

Building a sample Android App using Android Studio

Learning Objectives

After studying this module learner should be able to:

- How to create a project in Android Studio.
- How to create an emulated Android device.
- How to run your app on the emulator.
- How to run your app on your own physical device, if you have one.
- Know the structure of Android Project
- List the various types of Android Project files
- Understand anatomy of an android application

Step 1: Create a new project

1. Open Android Studio.
2. In the **Welcome to Android Studio** dialog, click **Start a new Android Studio project**.
3. Select **Basic Activity** (not the default). Click **Next**.
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**.

After these steps, Android Studio:

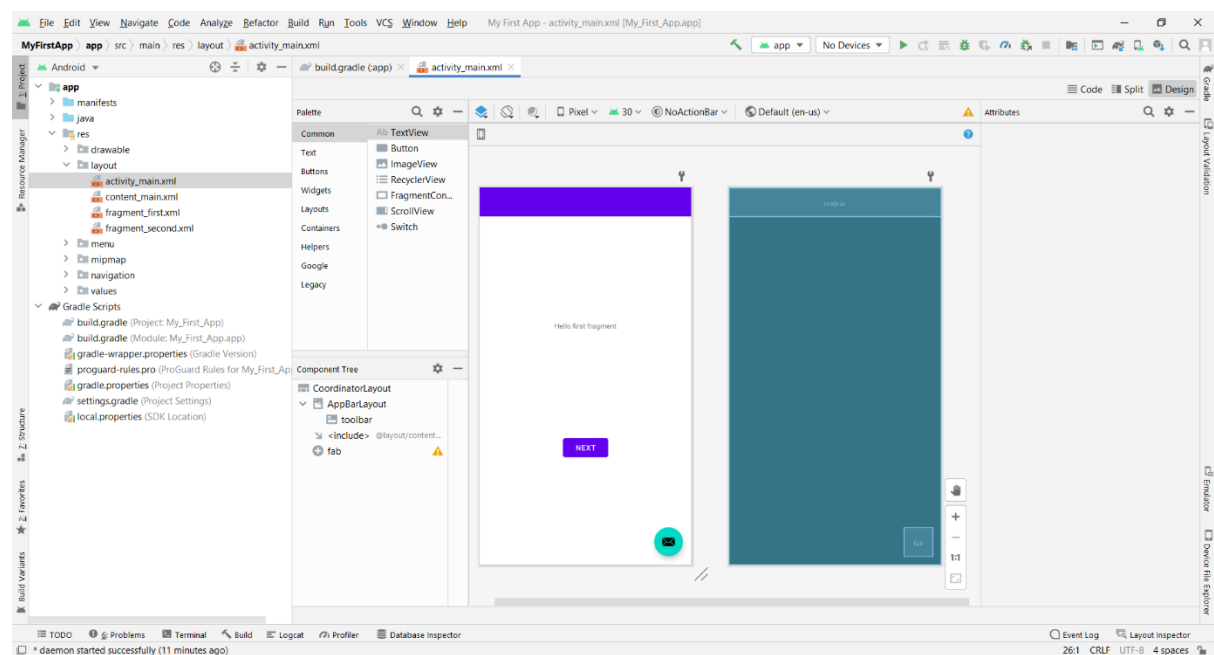
- Creates a folder for your Android Studio project called **MyFirstApp**. This is usually in a folder called **AndroidStudioProjects** below your home directory.
- Builds your project (this may take a few moments). Android Studio uses Gradle as its build system. You can follow the build progress at the bottom of the Android Studio window.
- Opens the code editor showing your project.

Step 2: Get your screen set up

When your project first opens in Android Studio, there may be a lot of windows and panes open. To make it easier to get to know Android Studio, here are some suggestions on how to customize the layout.

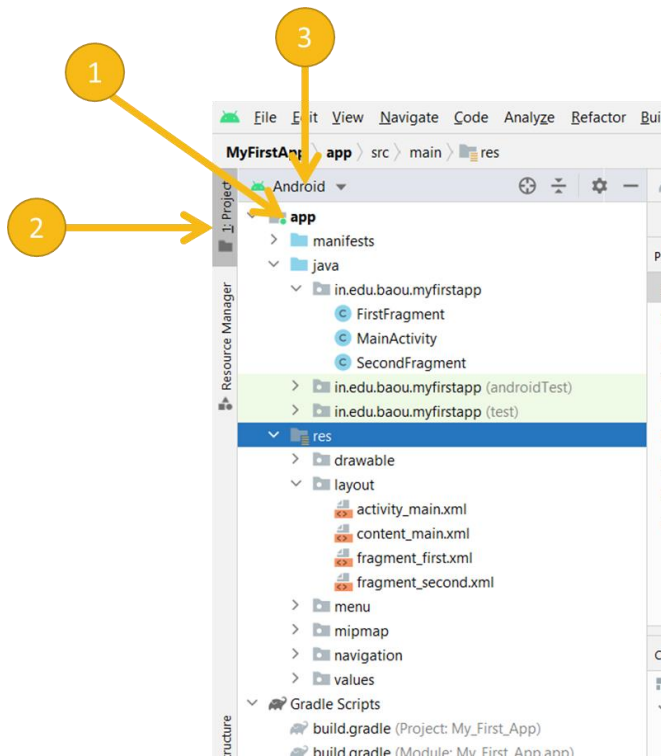
1. If there's a **Gradle** window open on the right side, click on the minimize button (—) in the upper right corner to hide it.
2. Depending on the size of your screen, consider resizing the pane on the left showing the project folders to take up less space.

At this point, your screen should look a bit less cluttered, similar to the screenshot shown below.



Step 3: Explore the project structure and layout

The upper left of the Android Studio window should look similar to the following diagram:



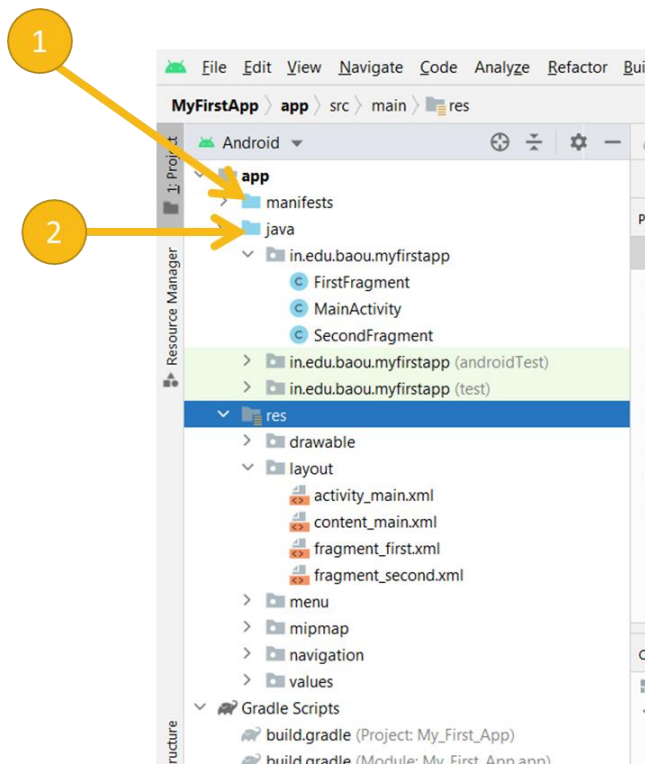
Based on you selecting the **Basic Activity** template for your project, Android Studio has set up a number of files for you. You can look at the hierarchy of the files for your app in multiple ways, one is in **Project** view. **Project** view shows your files and folders structured in a way that is convenient for working with an Android project. (This does not always match the file hierarchy! To see the file hierarchy, choose the **Project files** view by clicking (3).)

1. Double-click the **app** (1) folder to expand the hierarchy of app files. (See (1) in the screenshot.)
2. If you click **Project** (2), you can hide or show the **Project** view. You might need to select **View > Tool Windows** to see this option.
3. The current **Project** view selection (3) is **Project > Android**.

In the **Project > Android** view you see three or four top-level folders below your **app** folder: **manifests**, **java**, **java (generated)** and **res**. You may not see **java (generated)** right away.

1. Expand the **manifests** folder.
 - This folder contains **AndroidManifest.xml**. This file describes all the components of your Android app and is read by the Android runtime system when your app is executed.
2. Expand the **java** folder. All your Java language files are organized here. The **java** folder contains three subfolders:
 - **in.edu.baou.myfirstapp**: This folder contains the Java source code files for your app.

- **in.edu.baou.myfirstapp (androidTest):** This folder is where you would put your instrumented tests, which are tests that run on an Android device. It starts out with a skeleton test file.
- **in.edu.baou.myfirstapp (test):** This folder is where you would put your unit tests. Unit tests don't need an Android device to run. It starts out with a skeleton unit test file.



drawable: All your app's images will be stored in this folder.

layout: This folder contains the UI layout files for your activities. Currently, your app has one activity that has a layout file called `activity_main.xml`. It also contains `content_main.xml`, `fragment_first.xml`, and `fragment_second.xml`.

menu: This folder contains XML files describing any menus in your app.

mipmap: This folder contains the launcher icons for your app.

navigation: This folder contains the navigation graph, which tells Android Studio how to navigate between different parts of your application.

values: This folder contains resources, such as strings and colors, used in your app.

Step 4: Create a virtual device (emulator)

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

The first step is to create a configuration that describes the virtual device.

1. In Android Studio, select **Tools > AVD Manager**, or click the AVD Manager icon in the toolbar.
2. Click **+Create Virtual Device**. (If you have created a virtual device before, the window shows all of your existing devices and the **+Create Virtual Device** button is at the bottom.) The **Select Hardware** window shows a list of pre-configured hardware device definitions.
3. Choose a device definition, such as **Pixel 2**, and click **Next**.
4. In the **System Image** dialog, from the **Recommended** tab, choose the latest release.
5. If a **Download** link is visible next to a latest release, it is not installed yet, and you need to download it first. If necessary, click the link to start the download, and click **Next** when it's done. This may take a while depending on your connection speed.



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

6. In the next dialog box, accept the defaults, and click **Finish**.

The AVD Manager now shows the virtual device you added.

7. If the **Your Virtual Devices** AVD Manager window is still open, go ahead and close it.

Step 5: Run your app on your new emulator

1. In Android Studio, select **Run > Run 'app'** or click the **Run** icon in the toolbar.  The icon will change when your app is already running. 

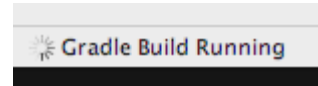
If you get a dialog box stating "Instant Run requires that the platform corresponding to your target device (Android N...) is installed" go ahead and click **Install and continue**.

2. In **Run > Select Device**, under **Available devices**, select the virtual device that you just configured. This menu also appears in the toolbar.

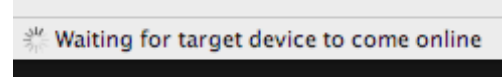
The emulator starts and boots just like a physical device. Depending on the speed of your computer, this may take a while. You can look in the small horizontal status bar at the very bottom of Android Studio for messages to see the progress.

Messages that might appear briefly in the status bar

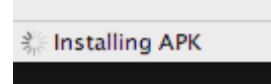
Gradle build running



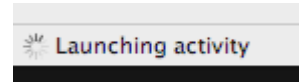
Waiting for target device to come on line



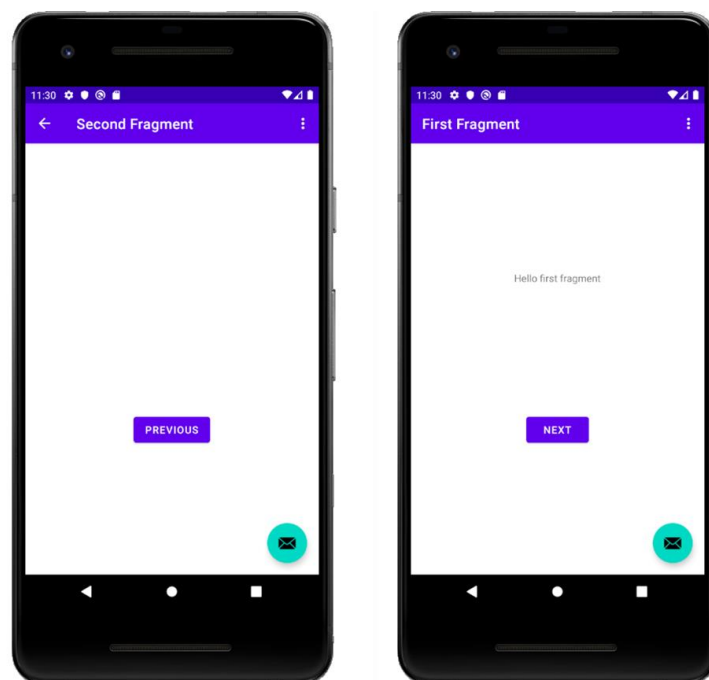
Installing APK



Launching activity



Once your app builds and the emulator is ready, Android Studio uploads the app to the emulator and runs it. You should see your app as shown in the following screenshot.



Note: It is a good practice to start the emulator at the beginning of your session. Don't close the emulator until you are done testing your app, so that you don't have to wait for the emulator to boot again. Also, don't have more than one emulator running at once, to reduce memory usage.

Step 6: Run your app on a device (if you have one)

What you need:

- An Android device such as a phone or tablet.

- A data cable to connect your Android device to your computer via the USB port.
- If you are using a Linux or Windows OS, you may need to perform additional steps to run your app on a hardware device. Check the Run Apps on a Hardware Device documentation. On Windows, you may need to install the appropriate USB driver for your device.


Run your app on a device

To let Android Studio, communicate with your device, you must turn on USB Debugging on your Android device.

On Android 4.2 and higher, the Developer options screen is hidden by default. To show Developer options and enable USB Debugging:

1. On your device, open **Settings > About phone** and tap **Build number** seven times.
2. Return to the previous screen (**Settings**). **Developer options** appears at the bottom of the list. Tap **Developer options**.
3. Enable **USB Debugging**.

Now you can connect your device and run the app from Android Studio.

1. Connect your device to your development machine with a USB cable. On the device, you might need to agree to allow USB debugging from your development device.
2. In Android Studio, click **Run**  in the toolbar at the top of the window. (You might need to select **View > Toolbar** to see this option.) The **Select Deployment Target** dialog opens with the list of available emulators and connected devices.
3. Select your device, and click **OK**. Android Studio installs the app on your device and runs it.

Note: If your device is running an Android platform that isn't installed in Android Studio, you might see a message asking if you want to install the needed platform. Click **Install and Continue**, then click **Finish** when the process is complete.

Troubleshooting

If you're stuck, quit Android Studio and restart it.

If Android Studio does not recognize your device, try the following:

1. Disconnect your device from your development machine and reconnect it.
2. Restart Android Studio.

If your computer still does not find the device or declares it "unauthorized":

1. Disconnect the device.
2. On the device, open **Settings->Developer Options**.
3. Tap **Revoke USB Debugging authorizations**.
4. Reconnect the device to your computer.
5. When prompted, grant authorizations.

If you are still having trouble, check that you installed the appropriate USB driver for your device.

Step 7: Explore the app template

When you created the project and selected **Basic Activity**, Android Studio set up a number of files, folders, and also user interface elements for you, so you can start out with a working app and major components in place. This makes it easier to build your application.

Looking at your app on the emulator or your device, in addition to the **Next** button, notice the *floating action button* with an email icon. If you tap that button, you'll see it has been set up to briefly show a message at the bottom of the screen. This message space is called a *snackbar*, and it's one of several ways to notify users of your app with brief information.

At the top right of the screen, there's a menu with 3 vertical dots. If you tap on that, you'll see that Android Studio has also created an options menu with a **Settings** item. Choosing **Settings** doesn't do anything yet, but having it set up for you makes it easier to add user-configurable settings to your app.

Acknowledgement: “The content in this module is modifications based on work created and shared by the Android Open-Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.”

