**Summary:**

I was given three different assignments over different features Grand Strand Systems wanted. The first feature was to make appointments. The appointment system was put into place to schedule and make appointments given certain criteria. My process for this assignment was to go through the criteria and code that in first. Once I was sure I coded it right, I debugged it to make sure the code ran without issues. After this, I had to go through the process of adding Junit tests to test to see if there are any exceptions of errors that would throw out. Once I was sure there was none for this, I compiled it and submitted it for approval.

The next assignment I had to make was task service. This was created to make a task (example: do chores). This was given to be with a few restrictions such as name characteristic limit. In this code I put an illegal argument exception in it to catch what doesn’t match the current criteria. We had to make the task could be updated, deleted, and created. When working on this code, I made sure to go through the process of checking to create a task, update a task, and deleting a task. The Junit tests were created to make sure this worked as intended without issues appearing.

The final assignment I was given was a contact service. The purpose of this was to store and find contacts of people that include their ID, name, phone number, and address. This needed to be updated, created, and deleted as needed. This assignment was given quite a few restrictions as I needed to make sure no results were void and matched a certain characteristic limit. I followed the same process as I did with the other two assignments on this and then created Junit tests to check and make sure that each instance was being handled as intended.

I made sure my code was technically sound by debugging it and verifying everything was up to date, as well as using the correct arguments and commands. I also ensured it was working properly through the debugging process, as well as using Junit testing. Junit testing was used to check and see if the exceptions would be accepted or not, as well as what type of limits would strain the code.

**Reflection:**

The main software testing technique I used would have to be Junit testing. This allowed me to input my own type of tests that I would believe violate the restrictions, and test to see if the code actually did this or not. If the code ended up accepting the violations, I’d have to go back and modify the code again until it worked as intended. Junit testing is a special testing used to test to see if illegal arguments are accepted are not. In simple terms, Junit testing is testing the software to check and see if it meets all the requirements you’ve put into the code. If something seems to not work with the code, or there’s an error, the Junit test will show you which part failed, and what’s the issue with that part.

The other software testing technique I used would be debugging. Debugging is one of the best ways to test software in the first place, and it’s necessary in order to make sure the code works as you would want it to. It’s important to always debug the code as you go through making changes, creating new code, and deleting code. If something goes wrong, debugging will typically be the first place to look to see why it’s not working. I ran into a few issues with the code when I first wrote it, but the debugging process allowed for me to check and see that I forgot to add the Junit library to the project I was working on. Overall, testing the software using debugging has saved me some time with testing issues and seeing what’ll break if it’s ran.

When it comes to being a software tester, I like to take as much cation as possible. I like to be prepared in the event of something messing up or failing. It’s important to appreciate the complexity of the code I’m testing as I have to understand the process of the code and learn what interacts with what else. Understanding this can allow for me to test the software more in dept and see if there’s any underlying issues I could be missing out on if I didn’t understand it. When it comes to bias in code reviewing, I steer clear of it. I like to view code in one of two ways to start: does it work, yes or no? If it does that’s great, and we move on to see if there’s redundant code or excess code that unnecessary and could be shortened. Coding is a valuable skill and not everything is perfectly spaced, explained, or written. I’ve learned from personal experience that if you try to code like this, it leads to issues and errors much faster than anything else. With coding, cutting corners can end up to errors. These errors can lead to leaks, crashes, or even breaking the software permanently. An example I can give to the last two mentioned claims about bias and code cutting is when I worked on a google form for automation. If you copy paste code, sometimes it doesn’t space out at all, and sometimes you’ll find you have an extra “}” or something similar in the code somewhere that doesn’t seem to exist. In addition, using the same code twice can lead to issues with reading and updating the automation process. What typically ends up happening is when the form first imports the data to the code, it saves that specific piece, but if you make it import the data again, it’ll pull the newest code unless it’s specified to a specific line in the google sheets (as to not go into a rant, I will end this here). Overall, it’s best to do your due-diligence and make sure to take your time when working with code and not be bias when you’re working with messy code.