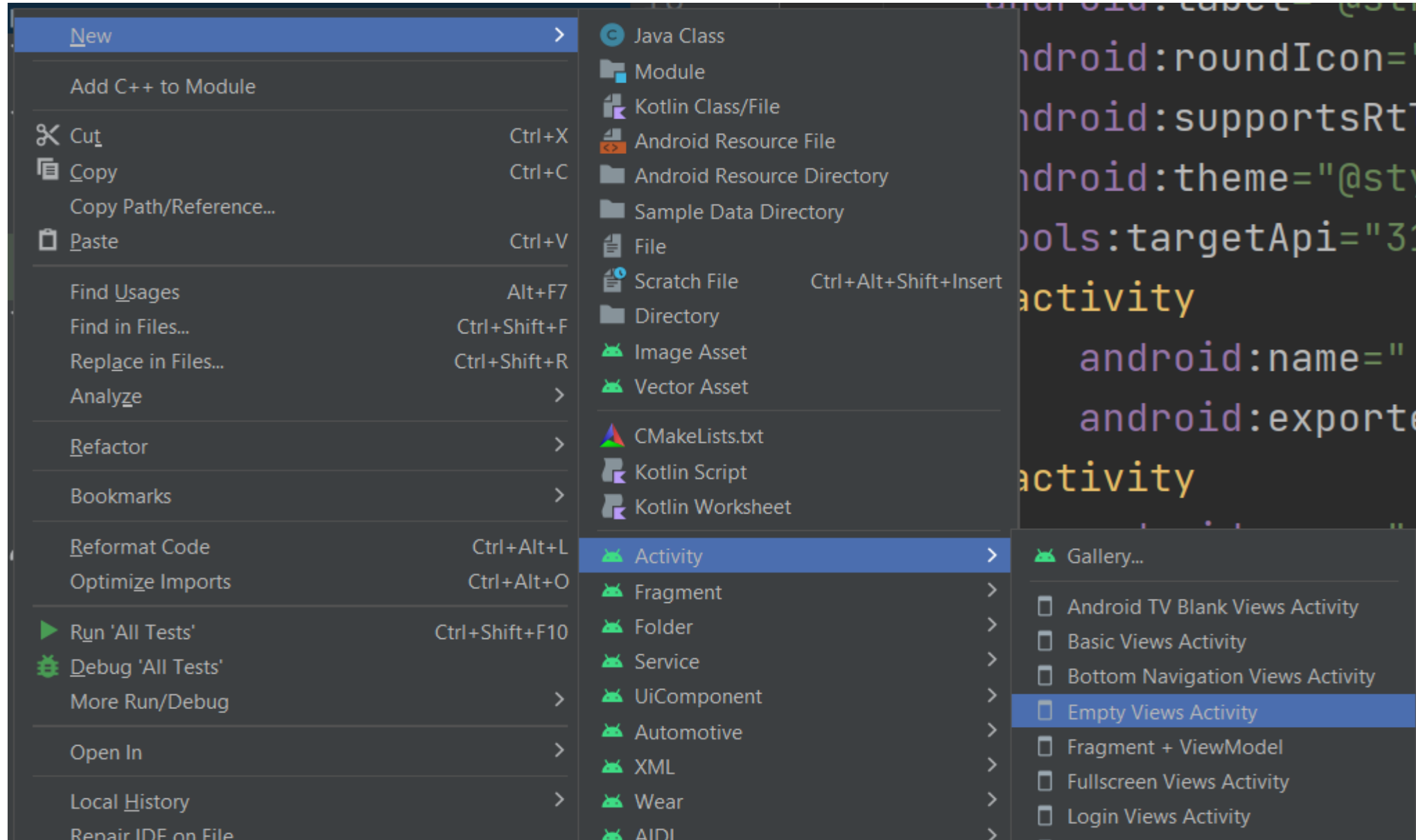
6. Leave the defaults for the other fields.

7. Click Finish.



Step 2: To create new activity

New->Activity->Empty Views Activity





Select/ Check
Launcher
Activity.
Click Finish.

New Android Activity

Empty Views Activity

Creates a new empty activity

Activity Name

MainActivity2

☒ Generate a Layout File

Layout Name

activity_main2

☒ Launcher Activity

Package name

com.example.myapplication

Source Language

Java

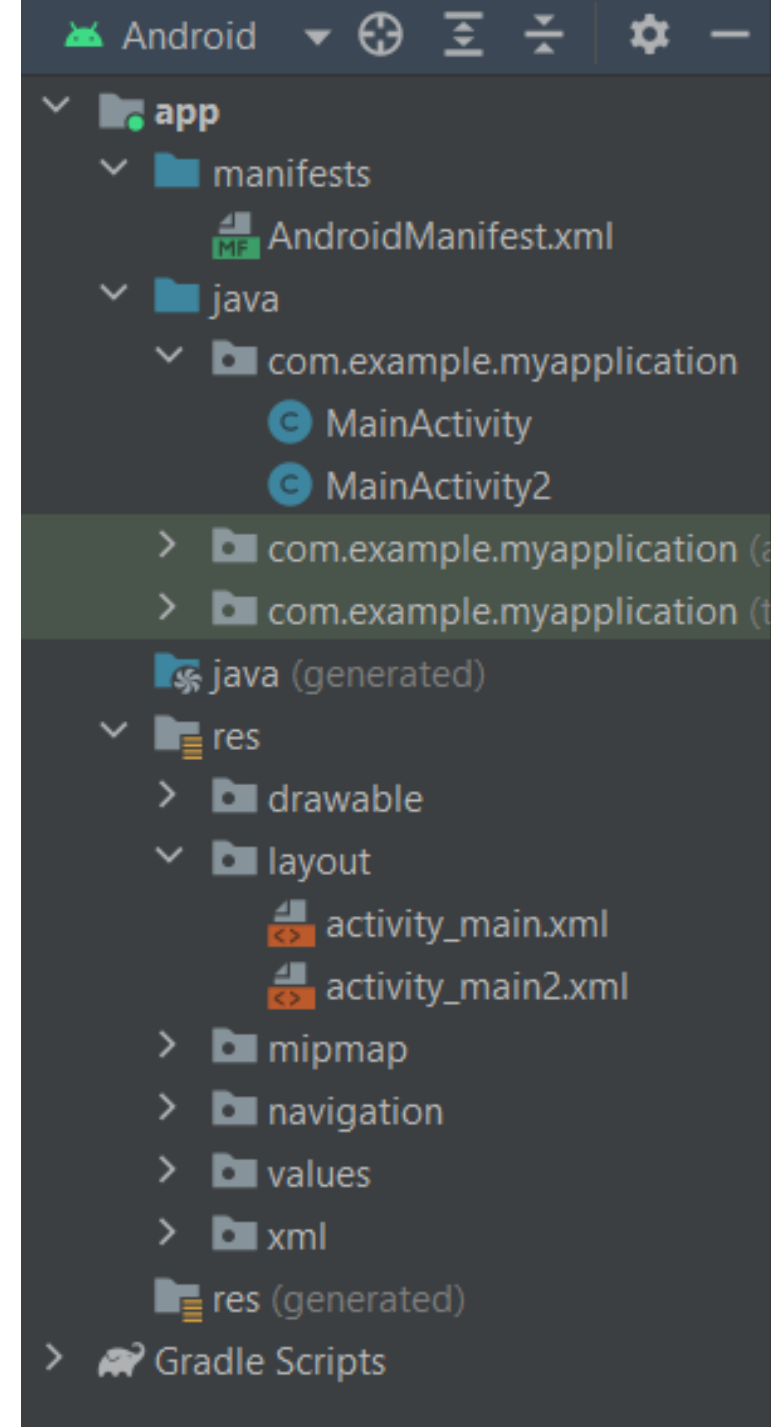
Previous Next Cancel Finish

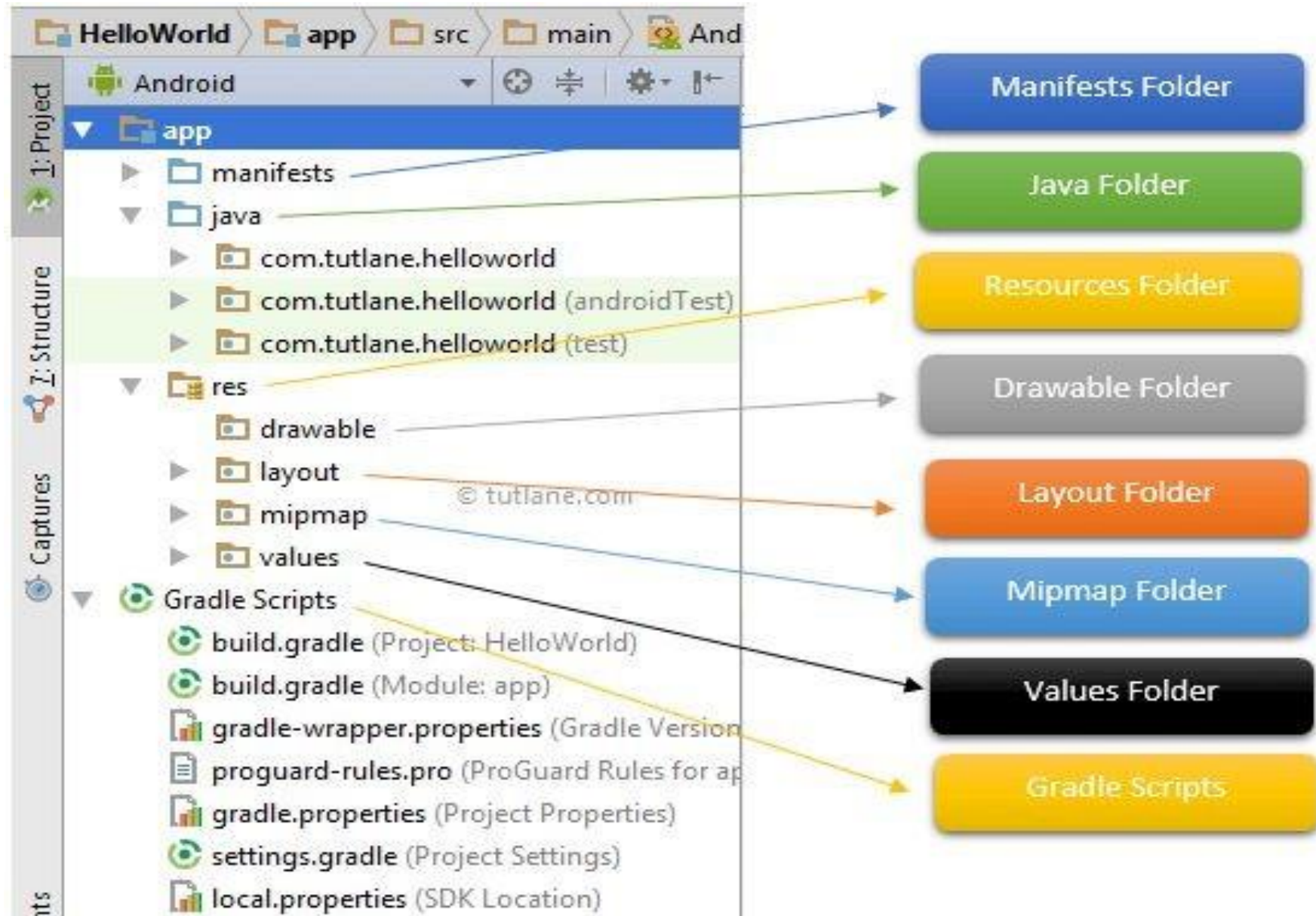
AndroidManifest.xml



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/title_activity_main"
            android:theme="@style/Theme.MyApplication">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```


Explore the project structure and layout







UNDERSTANDING THE ANDROIDMANIFEST.XML FILE

- Every project in Android includes a manifest file, which is [AndroidManifest.xml](#), stored in the root directory of its project hierarchy.
- The manifest file is an important part of our app because it defines the structure and metadata of our application, its components, and its requirements.
- This file includes nodes for each of the Activities, Services, Content Providers and Broadcast Receiver that make the application and using Intent Filters and Permissions, determines how they coordinate with each other and other applications.



UNDERSTANDING THE ANDROIDMANIFEST.XML FILE

The AndroidManifest.xml file contains information of your package, including components of the application such as

- **Activities**
- **Services**
- **Broadcast**
- **Receivers**
- **Content providers etc.**

It performs some other tasks also:

- 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. These information are removed just before the application is published etc.



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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="TwoActivities"
        android:roundIcon="@mipmap/ic_launcher_round"
        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>
        <activity android:name=".SecondActivity"></activity>
    </application>

</manifest>
```

[<uses-permission />](#)
[<permission />](#)
[<permission-tree />](#)
[<permission-group />](#)
[<instrumentation />](#)
[<uses-sdk />](#)
[<uses-configuration />](#)
[<uses-feature />](#)
[<supports-screens />](#)
[<compatible-screens />](#)
[<supports-gl-texture />](#)



ELEMENTS OF THE ANDROIDMANIFEST.XML FILE

<manifest> : manifest is the root element of the AndroidManifest.xml file.

It has package attribute that describes the package name of the activity class.

<application> : application is the sub element of the manifest.

It includes the namespace declaration.

This element contains several sub elements 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.



ELEMENTS OF THE ANDROIDMANIFEST.XML FILE

<activity> : activity is the sub element 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.

<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.

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

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



- **Java:** The Java folder contains the Java source code files. **These files are used as a controller for controlled UI (Layout file).** It gets the data from the Layout file and after processing that data output will be shown in the UI layout. **It works on the backend of an Android application.**
- **drawable:** A Drawable folder contains **resource type file (something that can be drawn).** Drawables may take a variety of file like Bitmap (PNG, JPEG), **Nine Patch, Vector (XML), Shape, Layers, States, Levels, and Scale.**
- **layout:** **A layout defines the visual structure for a user interface, such as the UI for an Android application.** This folder **stores Layout files that are written in XML language.** You can add additional layout objects or widgets as child elements to gradually build a View hierarchy that defines your layout file.

Sample layout file



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:backgroundTint="#9F2D9D"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="To"
        android:inputType="text"
        android:padding="10dp" />
```



```
<EditText
```

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:hint="Message"  
    android:inputType="textMultiLine"  
    android:padding="10dp" />
```

```
<Button
```

```
    android:id="@+id/ba"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="right"  
    android:backgroundTint="#85445A"  
    android:padding="10dp"  
    android:text="SEND" />
```

```
</LinearLayout>
```



mipmap

Mipmap folder contains the **Image Asset file that can be used in Android Studio application.**

You can generate the following icon types like **Launcher icons, Action bar and tab icons, and Notification icons.**



colors.xml

colors.xml file **contains color resources of the Android application.**

Different color values are identified by a unique name that can be used in the Android application program.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```



strings.xml

- The strings.xml file **contains string resources of the Android application.**
- The different **string value is identified by a unique name** that can be used in the Android application program.
- This file also stores string array by using XML language.
- Putting the strings in string resource
 - **allows your app to support multiple languages**(You can define string resource file for each language that you want to support)
 - **prevents duplication in the case wheres you use the same string multiple times**



```
<resources>
    <string name="app_name">Linear Layout</string>
    <string name="title_activity_main">MainActivity</string>
    <!-- Strings used for fragments for navigation -->
    <string name="first_fragment_label">First Fragment</string>
    <string name="second_fragment_label">Second Fragment</string>
    <string name="next">Next</string>
    <string name="previous">Previous</string>
    <string name="to">
        To
    </string>
    <string name="subject">
        Subject
    </string>
</resources>
```

themes.xml



The themes.xml file contains **resources of the theme style in the Android application.**

This file is written in XML language.

```
<resources xmlns:tools="http://schemas.android.com/tools">

    <style name="Theme.MyApplication" parent="Base.Theme.MyApplication">
        <!-- Transparent system bars for edge-to-edge. -->
        <item name="android:navigationBarColor">@android:color/transparent</item>
        <item name="android:statusBarColor">@android:color/transparent</item>
        <item name="android:windowLightStatusBar">?attr/isLightTheme</item>
    </style>
</resources>
```



- Android Studio uses **Gradle to build android projects.**
- Gradle is a **build system (open source)** which is used to **automate and manage the build process.**
- Gradle is responsible for **code compilation, testing, deployment and conversion of the code into .dex files** and hence running the app on the device.
- A gradle **takes all the source files (java and XML)** and apply appropriate tools, e.g., **converts the java files into dex files** and **compresses all of them into a single file known as apk** that is actually used.
- “Build.gradle” are **scripts where one can automate the tasks.**
- For example, the simple task to copy some files from one directory to another can be performed by Gradle build script before the actual build process happens.

Note: APK stands for Android Package Kit and it's the file format that Android uses for its apps.



Types of build.gradle scripts

- **Top-level build.gradle**

It is **located in the root project directory** and its main function is to **define the build configurations that will be applied to all the modules in the project.**

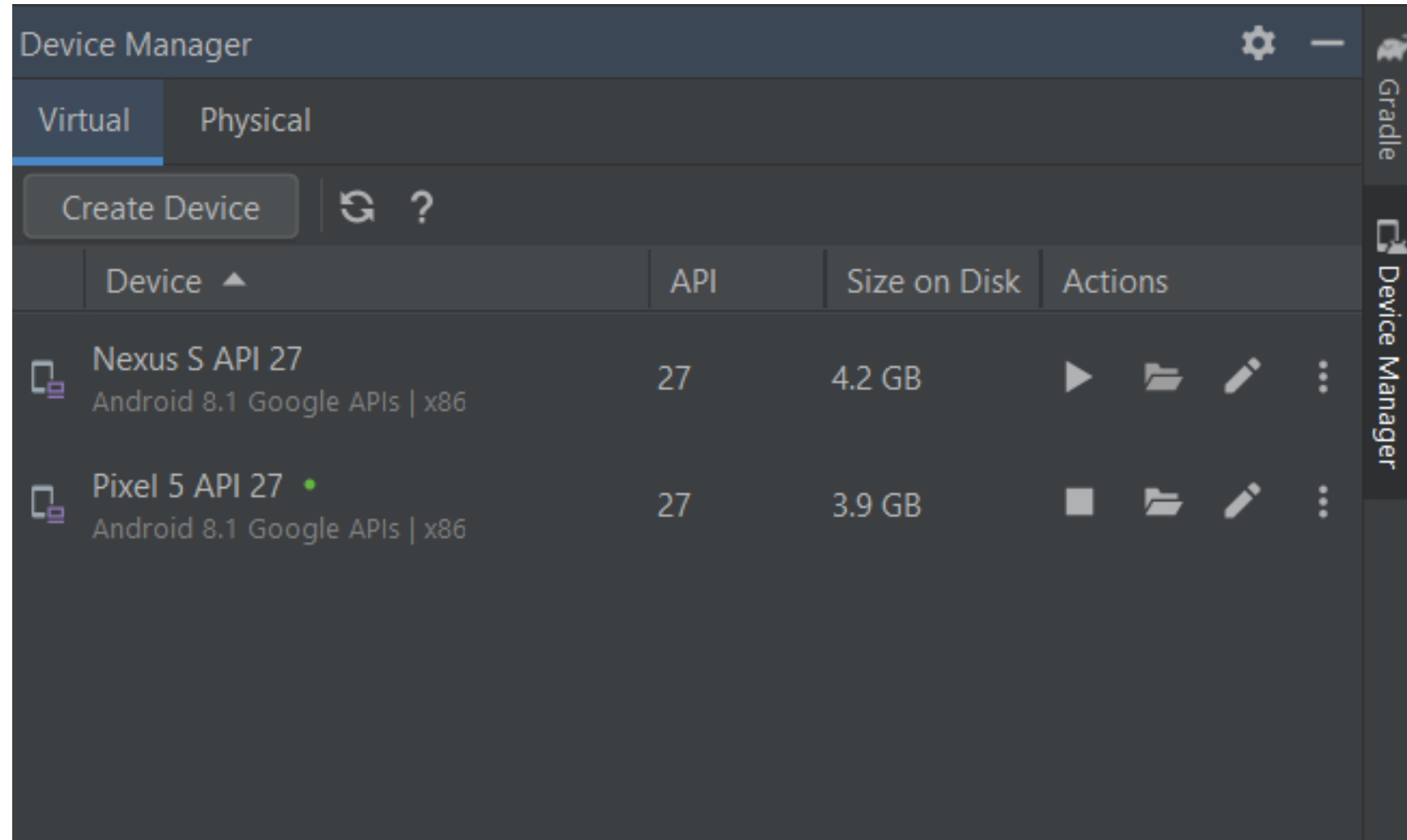
- **Module-level build.gradle**

- **Located in the project/module directory of the project** this Gradle script is where **all the dependencies are defined and where the sdk versions are declared.**
- This script has many functions in the project which includes additional build types and override settings in the main/app manifest or *top-level build.gradle* file.



Create a virtual device (emulator)

- Click Device Manager
- Click Create Device





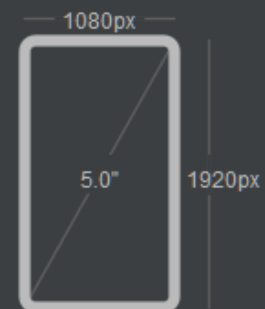
Select Hardware



Choose a device definition

Category	Name ▾	Play Store	Size	Resolution	Density
Phone	Resizable (Experiment...		6.0"	1080x2340	420dpi
Tablet	Pixel XL		5.5"	1440x2560	560dpi
Wear OS	Pixel 7 Pro		6.71"	1440x3120	560dpi
Desktop	Pixel 7		6.31"	1080x2400	420dpi
TV	Pixel 6a		6.13"	1080x2400	420dpi
Automotive	Pixel 6 Pro		6.7"	1440x3120	560dpi
	Pixel 6		6.4"	1080x2400	420dpi

Pixel 2



Size: large
Ratio: long
Density: 420dpi

New Hardware Profile

Import Hardware Profiles



Clone Device...



Previous

Next

Cancel

Finish



Select Hardware



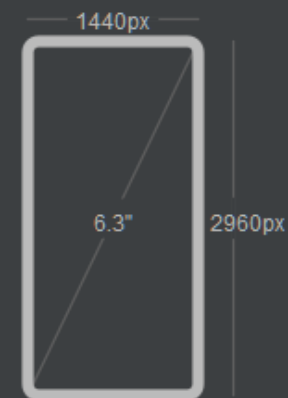
Choose a device definition



Category	Name ▾	Play Store	Size	Resolution	Density
Phone	Pixel 4 XL		6.3"	1440x3040	560dpi
Tablet	Pixel 4		5.7"	1080x2280	440dpi
Wear OS	Pixel 3a XL		6.0"	1080x2160	400dpi
Desktop	Pixel 3a		5.6"	1080x2220	440dpi
TV	Pixel 3 XL		6.3"	1440x2960	560dpi
Automotive	Pixel 3		5.46"	1080x2160	440dpi
	Pixel 2 XL		5.99"	1440x2880	560dpi



Pixel 3 XL



Size: large
Ratio: long
Density: 560dpi

New Hardware Profile

Import Hardware Profiles



Clone Device...




Previous

Next


Cancel

Finish

Select
System
image
You may
need to
download
it.



System Image




Select a system image

Recommendedx86 ImagesOther Images

Release Name	API Level ▾	ABI	Target
Sv2 ⬇	32	x86_64	Android 12L (Google APIs)
S ⬇	31	x86_64	Android 12.0 (Google APIs)
R ⬇	30	x86	Android 11.0 (Google APIs)
Q ⬇	29	x86	Android 10.0 (Google APIs)
Pie ⬇	28	x86	Android 9.0 (Google APIs)
Oreo	27	x86	Android 8.1 (Google APIs)
Oreo ⬇	26	x86	Android 8.0 (Google APIs)
Nougat ⬇	25	x86	Android 7.1.1 (Google APIs)
Nougat ⬇	24	x86	Android 7.0 (Google APIs)

↺

Pie



API Level

28

Android

9.0

Google Inc.

System Image

x86

We recommend these images because they run the fastest and support Google APIs.

! A system image must be selected to continue.

?

PreviousNextCancelFinish

Note: System images can take up a large amount of disk space, so just download what you need



License Agreement



Licenses

▼ android-sdk-arm-dbt-license

↓ Google APIs Intel x86 Atom System

Terms and Conditions

This is the Android Software Development Kit License Agreement

1. Introduction

1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: <http://source.android.com/>, as updated from time to time.

1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (<http://source.android.com/compatibility>) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).

1.4 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.

☐ Decline ☒ Accept

Previous

Next

Cancel

Finish



SDK Component Installer



Completing Requested Actions

SDK Path: C:\Users\admin\AppData\Local\Android\Sdk


```
Packages to install: - Google APIs Intel x86 Atom System Image  
(system-images;android-28;google_apis;x86)
```

```
Preparing "Install Google APIs Intel x86 Atom System Image API 28 (revision 12)".
```

```
Downloading https://dl.google.com/android/repository/sys-img/google_apis/x86-28_r12.zip
```

Downloading x86-28_r12.zip (2%): 19.0 / 948.4 MB ...

https://dl.google.com/android/repository/sys-img/google_apis/x86-28_r12.zip

 Please wait until the requested actions are completed.

Previous

Next

Cancel

Finish



SDK Component Installer

Completing Requested Actions

SDK Path: C:\Users\admin\AppData\Local\Android\Sdk

```
Packages to install: - Google APIs Intel x86 Atom System Image  
(system-images;android-26;google_apis;x86)
```

```
Preparing "Install Google APIs Intel x86 Atom System Image API 26 (revision 16)".  
Downloading https://dl.google.com/android/repository/sys-img/google_apis/x86-26_r16.zip  
"Install Google APIs Intel x86 Atom System Image API 26 (revision 16)" ready.  
Installing Google APIs Intel x86 Atom System Image in  
C:\Users\admin\AppData\Local\Android\Sdk\system-images\android-26\google_apis\x86  
"Install Google APIs Intel x86 Atom System Image API 26 (revision 16)" complete.  
"Install Google APIs Intel x86 Atom System Image API 26 (revision 16)" finished.
```

Done

Previous

Next

Cancel

Finish



Android Virtual Device (AVD)



Verify Configuration

AVD Name

	Pixel 3 XL	6.3 1440x2960 560dpi	<button>Change...</button>
	Oreo	Android 8.1 x86	<button>Change...</button>

Startup orientation

Portrait

Landscape

Emulated Performance Graphics:

Show Advanced Settings

AVD Name

The name of this AVD.



Previous

Next

Cancel

Finish

Click **Finish**. The AVD appears in the **Virtual** tab of the Device Manager



Run your app on your new emulator

After you have created an AVD, you can start the Android Emulator and run an app in your project:

1. In the toolbar, select the AVD that you want to run your app on from the target device menu.
2. Click Run. The emulator might take a minute or so to launch for the first time, but subsequent launches use a [snapshot](#) and should launch faster.

