Maksim Afichuk

C196 Mobile Development

5/20/24

To maximize the application’s efficiency on a tablet, I would increase the number of fragments to take advantage of the larger screen size. The number of fragments used for the mobile application is limited due to the smaller screen size; increasing fragments would clutter the screen space hindering on the user’s experience. In addition, as opposed to the hamburger menu implemented for the mobile application, I would consider using a fixed navigation fragment that would be displayed throughout the life cycle of the application. In consideration of layouts, I would give the user customization options in the settings to increase/decrease font and element sizes to accommodate the variable screen sizes of tablets. Although I implemented landscape variations on the mobile application, I would further optimize landscape orientation by changing the layout from vertical strip style to a horizontal layout for the tablet.

The minimum operating system is 26 Sdk and the target operating system is 31 Sdk. This application will run on Android 8.0 or higher.

One of the challenges I faced when starting this project was adjusting to the android syntax. My knowledge on recycler views was nonexistent to extremely basic prior to this mobile application class. There were several moving parts to get a recycler view to become functional such as an adapter, a view-holder, and an interface, and I was not confident with these aspects. As with any new syntax, there is a learning curve, and this project was my first introduction to mobile development. While the base is Java, which I am familiar with, the Android libraries were new which required me to study the documentation.

Another problem I came across was working with a limited screen size and figuring out responsivity. As an analogy, my mind would want to draw a line or a shape in a certain spot on a piece of paper. However, I quickly came to realize that Android Studios did not work the same way. For example, when I would try to place a button centered to the screen, it would not end up in the location I had anticipated when I would run the application. I had many issues with constraint layouts and elements getting clipped off screen.

It took time and patience to execute a functioning recycler view. I began by understanding what a recycler view was in its entirety and how it would benefit my application. The way I think about a recycler view is as a responsive list where I can make changes to individual aspects without the need of reusing code. If you want to change the layout, you will need to adjust the xml file and the whole list would copy your format. If you need to adjust the content of the list, adding an array through the adapter will do just that. I was able to achieve a functioning recycler view by applying my newfound knowledge of what a recycler view was, along with help from documentation and a few tutorials I found online to guide me through syntax and structure. After making my first view with it, I was able to make a second in a fraction of the time, overcoming my first major challenge in this project.

Before this class, my knowledge of building functional UIs was insufficient. While studying for this course, I found that many resources used constraint layouts to make their views. I started on this path too, but things quickly deteriorated as I tested my program on various screen sizes and in landscape mode. Elements generally were not where I wanted them to be and arranging them that I had envisioned was challenging. Two big solutions that I utilized to overcome these difficulties were changing my layout constraints to linear layouts where I focused less on the horizontal spacing of elements, and more on centering the elements and organizing them in rows. Each element had its own row where I could align the item to the center, wrap content for its height, and fill parent for the width. This combination would lead to an easy formula and consistent layout I could rely on to work more responsively to screen sizes and screen orientations. Lastly, with certain activities that contained a considerably larger amount of content, instead of opting to shrink the content, I chose to add a scroll view for the user to maintain the benefit of full-sized elements. This allowed the user to simply swipe up on the screen to scroll to the part of an activity not actively visible.

If I did the project again, I would strive to maintain consistency throughout the project as well as emphasize on designing a more UX inspired UI. This was my first android application, so a fair amount of experimentation took place. Some layouts, for example, are left in constraint view and some were migrated to a linear view. When doing this project again, I would make all my layouts linear. I may even opt for relative sizing options for text and other elements to create views that would respond and fill in screen size more efficiently. Regarding UX/UI, swipe gesture functionality would be an important addition I would add as most applications I use on a day to day utilize swipe gestures and feel natural to many users in 2024, helping the flow of the application. Lastly, to increase user satisfaction I would focus more on the details such as button sizing and shape. I would also create on-click effect to notify the user with feedback that confirms their input was registered. While these features were not the focus on this project, as I was learning to navigate Android Studios, these changes would be the next logical steps to take my application to the next level if I were to do it again.

Emulators are programs that imitate a device, like a clone. With android emulators, you can experience just about every android variant’s screen size and operating system to see how it interacts with your software. A pro to emulators is their availability and accessibility. There are countless free-to-use emulators online, making them an essential for android application development. Emulators allow application developers to quickly test their software without needing a development device on hand. A downside to emulators is that they can be rather buggy and slow since they utilize a virtual machine to work. Another con is that they also lack certain features such as messaging/calling or connecting to USB which may be necessary depending on what the user’s software needs are. Meanwhile, with Android Studios, there are options to connect our own android device and test our application through this development device. Using this has its pros, such as faster response times and a more accurate representation of our software. It may showcase a real user environment since emulators do not normally come installed with an average user’s app library. Therefore, the testing done in this environment more closely resembles real-life usage. As with emulators, this method also its drawbacks since these devices can be costly. Although it may be more realistic and convenient, it would not hit the variations or quantity of operating systems that an emulator can. Using both emulators and development devices in conjunction is what I believe to be the best way to approach mobile application development.