UNIT 6 ANDROID DEVELOPMENT

6.0 INTRODUCTION

Now you are aware that Android applications can be developed using Android Studio. You have also learnt fundamentals of Android App design. How to build a simple user interface and handle user input will be described further in this unit.

Particularly, you will apply Java features in the context of core Android components (such as Activities and basic UI elements) by applying common tools (such as Android Studio) needed to develop Java programs and useful Android apps.

6.1 OBJECTIVES

After studying this unit, you should be able to :

- Develop and run an Android application.
- Run the developed application in both on the actual device and in the emulator.

Outcomes

AVD:	Android Virtual Emulator
UI:	User Interface
XML:	eXtensible Markup Language

Terminology

6.2 CREATING YOUR FIRST PROGRAM

This unit shows you how to create a new Android project with Android Studio and describes some of the files in the project.

In Android Studio, create a new project

- In the New Project screen, enter the following values:

Application Name: "My First App"
Company Domain: "example.com"

Android Studio fills in the package name and project location for you, but you can edit these if you'd like.

- Click Next.
- In the Target Android Devices screen, keep the default values and click Next.

The Minimum Required SDK is the earliest version of Android that your app supports, which is indicated by the API level. To support as many devices as possible, you should set this to the lowest version available that allows your app to provide its core feature set. If any feature of your app is possible only on newer versions of Android and it’s not critical to the core feature set, enable that feature only when running on the versions that support it.

- In the Add an Activity to Mobile screen, select Empty Activity and click Next.
- In the Customize the Activity screen, keep the default values and click Finish.

After some processing, Android Studio opens and displays a "My First App" app with default files. You will add functionality to some of these files in the following lessons.

Now take a moment to review the most important files. First, be sure that the Project window is open (select View > Tool Windows > Project) and the Android view is selected from the drop-down list at the top. You can then see the following files:

app > java > com.example.myfirstapp > MainActivity.java

This file appears in Android Studio after the New Project wizard finishes. It contains the class definition for the activity you created earlier. When you build and run the app, the Activity starts and loads the layout file that says "Hello World!"

app > res > layout > activity_main.xml

This XML file defines the layout of the activity. It contains a TextView element with the text "Hello world!

app > manifests > AndroidManifest.xml

The manifest file describes the fundamental characteristics of the app and defines each of its components. You'll revisit this file as you follow these lessons and add more components to your app.

Gradle Scripts > build.gradle

Android Studio uses Gradle to compile and build your app. There is a build.gradle file for each module of your project, as well as a build.gradle file for the entire project. Usually, you're only interested in the build.gradle file for the module.

6.2.1 Check Your Progress

- Create a simple “Hello World Program”

6.3 BUILDING AND RUNNING THE APPLICATION

By default, Android Studio sets up new projects to deploy to the Emulator or a physical device with just a few clicks. With Instant Run, you can push changes to methods and existing app resources to a running app without building a new APK, so code changes are visible almost instantly.

To build and run your app, click Run ▶. Android Studio builds your app with Gradle, asks you to select a deployment target (an emulator or a connected device), and then deploys your app to it. You can customize some of this default behaviour, such as selecting an automatic deployment target, by changing the run configuration.

If you want to use the Android Emulator to run your app, you need to have an Android Virtual Device (AVD) ready. If you haven't already created one, then after you click Run, click Create New Emulator in the Select Deployment Target dialog. Follow the Virtual Device Configuration wizard to define the type of device you want to emulate. For more information, see Create and Manage Virtual Devices.

Select and build a different module

If your project has multiple modules beyond the default app module, you can build a specific module as follows:

- Select the module in the Project panel, and then click Build > Make Module *module-name*.

Android Studio builds the module using Gradle. Once the module is built, you can run and debug it if you've built a module for a new app or new device, or use it as a dependency if you've built a library or Google Cloud module.

To run a built app module:

- Click Run > Run and select the module from the Run dialog.

Change the run/debug configuration

The run/debug configuration specifies the module to run, packageto deploy, activity to start, target device, emulator settings, logcat options, and more. The default run/debug configuration launches the default project activity and uses the Select Deployment Target dialog for target device selection. If the default settings don't suit your project or module, you can customize the run/debugconfiguration, or even create a new one, at the project, default, and module levels. To edit a run/debug configuration, select **Run** > **Edit Configurations**.

Change the build variant

By default, Android Studio builds the debug version of your app, which is intended only for testing, when you click Run. You need to build the release version to prepare your app for public release.

To change the build variant Android Studio uses, select **Build** > **Select Build Variant** in the menu bar (or click Build Variants 🤖 in the windows bar), and then select a build variant from the drop-down menu. By default, new projects are set up with a debug and release build variant.

Using *product flavours*, you can create additional build variants for different versions of your app, each having different features or device requirements.

Generate APKs

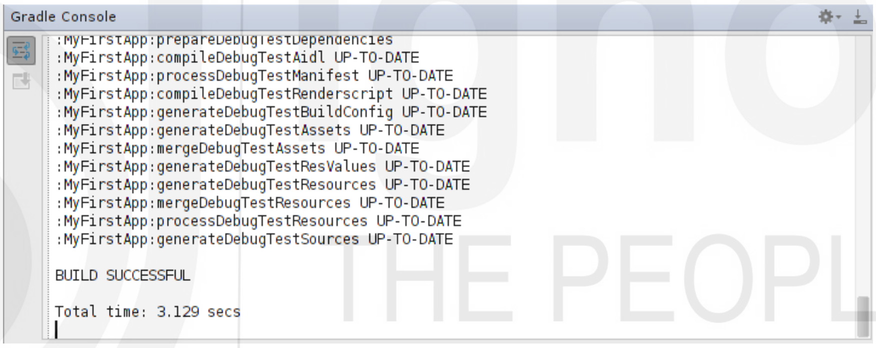
To create an APK for your app, follow these steps:

- Select the build variant you want to build from **the Build Variants** window.
- Click **Build > Build APK** in the menu bar.
- To instead build the APK and immediately run it on a device,click **Run** in the toolbar.

All built APKs are saved in ***project-name***/***module-name***`/build/outputs/apk/`. You can also locate the generated APKs by clicking the link in the pop-up dialog that appears once the build is complete, as shown in figure 2

Monitor the build process

You can view details about the build process by clicking **View > Tool Windows >Gradle Console** (or by clicking **Gradle Console** in the tool window bar). The console displays each task that Gradle executes in order to build your app, as shown in Figure 6.1.



```
:MyFirstApp:prepareDebugTestDependencies
:MyFirstApp:compileDebugTestAidl UP-TO-DATE
:MyFirstApp:processDebugTestManifest UP-TO-DATE
:MyFirstApp:compileDebugTestRenderscript UP-TO-DATE
:MyFirstApp:generateDebugTestBuildConfig UP-TO-DATE
:MyFirstApp:generateDebugTestAssets UP-TO-DATE
:MyFirstApp:mergeDebugTestAssets UP-TO-DATE
:MyFirstApp:generateDebugTestResValues UP-TO-DATE
:MyFirstApp:generateDebugTestResources UP-TO-DATE
:MyFirstApp:mergeDebugTestResources UP-TO-DATE
:MyFirstApp:processDebugTestResources UP-TO-DATE
:MyFirstApp:generateDebugTestSources UP-TO-DATE

BUILD SUCCESSFUL

Total time: 3.129 secs
```

Figure 6.1 The Gradle Console in Android Studio.

If your build variants use product flavours, Gradle also invokes tasks to build those product flavours. To view the list of all available build tasks, click **View > Tool Windows > Gradle** (or click **Gradle** in the tool window bar).

If an error occurs during the build process, the *Messages* window appears to describe the issue. Gradle may recommend some command-line options to help you resolve the issue, such as `--stacktrace` or `--debug`. To use command-line options with your build process:

- Open the **Settings** or **Preferences** dialog:
- Navigate to **Build, Execution, Deployment** > **Compiler**.
- In the text field, next to *Command-line Options*, enter your command-line options.
- Click **OK** to save and exit.

Gradle will apply these command-line options the next time you try building your app.

Running on the Emulator

Before you run your app on an emulator, you need to create an Android Virtual Device (AVD) definition. An AVD definition defines the characteristics of an

Android phone, tablet, Android Wear, or Android TV device that you want to simulate in the Android Emulator.

Create an AVD Definition as follows:

- Launch the Android Virtual Device Manager by selecting **Tools > Android > AVD Manager**, or by clicking the AVD Manager icon in the toolbar.
- In the Your Virtual Devices screen, click Create Virtual Device.
- In the Select Hardware screen, select a phone device, such as Nexus 6, and then click Next.
- In the System Image screen, choose the desired system image for the AVD and click Next. (if you don't have a particular system image installed, you can get it by clicking the download link.)

Verify the configuration settings (for your first AVD, leave all the settings as they are), and then click **Finish**.

Create and Manage Virtual Devices

An Android Virtual Device (AVD) definition lets you define the characteristics of an Android phone, tablet, Android Wear, or Android TV device that you want to simulate in the Android Emulator. The AVD Manager helps you easily create and manage AVDs

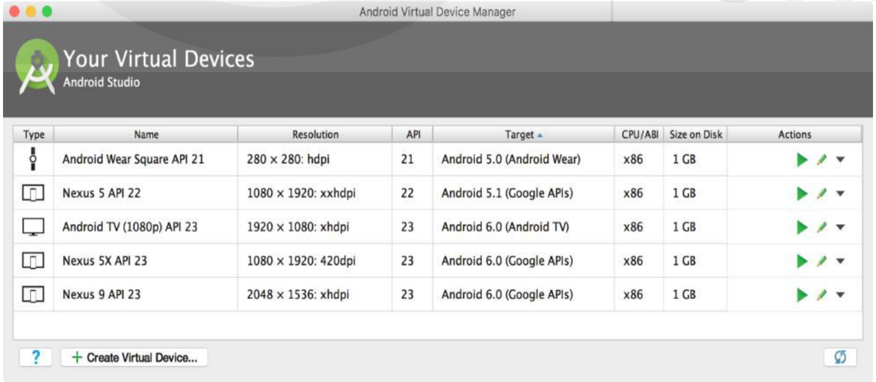
VIEWING AND MANAGING YOUR AVDS

The AVD Manager lets you manage your AVDs all in one place.

To run the AVD Manager, do one of the following:

- In Android Studio, select **Tools** > **Android** > **AVD Manager**.
- Click **AVD Manager** in the toolbar.

The AVD Manager appears as shown in Figure 6.2.



Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
⌚	Android Wear Square API 21	280 × 280: hdpi	21	Android 5.0 (Android Wear)	x86	1 GB	▶ ✎ ▾
▯	Nexus 5 API 22	1080 × 1920: xxhdpi	22	Android 5.1 (Google APIs)	x86	1 GB	▶ ✎ ▾
🖵	Android TV (1080p) API 23	1920 × 1080: xhdpi	23	Android 6.0 (Android TV)	x86	1 GB	▶ ✎ ▾
▯	Nexus 5X API 23	1080 × 1920: 420dpi	23	Android 6.0 (Google APIs)	x86	1 GB	▶ ✎ ▾
▯	Nexus 9 API 23	2048 × 1536: xhdpi	23	Android 6.0 (Google APIs)	x86	1 GB	▶ ✎ ▾

Figure 6.2 AVD manager

It displays any AVDs you have already defined. When you first install Android Studio, it creates one AVD. If you defined AVDs for Android Emulator 24.0.*x* or lower, you need to recreate them.

From this page, you can:

- Define a new AVD or hardware profile.
- Edit an existing AVD or hardware profile.
- Delete an AVD or hardware profile.
- Import or export hardware profile definitions.
- Run an AVD to start the emulator.
- Stop an emulator.
- Clear data and start fresh, from the same state as when you first ran the emulator.
- Show the associated AVD .ini and .img files on disk.
- View AVD configuration details that you can include in any bug reports to the Android Studio team.

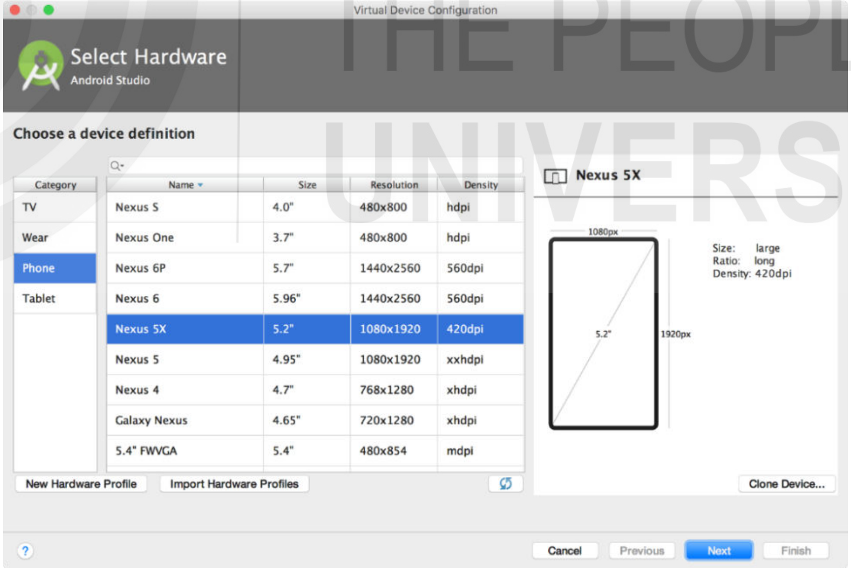
Creating an AVD

You can create a new AVD from the beginning, or duplicate an AVD and change some properties.

To create a new AVD:

- From the Your Virtual Devices page of the AVD Manager, click Create Virtual Device.
- Alternatively, run your app from within Android Studio. In the Select Deployment Target dialog, click Create New Emulator.

Then Select Hardware page appears as shown in Figure 6.3



Choose a device definition

Category
TV
Wear
Phone
Tablet

Name	Size	Resolution	Density
Nexus S	4.0"	480x800	hdpi
Nexus One	3.7"	480x800	hdpi
Nexus 6P	5.7"	1440x2560	560dpi
Nexus 6	5.96"	1440x2560	560dpi
Nexus 5X	5.2"	1080x1920	420dpi
Nexus 5	4.95"	1080x1920	xxhdpi
Nexus 4	4.7"	768x1280	xhdpi
Galaxy Nexus	4.65"	720x1280	xhdpi
5.4" FWVGA	5.4"	480x854	mdpi

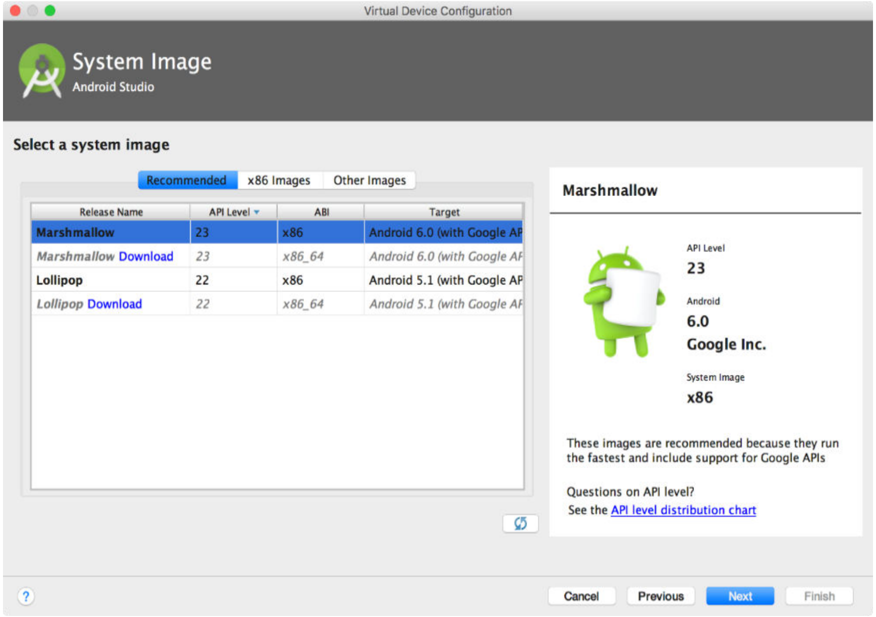
Nexus 5X

Size: large
Ratio: long
Density: 420dpi

Figure 6.3: Virtual Device Configuration(Select Hardware)

- Select a hardware profile, and then click ‘Next’.

If you don't see the hardware profile you want, you can create or import a hardware profile.Then System Image page appears as shown in Figure 6.4.



Select a system image

Release Name	API Level	ABI	Target
Marshmallow	23	x86	Android 6.0 (with Google AP
Marshmallow Download	23	x86_64	Android 6.0 (with Google AP
Lollipop	22	x86	Android 5.1 (with Google AP
Lollipop Download	22	x86_64	Android 5.1 (with Google AP

Marshmallow

API Level
23

Android
6.0
Google Inc.

System Image
x86

These images are recommended because they run the fastest and include support for Google APIs

Questions on API level?
See the API level distribution chart

Figure 6.4: Virtual Device Configuration(System Image)

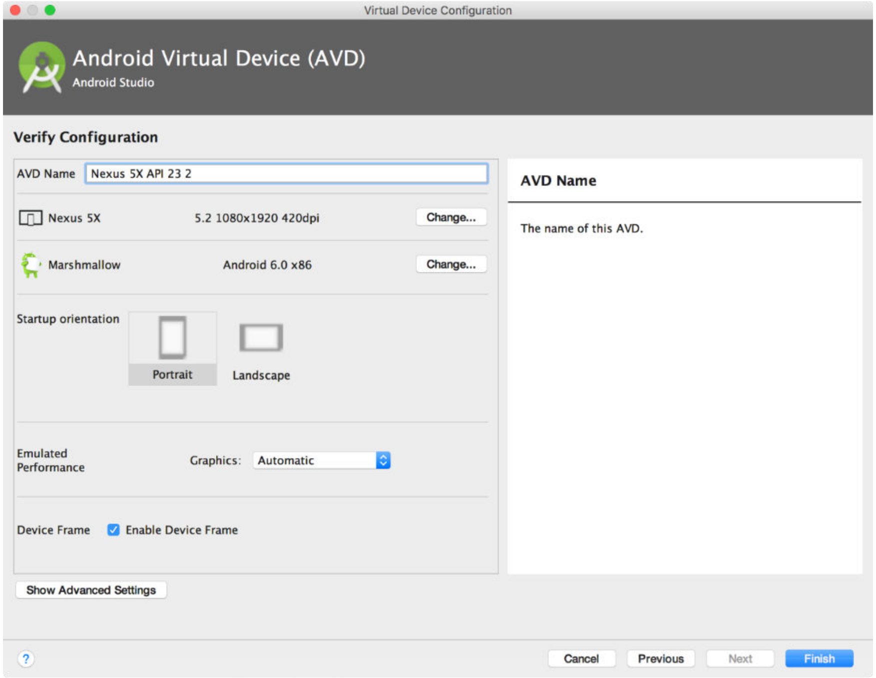
- Select the system image for a particular API level, and then click 'Next'.

The Recommended tab lists recommended system images. The other tabs include a more complete list. The right pane describes the selected system image. x86 images run the fastest in the emulator.
If you see Download next to the system image, you need to click it to download the system image. You must be connected to the internet to download it.

The API level of the target device is important, because your app won't be able to run on a system image with an API level that's less than that required by your app, as specified in the minSdkVersion attribute of the app manifest file. For more information about the relationship between system API level and minSdkVersion, see Versioning Your Apps.

If your app declares a <uses-library> element in the manifest file, the app requires a system image in which that external library is present. If you want to run your app on an emulator, create an AVD that includes the required library. To do so, you might need to use an add-on component for the AVD platform; for example, the Google APIs add-on contains the Google Maps library.

The Verify Configuration page appears as shown in Figure 6.5.



Virtual Device Configuration

Android Virtual Device (AVD)
Android Studio

Verify Configuration

AVD Name: Nexus 5X API 23 2

Nexus 5X 5.2 1080x1920 420dpi

Marshmallow Android 6.0 x86

Startup orientation: Portrait

Emulated Performance Graphics: Automatic

Device Frame ☑ Enable Device Frame

AVD Name

The name of this AVD.

Figure 6.5: Virtual Device Configuration(AVD)

- Change AVD properties as needed, and then click Finish.
- Click Show Advanced Settings to show more settings, such as the skin.

The new AVD appears in the Your Virtual Devices page or the Select Deployment Target dialog.

To create an AVD starting with a copy:

- From the Your Virtual Devices page of the AVD Manager, right-click an AVD and select Duplicate.

Or click Menu ▼ and select Duplicate.

The Verify Configuration page appears (See previous image).

Click Change or Previous if you need to make changes on the System Image and Select Hardware pages.

Make your changes, and then click Finish.

The AVD appears in the Your Virtual Devices page.

Running and Stopping an Emulator, and Clearing Data

From the *Your Virtual Devices* page, you can perform the following operations on an emulator:

- To run an emulator that uses an AVD, double-click the AVD. Or click Launch ► .

- To stop a running emulator, right-click an AVD and select **Stop**. Or click Menu ▼ and select **Stop**.

- To clear the data for an emulator, and return it to the same state as when it was first defined, right-click an AVD and select **Wipe Data**. Or click Menu ▼ and select **Wipe Data**.

Running on the actual device

When building an Android app, it's important that you always test your application on a real device before releasing it to users. This page describes how to set up your development environment and Android-powered device for testing and debugging on the device.

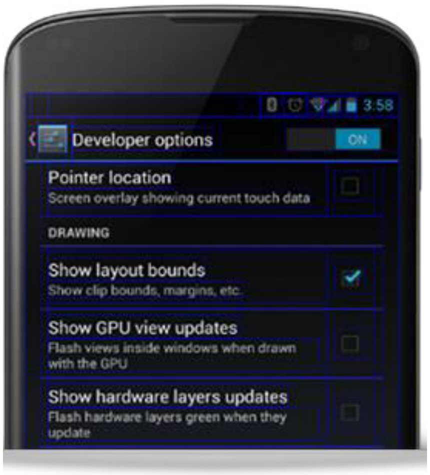
You can use any Android-powered device as an environment for running, debugging, and testing your applications. The tools included in the SDK make it easy to install and run your application on the device each time you compile. You can install your application on the device directly from Android Studio or from the command line with ADB. If you don't yet have a device, check with the service providers in your area to determine which Android-powered devices are available.

When building an Android app, it's important that you always test your application on a real device before releasing it to users. This page describes how to set up your development environment and Android-powered device for testing and debugging on the device.

Note: When developing on a device, keep in mind that you should still use the Android emulator to test your application on configurations that are not equivalent to those of your real device. Although the emulator does not allow you to test every device feature, it does allow you to verify that your application functions properly on different versions of the Android platform, in different screen sizes and orientations, and more.

Enable Developer mode

- Android-powered devices have a host of developer options that you can access on the phone, which let you
- Enable debugging over USB.
- Quickly capture bug reports onto the device.
- Show CPU usage on screen.
- Draw debugging information on screen such as layout bounds, updates on GPU views and hardware layers, and other information.
- Plus many more options to simulate app stresses or enable debugging options.



3:58

Developer options ON

Pointer location
Screen overlay showing current touch data ☐

DRAWING

Show layout bounds
Show clip bounds, margins, etc. ☑

Show GPU view updates
Flash views inside windows when drawn with the GPU ☐

Show hardware layers updates
Flash hardware layers green when they update ☐

Figure 6.6: Enabling Developer Mode in Mobile

To access these settings, open the Developer options in the system Settings as shown in Figure 6.6. On Android 4.2 and higher, the Developer options screen is hidden by default. To make it visible, go to Settings > About phone and tap Build number seven times. Return to the previous screen to find Developer options at the bottom.

Running app on the actual device

- In Android Studio, select your project and click **Run** ▶ from the toolbar.
- In the **Select Deployment Target** window, select your device, and click **OK**.

Android Studio installs the app on your connected device and starts it.

6.3.1 Check Your Progress

- Build a simple app and run the application on

1. Emulator
2. Actual Device

6.4 SUMMARY

In this unit,you learnt how to develop maintainable mobile apps that include the core Android components discussed in the previous unit. You need to watch the provided video on how to create an app from scratch using Android Studio.We also discussedabout creating the Android Virtual Devices to run, testing and debugging the application.

Configuring and saving the launch configuration and associating an AVD with your project as discussed here will makeyour debugging and testing task easier. You also learnt about different project files that are created during the development process of an Android application as well as different development tools, Android application components and adding permissions to an application.

6.4 FURTHER READINGS

- https://developer.android.com/training/basics/firstapp?authuser=1
- https://developer.android.com/studio/run/managing-avds