Design Document

My partner and I decided to use Android Studios because it provides an editor that is user friendly. It literally allows us to drag widgets into our layout. We are also given the benefit of previewing our layout while editing our XML file, which is basically equivalent to an HTML file. We quickly realized that Android Studios utilized the compiler language, Java, so we had to learn the basics of this programming language. We decided to use an emulator to emulate an Android phone because we found it more efficient to have the phone on our laptop screen. We chose to use the Genymotion Emulator only because it was recommended to us by the Android 101 seminar on the CS50 website.

The basic layout of our project begins in our MapsActivity.java file, which generally ensures that the map that was borrowed from the Google Maps API is displayed onto the screen along with the necessary markers. Looking in a more in-depth view at this file, it begins with the importing of various other files that allow us to have access to many different functions and implementations, such as the Google Maps API; a function that allows us to add the markers; a function that allows to choose the color of our markers; a function that allows us to use latitudinal and longitudinal coordinates; a camera function that allows us to set the focus of the map to wherever we want; and other files that Android Studios automatically provides in order to ensure that the app works as intended. The file then goes into an Activity that makes it so that the camera centers on Harvard Yard instead of Africa (where it was originally centered for default). We then set-up the code that creates the map. We have two if-statements that make sure that we have not already instantiated the map. From there, if the map has been successfully loaded, then automatically zoom into Harvard Yard. This is also where we implement a function that allows us to locate the user's GPS location on the map. Regarding the markers, the last portion of code within the MapsActivity.java file places the markers with the coordinates that are within the hash-map that we created in the file Toilet.java. My partner and I had to manually go to each building and locate each bathroom that is present within the building, which leads to a sense of uncertainty since we do not know if we covered every single public bathroom in the Yard. We then use an event listener function that allows the app to detect whenever the user clicks on a marker. If the user clicks on a marker, the desired information (name of building, the floor that the bathroom is on, whether a student id is required to enter the bathroom) displays on another page.

The next important file that we have is Toilet.java, which essentially creates Java's version of a hash-table called a hash-map. Our hash-map contains 5 fields: title, snippet, lat, lng, and needId. The title field contains the name of the building while the snippet contains the floor that the bathroom is on. The lat and lng fields contain the GPS coordinates of each building and the needId field will display whether the user needs a student ID or not to access the bathroom. The next block of code is the actual information that we insert into the hash-map.

The XML file that we have, called activity_display_info.xml, takes care of the aesthetics of the page that the user is directed to whenever they click on a marker. We are given the choice of manually designing the file or working with the actual code instead, which tends to be a lot more precise. If we take a look at the "Text" version of the file (which holds the XMl code) we can see that we set the padding for the text and the color of the background. We chose the color red for obvious reasons. We

next set the color and size of the text that indicates the title of the building that the bathroom is in. We set all text within this page to white so that it is easily readable. Afterwards, we did the same for the text that indicates the floor that the bathroom is on. The last thing on the XML file is the code for the picture that we implemented. We downloaded the picture from the Internet and then uploaded it onto Android Studios. We set its size and alignment so that it would not interfere with the text. The AndroidManifest.xml file sets up the aesthetics of the Google Maps that we use in our app. This code was automatically implemented for us when we uploaded the Google Maps API. The google_maps_api.xml is where our Google Maps API key is stored. This key is what gives us permission to use the Google Maps API.