Madeline Meyers

Southern New Hampshire University

CS 320: Software Test Automation and QA

Kalysa Wilson

October 15, 2023

Project 2: Summary and Reflections

My testing approach was implemented based on the requirements presented in the rubric for the project. Each task in the requirements had its own test designed to ensure that the potential exceptions would be tested as well as potential successes. This can be seen in each milestone of the project as the requirement that the ID cannot be longer than 10 characters long or null exists in each application I created and the tests for these applications show that if these requirements are satisfied then they will pass the test and if the requirements are not satisfied then a separate test will show that an exception will be created and thus the attempted creation of an invalid ID will fail. I made sure to create a test for the setters of each application’s main class as well that would check if they were functioning correctly as well as if these simple commands failed to work correctly it would cause the related commands in the service classes for each of the main classes.

My JUnit tests were performed using Eclipse’s built in “run as JUnit test” function and the coverage was calculated using Eclipse’s “Coverage as” tool which checks the tests against the methods and functions that are related to each. Each test that was created ran successfully which included a test for valid inputs as well as tests for invalid inputs and thus valid inputs have been proven to work correctly and invalid inputs threw exceptions where they would be expected to do so. When examining the coverage percentage in the “Coverage as” tool in Eclipse the class that the test was created for showed as 100% coverage of potential branches which shows that my tests covered each function that could throw an exception, the exception was triggered correctly, and when the input would be valid the exception would not be thrown, and the function would behave as expected. I followed these steps for each milestone and the final project submission to ensure that I would have created and performed enough tests to feel satisfied with my work.

My code is technically sound because of how thorough I was in the creation of exceptions and the tests associated with each exception. For example, with the function in the TaskService class “updateName()” I set up the function to throw an Illegal Argument Exception if the input name was null or too long and created a JUnit test that created a scenario where both branches could be triggered to ensure the functionality of the exceptions. The following code block is the test for the update function in the test service class:

@Test

public void testInvalidUpdateName() {

assertThrows(IllegalArgumentException.class, () ->

{task.setName(null);});

assertThrows(IllegalArgumentException.class, ()->

{task.setName("Sample Invalid Name Too Long");});

When this test is run the functionality of the exceptions is affirmed and shows that these exceptions are thrown in both cases. Because I coded similar test cases for each function I was able to test each possible exception that I created and have been able to affirm each of them as functioning as expected which shows that my code is technically sound as the input function will work correctly according to the tests and exceptions will be thrown as expected according to the tests. This also extends to each application created for this project as each set of classes for the applications has their own tests and exceptions that were coded to affirm the functionality of the code as well as to show that errors would be thrown as expected. For example, the contact service class has a test class that affirms that the ability to delete contacts works properly as shown below:

@Test

public void testDeleteContact() {

Contact contact = new Contact("Contact123", "John", "Doe", "1234567890", "123 Main St.");

contactService.addContact(contact);

contactService.deleteContact("Contact123");

assertNull(contactService.getContact("Contact123"));

}

When I wrote this test I also noticed, if I failed to format the contact information correctly in the initial creation of the test contact to delete I would have an error before the test could run properly and once the code was redone to change any issues I had caused I was able to prove that the deletion function was indeed working correctly. This proved conclusively that my code was checking for errors thoroughly and with as little room for error as I could manage.

The way I ensured that my code was efficient was to set up a standard test case for each test before each test rather than needing to type out the full constructor for each test case. This allowed me to focus on creating the test cases where I would need to check for possible exceptions while allowing me to have a standard input for testing valid functions. The following code block is this set up function as shown in my “TestTask” JUnit class:

@BeforeEach

public void setUp() {

task = new Task("Task12345", "Sample Task", "Sample Description.");

}

Because I was able to set up this standard test case I was able to lower the amount of copying this code into each test case where it would be relevant and if I made a mistake when writing the test case, I would only need to change one line rather than looking through multiple lines of code. This only changed if I needed to create an invalid version to test against or if I needed to create a second entry for any reason but ultimately I was able to save myself on needing to rewrite code when creating each test.

During the development of each milestone of this project I worked through I created functional tests to ensure that each error would be thrown as expected and that the entry methods would work as expected for valid entries where no error would be expected. Specifically, I created Unit tests to test the components of the milestones which covered each error I wrote into the code and each method and function created in each milestone. Because every function had a test associated with it and each error had a test to check for its functionality, I was confident in my code’s conformance to the requirements listed in each milestone. These unit tests were each small in scope and were made to test for errors and valid entries, as such they helped me to ensure that my code was functional and could potentially move to another level of testing as my code was shown to throw errors where they were expected and would show that the entries would be made correctly when the inputs were valid. I also worked with Integration testing as I created tests using both classes in each milestone and proved that they work together and created the expected outputs. Integration testing focuses on how multiple modules of code work together and because at least one of my classes was using a function from the other class these tests would throw errors from the other class that was not being tested if I made a mistake when creating a test for the class I was testing.

During development of the applications in this project, I did not move past Unit testing and Integration testing and as such I have not moved into Systems testing, or non-functional testing such as Performance testing. These types of testing require testing based on the hardware that they would be used on, and I was only testing the code I wrote and tested using an IDE and Unit tests. Performance testing, likewise, would test how the memory of the program would be managed and how fast information could be retrieved if there were many entries, these types of tests were outside the scope of this project however they are important to remember going forward.

For many different projects at least one of these types of testing can be important. Unit testing is best used for testing lesser amounts of code to ensure that they are functioning correctly and will lower the cost of bug fixing as errors would be caught quickly as the software is being developed rather than finding the errors after a lot of development has already been done. Integration testing is most useful when a piece of software would have multiple components that interact with each other, and the developer would need to ensure that each of the pieces is working correctly and any information that was created by one component that needs to be accessed by another component can access the information and if needed, modify the information. Systems testing is important to ensure that the performance of a piece of software works correctly in a more realistic scenario and verifies that all requirements are being met. Performance testing is important to show that the software can manage the resources of the system correctly and can help identify if there are any bottlenecks in the performance that need to be cleaned up during development. This type of testing is important for projects that work with large amounts of data or data which needs to be accessed quickly and updated quickly. Each of these types of tests are important to prove to the stakeholders and the developers that the software is functioning correctly, and the level of risk involved with progressing the development of the software is as low as possible.

I was cautious throughout this project as I took the care to create my tests for each milestone of this project. I wanted to ensure that each separate part of the project would work as expected, match the requirements for each class, and that the proper errors would be thrown when expected. I was careful to check through each class and see that each function was tested and that each possible branch of a function was tested thoroughly. This was especially important for the constructor functions of each class as well as the setter functions. This is because these functions would be used by the service classes for each main class and without these functions correctly working the service class would not only fail to work but also they would be harder to test as their success is built upon the main class’s success.

In my attempts to limit my own biases I worked to show specific lines of code to illustrate how my code was effective rather than only talk about how my code succeeded in meeting the requirements and testing efficiency. When creating the test cases, I started from the thought of where I could have failed in defining a function and working to prove how it would work rather than assuming any individual function would work without testing. It is best to limit bias when writing test cases, especially when the tests are for your own code, as overlooking a possible error point could lead to more work when testing the code as you may have to check through each function to determine what could have gone wrong. Instead, if you take the assumption that the code may fail you create thorough tests that show objectively where errors exist or where they may not when they are expected which can identify what needs to be changed and if a condition was set correctly. This can be clearly seen in how in each part of this project I created error statements for each possible point of failure, even in simple functions such as the setter functions in each of the main classes. By testing as thoroughly as possible I was able to prove to myself that my code was functioning correctly and up to the requirements set in the rubric.

It is important to be disciplined while writing code and tests as your code needs to be of the highest quality that you can commit to. The requirements of any project need to be met and proven to work correctly and it is important to be thorough in how a project is coded. If you cut corners while working on a project you will eventually run into a problem that should have been solved earlier in development which could range from a numerical error or a potentially dangerous situation that should have been avoided depending on the project being worked on. In order to avoid technical debt as much as possible, I plan to work diligently and as slow as I can while still being sure to keep in time with deadlines so I don’t rush the coding involved with each stage of development. I will ensure that my code is at the highest level of quality I can provide and if I need to collaborate with another professional I will do my best to communicate expectations, gather requirements about the project, and work to create solutions to any problems that arise during development and create thorough tests that prove that functions work as expected and fail as expected if needed.