Mobile application development has become very popular in recent years and thus has led to a whole new era of software development. With mini computers within our hands the possibilities of application development seems endless. However, developing a mobile application isn’t as easy as it seems. There are hardware limitations such as space, processing power, and RAM to name a few. This may cause some applications to run much better on higher-end devices, while other devices cannot run or struggle to run some applications.

I have built my application to target Android 9 (API (Application Programming Interface) level 28), also known as Android Pie. However, my application will support a minimum of Android 7.0 (API level 24), also known as Android Nougat. With that note, building to a target SDK (Software Development Kit) version is extremely important. With new devices, OS (Operating System) upgrades, and other constantly evolving improvements the target SDK your application is built to support can drastically effect the number of devices your application will be able to run on. We have to remember that most people with devices only upgrade every few years. Depending upon the SDK versions that you want to build your application to target this may affect the features within the API available to you. For example, devices that use SDK Android 4.4 (API level 19), also known as KitKat, are able to run applications built using Java 7. Any device that runs on Android 4.3 (API level 18), also known as Jelly Bean, is only able to utilize Java 6 features and therefore may not be able to run a large number of applications available today, including mine!

During the development of my mobile application I did face challenges. I struggled with getting my ListView components to operate in the matter that I wanted them to work. The implementation of the SQLite Database was also a challenge because I had only ever implemented this for non-mobile applications. There were other challenges but I will focus on these two.

I relied heavily upon Google and StackOverflow to overcome my challenges. I learned that ListView components had to have a specific ID (android:list) when the ListActivity was the super class for your activity. I also learned how to implement a CustomAdapter with the help of some examples I found on StackOverflow. This allowed me to build custom adapters for each of my objects so that I could make the ListView items appear and look how I wanted them to look.

The implementation of the SQLite Database taught me about the assets folder. I was adamant upon storing my database creation script within a .sql file rather than hardcoding it within my application. I created a custom DatabaseManager class to house my database functionality. This allowed me to pass my application context so that I could access the assets folder where my .sql file for the database creation was stored. After getting this far I started to feel comfortable with the database implementation and went on to implement it in a fairly clean way using multiple objects to manage Terms, Mentors, Courses, and Assessments.

If I had to do the project again I would’ve spent more time learning about the components I would have been using to build my application. I feel that in doing that I would have probably spent a lot less time figuring things out and could have focused more of that energy on getting the application built. I would also have spent additional time researching the best way to write object oriented code when building an android application. I wanted to make my code as clean as possible, but the way things are passed between objects when running an android application is different. If I had spent more time researching how to do this efficiently I would have probably written a lot less code.

As far as using emulators and actual devices are concerned I would highly recommend using an actual device. Emulators are great for allowing you to see how your application will look visually across a wide array of devices, but I did notice that scrolling and other things don’t seem to work as well in an emulator. I particularly noticed issues with my ListView components when using the emulator because it seemed to jump around when scrolling. When using my actual device I did not experience that issue. In addition to more stability with my actual device, I was also more comfortable testing on my device because I use it every day. The only cons I experienced with my actual device were my battery life, and only being able to see my application on one device.