Jonathan Vest

2/18/2023

CS 320 Software Test Automation and QA

**Project Two**

* 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. When I was creating the required classes and services, I worked directly from the software requirements provided by Grand Stand in the rubric. I worked directly with the requirements outlined in the rubric to stay within the parameters outlined, such as the length of variables etc.

* 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?  
  When creating my Junit tests, I worked my way through my classes and attempted to create a test for every constructor and every method in the class. My primary goal when creating my tests was to have complete coverage of my classes and objects. Once my test were built and I ran my JUnit tests, I then came back and ran as coverage for each service class and made sure I had the proper coverage. The requirements called for a minimum of 80% coverage on the service tests, but I always want to shoot for one hundred percent.



A screenshot of a computer

Description automatically generated with medium confidence









* How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.  
  I ensured that my code was technically sound was by debugging my code and ensuring that my variables held the expected values while I stepped through my code. For example, in my TaskService class, creating a unique ID that was no longer that 10 characters proved to be a bit difficult. The first thought was to use javas util.UUID to create a unique ID but this returns a 36 character value. I could have simply taken a substring of that UUID’s first ten characters but then there would be no guarantee that the id would remain unique. What I did was create a counter that would increment after the creation of each unique task ID. I then make a substring of the host address of the users’ machine and subtract the length of the counter from the desired task ID (10), then concatenate the counter to the end. This will ensure that the task ID is indeed unique but will keep the length at 10 characters. To ensure this was working as intended I set a break on the newUniqueTaskId method and stepped through it watching my variables as I did so.
* How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.  
  To ensure my code was efficient I directly assign constant variables with a value. Then I reused those variables throughout my code instead of hard coding those values. This will ensure that if a change were made to one of these values I would only have to change it in one location instead and having to try to remember or find everywhere that value was set. One example of this is the task ID length. If this value were to ever need modified to account for a longer ID I would only need to change it in the TASKID\_LEN variables assignment and it would then be changed for the entire program.
* What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.

I would describe the primary method of testing done for this project as white box testing. White box testing techniques are used when examining the lower level components of the application and when the tester needs an intimate understanding of the product being tested. For example, understanding classes, data structures, encapsulation and modularity are vital when creating Junit test cases for these classes and their methods. Without that knowledge and knowing how objects are created and stored internally you would be unable to develop a proper test or be able to apply the proper test coverage to the application.

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

Another testing technique that I do not feel applied to the project is black box testing. Black box testing occurs when the tester does not posses the intimate knowledge of the internal components to test them at this lower level. Black box testing would be considered a higher level testing where the tester applies tests such as input and output testing and verification where they would expect a certain output depending on an input but are not necessarily aware of, or need to be knowledgeable of the underlying functions being performed to receive that output.

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

White box testing is useful for software development projects and for developing applications such as what we are working on with these milestones. When developing Junit tests a lower level and more intimate understanding of object-oriented programming is necessary. When the tester needs to be knowledgeable of the internal components of a project and how they interact with one another and how they interact with the program as a whole. This knowledge is necessary to ensure we are developing the proper tests and that we are getting complete coverage of the application.

Black box testing would be more useful at a later stage in the development of the application. For instance, once we have a user interface in place, black box testing would be useful while performing tests on the inputs and examining for the proper output. Let’s take the module five milestone for example. A user could an interface to create a new appointment for a customer. They could then examine the appointment that was created to ensure the proper output of ID, Date, Name, Description etc. The tester would not need to be aware of the underlying components of the appointment, they would simply need to know if they receive the proper output for the input they entered.

* 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.

While working on this project as well as the milestones throughout this course I had in the front of my mine for every object, every class, every constructor, every method etc. that I would need to be able to create a test to demonstrate the functionality of what I was creating. I think this does cause a developer to employ caution to some extent as you don’t want to create anything overly abstract or eccentric that can not be tested. You want to be sure that what you are creating can be tested and included in your coverage evaluation. For example, when researching how to created a unique id but keep it limited to 10 characters, I came across a lot of ideas that I would have likely worked but would have been difficult to test and if I could not create a test for the method creating the unique ID It would take away from the coverage test.

* 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.

To limit bias I would try to switch my mindset from a development mindset to a tester mindset. While in development mode you want to believe that what you are creating, however wrong it is to do so, is flawless or unbreakable. We know this is an irrational belief but we are still human and deep down we for it to be true, we all want to be the next prodigy or “Neo” if you will. This can cause hesitation when testing code that you have developed as you don’t want to break your creation. A testers roll is to try to break that creation. To stress and test the code and find faults and defects. So, while I had in mind that the code I was writing would need to be tested while writing it, once it was time to test I tried to switch into test mode and consider the code from an unbiased viewpoint where the only thing that matter was testing the code in front of me.

* 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.

As software engineering professionals we should always strive to deliver code that is of the upmost quality. This means ensuring that what we are putting out into the world is as solid as possible from all viewpoints no matter what the project. If we are developing or testing software for a coffee pot or for a multimillion-dollar aircraft, we should approach the development and testing processes the same way. At the very least we want to be proud of the product we are developing, knowing that we are putting out a product that is as solid and as safe as possible. We want our code to be solid, meaning that it is going perform its function and not break once deployed. We want our code to be safe, meaning its protected from attack, and also that its going to perform when it is needed to. This is as important in the family coffee pot as it is in the aircraft. No matter what the product we never want to cut corners as doing so could cause subpar code to be implemented and this could be disastrous and cause the coffee pot not to perk or the aircraft to crash.

As a software engineering professional I will commit to developing and testing code in a way that will ensure it is as solid and as safe as possible. I will strive to create code that can be understood and tested and strive to be as thorough as possible and provide complete coverage of my test subjects.