For each project we were tasked with different testing methods. For the first project with the contact service and contact we did not use Junit testing. For this project I ended up creating a new file called invalid check to display an error if the tests did not pass in the main class. The test files for this project also had hard coded values that is used to test different scenarios. Though I wouldn’t entirely recommend doing testing this way, there wasn’t a whole lot needed to be tested so it should be fine. For the other two projects we were tasked with using Junit tests. On top of this we were to achieve a certain threshold of coverage percentage with the last project being 80%. Both of my projects, Appointment Service and Task Service fulfill this criterion and I would argue in saying that my overall quality for my Junit tests are effective. One thing I really liked about my code is in project one for Contact. On line 10 in my Contact Class, I added an extra snippet to call my invalid check so that it can throw an error if my tests do not work. Since we weren’t using Junit testing in this part of the project. I still wanted to get an error to display if an invalid input was being thrown. I think this was good example of technical sound code from the project. I would bring up my tests in Appointment Service and Task Service, however those were just following Junit tests. One thing I also tried to do is make sure every project was created efficiently. No unneeded lines of code and testing every scenario but not over testing to make it cluttered. I also really like making my code easily readable with spacing, too much jumbled code on one line or even together makes it seem like you’re reading a novel. It also makes it a lot harder to attempt to debug and may stress another developer out if they need to read my code. A good example of this could be my nested if else statements in my contact class lines 12 through 52. Spacing everything out on each line makes the code easier to read, you know where the brackets are, and you can clearly see what the code is doing. Though it may take more lines, you can efficiently see what the code is doing and solve an issue faster than if it was all on one line.

I’m going to first discuss the testing techniques that we didn’t really use in this project. I would have to say that we didn’t really employ any static testing methods. According to our book, these are tests that are done without the execution of code. This is usually things like reviewing a drafted document of the ticket or even code reviews. I feel like we didn’t really do this, simply because we had our guided prompt to follow. This takes away from reviewing the ticket process since it has already assumed to be done for us. We also did not do peer code reviews, though it can also be argued that our teacher is our final code reviewer. Another technique we didn’t really use is the white box technique. This test the overall structure of the code being submitted into the code base. We used Junit tests to create test cases for certain scenarios and to verify we were hitting a certain coverage percentage of the code base. I would say that the testing process we used the most would be the black box technique; this is because this type of testing is about test cases. Even when I didn’t use a Junit test like in the first project Contact. I still implemented several test cases in my code that could be tested, line by line to verify input values were being received correctly. I would also go to argue and say that even though we did only use black box techniques, it was the only technique necessary for the given project. We built the application for the company, so it’s not like it was submitted into a huge code base that needed checking for structural errors. We also knew exactly what the task was needed, so no meetings or reviews needed to really be done which crosses us off static testing as well. The only main thing was for us to verify bad inputs into each of our section and be sure that a user could not insert bad data into the code base. As this itself would cause errors, or for the client to crash.

As a developer already, one of the things I sometimes struggle with developing is putting myself in the shoes of the end user. Or thinking of ways that my code can be broken with different options being set or input values. A lot of time I get into the habit of just making sure the code works without putting a whole lot of time to seeing if it breaks. These projects were quite eye opening for me to see every little aspect QA must test and the amount of test cases that go into even just a tiny bit of code. While I like to believe everything I code is completely efficient, and I do some testing I know it’s nothing near what QA does. As a current developer I don’t need to imagine the bias I have for my code since I live it. Unfortunately, most of it is time and amount of work. When spending hours reading the same lines of code and getting something to work you get to a point you just don’t want to look at it anymore. You may end up cutting corners and not even think about other ways that could break the ticket you’ve been working on. However, it’s best to just take a step back take a breather and handle the ticket you’ve been task with as much as possible. When you cut corners, you can create technical debt. This is bad, since it not only wastes QA’s time, but your time as well since you will have to go back and fix your mistakes. When it could have been submitted and completely done the first time around without all the extra hassle. What I like to do when I’m getting fed up with a ticket, is I’ll set it aside and start working on something else till I’ve calmed down. In most cases this gives me the breathing room I need to go back to that ticket and come back with a level head. This also refreshes my eyes to the code I’ve been looking at for hours and I may see mistakes or other test cases I need to account for. In doing so this keeps my disciplined and I know that the code I’m outputting will not create any debt. However, I do know that there is still a lot I need to account for prior to development instead of jumping right into it. Especially with the software I’m currently developing, each state and county has different system options that must be evaluated and tested before the code is checked in. I believe this course has given me an even better level head in taking a step back and doing better testing on my end as a developer to not only help QA out, but my self in the end.