Project 2:

Summary and Reflections Report

Matt Smith

6/16/2023

In this report, I will discuss the various software testing techniques utilized during the development of the contact service, task service, and appointment service milestones. I will talk about the specific testing approaches employed and explore more techniques that were not utilized. I will also examine the practical applications and implications of each technique in diverse software development projects and scenarios.

My testing approach aligned with the software requirements because I used unit testing to test functionality of my software. Unit testing is a software testing technique that allows me individually evaluate functions, methods, and classes in my software. I used the JUnit framework because it supplies a structure from writing and executing unit tests. The tests I wrote covered the basic functions of adding, deleting, and updating contacts and tasks. I also made sure that the input data was correct and met the requirements. An example of this is me implementing input validation in the Contact and Task constructors as well as the setters for the Contact and Task class.

I believe my tests are effective because they cover a variety of scenarios, and I used assertions to make sure that the expected results were obtained. I also used the code coverage feature in Eclipse to measure the percentage of code being covered by my tests. The overall coverage I was able to achieve was 84 percent, which I believe to be high enough.   
 I made sure my code was technically sound by using best practices and making the code easy to read and understand. I also included error handling in my code to prevent the program from crashing. One example of this is in my Contact constructor where it checks if each value is null or over the max string length and throws an exception if any value is.

I made sure my code was efficient by using ArrayLists to store contact and tasks. ArrayLists are efficient for adding, deleting, and updating elements. One example from my code is the deleteContact method which uses an iterator to go through the list and remove a contact if a match is found. I also included an exception in the event a contact id was not found in the list.

For each milestone, two primary testing techniques were utilized: unit testing and functional testing. Unit testing involved thorough examination of individual components in isolation to ensure their accurate functioning. It played a crucial role in validating the behavior and functionality of the contact service, task service, and appointment service. Through the creation and execution of varied test cases covering different scenarios, I could verify expected outputs and identify any defects or inconsistencies. On the other hand, functional testing aimed at validating the overall functionality of the services. It involved testing the services against predefined requirements and specifications to ensure they aligned with the desired behavior. By simulating various user interactions and inputting various data, I could validate the functionality of the services and identify any functional defects.

While these milestones primarily focused on unit testing and functional testing, other testing techniques were not employed. Performance testing, for instance, evaluates the responsiveness, scalability, and resource usage of software systems. It is particularly valuable for projects that handle a large volume of simultaneous users or complex computations. Security testing ensures the integrity of software systems and safeguards sensitive data from potential threats. This involves identifying vulnerabilities, testing authentication and authorization mechanisms, and assessing the system's resilience against security breaches. Usability testing focuses on assessing the user-friendliness and effectiveness of software from an end-user perspective. By observing user interactions, collecting feedback, and analyzing usability metrics, usability testing uncovers areas for improvement and enhances the overall user experience.

Each testing technique serves a specific purpose and can be applied based on project requirements and priorities. Performance testing proves essential for projects that demand optimal system performance and scalability, such as high-traffic web applications or data-intensive systems. Security testing becomes crucial for projects handling sensitive data or striving to meet stringent security standards. Usability testing ensures that software meets user expectations and delivers a satisfactory experience.

The implications of employing these testing techniques are various, ranging from improved system performance and enhanced security posture to increased user satisfaction. By incorporating these techniques into software development projects, the risk of defects can be mitigated, software quality can be enhanced, and a more robust and reliable product can be delivered.

I had a somewhat cautious mindset while I was working on the project. I made sure to closely review the requirements while I was creating the program and I also reviewed the requirements a final time after completion. It’s important to appreciate the complexity and interrelationships of the code I was testing because it allows me to get better testing coverage and results. An example of this is seen with my ContactServiceTest class with the TestAddContactWithUniqueID test. I had to utilize methods from the Contact and ContactService class to properly test that adding a new contact worked as intended.

I tried to limit bias in the review of my code by keeping my focus on the specific requirements for the project. It was important for me to not assume that I had correctly implemented code that matched the requirements, so I reviewed them multiple times during my code review. An example of this is in my Contact class, specifically my constructor method. The requirements state that each field cannot have a null value and cannot be over a certain length. While reviewing the code for the Contact constructor, I realized I had incorrectly implemented the IF statement that checks the phone number length. I originally was checking if the phone number was over ten characters when the requirements stated it must be exactly ten. I can imagine bias being a concern if I was responsible for testing my own code if I had a poor mindset. If I believed I coded everything correctly the first time, I wouldn’t have carefully reviewed my code to make sure it matched requirements.

Being disciplined in my commitment to quality as a software engineering professional is incredibly important. It’s important not to cut corners when it comes to writing and testing code because it could result in technical debt, poor code quality, system vulnerabilities, and increased maintenance. It’s essential to invest the time and effort to adhere to coding best practices to avoid or at least minimize these issues. I plan to avoid technical debt by conducting regular code reviews and investing in automated testing frameworks. I will continuously assess and address code quality and strive for comprehensive test coverage to deliver reliable software while minimizing the technical debt.