Adam Resemius

CS 230

2/15/2022

**Project 2: Testing Summary and Review**

**Summary**

As a software developer, I understand the critical role that testing plays in software development. It is essential to verify that the software works as intended and that parameters are in place to protect against injection-based attacks, bugs, and other issues. In my most recent project, I took the necessary steps to ensure that the backend had parameters that worked as intended.

To verify the functionality of the software, I used various tests. Firstly, I tested the methods that I had created with verification tests. This allowed me to verify that the software would work as expected. A crucail part of this process was testing the add and delete methods to ensure that the user could add and delete objects as intended.

I also tested the parameters to verify that errors were thrown when an invalid input was given. This process involved verifying that the strings had upper and lower limits and were not set to be null. By testing these parameters, I was able to ensure that the required parts of the object were filled properly and that the string was within the provided parameters.

Overall, the quality of my JUnit tests covered most of the software requirements, effectively testing the majority of the cases in each method. However, I did overlook one critical aspect of testing, which was ensuring that an object's field could not be updated to null. Nonetheless, I was able to minimize injection-based attacks by checking the parameters implemented beforehand. I also made sure to check that a non-existent item could not be deleted, which may cause bugs.

To ensure that my code was technically sound, I utilized JUnit to perform tests on the code and verify that the objects met their requirements and constraints. For example, I conducted tests on the taskID and taskService methods to ensure that they were functioning correctly. To illustrate this process, I have included screenshots of some of the JUnit test cases for these methods below. These tests ensured that the methods were able to handle different scenarios and provided the expected results. By conducting these tests, I was able to identify and fix any issues in the code, ensuring that the program was working as intended.





To improve the efficiency of the code, I used hashmaps to store data. Hashmaps are effective because they allow the user to add, search, and delete object IDs in O(1), ensuring quick updates when necessary. While these tests were valuable, I failed to verify that my code was entirely efficient. To address this, I could add tests that measure the time it takes to execute the methods and compare it to other methods to determine efficiency accurately.

**Reflection**

During my time in developing and testing my backend software project, I employed a range of testing techniques to ensure that my code met the expected specifications and functioned properly. One of the first types of testing that I performed was static testing, which involves reviewing the software requirements to ensure that I had the proper requirements to begin coding. I conducted thorough test case reviews to ensure that I covered all possible scenarios in my code, and checked for gaps and inconsistencies to ensure that my code had a solid foundation. This approach helped me to identify any issues early on and establish a solid base for my project.

After completing the static testing, I moved on to dynamic testing. More specifically Junit tests. Junit tests are a type of unit testing that are used to validate the functionality of each method and ensure that there were no syntax errors or logical flaws. I created test cases for various sections of my code, such as the Contact class, to test different scenarios, including valid input, invalid input, and edge cases. For example, in the Contact class, I tested for the correct formatting of phone numbers and email addresses, as well as the proper handling of special characters.

Another type of testing that I performed was acceptance testing. Acceptance testing is a type of testing that helps to ensure that the software meets all the business requirements and performs as expected from an end-user perspective. This type of testing can be performed manually or through automated tools, and can be used to validate that the system performs as expected under a variety of conditions.

While I performed a range of testing techniques, there were other types of testing that I did not implement in this project. For example, performance testing could have been implemented to verify that my code was efficient and performing optimally. Security testing is also a critical type of testing that is typically used in front-end and full-stack development to test for vulnerabilities and ensure that injection-based attacks are not applicable. In my other course, CS 305, I learned how to perform dependency tests using Maven to check for issues in dependencies.

Overall, by implementing various types of testing, I was able to deliver a high-quality project that met the requirements and expectations for this project. Testing helped me to identify issues early on and provided a safety net for any additional updates. It also helped me to improve the quality of my code over time.

Mindset

During the project, I was committed to limiting errors and achieving high test coverage. This was important to ensure that the completed project would function as intended. For instance, I tested deleting a task using the task ID and then verified if the task was indeed deleted. I also made sure to check the interrelationships of the code to ensure that everything was working as intended.

As a software developer, I recognize the potential for bias in testing my own code. To avoid this, I made sure to ask myself why I wrote the code and what it was intended to do. I also utilized JUnit testing and other forms of automated testing to identify broken methods and security concerns. It is important not to get lazy during the development process, as this could compromise industry best practices. An example of where I could have improved my testing approach is in evaluating the performance and efficiency of my code, which would help to ensure scalability in a live environment.

Being disciplined in my commitment to quality as a software engineering professional is crucial, as it ensures that clients receive products that meet their expectations. Neglecting quality during the development phase can result in a snowball effect of vulnerabilities and bugs that can cause maintenance errors in the future. Cutting corners in code writing or testing can lead to technical debt, which can negatively impact software efficiency, maintenance costs, and reliability. For example, inadequate documentation can lead to time and money being wasted in understanding and maintaining the code.

By implementing best practices and utilizing automated testing we can limit technical debt. Any issues can be identified and fixed early in the development process, reducing the chance of future risks.

References:

Benko, A. (2018, September 10). *The Software Tester's mindset*. SoftwareTester.Careers. Retrieved February 15, 2023, from https://softwaretester.careers/the-software-testers-mindset/

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). Software testing : An istqb-bcs certified tester foundation guide - 4th edition. BCS Learning & Development Limited.

Heusser, M. (2020, November 12). *Why is unit testing important for developers?: TechTarget*. Software Quality. Retrieved February 16, 2023, from https://www.techtarget.com/searchsoftwarequality/answer/Is-unit-testing-an-important-aspect-of-software-development

Manico, J., & Detlefsen, A. (2014). *Iron-clad java: Building secure web applications*. McGraw-Hill Education.

*What is technical debt and why QA testers should be concerned about it?* (2023, January 12). Software Testing Help. Retrieved February 14, 2023, from https://www.softwaretestinghelp.com/technical-debt-and-qa/#:~:text=During%20a%20typical%20software%20design,integration%20and%20other%20out%20of