COMP 3000: Project Report

Evan Cooper 100975299
Cole Dorma 100965295

1 Introduction

1.1 Context

Our technological world is changing very quickly, resulting in the majority of the types of cell phones available to people being smartphones. We are at a very unique time in our world, in which the baby boomers (a large population of people born between the years 1946 and 1964) are having to switch from their regular, basic functioning cell phones to a new cell phone, without having much choice other than a smartphone. The fact that smartphones are one of the only types of cell phones available to people is a major issue because these baby boomers are being forced into a smartphone. The first cell phone didn't exist until 1973, when it was released by Motorola [1]. Alongside that, the first smartphone wasn't released until 2000 by Nokia [2].

Prior to our project, Android 7.0 (Nougat) was released in August 2016, had implemented a feature to attempt to simplify smartphones [3]. This feature allowed users to create their own custom quick setting tiles accessible from the notification dropdown menu. Quick setting tiles are mainly on and off toggles for the smartphone settings such as bluetooth, airplane mode, wireless and much more. Unfortunately, Nougat is only available on 0.4% of Android phones in the world and this feature is complicated to unlock, understand, setup and to work with [11]. Additionally, third party developers have released two applications to try and implement these custom quick setting tiles as well.

1.2 Problem Statement

With the baby boomers now having to change from their regular, basic functionality cell phones (functionality only consisting of phone calls, emailing, text messaging) and the only cell phones available to them on the market being smartphones, the issue is with the smartphones themselves. Smartphones have quite a steep learning curve coming from a basic cell phone. They are quite difficult to get used to and are hard to operate. Even the learning curve from switching from an iPhone to an Android smartphone is fairly steep as well. On top of this, baby boomers have been using regular, basic functionality cell phones for, on average, at least 30 years. Therefore, the switch from a basic cell phone to a smartphone for these baby boomers is going to be that much harder and the learning curve that much steeper. In summary, baby boomers are being forced into switching to smartphones and it is quite frustrating for them to get accustomed to these smartphones, when they only want their old cell phones basic functionality back.

In regards to the ease of accessibility for the basic functionality that these baby boomers will want back in these new smartphones they are being forced into, our overall goal is to give that to them, on Android phones, before walking out of the store. To do this, we wanted to give these baby boomers very easy access to the basic, core functionality applications - phone, text messaging, contacts, and emailing - that they only used in their old cell phones.

We wanted to achieve this goal by giving access to these basic, core functionality applications somewhere in the Android operating systems (OS) user interface (UI) homepage. We wanted to have all four of these basic functionality applications together and in a easily accessible tray. That tray being in the notification dropdown menus quick setting tiles that is accessible by a simple double swipe down of a finger from the top of the screen.

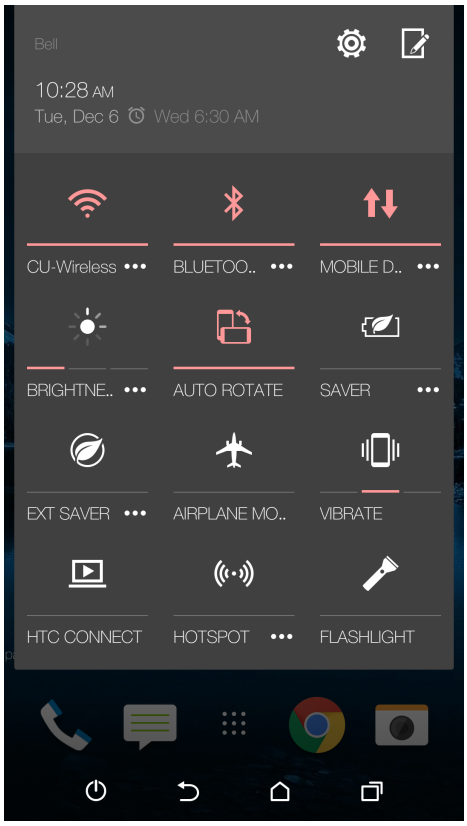

Figure 1: Screenshot of quick setting tiles on an Android phone.

Figure 1 is a real Android phone screenshot of all the quick setting tiles that are accessible on the Android OS by swiping down twice from the top of the screen from any page within the OS.

Having a collective of 3 years of experience between us in IT, we are consistently seeing this same issue arise for people that seem to be around the baby boomer generation. They are frustrated with these new smartphones that they are being forced into and just want their old, basic functionality that they had with their previous cell phones back. They are confused on how

to use their new smartphone, they can't understand how it works and they are struggling with simple tasks on it.

1.3 Result

We were able to create custom quick setting tiles for the stock Android 6.0 OS, which is available to around 26.3% of Android devices in the world [11]. These custom quick setting tiles were made for the phone, the text messaging, the contacts, and the email applications. These custom quick setting tiles' main functionality is that they are a shortcut to the actual default application. The quick setting tiles are located inside the notification dropdown menu that you can see in Figure 1, in which they can be accessed from there, on any UI page of the phone by sliding your finger down twice from the top of the screen. After creating these custom quick setting tiles, this allowed for us to load our custom quick setting tiles into the notification dropdown menu by default. Meaning that when the user takes the smartphone out of the box in the store and turns the phone on for the first time, they will be able to swipe down from the top of the screen twice and simply tap either one of our custom tiles to bring them directly to either their phone, text messaging, email or contacts applications.

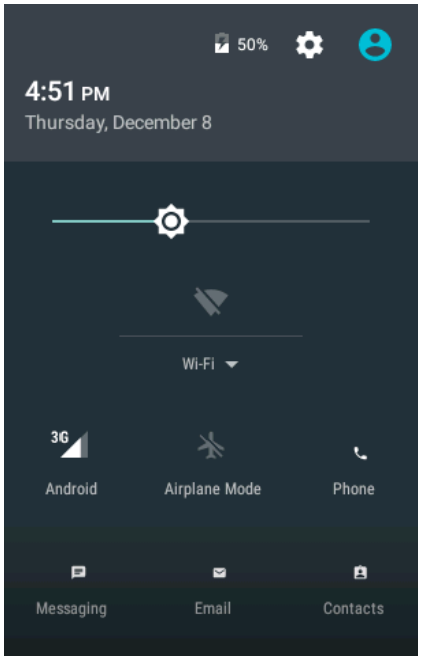

Figure 2 : Quick settings with custom tiles included.

Figure 2 : A screenshot of the quick setting tiles tray including the custom tiles for fundamental smartphone applications. This screenshot shows default tiles (Android and Airplane Mode), alongside the four custom accessibility tiles that we have implemented and added. Those being the phone, messaging, email and contacts tiles.

The solution that we came up with to this issue consisted of many steps and many different key pieces of software. We initially needed to start by using either a Linux or Mac OS X machine. From there, we used a Java Development Kit to handle any Java code that we needed to implement, add or change. We then had to download a stock Android 6.0 OS and download all of the specific binaries for that version of Android to a case-sensitive disk image. Alongside this, most of the downloading, decompiling, compiling and running was done in the bash terminal on a Mac OS X machine.

1.4 Outline

The following content of this report will be structured very much like how it has been previously structured. Section 2 will be more on the background information of this project and describing past software more in depth. Section 3 will consist of a more in depth explanation and analysis of our final result with interface screenshots and explanations. Section 4 will have our honest and truthful evaluation of our project, the different user scenarios, experimental results, and claims with explanations. Lastly, Section 5 will consist of an in depth conclusion involving the summary of our work, the relation of our work to the course and future work with this project.

2 Background Information

In August of 2016, Android 7.0 (Nougat) was released. In Nougat, and many other versions of Android, the user has the ability to swap in and out only a small, very specific list of quick setting tiles. The user is very limited as to the quick settings tiles that they can swap in and out themselves in the sense that those tiles are more of quick settings, rather than shortcuts to any applications. But in Nougat, there was additional functionality that was added that hasn't been added in any previous version of Android before. This additional functionality enabled the ability for the user to be able to make a lot more applications on their phone, default or downloaded, into a quick settings tile with the help of some additional downloadable applications. In order for the user to achieve this the limited way through Nougat's implementation, there is many steps involved. The first step being that the user needs to enable that applications quick setting toggle, if it has one. The user then needs to go into their quick setting tiles UI settings page, find the quick setting tile they would like to add, tap and hold it, then drag it into their quick setting tiles page. If the user wants to have access to a lot more applications and settings that can be added to their quick setting tiles page, they need to install an application called "Tiles", an application that costs $2.89 in the Google Play Store. In which once "Tiles" is downloaded, that opens up a lot more possibilities for the user to add a lot more applications and quick settings to their quick setting tiles page [4].

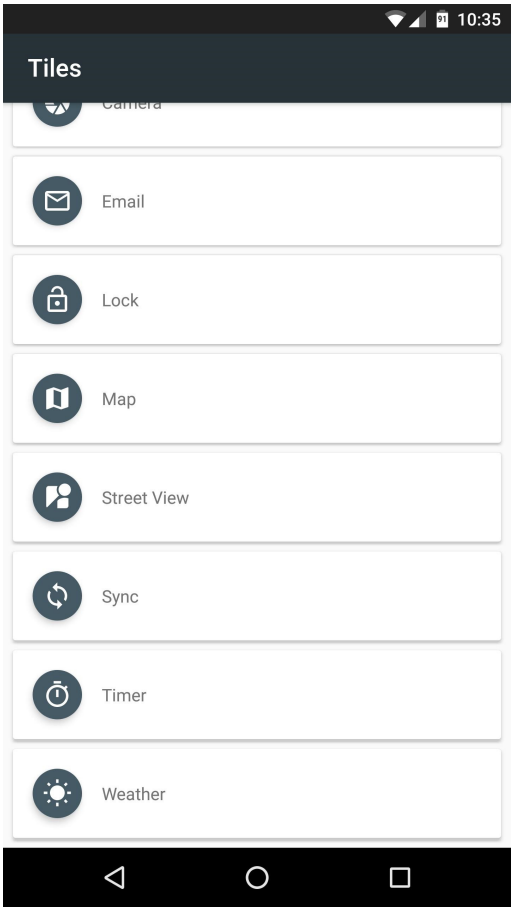

Figure 3: “Tiles” application UI screenshot [4].

Figure 3 is a screenshot of the “Tiles” downloadable application on an Android phone running Nougat. The screenshot is showing the possible additional quick setting tiles that “Tiles” offers to the user that can be added on top of Nougat’s built in quick setting tiles added functionality explained above [4].

Nougat has added quite a bit more accessibility for the user to modify their quick setting tiles, as well as adding more possible options for applications and quick settings that can be tiles with the help of the application, “Tiles”. But, the process for the user to achieve this from start to finish isn’t a very simple task. It takes knowledge and experience working with Android devices. Baby boomers who are used to only the simple and basic functionality cell phones struggle to navigate to their phone application on smartphones, let alone perform each step stated above to customize their quick setting tiles themselves. On top of this, to unlock the ability of adding a lot more applications and quick settings as quick setting tiles, “Tiles” needs to be bought and downloaded from the Google Play Store.

In regards to Nougat's additional implemented functionality with the quick setting tiles in linkage with the downloadable application, "Tiles", this does start to solve the issue that we have solved in this project. It allows the user to add applications to their quick setting tiles but it is limited to only specific applications and limited to Android 7.0 Nougat. We have built upon this same concept by adding the default, basic functionality applications as quick setting tiles to the Android 6.0 OS. On top of this, we made sure that those applications are added to the quick setting tiles to the stock Android 6.0 OS by default. Therefore, when the user first takes the phone out of the box, turns on the device, and swipes down from the top of the screen twice, they will see the phone, text messaging, email, and contacts applications as quick setting tiles. We did this for the simplicity of the user not having to do anything extra in the settings or download any applications to have the ease of accessibility to these core applications.

The above previously talked about softwares that are related to our project are specifically for Android 7.0 Nougat, but there is another piece of software that is achieving something similar to what we are doing for Android 6.0 as well. This piece of software is an application called "Custom Quick Settings" that is downloadable through the Google Play Store. Once this application is downloaded, the user needs to perform a few steps in order to be able to use "Custom Quick Settings" for it's intended use. The user needs to activate their System UI Tuner by holding the settings shortcut in the notification pull down menu first, and then go to their default quick setting tiles and add the broadcast tile, then the user is able to go back to the "Custom Quick Settings" application and add custom applications and quick setting toggles to their quick setting tiles [5].

The "Custom Quick Settings" application is definitely a stepping stone in regards to solving the problem with the baby boomers by allowing them to add applications as their quick setting tiles. We have an Android phone running Android 6.0, therefore we attempted to test this application but it didn't work for us. "Custom Quick Settings" prompted us with an error message telling us that it wasn't compatible with our device. We attempted to try to fix this error but in the end, we couldn't get "Custom Quick Settings" to work on our Android device due to this compatibility issue. Unfortunately, the steps required to get "Custom Quick Settings" working aren't very simple. Most baby boomers will struggle with these types of steps as they are only used to simple and basic functionality cell phones and do not have much experience with smartphones, let alone the Android OS. On top of this, we have many years of experience using Android phones and we could not get the application to work on any of our Android devices due to this issue. Therefore, it seems that this application is specific to a certain version of the Android 6.0 OS.

In relation to the downloadable application, "Custom Quick Settings", this application has partly solved the issue that our project has solved as well. It allows the user to add applications as quick setting tiles. We have built upon this idea by adding the core, basic applications directly to the quick setting tiles in the stock Android 6.0 OS by default. This ensured that when the user first took the phone out of the box, they would be able to swipe down from the top and see those core, basic applications. With this, it wouldn't require the user to

implement any setup steps or download any applications. The smartphone would be setup for the ease of access to these core, default applications for baby boomers right from the start.

3 Result

Our goal for the beginning of this project, to enhance the accessibility of today's complicated mobile devices, has been achieved. The fundamental components of cell phones that boot from our custom distribution of Android 6.0 OS are now much more accessible. The quick settings tiles, acting as a short cut to these core applications, are available from every running state of the device. Additionally, since these tiles are embedded into the system's notification tray, they appear to be very native to the device.

Using existing code from the stock Android distribution as reference, we have successfully been able to create our own independent quick settings tiles that perform our own custom actions. Each of these tiles is dedicated to opening a separate application when activated.

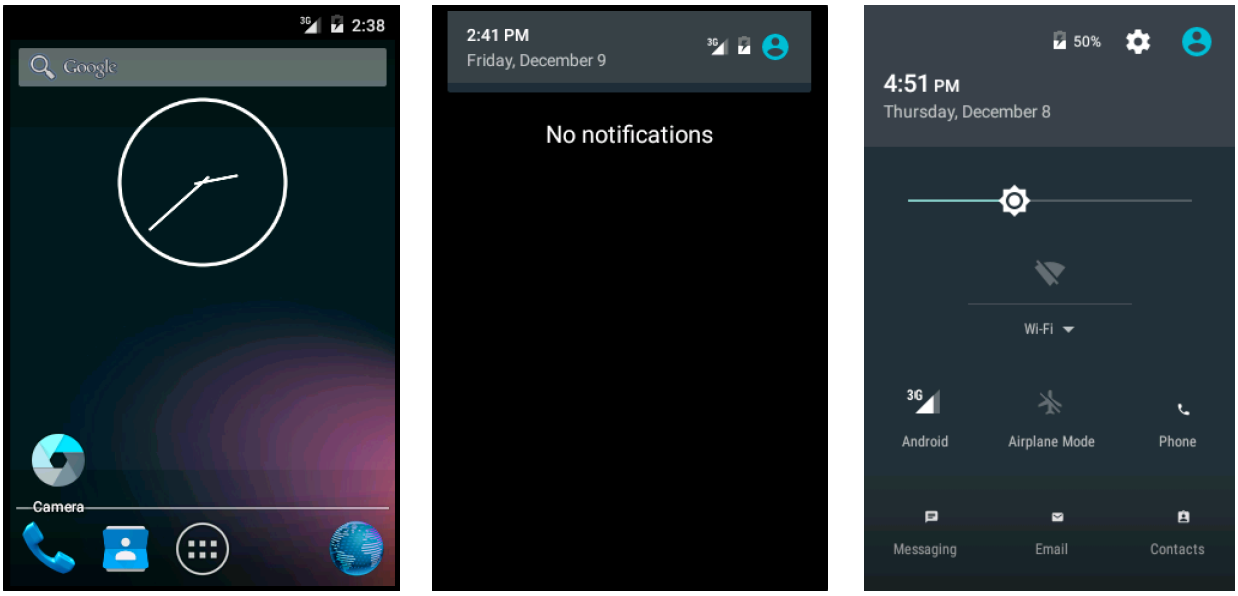

Figure 4 : A series of screenshots displaying the views when using the custom quick settings tiles.

Figure 4 shows different stages of the device when using the quick settings tiles. The left screenshot is the device's homepage. The middle screenshot is the result of the user swiping down from the top of the screen, opening the notification tray. The right screenshot is the result of a second swipe down which will expand the tray and display the quick setting tiles nested within.

From any state of the device, users are able to open the notification tray as displayed in Figure 4. Opening the notification tray will naturally display any notifications that the device's applications offer to the user. With a second swipe, the the screen presents the quick settings tiles embedded within the already opened tray. Visible are the phone, messaging, email and contacts tiles, which act as shortcuts to those applications. Clicking on one of these custom tiles will dismiss the tray and open the related application, and thus dropping the user directly where they want to be.

Once activated, a tile will prepare to launch the given application. The first step is for the tile to use it's own environment information, or Context [6], as the Context for the soon-to-be launched application. Provided by the Android system, this environment can range anywhere from the lock screen to anywhere deep inside the Activities of any application. This information was the key factor in brining the accessible experience everywhere within the device. The next step is to retrieve the launching functionality of the application it's self. By using each application's own launch services, the tiles are acting as a shortcut to their paired application. This allows for re-opening applications to their previous state, if they have already been used. Finally, the application is launched, using the above Context and launching functionality together. In order to further augment this function's accessibility, it was critical that the notification tray be dismissed while the application is being launched. Doing so has created a seamless transition from the tray to the destination, and has diminished the amount of steps required to reach the same destination.

In order to inject these custom tiles into the notification tray, there were other tiles which had to be removed or shuffled around. It is not enough to create a these tile classes in Java, but it was necessary that the system's resource files include them in a list to be pre-compiled and ready for display in the notification tray. In addition, there are many system Java classes that must be modified in order to include each custom tile. Most of these modified classes lie within the SystemUI package, which is included by default in the Android 6.0 OS. For example, QSTileHost, a class that manages the compilation of each required tile, must have the means to create new instances of each custom tile. This meant importing newly created resources into the default resource managers, so that they were included at compilation time. The fact that the integration of the accessibility tiles occurs at the system level, means that they are created and dealt with during the compilation of the OS, as opposed to running processes onto of the OS from a mobile application such as "Tiles". Theoretically, once the system has been booted onto a device, the overhead of additional features will be completed, thus maintaining speed and responsiveness throughout the lifecycle of the system's uptime. This key advantage has been displayed in our accessibility survey, described in the evaluation section. By nature of the tiles, they can be re-arranged to any combination, and even can be shown or hidden. As each Android device can vary in screen size, and thus have varying real-estate for quick settings tiles, it was important that our custom tiles would always be directly visible to the user. In order to achieve this goal, it was once again the system's resource files that needed to be edited. By prioritizing

our custom tiles to appear first, we have created a uniform and simple experience across all Android devices booted from our distribution.

There are various different scenarios where our software is used. The fact that the visual components and activation methods are the same throughout every scenario creates a feeling of unity throughout the system.

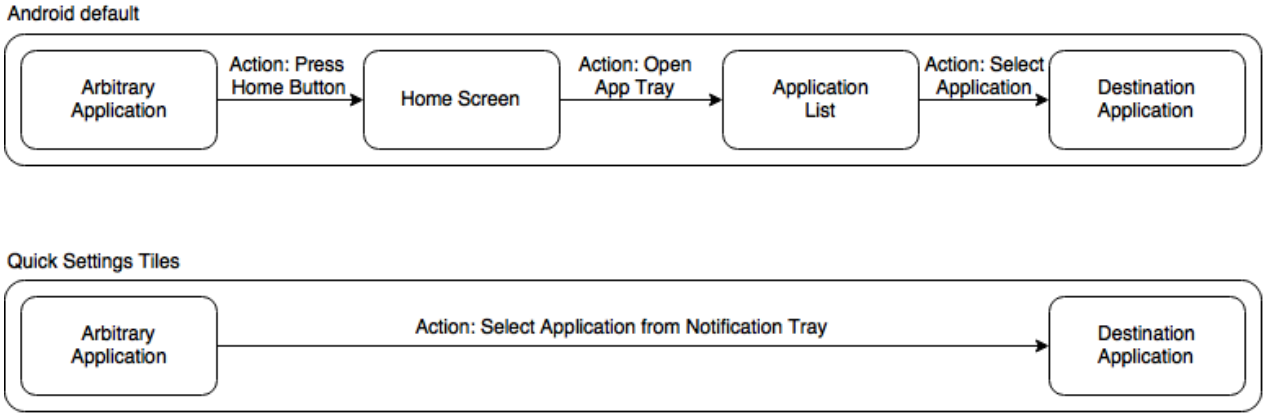

Figure 5: Flow of reaching one application from another application.

Figure 5 compares the different parts of the OS that must be accessed when a user is using one application and is trying to navigate to another one.

Baby boomers, who are mostly unfamiliar with the advancement of smartphones and mobile devices, are the target audience. When they are unwillingly forced by the market into purchasing a smartphone, our feature will allow them to have direct and easy access, to what they want our of their new device. As shown in Figure 5, and according to "A Beginner's Guide to Android" [7], the common method users use to navigate their device from wherever they are to their desired application, they must pass through a series of steps. These include returning to the home screen, opening the ever expanding application list and finding the desired application to launch. With our feature, users are able to launch any of the four applications immediately. By eliminating these intermediary steps, users will be able to more easily remember how to independently use their smartphone. Additionally, these features are readily available straight from the lock screen. Without even having to unlock their device, users are able to open the applications directly.

The interface of our quick settings tiles has adopted the previously existing interface of the Android system notification tray. The custom tiles appear and feel natural to the system. Already existing within the notification tray, are the notifications themselves, being indicators of events happening within each application. Our integration of access to the fundamental components to a cellphone within this event space, has centralized the location where said events

are discovered, and where the related application is launched. With a quick double swipe from the top of the screen downwards, the user has access to everything they desire in a cell phone.

4 Evaluation

Critical evaluation criteria that we had presented in our proposal included the accessibility, performance, and visuals. These three domains are the primary benchmarks that this project is build and tested upon.

We conducted a brief survey including friends and family within our target audience who currently own smartphones, in order to evaluate the accessibility from the user's perspective. Survey participants were asked to use the device as if it was their own. After they gained familiarity with the emulator, we asked the users to perform simple tasks of opening the phone, text messaging, contacts and email applications. Only 24% (6/25) of the users we able to successfully navigate to each core application without difficulty. Afterwards, we introduced the users to our custom accessibility tiles in the quick settings tray. With a concise explanation on how they work and the motivations behind the tiles, participants were instructed to use them, and evaluate the difference in ease of use between the device's core applications and our custom accessibility tiles. In total, 84% (21/25) people agree that the integration of core applications within the custom settings tray is more simplistic than the default Android OS way of launching the same applications. The survey takers who disagree provided feedback of various proposed solutions or alternatives. A re-occurring comment from the minority group suggested to place the desired system applications on the first page of the home screen, and remove all other applications. As this could definitely be a solution to our addressing problem, we feel that there are still some flaws that may have been overlooked by the subjects. The initial setup of this solution might be a difficult task for some users. There are many complicated steps that a user must take in order to change and re-arrange application icons on the home screen. Included from installation of our custom Android 6.0 OS comes the placement and availability of the desired applications directly in the notification tray, with no re-arrangement required. Additionally, our implementation allows for application access even when the device is locked, whereas the home screen is only accessible after the device has been unlocked.

Integrating custom tiles into the already established environment of the notification tray allowed us to eliminate a lot of the overhead associated with adding our own tray-like system to the OS. Avoiding unnecessary overhead is evidently a way to maintain overall performance of the mobile devices. Based on the nature of the project, implementing our own system tray that was constantly running in the background would hinder the performance of the system. During testing purposes, we were forced into using the default virtual machine created during the compilation of our custom OS. Unfortunately, this emulator did not provide an exceptional benchmark for responsiveness and performance during a stock run of the system. This meant that the overall performance of our project was limited to the performance of the device. Given this

restriction however, it was observed through many tests that the overall speed and responsiveness of the system operation was not effected. On average, as seen in Figure 6 (below), between 10 trials each of booting the stock and modified OS's, the stock OS boots in 183.1 seconds, while our modified OS boots in 183.8 seconds. A slight difference of 0.7 seconds can be considered negligible, since it only counts for 0.3% of an average stock OS boot. These results serve as confirmation that the integration of custom tiles has been successfully implemented. During the accessibility survey, subjects were additionally asked to compare the responsiveness of the notification tray upon activation. During the accessibility questionnaire, users were asked to freely navigate the device as if it was their own. Throughout the course of this section, they were asked to experiment with activating the notification tray, and using the accessibility tiles. When prompted, an outstanding 96% (24/25) of responses denied any noticeable difference between the responsiveness of the stock and modified notification trays, when tested side by side. Results like these are considered to be legitimate, due to the hypothesized difference between compilation and run-time process of the notification tray and the tiles included.

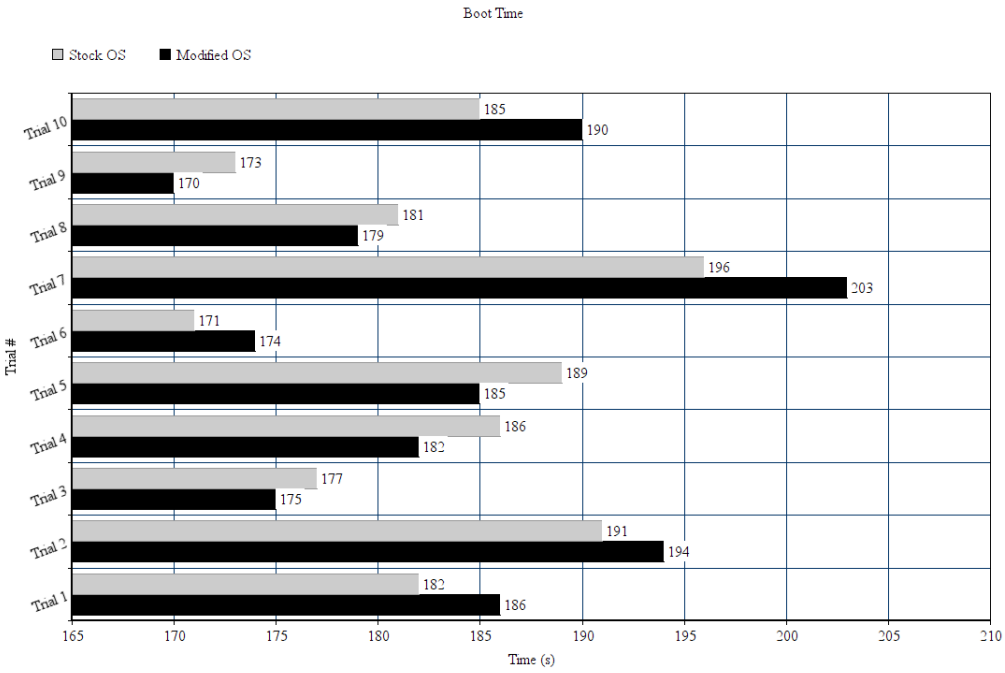

Figure 6: Boot time comparison of the OS.

Figure 6 compares the various boot time's recorded when booting the stock OS (red) and our modified OS (purple).

After making sure every aspect of the software was in working condition, and lied within our performance criteria, evaluating the interface was our next step. Since our custom tiles will be included on every installed device, the most important visual factor was to design the custom

tiles so that they appear to be natural to the system. It is critical that there be consistency between our custom tiles and the applications themselves, in order for users to easily recognize the relation between them. Consequently, we used icons included in the system, in order maintain the desired uniformity. The task of adding icons to our custom accessibility tiles became a side project in itself. Due to the structure of the system, adding custom icons was not a simple drag-and-drop process into the Android OS file system. The process of loading in an icon by default into the Android OS is an intricate web of Java, Android manifest, Makefile, and XML resources files, all reliant on each other. We first started by mimicking the native Android icons, converting them into SVG files then converting those files into vectors. After this, we learned that quick setting tile icons are animations, in which we created an animation XML file to hold an additional XML file which contained each specific icon vector and their animation attributes set to static. This finally achieved our side project of adding icons to our custom quick setting tiles. Unfortunately, since our icons needed by our tiles are not designed to be a part of the notification tray, the scaling of these icons does not match the default tile scaling, and can be observed in Figure 2. Although icon scaling does not create a functional difference, the look of the quick settings tray does not appear to be as native as we wanted it to. For mobile platforms, including both the OS and third party applications, material design [8], introduced by Google, has evolved to be just as necessary as the software itself. Material design emphasizes “a unified user experience across platforms and devices” [9], and due to our icon limitations, a small part of that philosophy has been broken. Putting the size aside, the style and colour of these icons integrate well with the system as a whole, to help unify our accessibility tiles within.

5 Conclusion

5.1 Summary

In all, the goals that motivated this project have been achieved. We have introduced a new feature within the Android OS, which is specifically targeted for baby boomers. Our feature is implemented into the native quick settings tray, and feels natural to the system. In order to support our project, we wanted to extend our reach to as many devices as possible. With so many options out on the market today, we chose Android 6.0 as a base operating system, and have achieved a reach of 26.3% of all Android devices, which have held on average 83.73% of the smartphone market share between 2015 Q3 and 2016 Q2 [9]. With a simple double swipe gesture from the top of the screen, users have the ability to open their most important applications included in the smartphone, that are essential to the cell phone. Our custom quick settings tiles have enhanced the usability of the Android device as a whole, as seen in our usability study. More specifically, as they are accessible from anywhere in the device, including the lock screen, these custom accessibility tiles allow our target market, baby boomers, to have instant access to the applications they want based off of their previous generation cell phone. Backed by our accessibility survey of friends and family falling within the baby boomer category, we believe

that with our distribution of the Android 6.0 operating system, our target market will easily be able to intuitively navigate their new Android smartphones directly out-of-the-box.

5.2 Relevance

In terms of relating this project to the course content, majority of the our work focuses within the domain of the OS structure and system calls. Specifically, we used the software aspect of the device in order to modify how the user mode interacts with the kernel mode. Naturally, in order to understand the way that the Android system works, it was necessary to filter through the open-sourced code. We learned about the software modules that interact with the kernel in order to perform actions within the user mode. Additionally, due to the fact that Android is a linux kernel, our project work pertains to the kernel domain of the course. Particularly, the custom accessibility tiles modify the way that a touch event that occurs within the notification tray is processed. Overriding the handleClick() function required by a tile allowed us to perform custom actions and modify the device state, by directly opening any of the four fundamental applications.

5.3 Future Work

Although we have reached our goals for this project, there is definitely room for further improvement. Primarily, the size of each icon for the custom quick settings tiles needing to be enlarged in order to blend in with the styles of the system tiles. Additionally, adding tiles for all other system and possibly third party applications would allow our project to target a wider audience. Instead of having the only core functional applications of a smartphone, users would be able to add their preferred apps to the quick settings tray in order to have a seamless and efficient experience with their device. Lastly, even though our current build covers a wide range of Android device that are currently on the market (26.3%), once perfected, applying the same design structure and motivations to other devices on the market running alternative major Android OS versions, we can extend our reach and bring mobile accessibility to more people.

Contributions

Evan Cooper

Responsible for writing the result, evaluation and conclusion sections of the project report, along with the final editing of the report, to ensure uniformity of styles. Brainstormed and created the project proposal slideshow. Contributed to building and establishing the android case-sensitive development environment. Navigated through the Android OS files to understand the

structure of the system. Created each individual custom tile class that was added to the file system.

Cole Dorma

Responsible for the introduction and background information sections for the project report, while also brainstorming on the contents of the conclusion, along with the final editing of the report. Brainstormed and created the project proposal slideshow. Contributed to building and establishing the android case-sensitive development environment. Navigated through the Android OS files to understand the structure of the system. Overrode the handleClick() callback function in order to launch applications and hide the quick setting and notification trays. Implemented custom icons into each custom tile.

References

[1] Richard Goodwin, “The History of Mobile Phones From 1973 To 2008: The Handsets That Made It ALL Happen.” www.knowyourmobile.com. Accessed: December 2, 2016.

[2] Nick T., “Did you know what was the first smartphone ever?” www.phonearena.com. Accessed: December 2, 2016.

[3] Omar Hamwi, “ANDROID 7.0 NOUGAT: RELEASE DATE, NEWS AND FEATURES.” www.androidpit.com. Accessed: December 2, 2016.

[4] Dallas Thomas, “Add Your Own Quick Settings Tiles in Android Nougat.” android.wonderhowto.com. Accessed: December 2, 2016.

[5] Ryan Whitwam, “[Hands On] Custom Quick Settings Adds New Tiles To The Android 6.0 Quick Settings Via UI Tuner.” www.androidpolice.com. Accessed December 3, 2016.

[6] Android Developers, “Context” https://developer.android.com. Accessed December 4, 2016.

[7] Joe Donovan, “A Beginner’s Guide to Android” http://www.digitaltrends.com. Accessed December 2, 2016.

[8] Google, Inc., “Material Design” https://material.google.com/. Accessed December 3, 2016

[9] indianappdevelopers, “5 Important UI and UX Design Trends for WordPress Mobile Apps” https://www.devsaran.com. Accessed December 2, 2016.

[10] International Data Corporation, “Smartphone OS Market Share, 2016 Q2”, https://www.idc.com. Accessed December 2, 2016.

[11] Google, Inc., “Platform Versions” https://developer.android.com, Accessed December 8, 2016.