

CSCI3310 Mobile Computing & Application Development

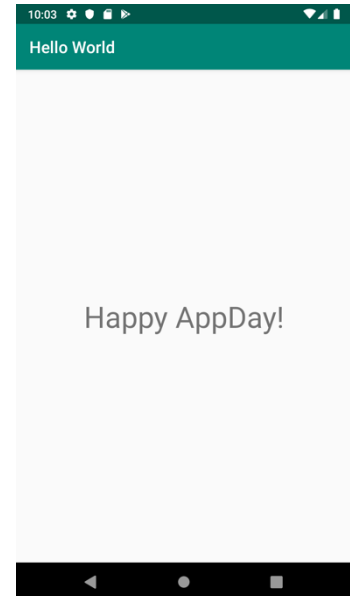
Pre-Lab – Hello World: Happy AppDay!

Objectives

- 1) To use the development process for building Android apps.
- 2) To create an Android project from a template.

ToDo

- 1) Install the Android Studio development environment.
- 2) Create an emulator (virtual device) to run your app on your computer.
- 3) Create and run the Happy AppDay app on the virtual and/or physical devices.



This simple app displays the greeting "Happy AppDay!" on the screen of the Android virtual or physical device. Note that Android Studio Arctic Fox is assumed in this doc.

1. Install Android Studio (if not yet done so)

- 1) What is Android Studio?

A complete IDE including advanced code editor tools for development, debugging, testing, and a set of app templates. With preconfigured emulators, you can test your apps, build production apps, and publish on the Google Play store.

What's required to work with Android Studio? Available for Windows, Linux or Macs. Latest OpenJDK is already bundled with Android Studio. Check more on the [system requirements](#).

How to install Android Studio? Follow instructions for [install Android Studio](#), default configurations shall be fine at the start. Additional components such as the Android SDK should be installed after finishing the Setup Wizard. In short, be patient! Then, you are ready to create your first project.

1.1 Setup Android Studio for the first time

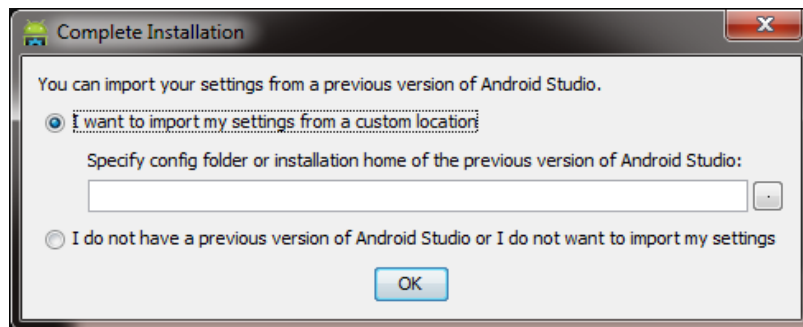
- 1) For **CSE users**, Android Studio 3.2.1 is pre-installed for access in **S:** drive from your CSDOMAIN login in our PC-Lab.
- 2) Before you run Android Studio, make sure the Android Studio SDK is already correctly set up in your **M:** drive:

M:\Android_SDK

- 3) Then open Android Studio from:

S:\android\android-studio\studio64.exe

Startup Configuration:



In the "**Complete Installation**" step, Select the option for Importing Studio Settings from Custom Location and type:

S:\android\android-studio\CSE-Settings\AndroidStudio3.2

If Android_SDK has been copied to **M:\Android_SDK**, the below steps will be by-passed. Otherwise, please get help from instructors:

Missing SDK -> Next

SDK Components Setup -> Next

Verify Settings -> Finish

After that, the required Directories will be set as below:

- **M:\Android:** Virtual Machine for Android Phones
- **M:\Android_SDK:** SDK for Android
- **M:\Gradle:** Android Build Toolkit
- **M:\.AndroidStudio:** System Created.
- M: is connected to **\\ntsvr1\android\%YOURUSERNAME%**, and the disk quota for each user is 25GB.

A FAQ for the setup or reconfiguration can be found here:

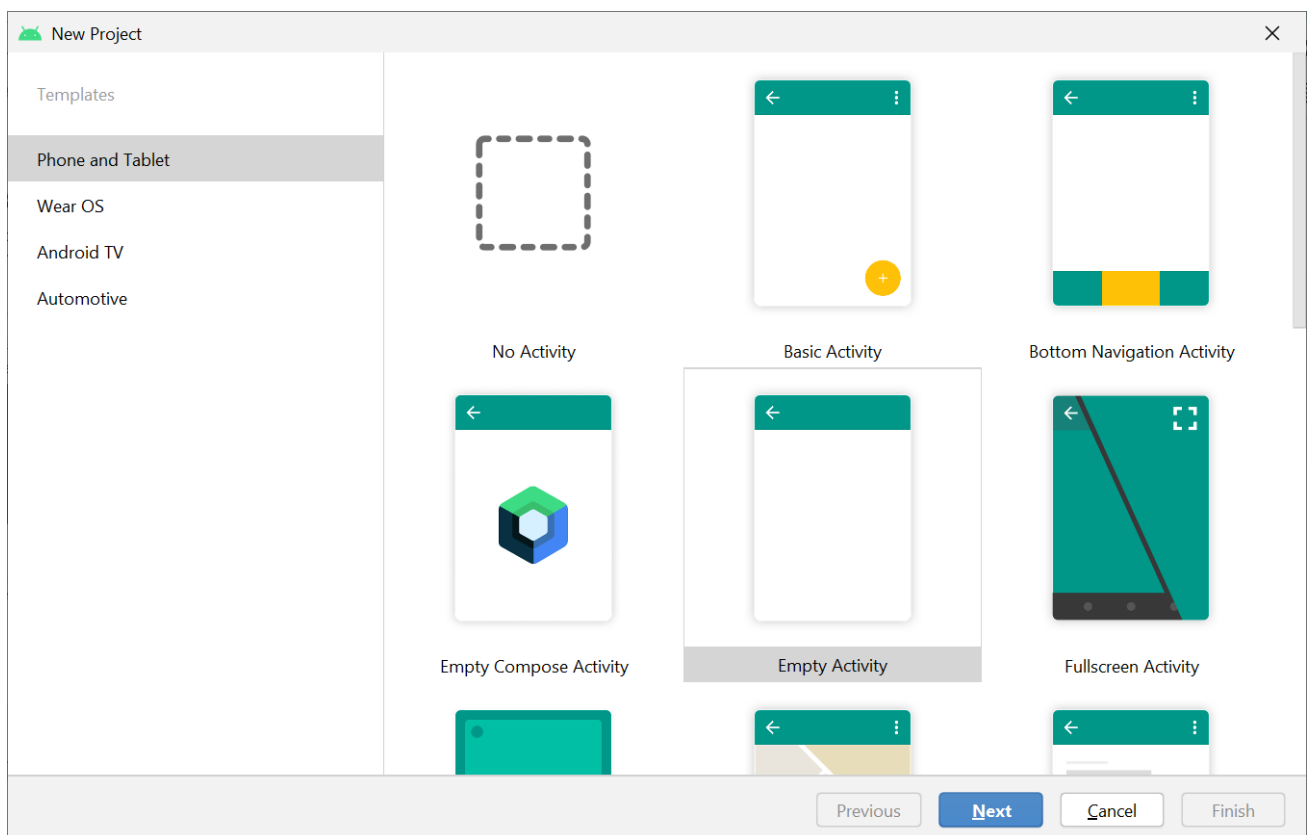
S:\android\android-studio\CSE-Readme.txt

2. Create the Happy AppDay app

In this task, you will create and run a default app to verify that the Android studio is correctly configured (and installed) and to learn the basics of developing with Android Studio.

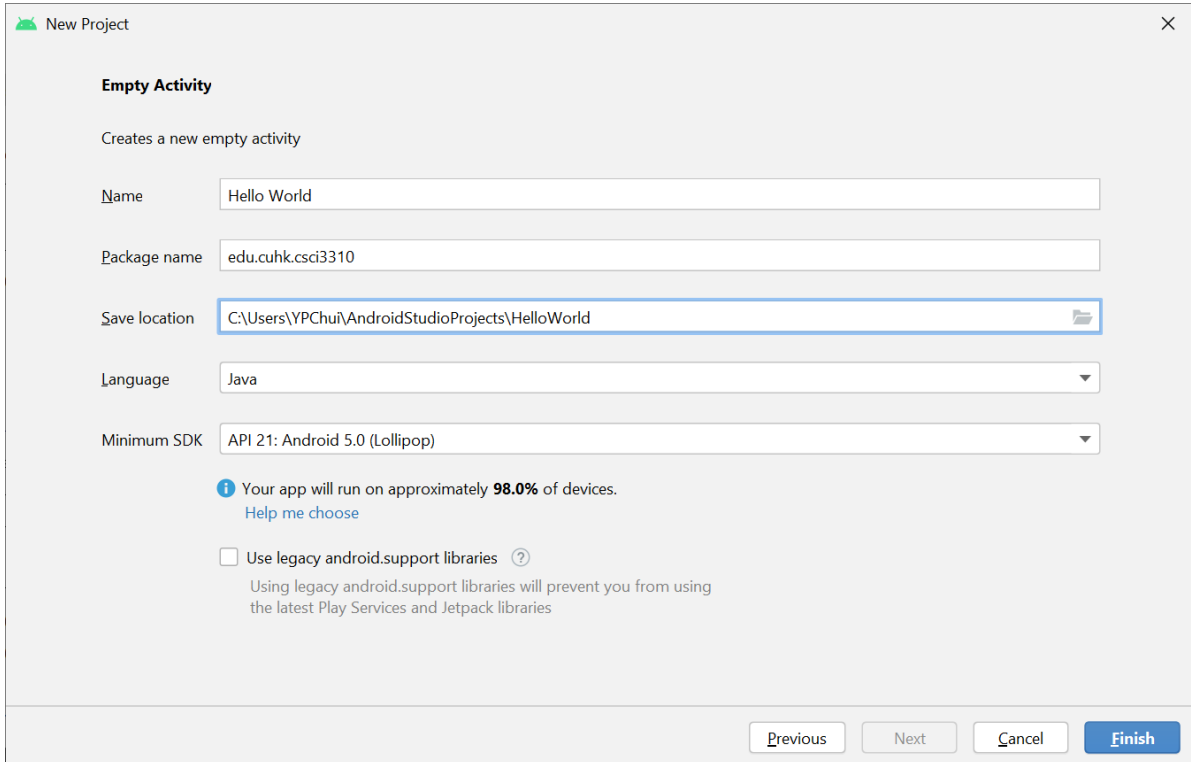
2.1 Create the app project

- 1) Open **Android Studio**, in the main **Android Studio** window, clicks **Create new Project**.
- 2) In the **Select a Project Template** window, choose **Empty Activity**.



- 3) In the **Configure Your Project** window, enter **Hello World** for the Application name.
- 4) Leave the default **com.android.example** for **Package Name**, or create an unique one such as **edu.cuhk.csci3310**.
- 5) Pick **Java** as the **Language**
- 6) Leave the default **API 21: Android5.0 (Lollipop)** is set as the **minimum SDK** unchanged.

- 7) Leave unchecked the **Use legacy android support**. Then click **Finish**. If your project requires additional components for your chosen target SDK, Android Studio will install them automatically.

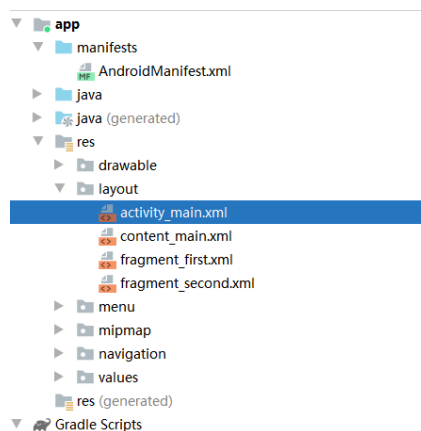


2.2 Explore the Android Studio interface

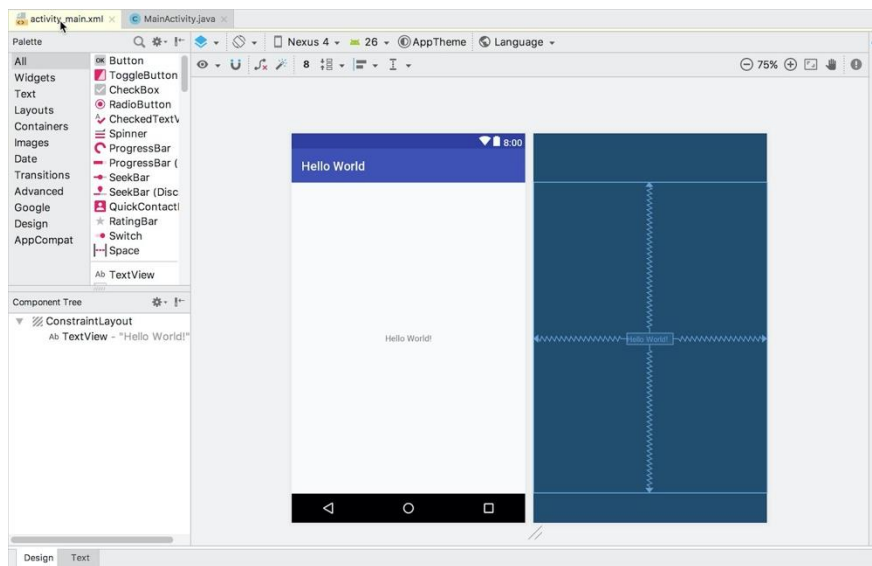
Android Studio creates a folder for your projects and builds the project with [Gradle](#) (this takes a while....). **Reference:** See the [Configure your build](#) for details.

The Android Studio editor appears. Follow these steps:

- 1) Click the **activity_main.xml** tab to see the layout editor.



2) Click the layout editor **Design** tab to show layout graphically.

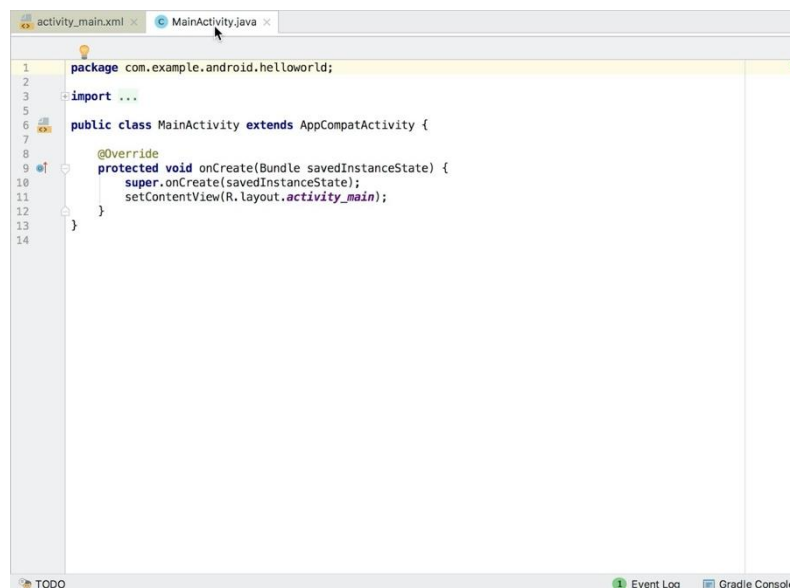
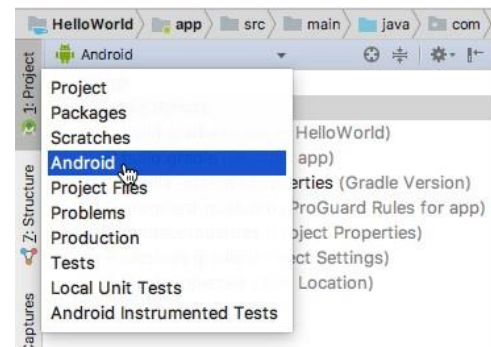


3) Click the **MainActivity.java** tab to see the code editor as shown below.

2.3 Explore the Project > Android pane

Explore how the project is organized in Android Studio:

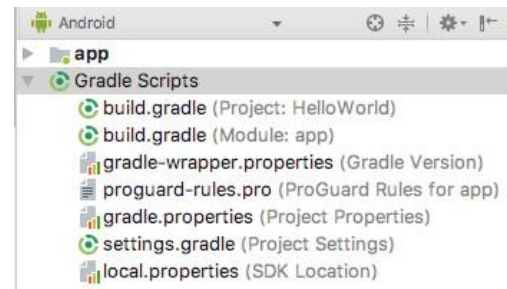
- 1) Click the **Project** tab in the vertical tab column on the left side of the Android Studio window. The Project pane appears.
- 2) To view the project in the standard Android project hierarchy, choose **Android** from the popup menu at the top of the Project pane, as shown below.



3) Explore the Gradle Scripts folder

The Gradle build system in Android Studio is for including external binaries or other library modules to your build as dependencies.

When you first create an app project, the **Project > Android** pane appears with the **Gradle Scripts** folder expanded as shown below.



Follow these steps to explore the Gradle system:

1. If the **Gradle Scripts** folder is not expanded, click the triangle to expand it. This folder contains all the files needed by the build system.
2. Look for the **build.gradle(Project: HelloWorld)** file.

Every Android Studio project contains a single, top-level Gradle build file with the configuration options that are common to all the modules that make up your project. Most of the time, you won't need to make any changes to this file, but it's still useful to understand its contents.

By default, the top-level build file uses the **buildscript** block to define the Gradle repositories and dependencies that are common to all modules in the project.

When your dependency is something other than a local library or file tree, Gradle looks for the files in whichever online repositories are specified in the repositories block of this file.

By default, new Android Studio projects declare **JCenter** and **Google** (which includes the [Google Maven repository](#)) as the repository locations.

3. Look for the **build.gradle(Module:app)** file.

In addition to the project-level **build.gradle** file, each module has a **build.gradle** file of its own, which allows you to:

- configure build settings for each specific module (this app has only one module) so as to allow custom packaging options, such as additional build types.
- override settings in the AndroidManifest.xml file or the top-level **build.gradle** file.

This file is most often the file to edit when changing app-level configurations, such as declaring dependencies in the dependencies section. You can declare a library dependency using one of several different dependency configurations. Each dependency configuration provides Gradle different instructions about how to use the library.

4. Click the triangle to close **Gradle Scripts**.

4) Explore the app and res folders

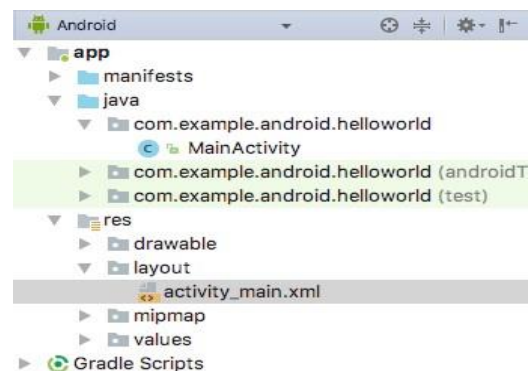
All code and resources for the app are located within the **app** and **res** folders.

1. Expand the **app** folder, the **java** folder, and the **com.example.android.helloworld** folder to see the **MainActivity** java file. Double-clicking the file opens it in the code editor.
 - The **java** folder includes Java class files in three subfolders, as shown in the figure. The **com.example.android.helloworld** (or the domain name you have specified) folder contains all the files for an app package.
 - The other two folders are for testing.

The name of the first Activity (screen) the user sees, which also initializes app-wide resources, is commonly called **MainActivity** (the file extension is omitted in the **Project>Android** pane).

2. Expand the **res** folder and the **layout** folder and double-click the **activity_main.xml** file to open it in the layout editor.

The **res** folder holds resources, such as layouts, strings, and images. An Activity is usually associated with a layout of UI views defined as an XML file. This file is usually named after its Activity.

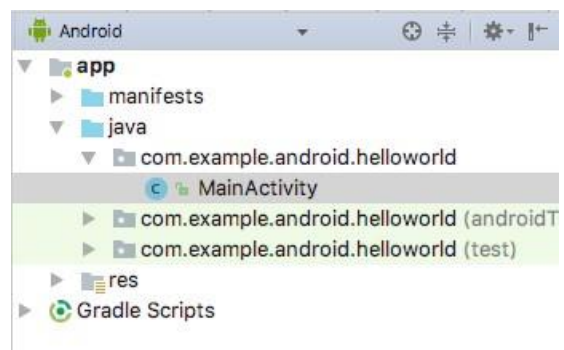


2.4 Explore the manifests folder

The manifests folder contains files that provide essential information about your app to the Android system, which the system must have before it can run any of the app's code.

- 1) Expand the **manifests** folder.
- 2) Open the **AndroidManifest.xml** file.

The **AndroidManifest.xml** file describes all of the components of your Android app. All components for an app, such as each **Activity**, must be declared in this XML file. More on **Android components** to discuss as we go through other lectures and labs. See more in [App Manifest Overview](#).



3: Use a virtual device (aka emulator)

In this task, you will use the **Android Virtual Device** (AVD) manager to create a virtual device that simulates the configuration for a particular type of Android device and use that virtual device to run the app.

Hardware characteristics of a device, its **API level** (corresponds to different **Android versions**), **storage**, **skin** and other properties can be defined and be saved as a virtual device.

With this, apps can be tested on different device configurations (tablets or phones) with different Android versions, without having to use real devices.

Note: Android Emulator has additional requirements beyond the basic system requirements for Android Studio.

3.1 Create an Android virtual device (AVD)

To run an emulator on your computer, you have to create a configuration that describes the virtual device.

- 1) In Android Studio, select **Tools > Android > AVD Manager**, or click the AVD Manager icon

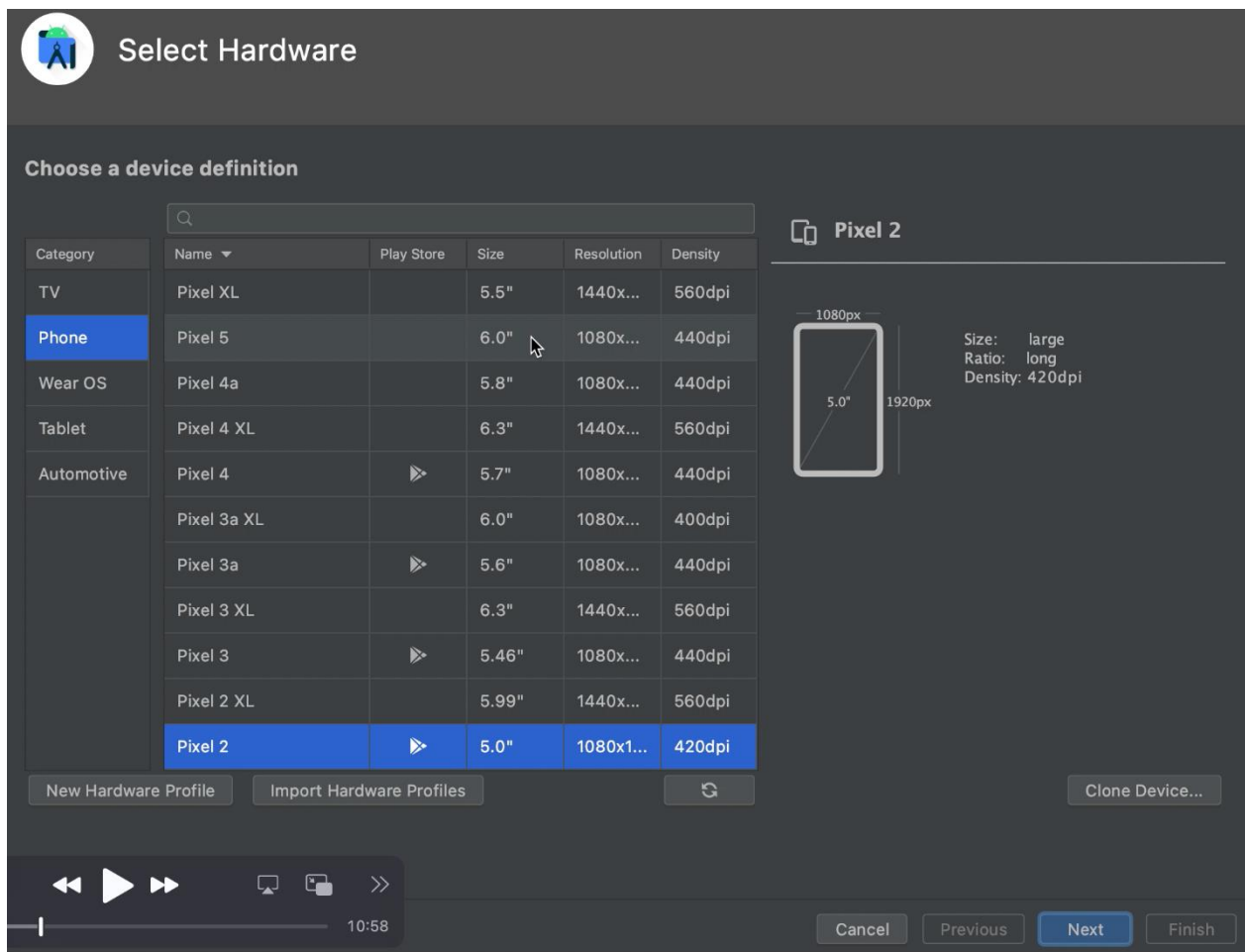


in the toolbar. The **Your Virtual Devices** screen appears. If you've already created virtual devices, the screen shows them; otherwise, you see a blank list.


- 2) Click the **+Create Virtual Device**. The **Select Hardware** window appears showing a list of pre-configured hardware devices. For each device, the table provides a column for its diagonal display size (**Size**), screen resolution in pixels (**Resolution**), and pixel density (**Density**).

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 28		1080 × 1920: 420dpi	28	Android 9.0 (Google...)	x86	9.4 GB	
	Pixel 2 API 28		1080 × 1920: 420dpi	28	Android 9.0 (Google...)	x86	8.9 GB	

+ Create Virtual Device...



- 3) Choose a device such as a **Pixel 2** and click **Next**. The **System Image** screen appears.
- 4) Click the **Recommended** tab if it is not already selected and choose which version of the Android system to run on the virtual device (such as Android 10.0 Q).




System Image

Select a system image

Recommended
x86 Images
Other Images

Release Name	API Level	ABI	Target
<i>R</i> Download	30	x86	Android 11.0 (Google Play)
<i>Q</i> Download	29	x86	Android 10.0 (Google Play)
<i>Pie</i> Download	28	x86	Android 9.0 (Google Play)
<i>Oreo</i> Download	27	x86	Android 8.1 (Google Play)
<i>Oreo</i> Download	26	x86	Android 8.0 (Google Play)
<i>Nougat</i> Download	25	x86	Android 7.1.1 (Google Play)
<i>Nougat</i> Download	24	x86	Android 7.0 (Google Play)



API Level
29


Android
10.0

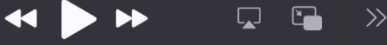
Google Inc.

System Image
x86

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?
See the [API level distribution chart](#)

 A system image must be selected to continue.



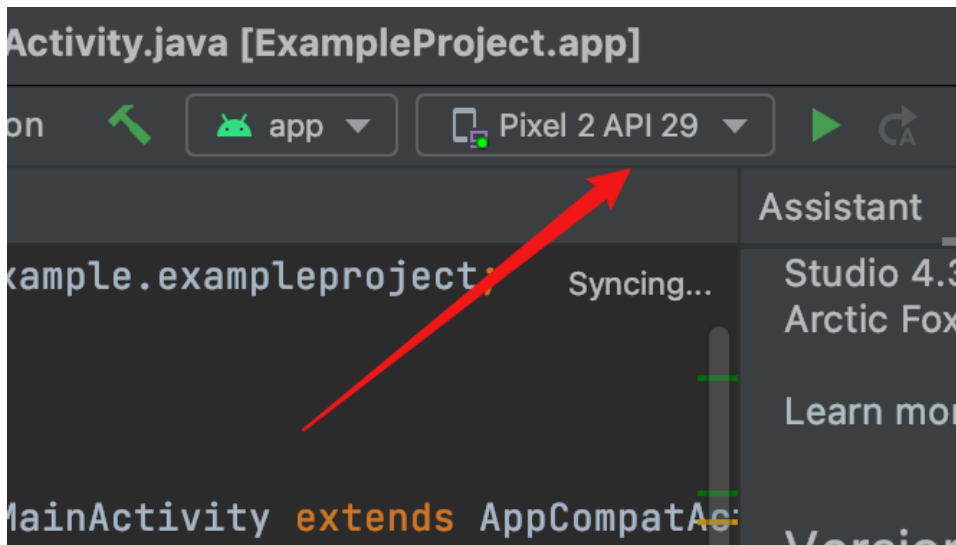
10:58

Cancel
Previous
Next
Finish

There are many more versions available than shown in the **Recommended** tab. Look at the **x86 Images** and **Other Images** tabs to see them.


If a **Download** link is visible next to a system image you want to use, it is not installed yet. Click the link to start the download, and click **Finish** when it's done.

- After choosing a system image, click **Next**. The **Android Virtual Device (AVD)** window appears. You can also change the name of the AVD. Check your configuration and click **Finish**.



3.2 Run the app on the virtual device

Finally, you run the Happy AppDay app.

- 1) In Android Studio, choose **Run > Run app** or click the **Run** icon  in the toolbar.
- 2) Use the drop-down to select the virtual device.

The emulator starts and boots just like a physical device. Depending on the speed of your computer, this may take a while. Your app builds, and once the emulator is ready, Android Studio will upload the app to the emulator and run it.

3.3 Customizing the greeting

Now that you are about to finish just change the "**Hello World**" greeting to "**Happy AppDay!**".

Tips: When testing on a virtual device, you may start it up once at the very beginning of your session and keep it until you are done testing your app so that your app doesn't have to go through the device startup process again. To close the virtual device, click the **X** button at the top of the emulator, choose **Quit** from the menu.

=== END of Pre-lab ===