Reflective Essay

Word Count: 987

Skills Learned

Overall, throughout this project I have learnt a handful of skills and tools in relating to app development. As I had always been meaning to develop my own application, this project was a big motivator in my decision to learn new languages and tools. The main skill that I learnt is developing with Kotlin, which I may have avoided if I was not invested into this project. However, as my aim was to learn new technologies, developing with Kotlin was a no-brainer. I have seen and read multiple articles online about the rise of Kotlin to replace Java for android development, so I always had an interest in it. This project allowed me to get familiar with the syntax of Kotlin as well as it's strengths and reasons why it is on track to replace Java in this domain. After developing my application, I can say that Kotlin really is a very easy to use language that is not only light but powerful, I did not regret choosing it over Java at any point. Not only did I feel like I was able to develop faster by typing quick and short code, I felt like my code was also more readable and understandable at a glance perspective. This would be more complicated with Java as the code tends to run long with boilerplate.

Another key skill that I learnt was Firebase, as part of the Google ecosystem, Firebase allowed for very easy and seamless integration with Android Studio. Setting up was just a couple clicks and the tools section even included example code for basic functions such as authenticating the user. I found that Firebase is another great lightweight utility for app development and useful when the data structures required to implement the app is not overly complex. Although, I did have to get used to thinking through a NoSQL methodology, which I have never done before. This took me some time to get used to and multiple reworks of my back-end structure in order to understand how to lay out my data. However, once I understand the concepts, developing additional functionality onto the application felt very quick since no back-end set up is required. I think that the combination of Kotlin and Firebase allows for very swift development as their nature both lies in efficient development rather than robust functionality. This makes the tools that I selected perfect for small-scale applications like my project.

Different Approaches

Different approaches that I would try next time would be to develop my Cloud Firestore with proper asynchronous function handling. Since I did not want to figure out how to handle waiting for server requests, I simply loaded all database data on a continuous thread. Although this does not lead to any errors in processing data due to delays, it leads users to wait load durations on every server upload request. This is also partly due to my lack on implementation of a loading screen, which could have aided the user in identifying that background process is still in progress.

Another possible different approach to this issue would be to load the database data once and cache it for the whole session. This will greatly reduce the number of times the user will need to wait; however, it means that the server uploads may be dealing with old data. This approach may not be as feasible due to the system being collaborative in nature, requiring real time data updates. However, it could be remedied by allowing uses to force refresh the cached data by a swipe down gesture on any each page. This solution may improve time performance, but the risks the user being unaware of real-time updates.

Effective Applications

An effective application that allowed my project to succeed would be the usage of GitHub for version control. Version control platform such as GitHub combines functionality with a social

media aspect. I find that this promotes productivity as I am constantly motivated to push a commit daily to get a green square of my past activity. In addition to the social aspect, the constant version control usage allows me to be fearless of experimenting with unknown code. This especially useful when learning Kotlin as I can test how a completely different function or syntax of code impacts my app without having to undo step by step. Overall, I found that GitHub is a great motivator in both a social and developing aspect that really contributed to my project.

Ineffective Applications

Throughout the project, all applications have been effective in some way, however, in terms of hinderances to development, XML would be most notable. Maybe because I am not too comfortable with XML, but I found that styling my layouts in Android Studio took up most of my time. The difficulty in translating the user interface in my designs into a markup language is far too tedious. Such a process should already be simplified to a drag and drop methodology. However, the drag and drop mode for XML in Android Studio is ineffective and often useless, other than just to view a library of the default widgets and layouts. Most of the time I use the drag and drop functionality the object I am moving always falls into the child of whatever group is above it. In addition, the properties panel on the right fails to suggest the common attributes that I want to set. In the end, most of the xml is done through copy and pasting and trial and error. Which I feel is very counterintuitive to how coding should be done. Perhaps due to my inexperience, but if I'm unable to tell what the code does without running it then my development would be greatly hindered. Which is the case with XML as developing without a preview window is practically impossible for me, even as I gain more experience in it.