CS407 Fall 2024

Pair Programming: **NOT ALLOWED**
Due 12:00PM CT on September 16*th*

Lab01 Introduction to Android Studio and Git

Overview

In this lab, you are going to set up your Android development environment. We will be using Android Studio which is the official integrated development environment for Google's Android operating system. This lab consists of the following **three main tasks**.

- **The first one** is to successfully install Android Studio on your personal computer and run a basic "Hello World" application.
- **The second one** consists on setting up `VCS (git)` and pushing some changes to `Github`.
- **The third one** is to make a clone of our class GitHub classroom repository and make a few changes that will be described in more detail below.

The successful completion of this lab will help you get more comfortable with Android Studio and version control with Git. It will also help you complete more complex Android projects in the future.

Learning Objectives

By completing this lab, a cs407 student will be able to:

1. Successfully **install** Android Studio on their personal machine,
2. **Get** familiar with the Android Studio interface,
3. Successfully **run** a first simple Android application,
4. **Get** familiar with Git and GitHub.

Directions

Milestone 1: Install Android Studio and Run Your First App

A Download and Install Android Studio

The first step is to **download** the Integrated Development Environment (IDE) called **Android Studio** and do the necessary setup. Android Studio is the official IDE for Android development provided by Google. It is a free software development tool and it includes everything you need to build your mobile Android apps from scratch.

What you'll need

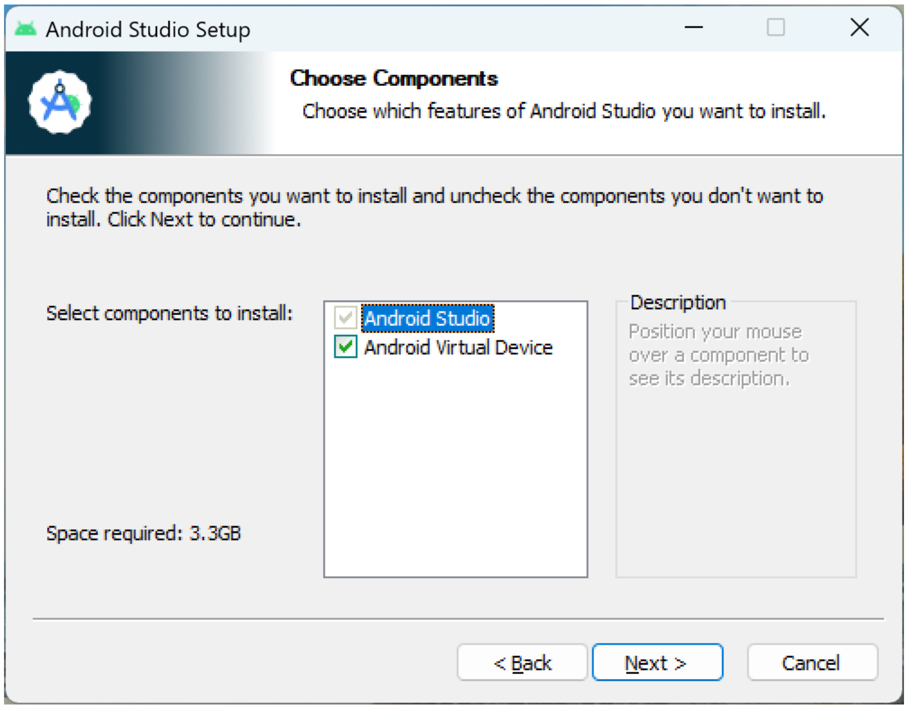
- A computer running a 64-bit version of Windows (8, 10, or 11), Linux, MacOS (10.14 Mojave or later), or ChromeOS.
  - Verify System Requirements (Windows)
  - Verify System Requirements (macOS)
  - Verify System Requirements (Linux)
- Internet access for your computer.

Follow the instructions below to download and install Android Studio on your own personal machine if you haven't done so already.

1. First, follow the Android Studio download page provided by Google.
2. Then, navigate to the **Android Studio Download** section and **download** the Android Studio package that matches your platform. The most recent **stable** version is actually **Android Studio Koala**: android-studio-2024.1.2.12. It should show up for you at the top of the download page.
3. **Accept** the terms and conditions and follow the instructions to download and install Android Studio.

   Note that since Android Studio 4.0, Android Studio does NOT support 32 bits operating systems. If your computer's operating system is 32 bits, download Android Studio 3.6.3 through the download archives link.
4. **Accept** the **default configurations** for all steps.
5. Make sure that **ALL components** are selected for installation as shown in Fig.1.
6. After finishing the installation, the **Setup Wizard** will download and install some additional components.
7. When the download and installation completes, click **Finish**.
8. Android Studio will start, the **Welcome to Android Studio** window will display as shown in Fig. 2 and you are ready to create your "Hello World" project.

- If you need further instructions on the setup process, refer to this Download and install Android Studio codelab online resource.



Android Studio Setup

Choose Components

Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

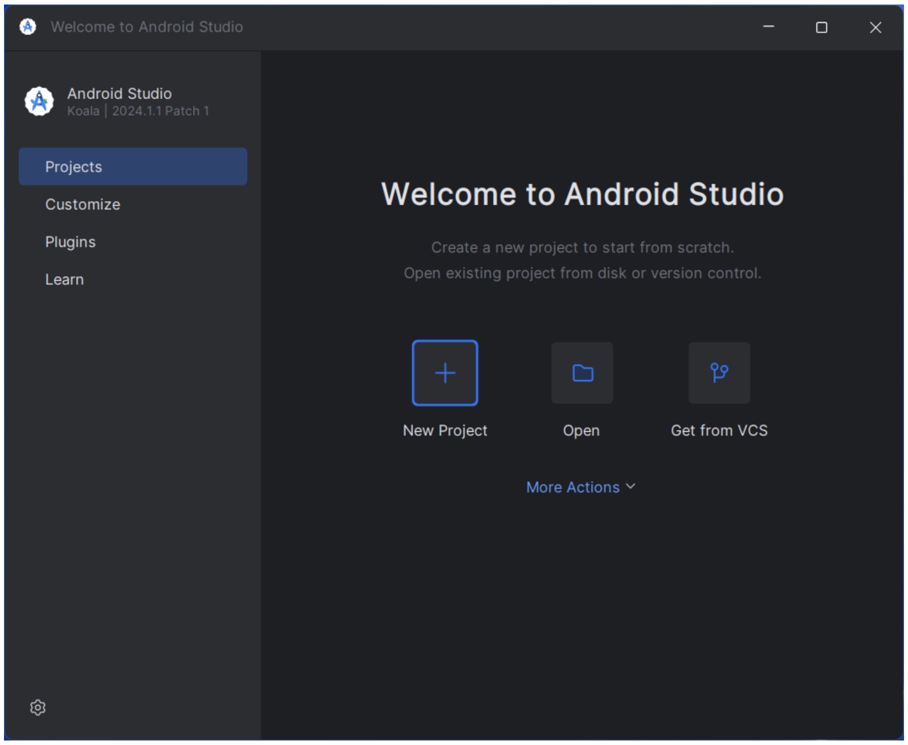
Select components to install:
- [✓] Android Studio
- [✓] Android Virtual Device

Space required: 3.3GB

Description

Position your mouse over a component to see its description.

Figure 1: Android Studio Setup Window



Welcome to Android Studio

Android Studio
Koala | 2024.1.1 Patch 1

Welcome to Android Studio

Create a new project to start from scratch.
Open existing project from disk or version control.

Figure 2: Welcome to Android Studio Window

B Create the "Hello World" app

1. **Launch** Android Studio if it is not already opened.
2. In the main **Welcome to Android Studio** dialog, **click** "**New Project** to create a new Android Studio project".
3. The New Project window opens with a list of templates provided by Android Studio as depicted in Fig. 3. In Android Studio, a project template serves as a blueprint for creating a specific type of mobile app. Templates create the structure of the project and the files needed for Android Studio to build your project. The template that you select provides a starter code, helping you to start the development of your app faster.

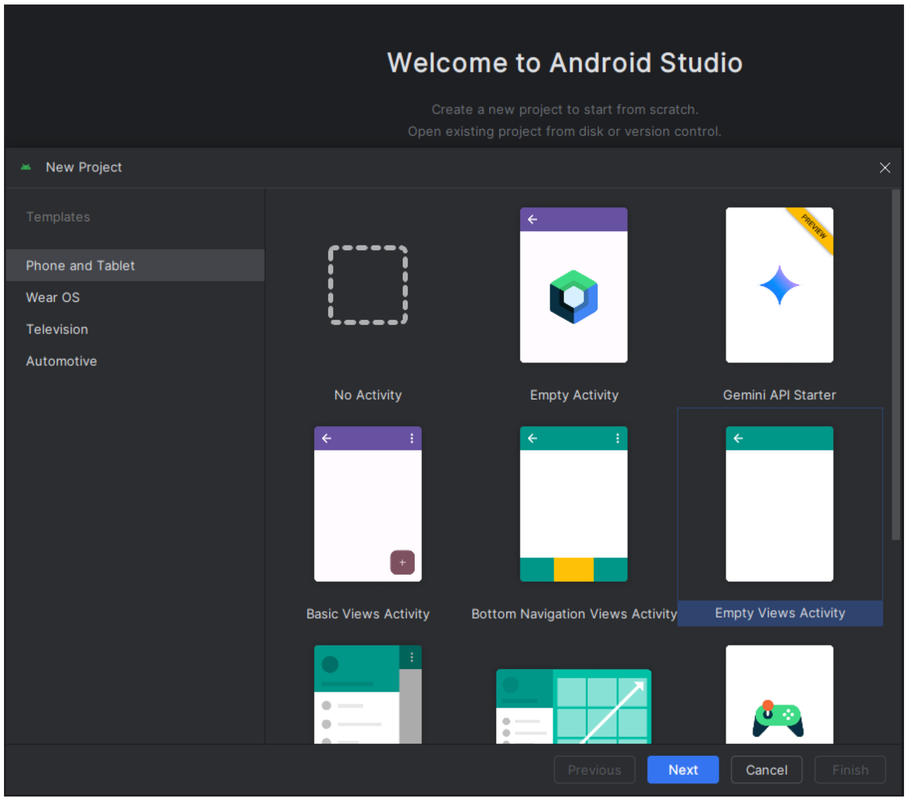
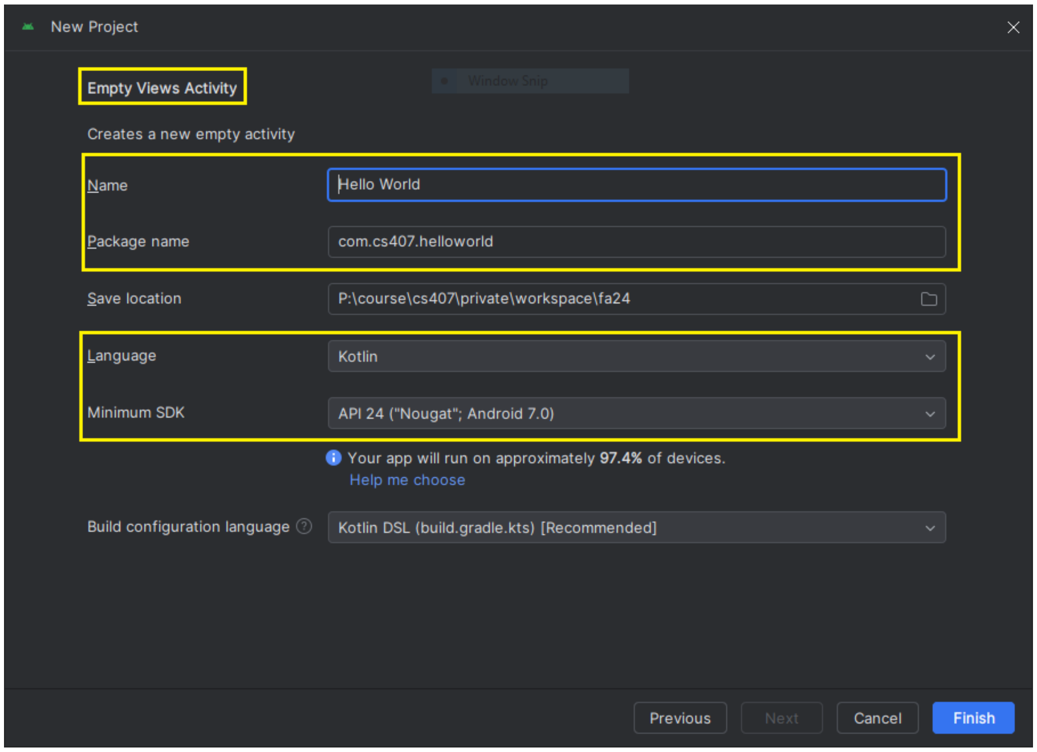


Figure 3: New Project Templates

4. Customize the **Activity window** under the **Phone and Tablet** section. Every app needs at least one activity. An **activity** represents a single screen with a user interface and Android Studio provides templates to help you get started. Android Studio offers many different templates under the Phone and Tablet tab. For the Hello World project, **choose** the **Empty Views Activity template**. The Empty Views Activity template is the template designed for creating a project with basic set of views using XML layouts.
5. As shown in the New Project window (Fig. 4), give your application a **Name**. Use the name "Hello World".


New Project

Empty Views Activity

Creates a new empty activity

Name: Hello World

Package name: com.cs407.helloworld

Save location: P:\course\cs407\private\workspace\fa24

Language: Kotlin

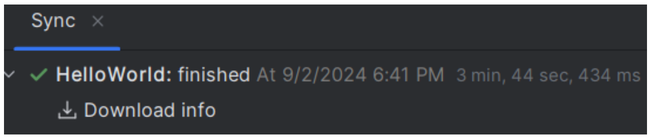
Minimum SDK: API 24 ("Nougat"; Android 7.0)

Your app will run on approximately **97.4%** of devices.
Help me choose

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

Figure 4: New Project Window

6. Edit the **Package name** by setting it to `com.cs407.helloworld`
7. Verify that the default Project location is where you want to store your `Hello World` app and other Android Studio projects, or **change it to your preferred directory**. Make sure to note the location of this directory on your computer so you can easily find and access your files later, if needed.
8. Select **API 24: Android 7.0 (Nougat)** as the **Minimum SDK**. This defines the minimum Android version that your application is going to run on. As we select Android 7.0, this means that the application is going to work on Android 7, 8, Android 9, 10, through Android 15 (the latest version of Android).
9. If your project requires additional components for your chosen target SDK, **Android Studio will install them automatically**.
10. Use the Kotlin DSL gradle as build configuration language (recommended).
11. Last step is to verify all the details that you have provided and click **Finish**. This may take some time. While your Android project is being built and set up, take a short break to refresh before continuing. A message that looks similar to the following informs you when the project setup is created.



Sync

HelloWorld: finished At 9/2/2024 6:41 PM 3 min, 44 sec, 434 ms

Download info

After these steps, Android Studio:

- **Creates** a folder for your Android Studio project called `HelloWorld` in the folder you selected as your default Project location.
- **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.
- You may see a **What's New** pane which contains updates on new features in Android Studio. Close it for now.
- **Opens** the code editor showing your project.

C Create an Android Virtual Device (AVD) Emulator

Now that you have started an Android project, you will need a **virtual device** to **run** the app. You will use the Device Manager to create an Android Virtual Device (AVD).

An AVD is an emulator of a mobile device that runs on your computer and mimics the configuration of a particular type of Android device. It is worth noting that the Android Emulator is an independent app used to set up a virtual device and it has its own system requirements. Virtual devices can **use a lot of disk space**. If you run into any issues, see Run apps on the Android Emulator.

To create an Android Virtual Device, follow these steps:

1. In Android Studio, select **Tools** > **Device Manager**. Then, select the **Virtual** tab.
2. The **Device Manager** dialog opens. If you created a virtual device previously, it's listed in this dialog.
3. Click the + **(Create Device)** button. The **Virtual Device Configuration** window appears.
4. Select **Phone** as category. A screen shows a list of pre-configured hardware devices, organized by category, from which you can choose. For each device, the table shows its diagonal display size (Size), screen resolution in pixels (Resolution), and pixel density (Density).

5. Choose a phone, such as the Pixel 6, and then click **Next** as shown in Fig. 6. If `Pixel 6` device is not shown, select any other device.

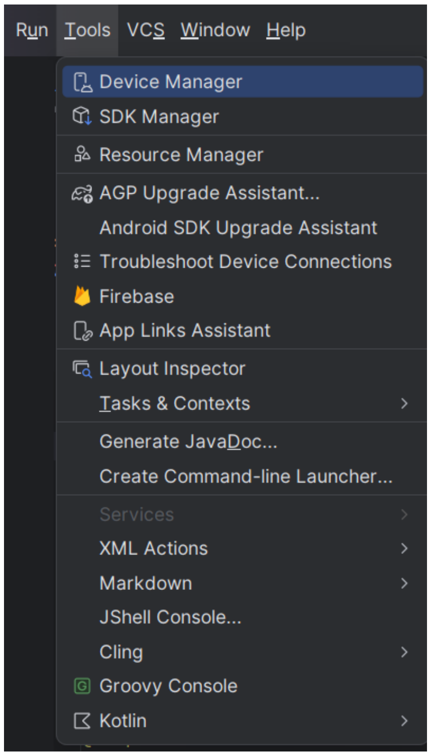
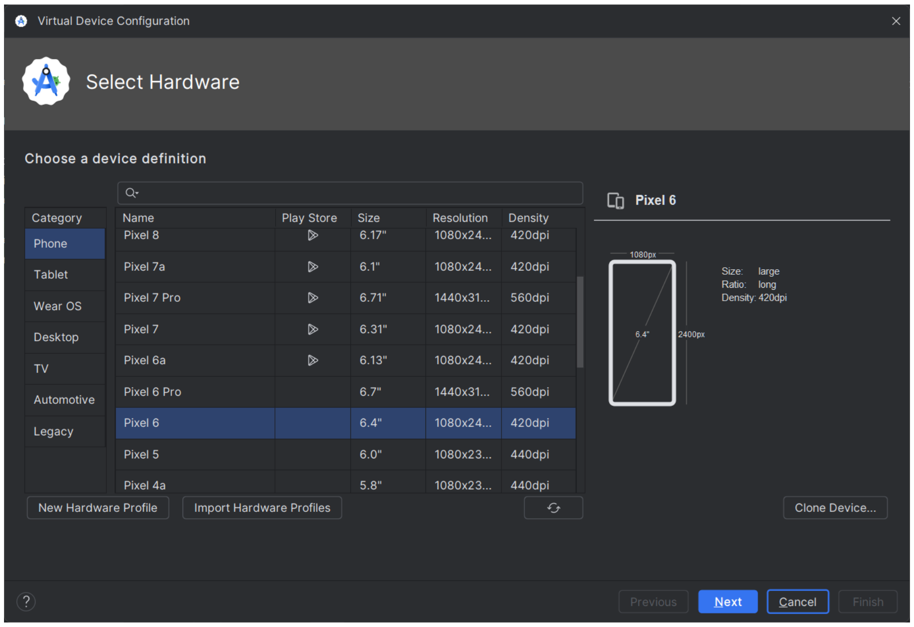


Figure 5: Device Manager

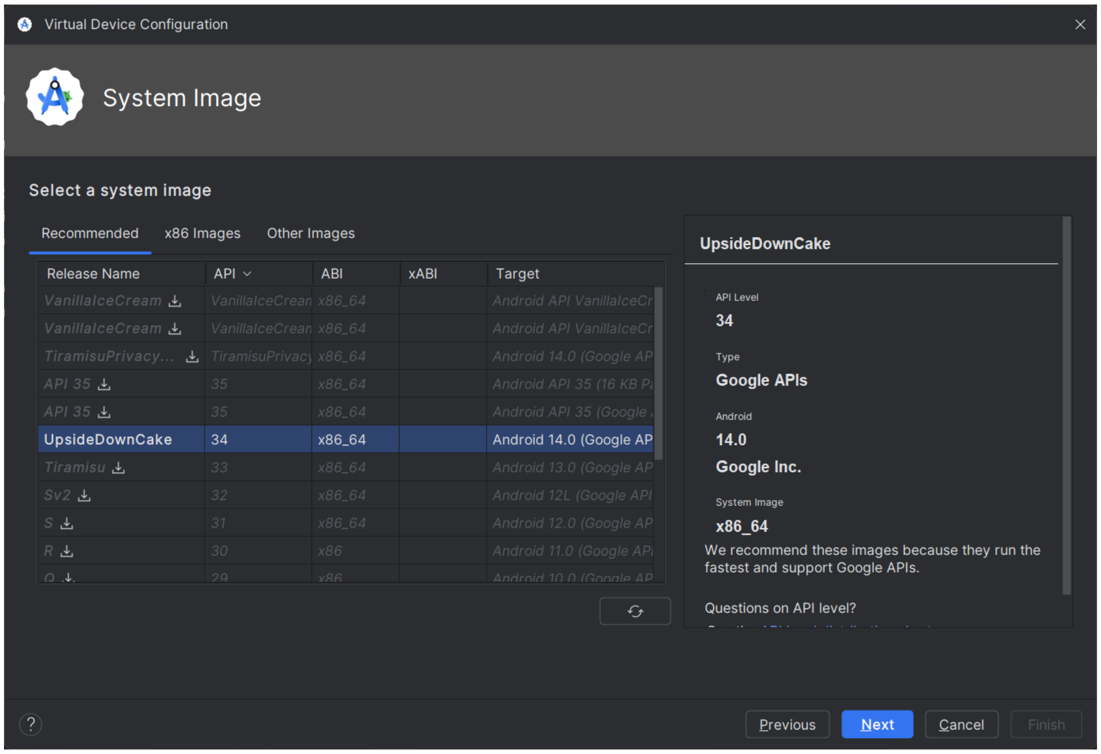


Name	Play Store	Size	Resolution	Density
Pixel 8	▷	6.17"	1080x24...	420dpi
Pixel 7a	▷	6.1"	1080x24...	420dpi
Pixel 7 Pro	▷	6.71"	1440x31...	560dpi
Pixel 7	▷	6.31"	1080x24...	420dpi
Pixel 6a	▷	6.13"	1080x24...	420dpi
Pixel 6 Pro		6.7"	1440x31...	560dpi
Pixel 6		6.4"	1080x24...	420dpi
Pixel 5		6.0"	1080x23...	440dpi
Pixel 4a		5.8"	1080x23...	440dpi

Pixel 6

Size: large
Ratio: long
Density: 420dpi

Figure 6: Select Hardware Window



Select a system image

Release Name	API	ABI	xABI	Target
VanillaIceCream	VanillaIceCrean	x86_64		Android API VanillaIceCr
VanillaIceCream	VanillaIceCrean	x86_64		Android API VanillaIceCr
TiramisuPrivacy...	TiramisuPrivacy	x86_64		Android 14.0 (Google AP
API 35	35	x86_64		Android API 35 (16 KB Pa
API 35	35	x86_64		Android API 35 (Google
UpsideDownCake	34	x86_64		Android 14.0 (Google AP
Tiramisu	33	x86_64		Android 13.0 (Google AP
Sv2	32	x86_64		Android 12L (Google API
S	31	x86_64		Android 12.0 (Google AP
R	30	x86		Android 11.0 (Google API
Q	29	x86		Android 10.0 (Google AP

UpsideDownCake

API Level
34

Type
Google APIs

Android
14.0
Google Inc.

System Image
x86_64

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

Questions on API level?

Figure 7: Select a System Image

6. This step opens another screen where you can choose the version of Android to run on your virtual device. This lets you test your app on different versions of Android.

7. On the **System Image** screen, from the **Recommended** tab to **Select a system image**, you will choose which version of the Android system to run on the virtual device.

8. If there's a download link next to `UpsideDownCake`, click **Download**>**Accept**>**Next**>**Finish**. The presence of the download link indicates that the image isn't installed on your computer, in which case you must install the image before you can configure the virtual device. Expect the download to take some time to complete.

9. In the Recommended tab, choose `UpsideDownCake` as the version of Android to run on the virtual device, and then click **Next**. Android 14 `UpsideDownCake` is the latest version of Android at this time, but feel free to choose any later stable version.

10. This action opens another screen, where you can choose additional configuration details for your device. **Note:** If you see a red warning about using a system image with Google APIs, you can disregard it for now.

11. On the **Verify Configuration** screen, in the AVD Name field, you can select a name for your AVD or use the default. Leave all the other settings unchanged (default) and click **Finish**.

D Run the App on the Android Emulator

Now that you have set up an Android Virtual Device and a “Hello World” project all that remains to do is run this application on the Emulator.

1. Select the virtual device that you created from the `dropdown` menu at the top of the Android Studio window.
2. In Android Studio, select **Run** > **Run app** or click the **Run icon** in the toolbar.

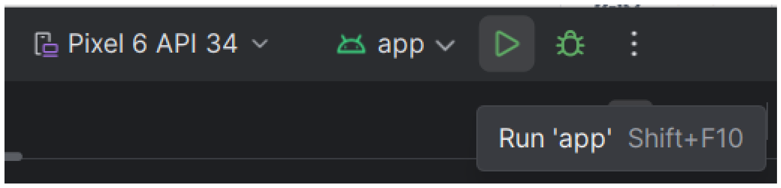


Figure 8: Run App Button

3. If you complete all these steps successfully, you should see the emulator showing something like Fig. 9. **Note:** The virtual device starts just like a physical device. Expect this to take a while—potentially several minutes—for the emulator to start for the first time. The virtual device should open beside the code editor.

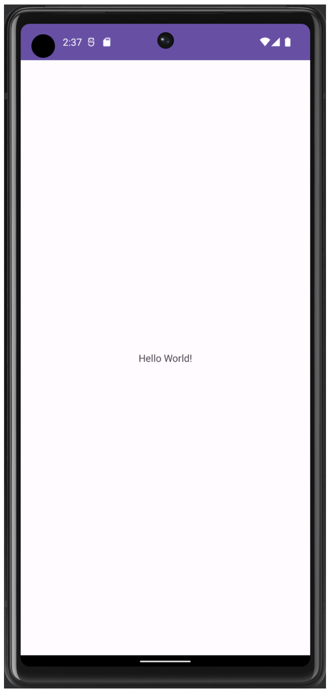


Figure 9: Hello World on Emulator

E Explore Project Files

Now, let's explore the Android Studio interface to get familiar with the file structure.

1. Take a look at the **Project** tab in Android Studio. This tab displays the files and folders of your project. You can see the package named `com.cs407.helloworld` in the **Project** tab. That was the exact name we assigned to this project package at setup. Android Studio organizes the project in a directory structure made up of a set of packages.

2. If necessary, select **Android** from the drop-down menu in the **Project** tab. This is the default view and file organization you'll use.

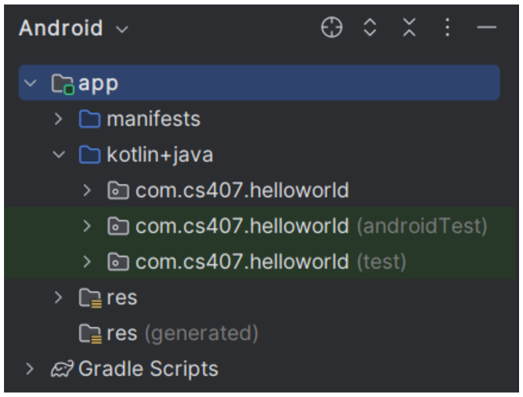


Figure 10: Android Studio Files Structure

3. If you look at the files in a file browser, such as Finder or Windows Explorer, you'll find that the file hierarchy is organized very differently. To browse the files in the same way as in your file browser, select **Project Source Files** from the **drop-down menu**.

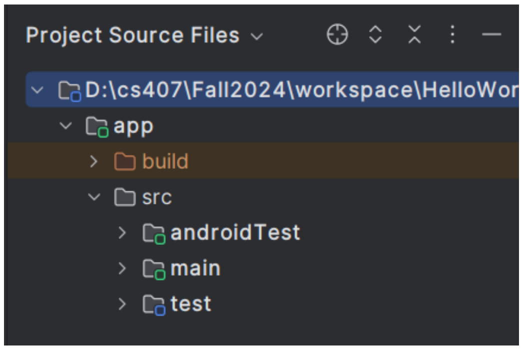


Figure 11: Project Source Files View

4. Select **Android** again to switch back to the previous view. We recommend using the **Android** view for this course. It's helpful when writing code for your project, as it allows you to easily access the files you'll be working on in your app. If your file structure ever looks strange, check to make sure you're still in **Android** view.

Milestone 2: Successfully setup VCS(Git) and push changes to github

`Version Control Setup` - Now you will be learning how to use and integrate a **Version Control System (VCS)** with Android development. We will be using `Git` as our version control software to track your progress. In this milestone, you will **connect** your **GitHub account** to your Android Studio project.

- **Install Git and sign-up for GitHub:** If you don't already have **Git** installed on your machine, download and install it: **Git Downloads**.
- If you do NOT already have a GitHub account, then you can **sign up** at this **link**.
- We will be using **GitHub** in this course to monitor your progress and showcase your projects. Below is an easy step-by-step tutorial on how to set up your first GitHub repository on Android Studio.

- **Install** `Git` on your computer: Visit this **official site** to download and install Git on your computer. Once you do that, **restart** Android Studio and you can start using it with Android Studio.
- **Enable** Version Control Integration on android studio: **Go** to **Menu>VCS>Enable Version Control Integration**

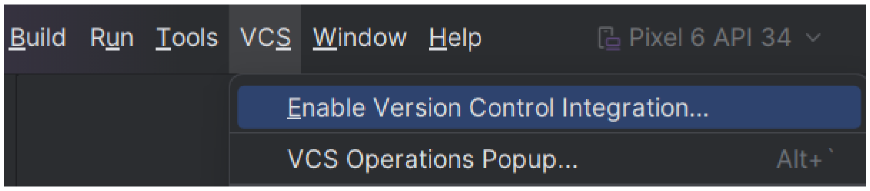
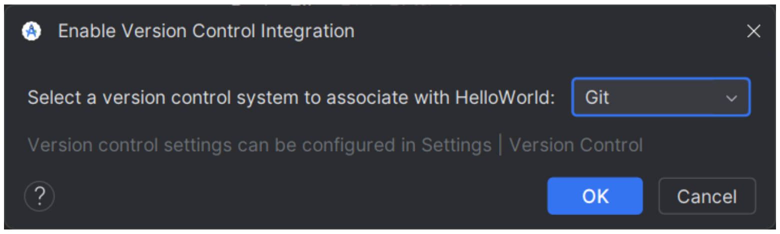


Figure 12: Enable Version Control Integration

- After that, **select** `Git` as your version control system.



Select a version control system to associate with HelloWorld: Git

Version control settings can be configured in Settings | Version Control

Figure 13: Select Git as Version Control System

On doing this, you will notice that all your files will turn red, as depicted in Fig. 14.

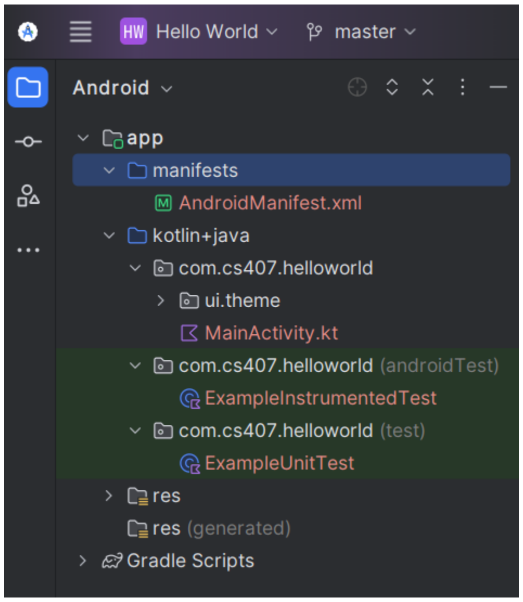


Figure 14: Files Turning Red

- **Share on Github:** Now, go to **Git** → **GitHub** → **Share project on Github** OR **VCS** → **Import into Version Control** → **Share project on Github**.

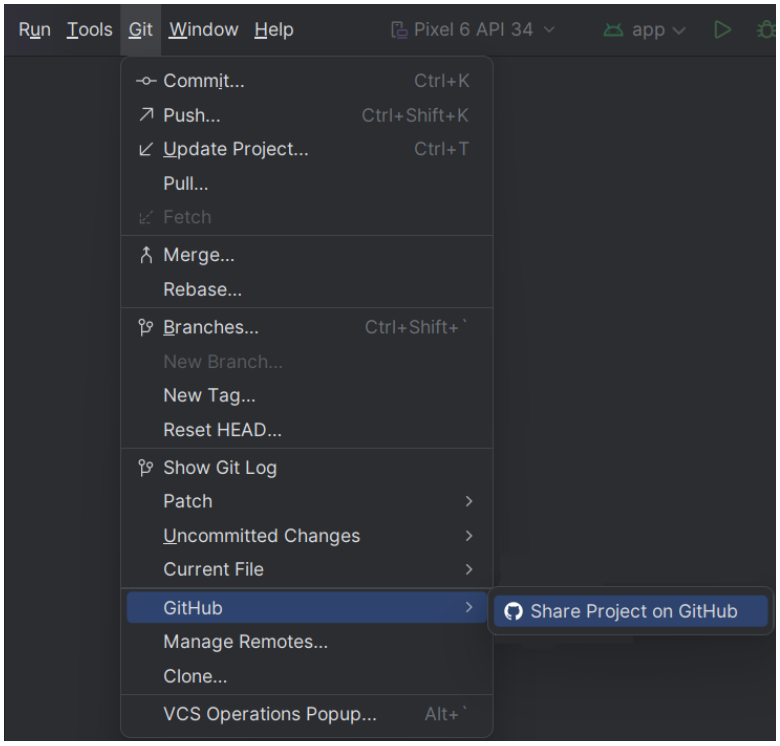
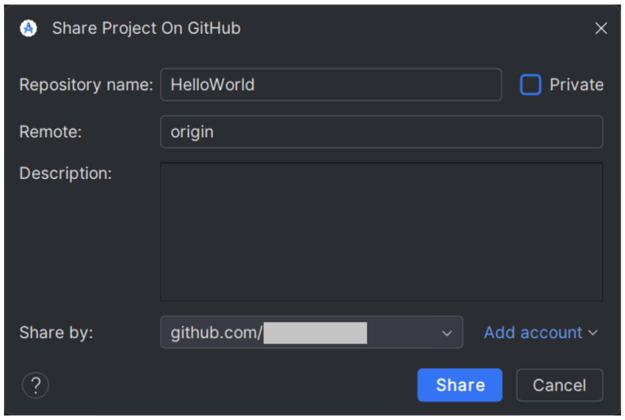


Figure 15: Share Project on GitHub

- Now **give a name** to your repository and write a description if you want. This step may require you to enter **github credentials**.



Share Project On GitHub

Repository name: HelloWorld ☐ Private

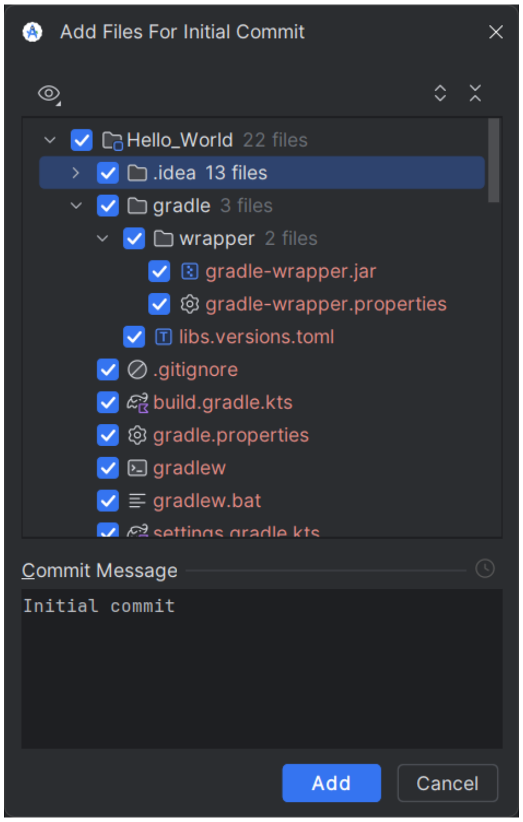
Remote: origin

Description:

Share by: github.com/[redacted] Add account

Figure 16: Name Your Repo

- **Commit and Push your project files on Github (Initial Commit):** Now select the files you want to share and press **Add**. You can also write a specific commit message. After you do so, all your files will turn white or green.



Add Files For Initial Commit

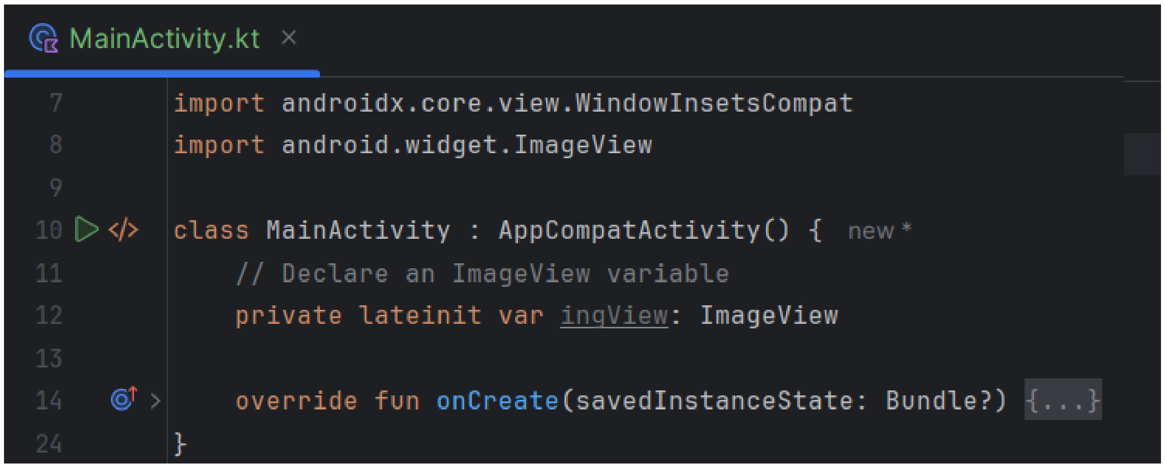
```
✓ Hello_World 22 files
  ✓ .idea 13 files
  ✓ gradle 3 files
    ✓ wrapper 2 files
      ✓ gradle-wrapper.jar
      ✓ gradle-wrapper.properties
    ✓ libs.versions.toml
  ✓ .gitignore
  ✓ build.gradle.kts
  ✓ gradle.properties
  ✓ gradlew
  ✓ gradlew.bat
  ✓ settings.gradle.kts
```

Commit Message

```
Initial commit
```

Figure 17: Select Files to Push to GitHub

- Your project is now under version control and shared on Github, you can start making changes to **commit** and **push**.
- **Commit Your Changes:** We'd like now to edit one file and learn how to commit the changes. Add a new `ImageView` reference to the `MainActivity.kt` as shown in Fig. 18[1]. Notice that after the changes are made, the `MainActivity.kt` file turned blue. That means the file now has **uncommitted changes** (and it doesn't match the file in the Github repository).



```
import androidx.core.view.WindowInsetsCompat
import android.widget.ImageView

class MainActivity : AppCompatActivity() {  new *
    // Declare an ImageView variable
    private lateinit var ingView: ImageView

    override fun onCreate(savedInstanceState: Bundle?) {...}
}
```

Figure 18: Reference of ImageView in MainActivity

- It is up to you now when you want to commit and push. You can do it after every new change or after a lot of changes.
- To **commit and push** the changes, go to the Version Control menu as shown in Fig. 19.
- After that, you will see a commit changes window/panel and you have to write a commit message (mandatory) and then press on **Commit and Push**.
- After this step, you will now push your changes to the GitHub repository. You will do this by navigating to **VCS** → **Git** → **Push** OR **Git** → **Push**.

[1]**Note:** In Kotlin, non-nullable properties (non-primitive type variables like objects) must be initialized at the time of declaration or in the constructor. Using `lateinit` allows you to resolve the compiler error and defer the initialization to a later point.

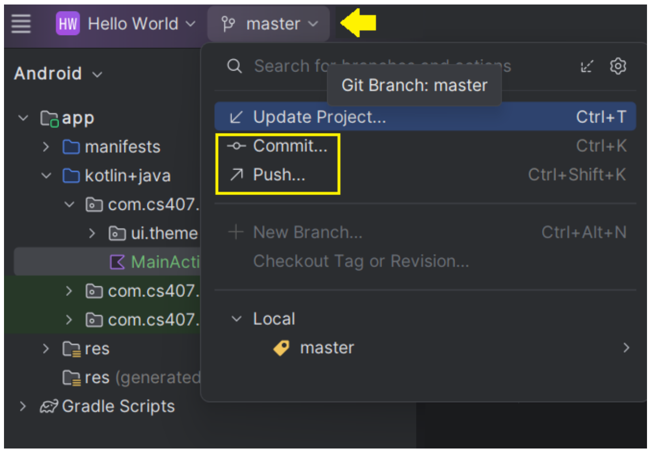
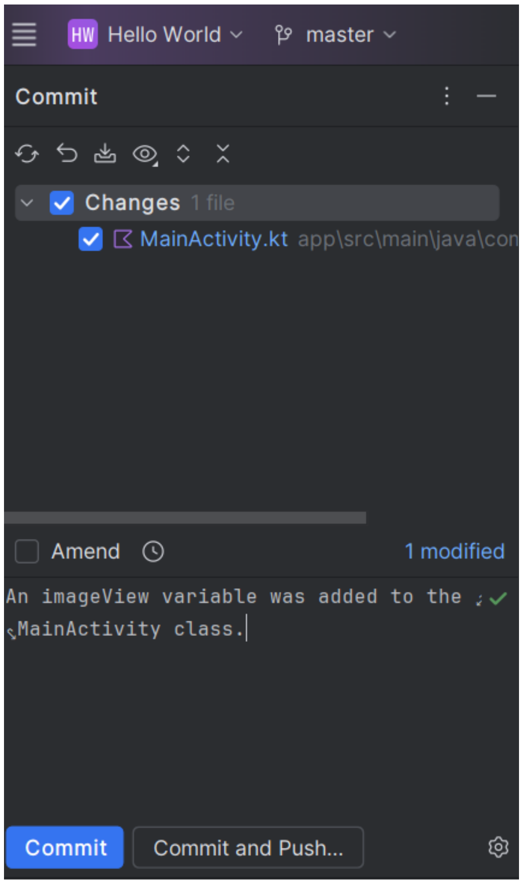


Figure 19: Git Menu at the Upper-Left Corner of Android Studio



```
An imageView variable was added to the
MainActivity class.
```

Figure 20: Git → Push

- Finally, you will see another push window. Just press the push button and all your changes will be reflected in your repository.

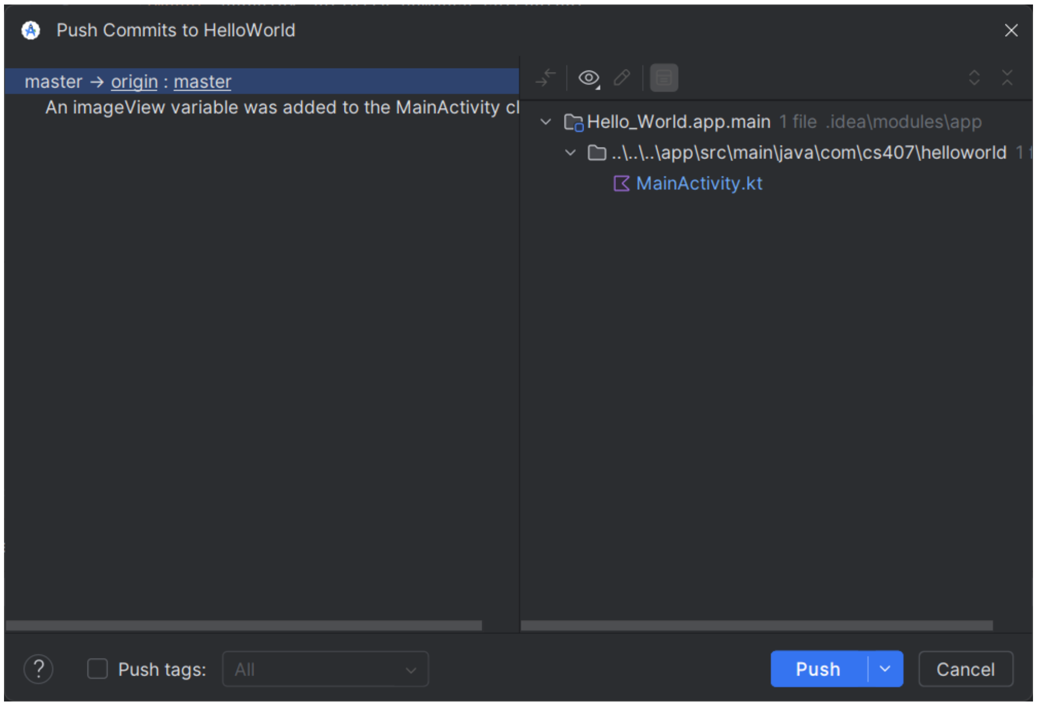

Push Commits to HelloWorld
master → origin : master
An imageView variable was added to the MainActivity cl
Hello_World.app.main 1 file .idea\modules\app
..\..\..\app\src\main\java\com\cs407\helloworld
MainActivity.kt
Push tags:
All
Push
Cancel

Figure 21: Click the Push Button

Congratulations on setting up your first project with version control.

Milestone 3: Share an Android Studio Project on GitHub classroom

In this milestone, we are going to create a new Android Studio Project and you will have to successfully **share** your **local repo** in the `GitHub classroom`.

Create a New Android Studio Project:

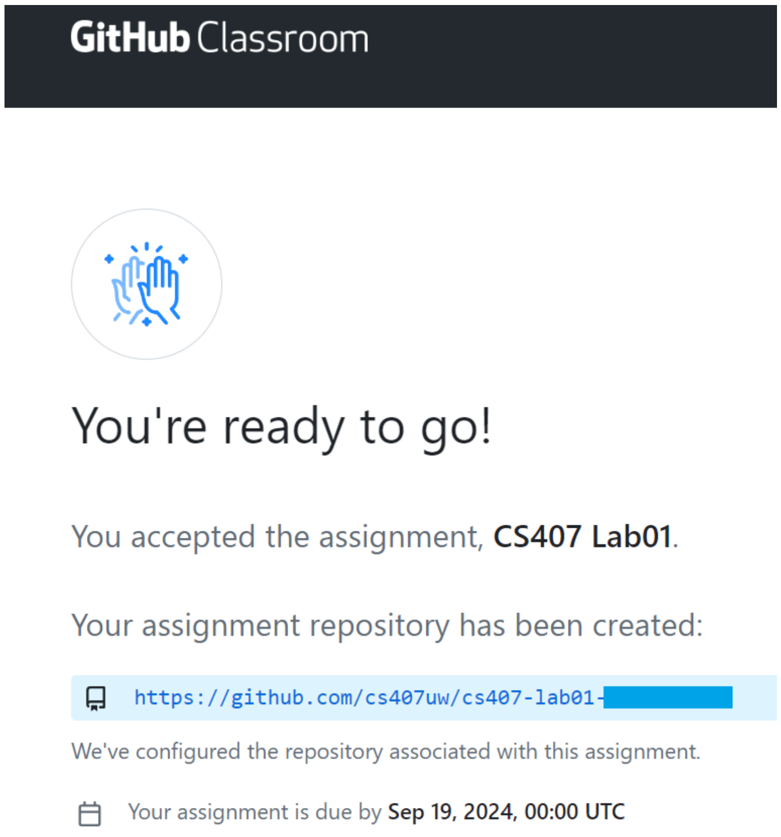
1. Start a **new project** by selecting **File** → **New** → **New Project**.
2. Select the "Empty Views Activity" template. Click Next.
3. Configure your project:
   - **Name** your project "Greetings".
   - Set the **Package Name** to "com.cs407.greetings'.
   - Choose **the Save Location** on your computer, set the **Minimum API Level** (API 24: Android 7.0 (Nougat)), and click **Finish** to create your project.

Setup GitHub Integration of Your Project and Link with Classroom Assignment:

1. **Click** the following link, which takes you to GitHub Classroom and will prompt you to accept the assignment and create a repository for your project.

2. Click **Accept** this assignment's invitation, and a new repository for your assignment will be created on GitHub. This will allow GitHub Classroom to track your work for the assignment.

3. Once you click **authorize** you will get the **repo** with a link (it will be something like https://github.com/cs407uw/lab1-yourgithubnamehere) as shown in Fig. 22.



GitHub Classroom

You're ready to go!

You accepted the assignment, **CS407 Lab01**.

Your assignment repository has been created:

https://github.com/cs407uw/cs407-lab01-[redacted]

We've configured the repository associated with this assignment.

Your assignment is due by **Sep 19, 2024, 00:00 UTC**

Figure 22: Configuration of the GitHub Classroom Repository

4. Next, you will need to **initialize** this project with **git** as done in Milestone 2.

5. Follow the directions provided in Milestone 2: In Android Studio, go to **VCS** → **Enable Version Control Integration** → select **Git** as version control system (as shown in Fig. 12 and Fig. 13). Do NOT share this *greetings* project to your personal github account!

6. Next, **set up** the **remote repository** with the url link of your GitHub Classroom assignment repository. In Android Studio, go to **Git** → **Manage Remotes** → **add**

as shown in Fig. 23.

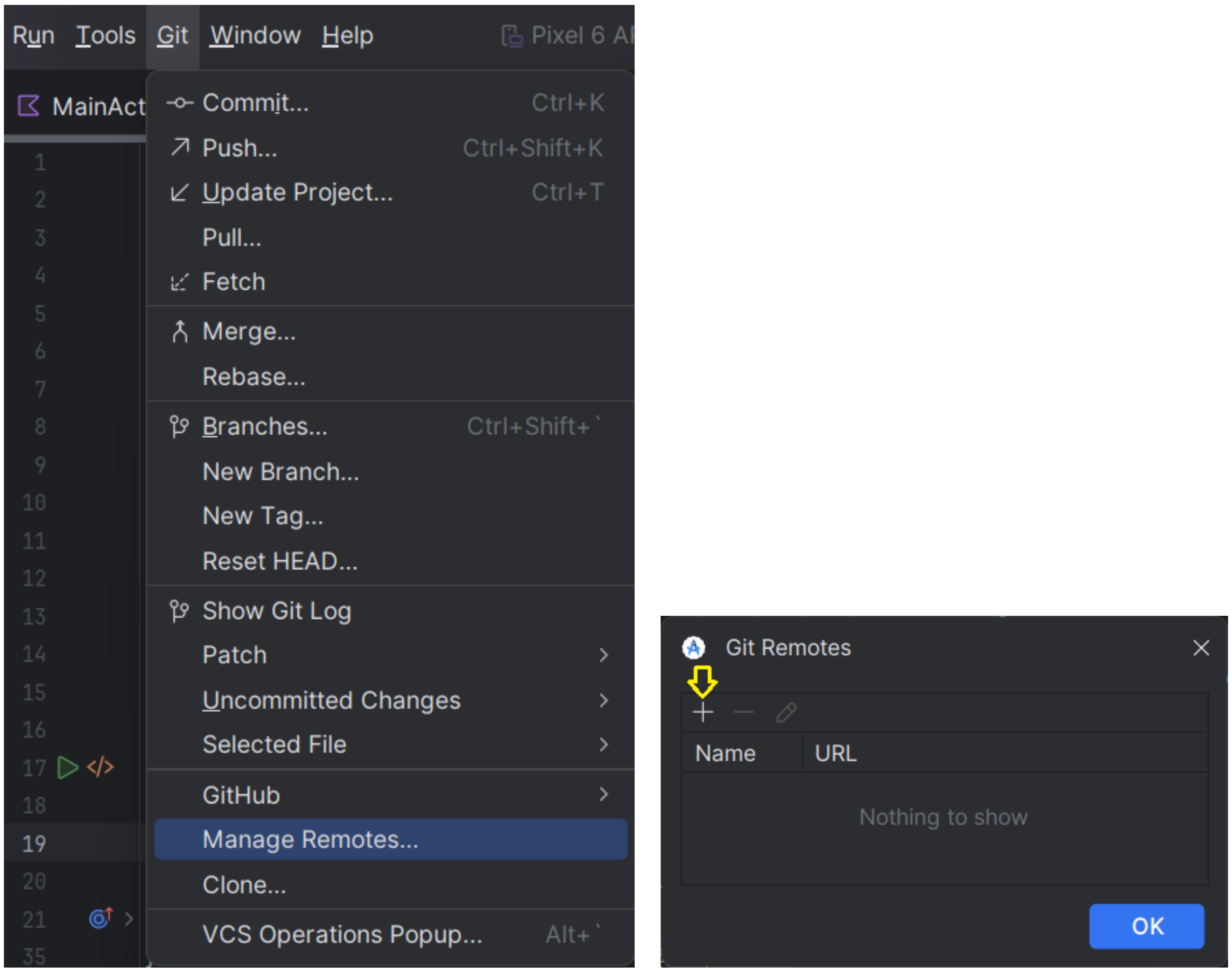
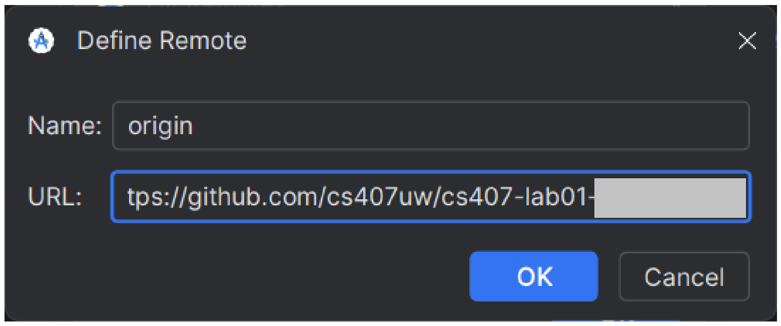


Figure 23: Setting up remote repository

7. Type the `https URL` of your GitHub Classroom repository (see Fig. 24 . Then, it may trigger github login.

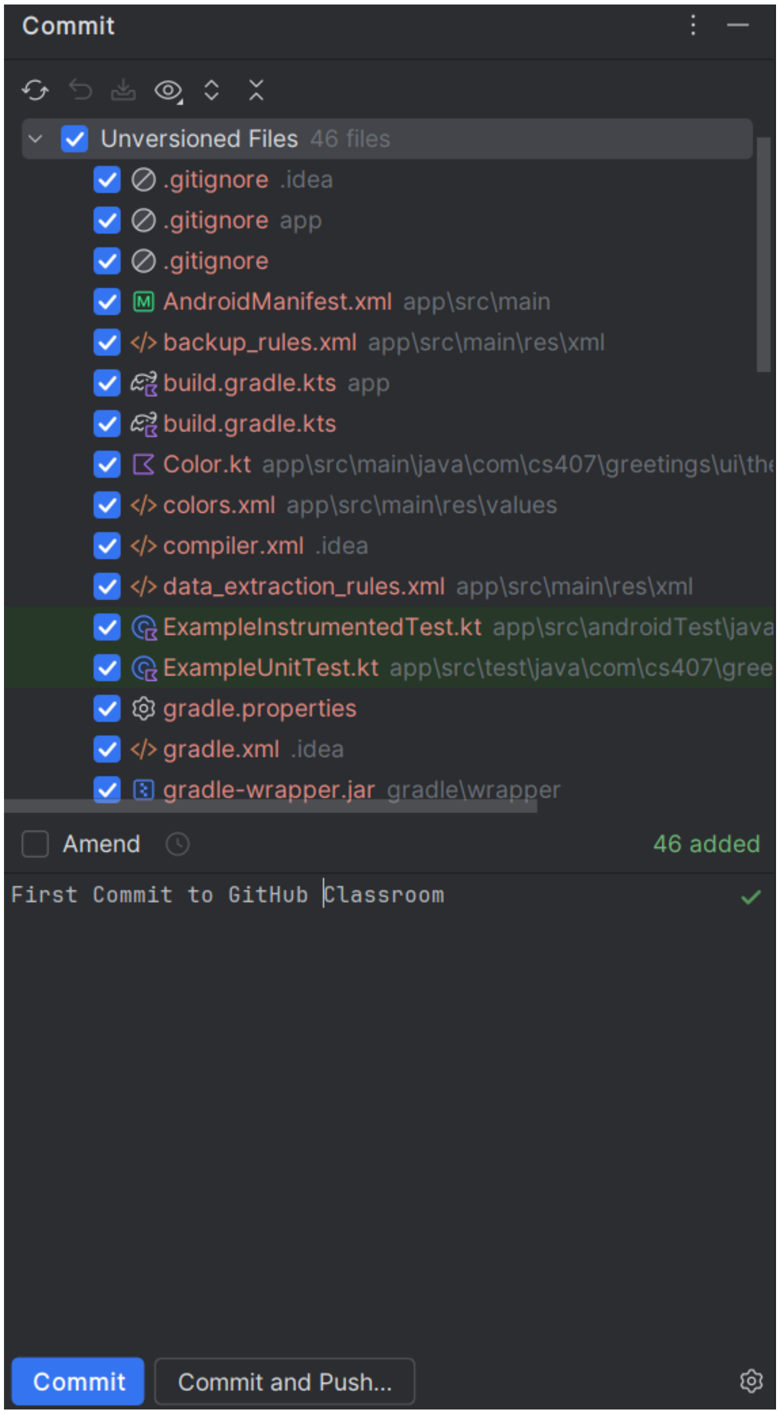


Name: origin

URL: tps://github.com/cs407uw/cs407-lab01-[redacted]

Figure 24: Adding your GitHub Classroom Assignment Repository

8. This will allow the Android Studio to have access to your GitHub classroom assignment repository. After the GitHub authentication is done, you can find that your local project can be found in the commit panel/window. Then commit and push it (reference to Fig. 19 and 20). Use **Git** → **Push** to push your work to the Classroom repository.



Commit

Unversioned Files 46 files

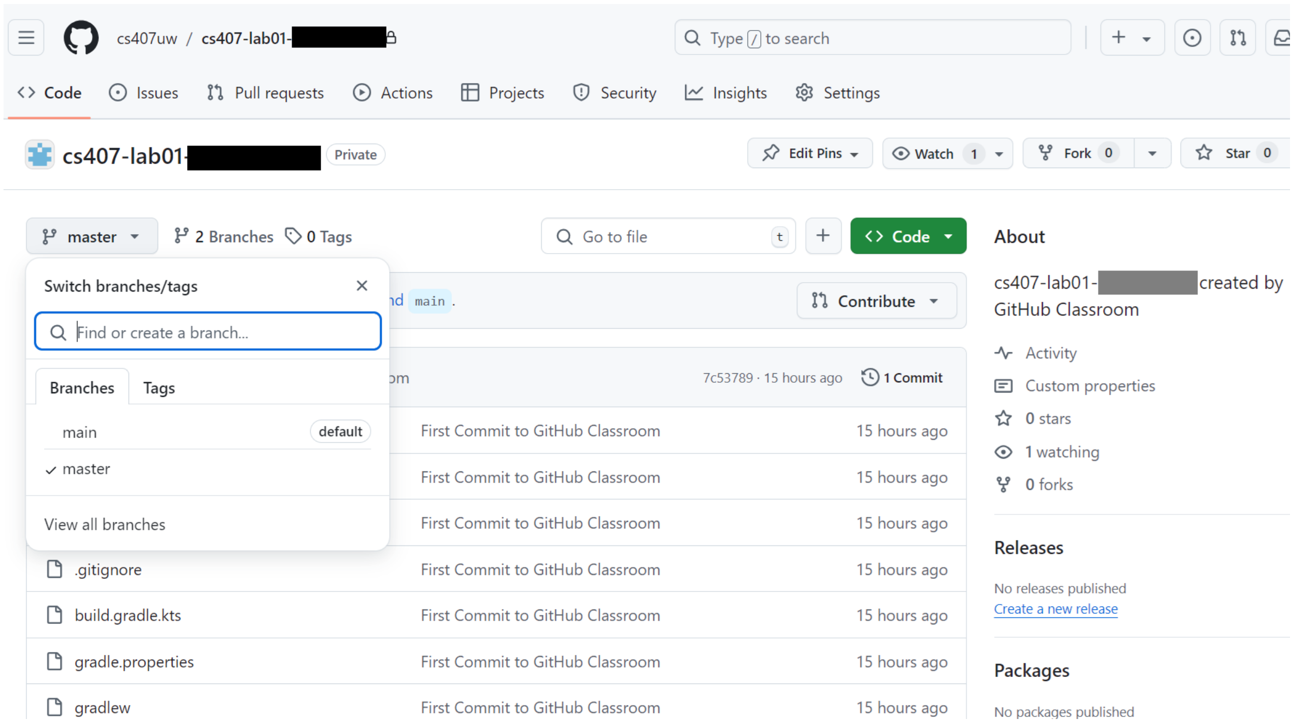
```
.gitignore .idea
.gitignore app
.gitignore
AndroidManifest.xml app\src\main
backup_rules.xml app\src\main\res\xml
build.gradle.kts app
build.gradle.kts
Color.kt app\src\main\java\com\cs407\greetings\ui\the
colors.xml app\src\main\res\values
compiler.xml .idea
data_extraction_rules.xml app\src\main\res\xml
ExampleInstrumentedTest.kt app\src\androidTest\java
ExampleUnitTest.kt app\src\test\java\com\cs407\gree
gradle.properties
gradle.xml .idea
gradle-wrapper.jar gradle\wrapper
```

Amend 46 added

```
First Commit to GitHub Classroom
```

Figure 25: You can see uncommitted changes here

9. You will find your update is checked in the GitHub classroom in branch master! (c.f. Fig. 26)


cs407uw / cs407-lab01-[redacted]

Code · Issues · Pull requests · Actions · Projects · Security · Insights · Settings

cs407-lab01[redacted] Private

Edit Pins · Watch 1 · Fork 0 · Star 0

master · 2 Branches · 0 Tags · Go to file · Code

Switch branches/tags

Find or create a branch...

Branches · Tags

main default

✓ master

View all branches

7c53789 · 15 hours ago · 1 Commit

	First Commit to GitHub Classroom	15 hours ago
	First Commit to GitHub Classroom	15 hours ago
	First Commit to GitHub Classroom	15 hours ago
.gitignore	First Commit to GitHub Classroom	15 hours ago
build.gradle.kts	First Commit to GitHub Classroom	15 hours ago
gradle.properties	First Commit to GitHub Classroom	15 hours ago
gradlew	First Commit to GitHub Classroom	15 hours ago

About

cs407-lab01-[redacted] created by GitHub Classroom

Activity · Custom properties · 0 stars · 1 watching · 0 forks

Releases

No releases published

Create a new release

Packages

No packages published

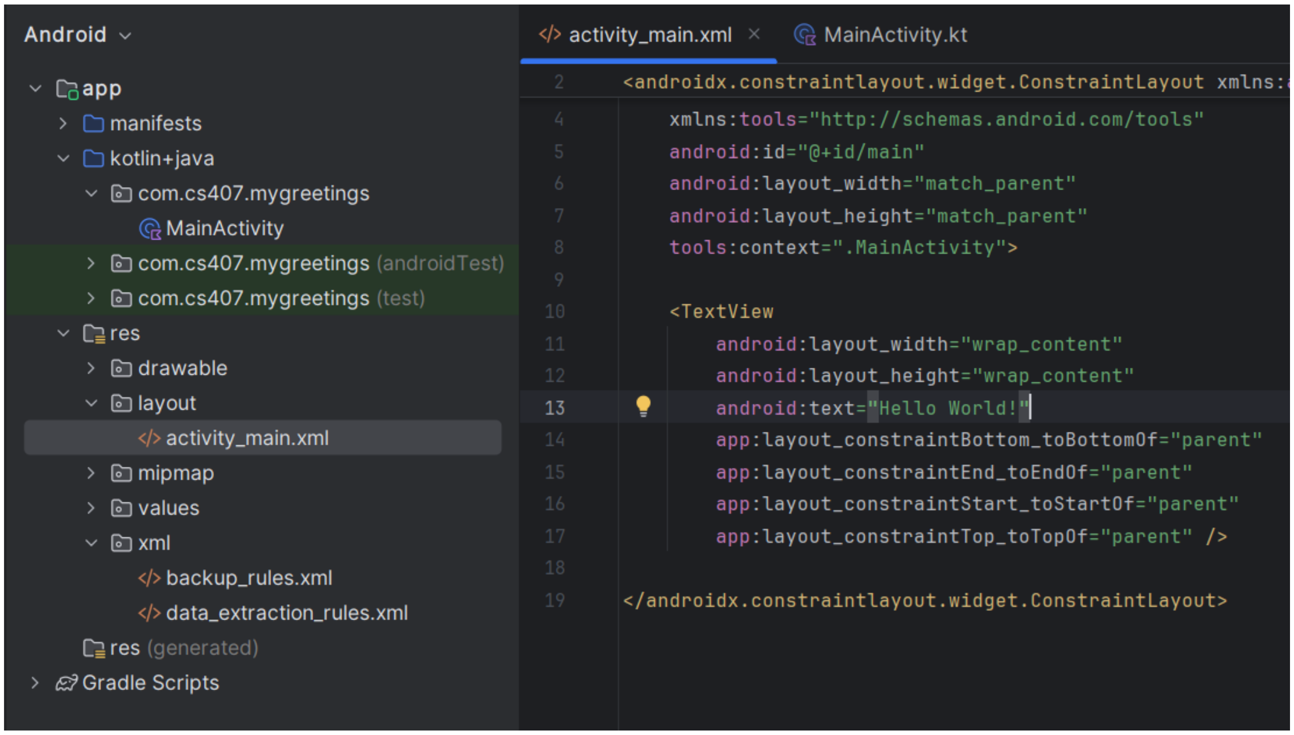
Figure 26: Github classroom local clone

Once you have completed this step, you will be left with a local clone of our GitHub classroom repository. As you make updates to your app in the next section, make sure to commit and push changes regularly by following the same steps. Always ensure you use descriptive commit messages to keep track of changes.

Update the App:

Now you have the most updated copy of the code you will have to make the following changes to the source code of your `greetings` Android project.

1. In the *activity_main.xml* file, found in the files on the left hand side: **app** → **res** → **layout**. Once you open this file from the file explorer you should be looking at some code like the one in the screenshot below.



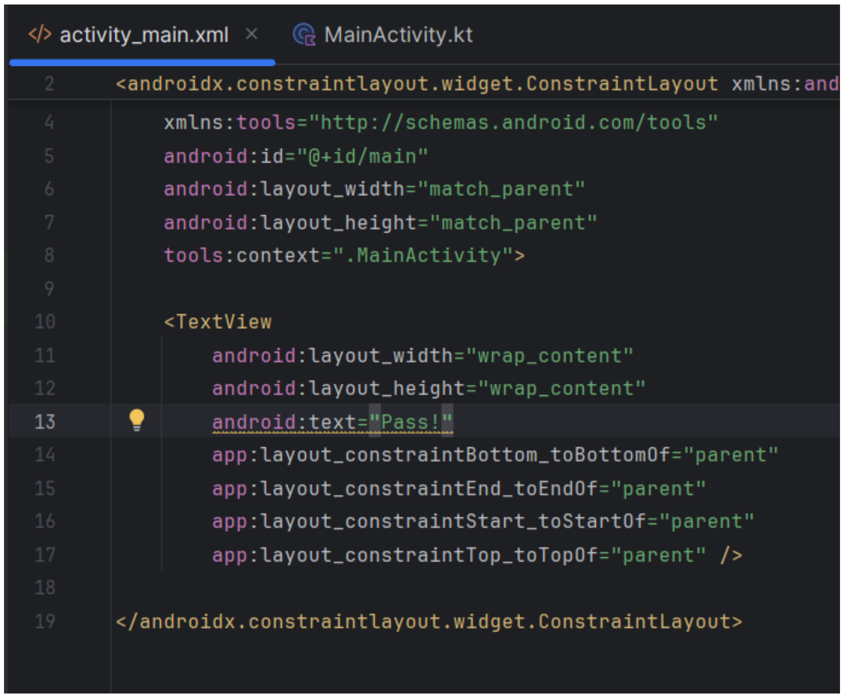
```xml
<androidx.constraintlayout.widget.ConstraintLayout xmlns:
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 27: Starting code for step 1

2. Your **first task** is to **change the text message from the original text to "Pass!"**. You will need to make this change in the following location.



```xml
<androidx.constraintlayout.widget.ConstraintLayout xmlns:and
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pass!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 28: Changing message text

3. **Your second task is to add a Button** by using the design tab. You can use the design tab to drag and drop various components onto your app. For this task, you should select **Common** → **Button** and drag and drop it onto the app as shown in the below figure.

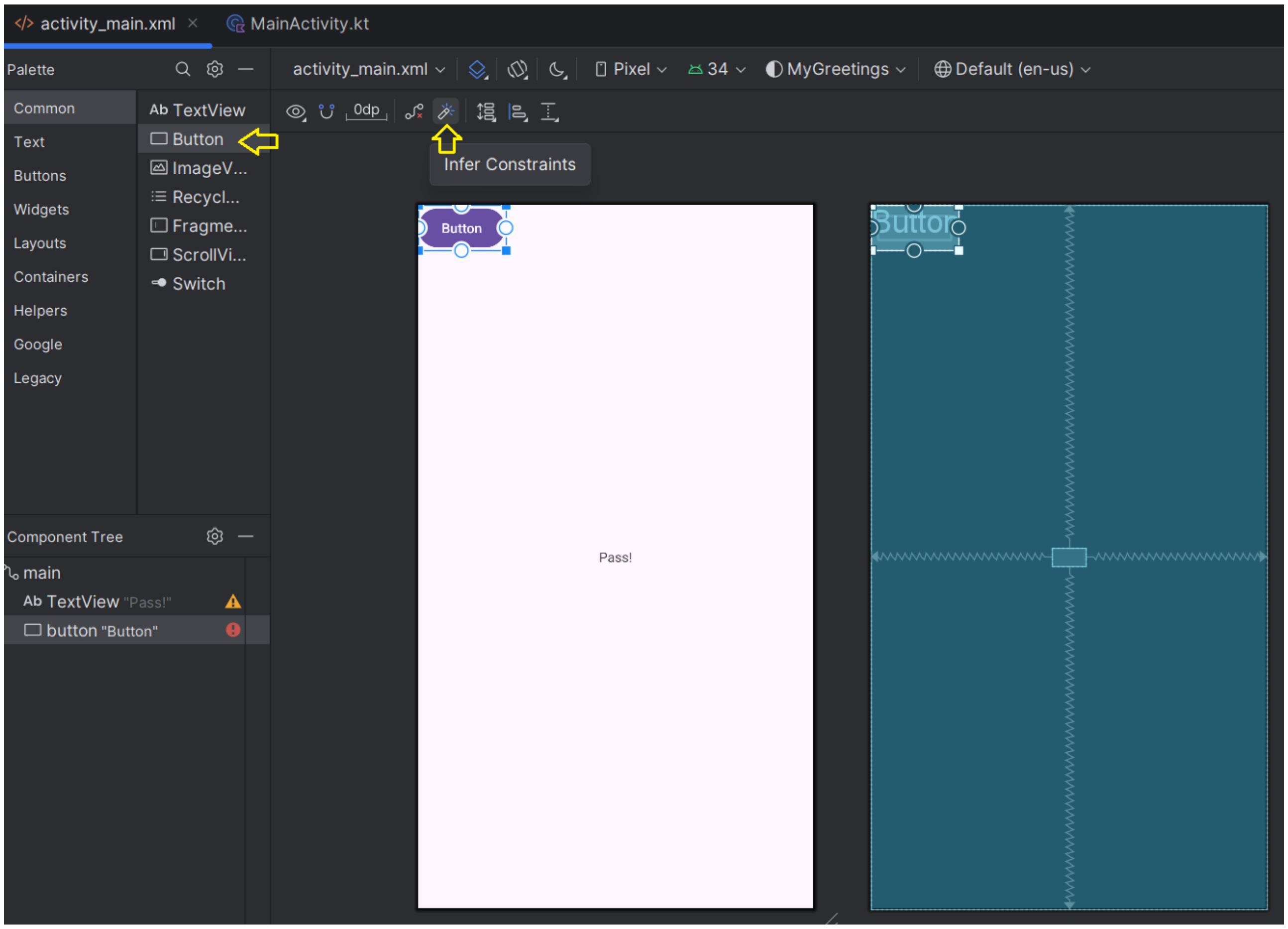
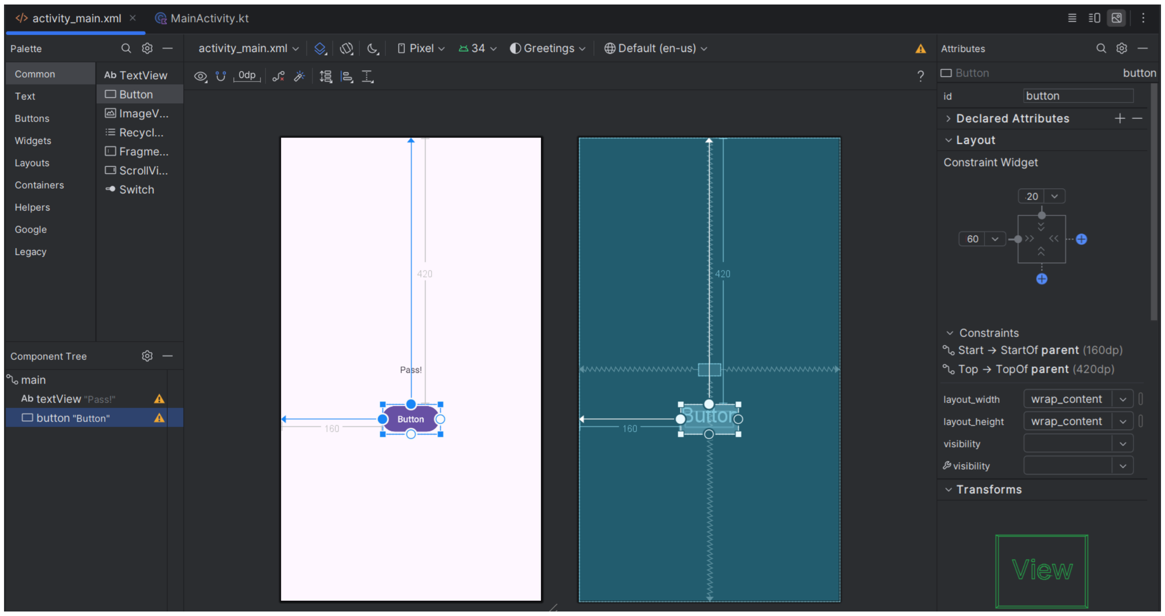


Figure 29: Inserting Button

4. You will now be able to see it on the app preview. You can move it around and place it wherever you'd like as shown in the screenshot below.

5. Once you've placed the button, click "**Infer Constraints**" in the layout editor toolbar (shown by arrow in the above screenshot). This will automatically adjust the layout constraints based on the position of your button, ensuring it stays properly aligned when the app is running.

6. Change the text that the Button displays by going back to the text/code tab or you can change it in the design tab by first selecting the button and then editing the common attributes (common attributes → text) as shown below.

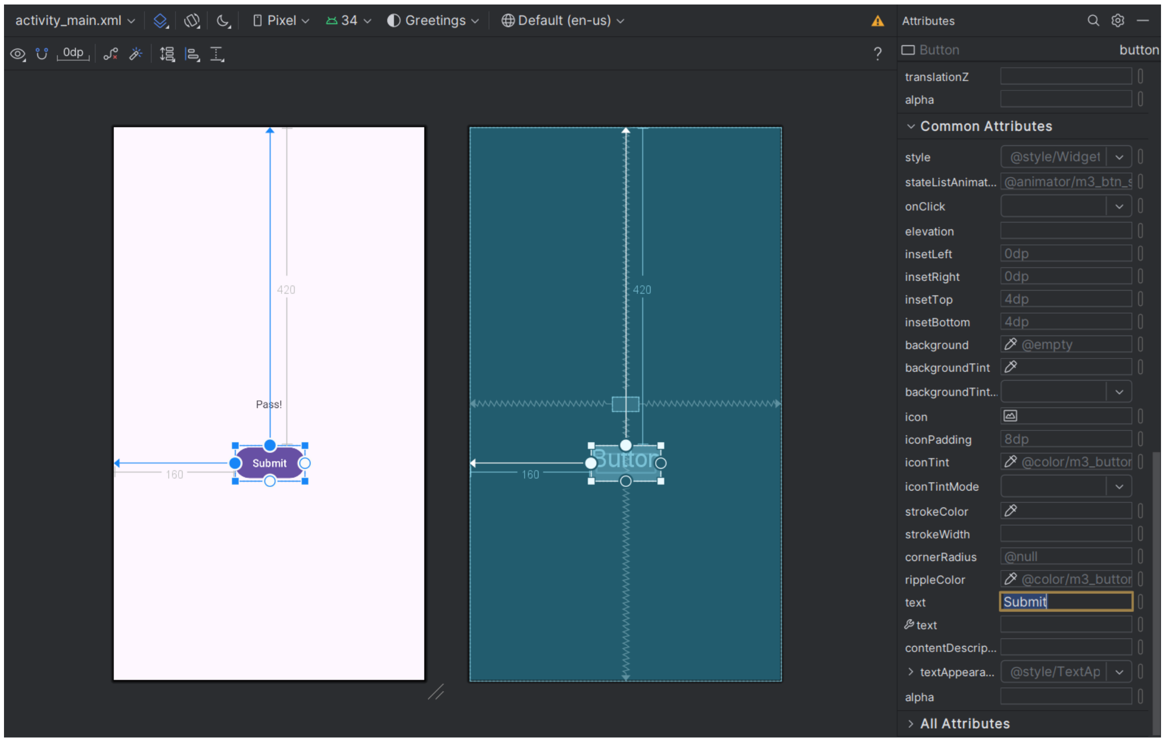


id	button

Constraints
Start → StartOf parent (160dp)
Top → TopOf parent (420dp)

layout_width	wrap_content
layout_height	wrap_content
visibility	
visibility	

Figure 30: Constraint Layout



translationZ	
alpha	
style	@style/Widget
stateListAnimat...	@animator/m3_btn_s
onClick	
elevation	
insetLeft	0dp
insetRight	0dp
insetTop	4dp
insetBottom	4dp
background	@empty
backgroundTint	
backgroundTint...	
icon	
iconPadding	8dp
iconTint	@color/m3_buttor
iconTintMode	
strokeColor	
strokeWidth	
cornerRadius	@null
rippleColor	@color/m3_buttor
text	Submit
text	
contentDescrip...	
textAppeara...	@style/TextAp
alpha	

Figure 31: Changing text for button

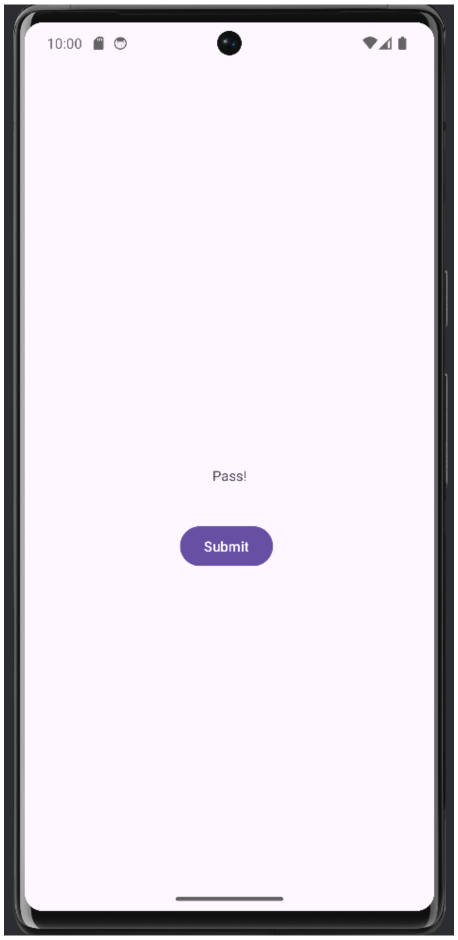


Figure 32: Running App

- As the semester progresses, you will get more and more comfortable with the various resources that Android Studio provides.
- Now that you have made these changes, **`commit` these changes** to your **local version control repo** as you have learned from the git tutorial above.
- After you have made these changes to your android project, the last milestone is to **`Git push` all these changes** to your **remote GitHub classroom repository**. We will be checking the time and date of the last push to your individual repositories and grading you accordingly.

Conclusion

You have done a **great job** to get this far. Generally, the setup is the most boring part of any software development. Now that you have successfully completed this you are ready to take on more complex challenges that are waiting for you in the next many weeks of this semester.

Useful References

- Kotlin for Java Developers: **this** is a good codelab to help you switching from Java to Kotlin.
- Github set up tutorial.
- **Git Tutorial**
- Google has great documentation for Android. This lab uses XML layouts views. If you would like to learn how to build a basic Android app with **compose**, this is a good link to help get you started with such a project.

Gradescope Submission and Demo Grading

Congratulations on finishing this CS407 Lab! The file that you must submit to Gradescope is:

- `activity_main.xml` from your **MileStone 1 app** (included in the `com.cs407.helloworld` repository).

This lab milestones will be graded upon request via a demo session during Thursday labs, Friday discussions, or TA office hours. The schedule of our TA assistance/grading office hours is available here. Please submit a grading request as soon as you are done with the development of your application. Do NOT wait for the due date of this lab (**Due 12:00PM CT on September 16th**) to get your lab graded if you are already done working on it. Penalties to your lab's grade are applied if the demo of your lab is not manually graded before **4:00PM CT on September 16th**.

Before Your Demo:

- Please ensure that you have launched the emulator.
- Make sure your mobile app is ready to run.
- Verify that all necessary permissions are granted (if applicable), and that the app is functioning as expected (w.r.t milestones' deliverables).

During Your Demo:

- Your app should be ready to demonstrate immediately to your TA without any delays.