1. Mobile application development is the set of processes and procedures embroiled in writing software for small, wireless computing devices, such as cellphones, tablets, and wearable devices. Mobile application development is dependent on the architecture of the hardware and software the application is running on. Mobile application architecture is a set of techniques and patterns used to develop fully structured mobile applications based on industry and vendor specific standards. Mobile applications are written specifically to take advantage of the unique features of a specified mobile device. Mobile devices have specialized capabilities that are not traditionally found on laptop or desktops, like accelerometers used for gaming applications. The design of the application have to take into account hardware limitations such as storage, memory, battery life, operating system used, screen size, and CPU. When developing for mobile applications, the application's performance must be optimized by minimizing power consumption and memory allocation. Typically, mobile application architecture elements include presentation, which is considered the user experience, business, which contains the components and entities, and the data, which contains the data access and utilities. These layers bring a consistent flow to the architecture and makes a little easier for the developer. Keeping this in mind, the four main things to keep in mind when building a mobile application architecture is the following:

1. The device types' screen size, resolution, storage space and memory, CPU, and development tools environment.
2. When connectivity is either intermittent or plain not available, designing for caching and data access mechanisms.
3. The user interface can make or break the application. Simple, intuitive interface is all you need. Do not over complicate it. (KISS)
4. For the navigation methods, think of how the users are going to use the application and the application requirements.

The application is developed for Android devices with a target API level 28 and minimum API level

2. Thankfully, information regarding developing Android applications within Android Studio is detailed, current, and contained many examples. Before I started writing the application, I could not find a place to start. I wanted to use my time efficiently and use the best practices for this project's scenario. This involves creating the most effective architecture for the project, not repeating calls and handlers when it can be accomplished in a more simple manner. I did not know if I wanted to use the Uri references or fragments utilizing DAOs.

At the time of creating the layouts of the application, I noticed that there was no simple date picker functionality framework like with JavaFX and SceneBuilder, where implementing it was extremely easy. I wanted to use the simplicity of the date picker to be used when the user is selecting the dates for the term, course, and assessments rather than the user manually entering the date, which can lead to errors and frustration. I then thought about when I implement a date picker and the user selects the date picker that the calendar displaying and the day selected by default should be the current month,

date, and year. I also had a hard time getting the alerts to work. I could not get the notification to appear on the phone's notification bar nor to display elsewhere.

3. I was able to overcome the all too common challenge of not knowing where to start writing the application with designing a logical/brainstorm map of how I wanted the application to flow, how the user is to interact with the application, and the application requirements. From there, I kept in mind the three elements of mobile application architecture: presentation, business, and data. This aided tremendously in my development plan and, once I hand my plan, I was able to write the application quickly. Development plans are friends not futile endeavors, like attempting to find a black cat in a dark room, especially if there is no cat. After searching the web for documentation and inquiring my course mentor about the methods I was considering, I wrote the application using both methods. Ultimately, I chose to use fragments utilizing DAOs because I was able to structure the project folders and files more pleasingly organized than Uri at my current (at the time of developing the application) understanding of Uri references and fragments with DAOs.
Through querying documentation, many trials, and errors, I was able to use the DatePickerDialog class available in Java android development. This method was just as simple as the JavaFX functionality just without the UI layout selector option. While writing these reflection and through testing and documentation, I know know I can include JavaFX within Android Studio by creating an empty android project then add another Java library module. When I was adding the DataPickerDialog class functionality, I thought to use the Calendar class to get the date instance as I did for another project (C195). This worked. Now the alerts troubleshooting was a little different but a common occurrence. I reviewed my alert code from the AlarmReceiver class to all files that the alerts are touched in the application. I searched documentations and stackoverflow to no avail. I reviewed activities that set the alerts again. This time I found it. The problem was with SimpleDateFormat pattern. I had it set to "MM/dd/yyyy" to set the alerts but I was saving the dates in the database as "MM-dd-yyyy".

4. If I were to do the project again (which is highly likely because of how I am), I would (will) add violation messages to display on the screens. This will improve user functionality and user experience of the application. Simply, when the user enters invalid information, they will see a violation message indicating that it was invalid instead of not being notified that it was invalid and thinking the application is broken. I would also add the functionality for the spinners used when editing the assessments and courses to default to the selected values when they were saved previously until changed by user. I would also change the theme of the date picker dialog to more closely match the theme of the rest of the application. Though the default theme is close to the theme of the application, it still is not consistent with the application as I would have liked. Most importantly, I would name the project root folder correctly.

5. Emulators are software programs that imitates as close as possible the features of other computers and mobile operating systems for testing by installing them to the users computer/mobile. They are more suitable when it comes to debugging purposes and testing the mobile's internal behavior. A development device is better for testing for longer periods of time and battery issues than emulators. Emulators are far more cost effective, generally speaking, than development devices. It is easier for unit testing to use an emulator rather than a development device. Not all emulators support complete gamut of mobile applications like a development device will. With an emulator, it is far easier to capture screenshots then quickly add them to documentation than a development device. There is a difference between a click and drag using a mouse than swiping and "clicking" with a finger. Emulators are slower than the development device due to the fact that the emulator has to run the simulation of the operating system and run the application.