Project 2

Summary

* Describe your unit testing approach for each of the three features.
  + I took the same approach but further refined it as I continued through the code of each section that we created, the way that I tested my code was as I created a section of code I would go to the JUnit and test each section. The reason that this was great to do was that it helped me ensure that the section of code that I created was working correctly before moving onto the next section of code so that I was able to ensure my code quality but also by doing this I helped to ensure that I remembered what that section of code did instead of just creating all the code and potentially forgetting what the code did for that section.
* To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.
  + I made sure to follow the software requirements that we were given and to ensure that the software I wrote worked as needed I would constantly test it to ensure it was working within the software requirements. The specific examples I would use would be in the coding sections for the contact service coding it asked for a firstName string that was less than 10 and not null, in the task service it asked for a description string that cannot be longer than 50 characters and not null, and finally in the appointment service it asked for description field that was not longer than 50 characters and not null. To ensure that this was happening when I created these, I would ensure by creating JUnit tests and testing for the length of the string and if they were null and if they were it would output an error message. I then also tested to make sure it was working with the correct parameters that we were given.
* Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were effective based on the coverage percentage?
  + By using the technique above I was able to ensure that my testing was effective and had the quality we needed since I would test as I went with the software. I would also test for failures and for when the software was supposed to work which helped ensure that everything was working as intended. By ensuring that I tested for the software was supposed to work and when it was supposed to fail, I ensured the quality of my code since by it failing in those parameters would mean I would have to go back and re-look at the code which is the purpose for testing software.
  + Text

    Description automatically generated
  + Text

    Description automatically generated

Reflection

* What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.
  + The only software testing techniques that I used in this project was JUnit tests, and JUnit tests is a way of testing the input of code as well as checking any errors that should happen when something messes up inside of the code. As I stated earlier, I tested each section of code as I wrote it to ensure quality with my JUnit tests and by testing for pass cases and fail cases I was able to ensure that my software was adequately tested and effective.
* What are the other software testing techniques that you did not use for this project? Describe their characteristics using specific details.
  + I did not use any other software testing techniques besides JUnit tests due to the fact that this was a smaller project and I only needed to test for inputs and error handling of the software, if this software project was much bigger and had more people working on it I would have needed to use other testing techniques but since it was a smaller more isolated project I only used JUnit tests to ensure my code was working as intended.
  + Honestly, I didn’t use a lot of different testing techniques besides JUnit tests but looking over our reading one that I didn’t touch or use any of the frameworks such as JBehave, or any other testing techniques like integration testing, benchmark testing. (My wording from journal 5-2)
* For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.
  + As I stated above our project was relatively small so we had no reason to use other testing techniques besides JUnit testing, if it was a much bigger project, we would have to had used frameworks like JBehave or testing techniques like integration testing for benchmark testing.
  + The reason that you would want to use these other techniques is since you must test for more and account for me when you work with a bigger team to ensure that everything is working as intended, for example I said integration testing is to check that integration is working with other people’s work. Since this is a smaller more contained project there is no need to do any other testing besides JUnit tests.
* Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ caution? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.
  + My mindset was to follow the clients specifications or in this case our rubric that had what we needed to do and what was wanted of us. To ensure that my code was working as intended together and correctly I would employ a more cautious testing method in which I would create a section of code and then I would immediately test it which helped ensure that I was ensuring quality but also making sure to appreciate the complexity and interrelationship these classes have with each other. We can see this through out my code but the first example I can think of was the appointment service with adding appointments in which I created the code to add the appointment   
    Text

    Description automatically generated  
    and then went to go test it right away to ensure it was working correctly  
    A screenshot of a computer

    Description automatically generated  
    by doing this I ensured that everything was working all together before moving onto my next section of code.
* Assess the ways you tried to limit bias in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.
  + I had a few biases in how I wanted to create the code thinking that they were the most optimal way when we first started in how I was creating the task java class in which I played around with a few ways to get the program working but as I continued to work I found out that I was making the software more complex that it needed to be. This can lead to things such as feature creep in which if I stayed with that course of thought I would have continued till I almost ran out of time. By reassessing my biases, I was able to go back and remove these changes and create the software within the timeline were given.   
    Text

    Description automatically generatedText

    Description automatically generated
* Finally, evaluate the importance of being disciplined in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.
  + There are a few reasons as to why I always want to be a disciplined software developer, the first reason is that I want to take pride in my work in which if I leave a company I want people to come and look at my code and be able to first understand what it does but also that it was tested, the second reason is that I never want to cut corners without ensuring the safety of what I am creating since this can lead to either buggy products or allow someone to access the system that shouldn’t have access to it which in turn causes issues down the line in time and money to fix, and finally I want to be an ethical programmer that ensures if I am dealing with customer data that it is as secure as I can possibly make it so that the end user never needs to worry about any of their data.
  + As long as a rigorous testing methodology is in place you can usually avoid technical dept since you have to test the code you create which in turn helps to ensure that anything that gets pushed through is quality code that you won’t have to go back down the line and fix. Another thing that I do to avoid technical dept is to stick to a schedule which helps to break down big problems to smaller more presentable problems that you can work on and slowly move towards getting the software together and by doing this you are able to test those smaller sections of code to make sure that they are quality.
  + From what I have seen in the industry technical debt will always happen due to either code not being tested enough which then cause the developers to push back that section of code since they need to move onto the next section of code on the schedule.