Kayla Kintchen

CS-320

Professor Toledo

6/12/2023

Project 2: Summary and Reflection

Summary

1. **Summary**
   1. Describe your unit testing approach for each of the three features.
      1. To what extent was your approach **aligned to the software requirements**? Support your claims with specific evidence.

In ContactService, AppointmentService, and TaskService, I followed a systematic unit testing approach to ensure alignment with the software requirements. Here's a breakdown of my unit testing approach for each feature:

1. ContactService: I wrote test methods to verify the functionality of key operations such as adding a contact, retrieving a contact, updating a contact, and deleting a contact. Each test method focused on a specific requirement or functionality, ensuring that the code meets the expected behavior as per the requirements. For example, in the ContactServiceTest class, I tested the addContact method to ensure that contacts are successfully added and the getContact method to ensure that the correct contact is retrieved. By testing these methods against the requirements, I verified that the ContactService meets the specified functionality.

2. AppointmentService: I designed test cases to cover different aspects of appointment management, including creating appointments, retrieving appointments, updating appointments, and deleting appointments. The test methods focused on validating the requirements related to appointment scheduling and manipulation. For instance, in the AppointmentServiceTest class, I tested the addAppointment method to ensure that appointments are added correctly and the deleteAppointment method to verify that appointments are successfully removed. Through these tests, I verified that the AppointmentService adheres to the requirements.

3. TaskService: I created test scenarios to validate the behavior of task management operations such as adding tasks, retrieving tasks, updating task names, and updating task descriptions. The test methods targeted specific requirements. For example, in the TaskServiceTest class, I tested the addTask method to ensure that tasks are added properly and the updateTaskName method to confirm that task names are updated correctly. These tests ensured that the TaskService aligns with the requirements.

To support the alignment of my approach with the software requirements, I incorporated assertions within the test methods. These assertions compared the actual results of the operations with the expected results based on the requirements. By verifying that the test outcomes matched the expected behavior, I ensured that the unit tests aligned with the software requirements.

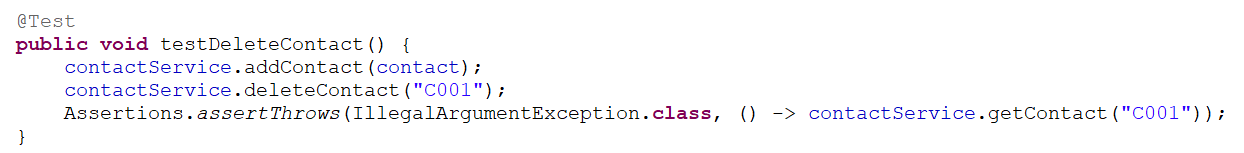
Overall, my unit testing approach closely followed the software requirements by testing the key functionalities and verifying that the code met the expected behavior specified in the requirements. The test methods and assertions provided specific evidence of this alignment, considering they targeted the individual requirements and validated the corresponding functionality.

* + 1. 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?

In my testing, I ensured that my coverage percentage exceeded the industry standard of 80%. Aside from exceeding the industry coverage baseline, I’m confident that my tests had other characteristics to guarantee effectiveness. My test designs covered several scenarios, I included assertions to verify the expected behavior of the code, tests were isolated, and my tests covered error handling. For these reasons,I am confident that my Junit tests were of top quality.

* 1. Describe your experience writing the JUnit tests.
     1. How did you ensure that your code was **technically sound**? Cite specific lines of code from your tests to illustrate. **<Write your answer>**

To ensure that my code was technically sound, I followed industry best practices and I implemented various techniques in creating my Junit tests. I used assertEquals to validate results of operations to ensure expected and actual values matched.



Aside from using assertEquals, I used the “setup” method to set up the necessary objects before each test is executed. By isolating the set up, I was able o prevent interference between tests and ensure accurate and independent results.

A picture containing text, font, screenshot, line

Description automatically generated

Another way I ensured my code was exceptionally sound was by implementing exception handling. By expecting exceptions, I was able to validate that code handles errors appropriately and throws the expected exceptions. An example of this is in the TastServiceTest class.

A picture containing text, screenshot, font, line

Description automatically generated

* + 1. How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate. **<Write your answer>**

To ensure efficiency, I minimized test set up, utilizing the setup method with the BeforeEach annotation to set up the objects before each test. Below is an example from ContactServiceTest

A picture containing text, font, line, screenshot

Description automatically generated

I also used assertions to validate the expected outcome of the code. Although I used this method numerous times, attached is an example from AppointmentServiceTest.

A screen shot of a computer code

Description automatically generated with low confidence

Throughout the program, to ensure efficiency, I also strived to reduce redundancy. In the following snippet from TaskServiceTest, I directly access the object after updating it using the “updateTaskName” method. Having this approach throughout the entire program reduces redundancy and improves efficiency.

A picture containing text, screenshot, font, line

Description automatically generated

1. **Reflection**
   1. Testing Techniques
      1. What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details. **<Write your answer>**

In this project I used several software testing techniques, including unit testing, boundary testing, and error testing.

Unit testing: I used unit testing to verify the functionality of individual components of my code. For example, in the ContactServiceTest class, I wrote code to validate the addContact method.

A picture containing text, screenshot, font, line

Description automatically generated

Boundary testing: Boundary testing checks the limits of input values to ensure that the code handles them properly. For example, in ContactTest, I included tests to verify that the input met the requirements. I tested for cases where input values are null, empty, too long, or exceed the allowed limits. This helps identify potential issues related to boundary conditions and ensures that the code handles them appropriately.

Error testing: I used error testing to test how the system handled errors or exceptional conditions. An example from my code is how in the ContactServiceTest class, I included tests to verify the behavior when attempting to delete a non existing contact. By using assertions and exception handling, I ensured that the system throws the expected exceptions in these scenarios. This helps verify that the code handles errors with the appropriate feedback.

A screen shot of a computer

Description automatically generated with low confidence

* + 1. What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details. **<Write your answer>**

In this project, I did not use security testing. Security testing would be testing the code and dependencies for vulnerabilities. I also did not use integration testing, which is testing to see that different modules interact with each other appropriately. I also did not conduct performance testing, which would test to ensure that the program is using resources optimally and running efficiently.

* + 1. For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations. **<Write your answer>**

The techniques I’ve mentioned, and their practical uses and implications are as follows:

Unit testing: Unit testing tests the functionality of individual components. Unit testing is extremely important since it allows for us to identify errors early in the software development process since we can conduct unit testing before integrating the components into a larger program.

Boundary testing: Boundary testing is used to test how a system handles input that meets or exceeds the boundaries. Proper boundary testing is extremely important in preventing vulnerabilities and maintaining system stability.

Error testing: Error testing specifically tests how a system handles errors or exceptional conditions. It involves triggering errors in the system to ensure that the system responds appropriately. Error testing plays a significant role in addressing potential issues related to error handling and reliability.

Security testing: Security testing focuses on identifying weaknesses and vulnerabilities in a program. This is essential in protecting sensitive data and ensuring a program’s integrity. Conducting thorough security testing will help an organization mitigate risks and maintain confidentiality.

Integration testing: Integration testing verifies that different components of a system work together and interact well. This ensures that a system functions properly as a whole. Integration testing can help identify issues related to data transmission, communication between modules, and overall system behavior.

Performance testing: Performance testing can identify inefficiencies and issues with resource allotments. Performance testing is essential in evaluating a programs resource utilization, speed, scalability, and responsiveness to ensure that the system delivers a smooth user experience.

* 1. Mindset
     1. 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, I adopted a mindset of thoroughness and caution as a software engineer. I assumed that unless a feature was tested, then it did not work. I recognized the importance of leaving no stone unturned and that there was no such thing as too much testing.

It was important to appreciate the complexity and interrelationships for numerous reasons. It allowed me to understand how different features interacted with one another and how changes to one part of the system could affect other parts of the system. Understanding this helped me to design thorough test cases that covered various scenarios and tested the overall integrity of the system.

For example, in ContactService, I considered the interactions between the Contact, Appointment, and Task modules to ensure that any dependencies were properly handled.

* + 1. 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.

In reviewing the code, I consciously made efforts to limit bias by testing my code as if I were not the one who wrote it. In doing so, I ensured a more objective evaluation of the code’s quality and reliability. As a software developer, I understand the impact that testing your own code can have, as some developers may be lenient or overconfident in their testing. However, as a growing developer, I am cautious and thorough in my approach.

For example, in my ContactServiceTest class, I tested the maximum inputs for each parameter to uncover potential flaws or vulnerabilities. Even though I had written the code, I wanted to ensure that it could handle and validate inputs that exceeded the maximum allowed length. This demonstrates my commitment to thorough testing and unbiased evaluation of the code.

* + 1. 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.

Regarding maintaining discipline and upholding quality as a software engineer, I firmly believe that I am the first line of defense when it comes to software security. I recognize that cutting corners during code development and testing can have extreme consequences, including reduced reliability, heightened vulnerability, and potential security breaches. To ensure that my code is robust, secure, and aligned with requirements, I am dedicated to adhering to rigorous code quality standards and conducting comprehensive testing.

While it may be tempting to take shortcuts for immediate relief, I am fully aware that such practices often lead to long-term challenges and an accumulation of technical debt. To counteract this, my plan is to follow industry best practices and diligently test every aspect of my code. I understand the importance of making informed decisions by considering multiple options and prioritizing quality over quantity.

An example from Project One that showcases my commitment to generating high-quality code by maintaining exceptional test case coverage. Despite moments of redundancy, I consistently ensured that my tests were meticulous and comprehensive, thereby delivering code of quality and maintaining consistently high code coverage. By carrying forward this commitment to excellence, I aim to translate these behaviors into a professional setting, effectively mitigating technical debt and contributing to the development of secure and reliable systems.

Through these principles and a commitment for producing high-quality code, I am confident in my ability to uphold discipline and avoid compromising on code quality. By doing so, I will continue to foster secure, efficient, and sustainable software solutions.