Sarah Prince

June 20th, 2021

CS-320

1. **Summary**

**To what extent was your testing approach aligned to the software requirements? Support your claims with specific evidence.**

My testing approach was directly aligned to the software requirements. To the best of my ability, I tried to make sure all requirements and goals were accounted for. It is very important to include every requirement mentioned by the client to ensure you deliver a high-quality product that they can be satisfied with. For example, the requirement for Task Service stated, “The task object shall have a required name String field that cannot be longer than 20 characters. The name field shall not be null”. To test this and make sure the requirement was sound, I executed this:  
“public boolean validateName(String name) {  
  if (name != null && name.length() <= 20)  
 return true;  
return false;”

**Defend the overall quality of your JUnit tests for the contact service and task service. In other words, how do you know that your JUnit tests were effective on the basis of coverage percentage?**

The contact service and task service unit tests were an essential part of the overall project. These tests were responsible for running the code and to check for accuracy. After completing these tests, I was more confident in the project overall. “JUnit is an open-source framework, which is used for writing and running tests, providing annotations to identify test methods, and also provides assertions for testing expected results” (Williams, 2021). Coverage percentage measures executed code within the test. The higher the percentage of coverage, the more likely the code will be of higher quality.

**How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.**

While completing these portions of the project, I wanted to ensure I was using the correct commands, functions, etc. Utilizing the proper syntax was also very important. After I would complete the code, I would go back several times and review my lines checking for any mistakes. The annotations also help understand the functions and commands used while going back over the code. If the code is able to run and deliver what is needed in a legible way, I consider it technically sound. Below is an example where I tried to stay organized and use the correct verbiage to stay on track:

  “  //getters

    public String getId() {

        return id;

    }

    //setters

    public void setId(String id) {

        this.id = id;”

**How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.**

Making sure my code was efficient was just as important as making sure it was sound. The overall goal was to ensure it included all requirements from the client in a readable way. I constantly went back to review portions of the files and then validate them against the requirements listed. I wanted to make sure I did not include anything extra or that was not necessary. Sticking to the guidelines were key. Again, ensuring all variables are properly declared and utilizing correct functions at the proper time is very helpful to create efficient code. Using the Contact code as an example, I had to make sure everything was in the correct spot and matched up:  
“   // parameterized constructor  
   public Contact(String contactID, String firstName, String lastName, String numberAddress) {  
       this.contactID = contactID;  
       this.firstName = firstName;  
       this.lastName = lastName;  
       this.numberAddress = numberAddress;”

1. **Reflection**

**What were the software testing techniques that you employed for each of the milestones? Describe their characteristics using specific details.**

I used static and dynamic testing for each of the milestones I completed and then again in the final deliverable. When using static testing, I went through each line of code to ensure I had all components and requirements. I also wanted to make sure that not only were they included, but if they would compile correctly to give the client what they need. For functionality, dynamic testing and black box testing was used. Black box testing does not deal with the internal workings of the project and primarily deals with the ability to run accurately.

**What are the other software testing techniques that you did not use for the milestones? Describe their characteristics using specific details.**

White box testing, part of the integration testing, was a technique that I did not utilize for the milestones in the beginning. However, while working on my final submission, I went back to implement this technique. White box testing involves focusing on the internal structures of a code and verification. Integration testing involves taking individual parts of a code are taken and then run together and tested as a group. This makes sure that everything not only works individually, but overall as well. “White box testing technique is used by both the developers as well as testers and it helps them to understand which line of code is actually executed and which is not” (Leonard, 2021).

**For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.**

Static and dynamic testing are vital techniques needed to be implemented in any project. For larger projects, it may be more difficult to complete static testing due to possibly dealing with working through thousands of lines of code. If this is the case, it would be wise to delegate portions of the code onto multiple team members to work through. Using the static technique ensures that the internal components and code are integral to the overall project. Dynamic testing is just as essential since that deals with the end goal of the project, as opposed to the internal structure. These techniques, as well as the box testing techniques, are vital parts of the software development process. Testing keeps all projects organized and compliant with any requirements. Black box testing, for example, can check different versions of software to check for regression, or other trends.

1. **Mindset**

**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.**

I tried to have a professional mindset at all times, as this project was something that would be performed in the real world. The guidelines, characteristics, and deliverables were similar to something I would expect as a developer within an actual team. Caution was always employed because it was imperative that all portions of the project flow together and build off of each other accurately. Each code segment and its mirroring testing code were complex in many ways to ensure the overall project was efficient. An example of how these different portions of code need to be in sync is as follows:

“ Appointment appointment =

new Appointment("11111",calendar.getTime(),"Appointment One";

}

}”  
  
As seen above, this class tests the fields names in the previous code and check certain requirements.

**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.**

As someone reviewing their own code, it is so important to limit bias to ensure no mistakes are overlooked. Since I wrote this code, I tried to have the mindset of another team member when going over what I wrote. I can imagine someone overlooking several issues with their own code because they assume they got it right the first time. It was the following line of code that I found an error in after reviewing it for the third time: “ new Appointment("More than 10 character",calendar.getTime(),"description");”. I realized I had spelled “description” incorrectly due to typing too fast which caused things not to work.

**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.**

Being disciplined in the commitment to prove quality work is of utmost importance. The goal of a software engineering professional is to create a program that the client is 100% content with. With any career, submitting work that is accurate and executed carefully is expected and extremely appreciated. Taking the time to read over each requirement and line of code carefully will hopefully reflect the work submitted and inhibit the amount of mistakes.

**Citations:**

Leonard, R. (2021). White Box Testing.   
 https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/

Williams, T. (2021). JUnit Tutorial – Tutorialspoint.

https://www.tutorialspoint.com/junit/index.htm