**Homework 1**

*“Hello Android World”*

*Due Friday, August 24th*

Summary:

This first assignment is meant to be a nice, gentle re-introduction to the rigors of the school year, as well as a forcing function to ensure you download and install Android Studio ASAP, so that you can get your development environment up and running!

Objectives:

Android Studio is the official integrated development environment (IDE) for developing Android applications, and it is what I recommend you use in this class. As is typical of IDEs, Android Studio has a lot of moving parts to it. Although you’ll learn many of the features through extended time and practice, this assignment will introduce you to several of the most fundamental components of Android Studio and a typical Android application. This includes the New Project Wizard, the project directory structure, editing resource files, creating an Android Virtual Device (AVD), gaining familiarity with layout files and both the Design and Text editors, and pushing your project to Gitlab. It will also hopefully inspire you to start thinking about the kind of application you might want to work on for your class project!

**Step 1: Download and Install Android Studio**

The journey of a thousand award winning applications begins with a single download. Go to <https://developer.android.com/studio/> to download and install the latest version of Android Studio appropriate for your device.

**Step 2: Open Android Studio and Make a New Project**

The first time you open Android Studio, it should be very insistent on helping you to create a new project. In general though, to make a new project, you’ll select the “New Project” option from the File Menu (namely File->New->New Project). This will begin the “Create New Project” process, aka the new project wizard. There are a series of dialogue windows that will be presented to you:

**“Create Android Project” window:**

*For Application name, call it “Hello World”, “Homework1”, or a similar demarcation.*

Feel free to accept the default company domain and project location, or change them to suit your tastes. Also leave the checkboxes for C++ support and Kotlin support unchecked.

*Then hit next.*

**“Target Android Devices” window:**

This section of the wizard lets you specify what kind of App you are planning on developing? Something for phone and tablets only? Something for a Google smartwatch? Thinking about diving into the Internet of Things? This is where you select that!

*For now, only check “Phone and Tablet”*

You’ll also have to specify what version of Android you want to use by selecting an API. Later APIs have more features, but also only work on the latest Android devices. Earlier APIs work on more devices, but are less feature rich. APIs have their own number, and are associated with different version numbers of Android, which themselves have code names based on candies and sweets.

*Select API 19: Android 4.4 (KitKat) from the dropdown box.* This is a nice middle ground of working on most devices, while still being sufficiently feature rich.

*Then hit next.*

**“Add an Activity to Mobile” window:**

We’ll be learning all about activities in the coming days, weeks, and months. An Activity can be thought of as any given “screen” in your application; ideally, each activity allows the user to do one specific task.

*Select “Empty Activity” from the options and hit “Next.”*

**“Configure Activity” window:**

Here you will name your activity, and its corresponding layout file. The layout file will specify the graphical user interface (GUI) elements (or “views”) your activity will have, and how those views are arranged in relation to one another.

*Feel free to keep the default names “MainActivity” and “activity\_main” for the Activity and Layout file, respectively. Keep Generate Layout File checked, but Uncheck Backwards Compatibility (AppCompat). (We’ll be talking about Backwards compatibility more later in the semester. If you forget to uncheck it, it’s actually not a very big deal and the rest of your program should work just fine).*

*Then click Finish, and let Android Studio work it’s magic!*

**Step 3: Familiarize yourself with Android Studio**

**I’ll have no way of knowing if you do this first part, but given that you’re gonna be using this for at least a semester, might as well take a few minutes to get to know your new tool, yeah?**

You’ve got a whole new world of stuff to play around with, as Android Studio (and the Gradle build tool) just generated a lot of files for you. There should be two files opened automatically for you: activity\_main.xml and MainActivity.java (or whatever you decided to call your Activity and its layout in Step 2).

The IDE is split into several different panes. The Project pane, on the far left by default, presents the directory structure of your project. That said, there are different views into the structure, that don’t necessarily correspond to how the files are actually stored in your filesystem. By default, you’ll be in the “Android” view, which surfaces the most common files you’ll be working with. However, if you want Android Studio’s file hierarchy to “match” reality, click the dropdown box and switch “Android” to Project. For now, I’m going to assume you’ll be using the “Android” view.

At the top level you should see “app” and “Gradle Scripts” expanding app reveals the folders “manifests”, “java”, and “res”

\*Manifests simply has the AndroidManifest.xml file – among other things, this file specifies what the starting activity is for your application.

\*Java has three folders with seemingly identical names, though you should see one has the word (androidTest) written in grey, and the other has the word (test) written after it. Those folders are for your unit tests. The folder with no “grey text” after it is where the code for your activities (and other Java classes you happen to make) will go. Right now there should just be a single file: MainActivity.java

\*res (short for “resources”) has four folders. **Drawable** is where you’ll place art assets that you hope to display to the screen. **Layout** is where your activities’ layout files will live. **Mipmap** is for your app launcher icons. And finally, **values** is where you specify values for things that will be used throughout your app, such as *colors* to give your app a consistent color scheme, *strings* which, when separated into their own file will ease translating/localizing your app to other languages, and *styles* to give each activity appropriate structural elements.

**Step 4: Create an Android Virtual Device (AVD)**

If you happen to already own an Android phone or tablet, then great! You’ll be able to test your work on a real life device! If you don’t own an Android device: not to worry, Android Studio has you covered! You can basically create an emulator for any existing Android device, allowing you to test your app on a variety of configurations, screen sizes and shapes and APIs.

**Go to Tools->AVD Manager or click on the “pink rectangle with Android head” button near the upper right corner of Android Studio.**

You’ll be presented with all of the Android Virtual Devices you have created. If you haven’t created any yet, you’ll be prompted to make one. Click the “Create Virtual Device” button. First, you’ll be asked to Choose your Device Definition.

**For now, select “phone” from the Category options, and Nexus 5X from the Name options.**

(Although you can feel free to play around and pick any device, I offer the above just as launching off point. For now, I really would recommend sticking to “phone” however). Once you’ve made your selections, click Next.

Now, you’ll prompted to choose what version of Android you want to run on your phone. You want to make sure that what you select here is AT LEAST as up to date as the minimum SDK you selected when you made your new activity. If you followed the instructions here, you selected API 19, KitKat (version 4.4) before, so anything later than that should be fine.

**For now, select “Nougat” (API Level 25).**

The first time you select a “new” API that you haven’t used before in an AVD, you’ll have to download it. So before you can hit next, push the Download button next to the appropriate version of Nougat.

This sadly can sometimes take a while :/

Once it has finished downloading, hit the *Next button.*

**This final screen allows for some final configuration options.**

Including the name of your AVD, its start-up orientation (i.e., does it begin in landscape or portrait mode), and how much the emulator should rely on your computer’s graphics card. For now, I would simply accept all the default options, and click Finish, and then Android Studio will make your new AVD!

Once you have your AVD, you have it forever. Or at least until you delete it. In other words: you don’t have to go through this process ever again (unless, of course, you’d like to test your app on a different device).

**Launch your new AVD by clicking on the green “play” button on the AVD manager window.**

This traditionally takes a \*painfully long time\* to boot up, though you only need to launch the emulator once per session, and then can leave it on and install different builds of your code to it.

**Step 5: Edit Your Layout**

By default, you were given a “Hello World” application when you completed the New Project Wizard. Let’s spice it up a little, and learn about editing layout files in the process.

**Open activity\_main.xml from the Layout folder (or, it is already open, simply click its tab).**

There are two different editors for editing layout files: The *Design Editor*, and the *Text Editor*. The tabs for these guys are near the bottom of Android Studio. Much like with the different “views” of the project hierarchy, the Design and Text editors two different “hooks” into your single application – changes that you make to your application using one editor will be reflected in the other.

In the Design Editor, you’ll see two views (or possibly one, depending on the size of your screen) – “Design View” and “Blueprint View.” The Design view is a real world preview of your layout, while the Blueprint view will only have outlines for each of your GUI elements.

**In the “Design Editor”, click and drag the Hello World “TextView” towards the top of the devices screen.**

Ta-da! You just edited your first app!

**In the “Text Editor” find the following line:**

*android:text="Hello World!"*

**And change the String “Hello World!” to your name.**

Go back to the Design Editor and confirm that, yes, indeed, changes persist between editors; the TextView element should now display your name instead of Hello World.

Now we are going to add an additional TextView to our application! Make sure you are in the “Design Editor” view. Then, in the upper left hand corner of the LayoutEditor is the Palette. The Palette contains many different types of GUI elements. For now, we’ll be using TextView.

**Click and drag “TextView” into either the “Design View” or the “Blueprint View”**

You should now have a second GUI element with the default text “Text View” Clicking on this will open up the “Attributes Pane” on the far right of Android Studio – this is yet another way to edit it!

**Find the “text” attribute, and change it from the default String “TextView” to a few sentences (at least 2-3, but more if you’d like!) about something fun/interesting/strange/whatever that happened over your summer vacation that you’d like to share.**

After doing that, you’ll probably have to resize or reposition your TextView. You can reposition the TextView simply by clicking and dragging it. You can resize it by clicking and dragging on the BOXES that appear on the corners of the TextView when you hover it.

(The Circles that appear when you hover over the TextView are connected to the notion of a “ConstraintLayout”, which we’ll talk about in more detail later).

**Add one more TextView, by again, clicking and dragging it down from the Palette into the Design View. Position it wherever you think it looks the most pretty.**

In this third TextView, using either the Design Editor or the Text Editor, I want you to write at least four sentences describing (one of) your favorite existing mobile applications. Although you can write anything you want about it, some potential discussion points include:

--What is the name of the app (important to include!) Who develops it?

--What “kind” of app is it? (Fitness? Organization? Recreation?) What does it help people to do? How does it do it?

--What about it appeals to you (Does it nurture good habits? Connect you with others? Is it easy to use? Is it beautiful to look at? Does it have personality?) Be specific!

--When and where do you use this application?

--If you could change something about it, what would you and why?

--If you don’t have a favorite app, talk a little bit about that instead. Do you not own a mobile device? Do you find existing mobile applications lacking in some way? What would your dream app be, what features would it have, what would it help people to do?

**After adding this third text view, rearrange all of your views so that there is enough space for all of them.**

**Step 6: Run Your Application on your AVD!**

You’ve just finished creating a heartfelt, personal, name-brand application… let’s get it up and running on a virtual device!

**Click the green play button near the upper right corner of Android Studio**.

You’ll be presented with a dialogue window asking you which AVD you’d like to use. Assuming you already launched the AVD you created in Step 4, you should see that AVD listed under “Connected Devices.” Select it and push OK.

*NOTE: Although in theory you should be able to selected an “Unconnected” device (i.e., an AVD that you haven’t launched yet) from this screen, on rare occasion I’ve found that Android Studio gets upset by this, spitting out weird errors and refusing to launch. Spinning up the AVD first, and then running your application on a “connected device” seems to alleviate these issues. Your mileage may vary!*

**What’s happening here is your project is compiling and building, resulting in an “APK” file (Android PacKage, or Android application package). That APK is then being installed on your AVD.**

Uh oh, odds are good that it actually looks a little ugly, with a lot of text bunched up in the corner! This is a result of using the **ConstraintLayout** mentioned above.

Return to the Design Editor, hover over your text views, and try clicking and dragging on the aforementioned circles of a View. Try connecting a view to the other views of the screen, or to the sides of the screen. Play around with this for a little bit until everything is connected in a way that you are happy with.

**Now push the Green Triangle Play button again, and select the same AVD.**

Hopefully now everything looks a little nicer! Again, we’ll talk about Constraint Layouts in more depth not too long from now.

Congratulations! You’ve finished your first app! All that’s left now is to turn it in…

**Step 7: Submit Your Assignment Using GitLab**

I confess, I am assuming some baseline knowledge of GitLab use on your part, since I know you were introduced to it back in the good ol’ days of 1583, but version control software is always tricky, and Android Studio brings its own quirks into the mix, so let’s go through it step by step.

(as a quick aside, this website: <https://code.tutsplus.com/tutorials/working-with-git-in-android-studio--cms-30514> has a nice overview of using Git with Android Studio, although for this assignment you’ll only need a small subset of what is outlined there).

**The first trick: Creating a new project on GitLab.**

Go to this URL: <https://gitlab.cs.uno.edu> and click on the “Sign in With UNO Active Directory” Button. You should be able to login with your UNO credentials. You’ll then be prompted to create a new project. Please do so, including your UNO user name and the course number in the name of the project.

**Add me as a developer on your project.**

From your project’s main page, click on the “Settings” button along the left hand side of the screen, then click “Members” beneath it. Search for me using the name “bmsamue1” and change the role permission to “Developer.” Finally, click the green “Add to Project” button.

**Give yourself permission to push/pull from the project.**

You can either use SSH or HTTPS, but both require steps. On your project’s main page, there should be orange bars of text across the top, informing you that before you can use SSH, you must first add an SSH public key to your profile, and before you can use HTTPS, you must first set up a password on your account.

*To go the SSH Route*: Assuming you’ve already generated a public/private key pair, you can add your public key by clicking on your “profile picture” in the upper right hand corner, then choosing settings, then clicking on “SSH keys” along the left hand side of the screen. Paste your public key into the box, give it a name, then click the ADD key button. If you don’t know how to generate SSH keys, this website provides a nice tutorial: <https://www.ssh.com/ssh/keygen/>

*To go the HTTPS Route*: The orange bar should let you click on the words “set a password”, which will then prompt you to create your password.

Based on which path you went down, on the main project page, select the appropriate protocol, and copy the URL to the clipboard.

**The second trick: Initialize your Android Studio project as a git repo.**

In Android Studio, go to the VCS Menu, Import into Version Control, Create Git Repo (VCS->Import into Version Control->Create Git Repo).

(At any point along this process, you may get a dialog window popping up asking you about a file named vcs.xml – this is a file that contains project specific settings, and is thus safe to leave out of version control. And indeed, should be left out!).

**The third trick: Adding and Committing your files.**

Along the bottom of Android Studio, in addition to “Terminal” and “Build” and “Logcat”, you should now also see a tab called “Version Control.” Right now, all of your files are unversioned. But right click on the root directory, go down to git, and then select “add” – this will now add all of your files to version control. (if you go this route, you will have to manually UNversion vcs.xml yourself. Sorry!). All of the files should now be in green.

Now, *click on the root directory (important – otherwise you will commit just a single file at a time)*, and click on the “commit” button along the left (three connected circles in a “v” shape, with the far left circle colored green and slightly larger than the others). This will bring up a commit dialogue window, where you can confirm that you are happy with all of the files being committed, and provide a commit message (which simply defaults to whatever the previous commit message was). Click the commit button.

Android Studio may complain about there being warnings. I do not know why these warnings appear… I believe they can be safely ignored.

And, ta da, you’ve now committed all of your files, but they still only live on your local repo…

**The fourth trick: Pushing your files upstream.**

From the VCS Menu, find the Git submenu, then select Push (VCS->Git->Push), which brings up the Push Dialogue window. Near the top you should a link that says “Define Remote” – clicking on that brings up another dialogue box, where you can specify the URL of your remote repo (this should be the URL that you copied to the clipboard at the end of “the first trick.”

It might ask you if you’d like to add this URL to a list of known, safe, places, which I recommend agreeing to.

After that, you should be able to push the “Push” button, and your commit should be pushed upstream!

**The fifth trick: Sanity Check.**

Go back to GitLab and confirm that all of your files did indeed make it up there. If they didn’t, I won’t be able to see them and give you a grade.

**Congratulations! That’s a whole homework assignment under your belt!**

**Welcome to the wonderful world of Android programming!**