Grand Strand Systems Summary and Reflection

Kate Doughty

Southern New Hampshire University

# Summary

For each feature of the application, I based my testing on the requirements given by the customer. I made sure each requirement had at least one test associated with it. I made sure that each unit in my code had at least one test. Making sure I had complete coverage of the code was important, so that I knew everything worked as it should.

I made sure to test each function worked as it should, such as in lines 131-220 of ContactServiceTest, where I tested the functionality of the update feature, making sure each function updated the correct variable with the correct information. Efficient code can be ensured by checking that code does what is required, without any unnecessary additions or mistakes. This is shown in TaskServiceTest lines 76-92, where the Delete Task function was tested, making sure that the correct information was deleted.

## Reflection

For all three milestones I used unit testing. In unit testing each ‘unit,’ which can be an individual class or method, is tested on its own to make sure they work as intended. Another method used was static testing. I spent some time as I was writing the code checking my work for errors and mistakes. While this method did not catch everything, it helped reduce the number of test failures later. Some of the testing techniques that I did not use for the milestones were regression testing and usability testing, mostly because they did not apply. Regression testing is where a new version of a software is tested using tests from the previous version to check that the software still does what is intended. Usability testing checks that the software interface works well, focusing on the user experience. Unit and static testing are practical for all software development projects. These methods are important for making sure the code is the highest quality it can be, and that the code works. Regression testing is important for when a software is being updated or remade. It is important to make sure that the software still does what it is supposed to. Usability testing is important for applications and other systems with user interfaces. Any program that will be used by anyone should be accessible and easy to navigate.

For this project I attempted to be cautious. It is important to remember how pieces of code interact with each other, as those interactions are what make the code fail or succeed. If the incorrect variable is passed to a function, the outcome will be incorrect, and potentially result in an error. I attempted to reduce bias by distancing myself from the code. I tried to avoid thinking about it as my work and approached it like I was checking a peer’s work. Bias is a bad thing for a developer to have when testing. Developers tend to see their own work as correct, and that can cause errors to slip past them, as they don’t want to see any errors in their own work. Testing should be thorough and carefully done. Cutting corners and sloppy work can lead to errors slipping through the cracks. In the example of Project One, the testing done on variables was an important thing to not skip. If a developer tested the system’s response to a variable that was too long and decided that was good enough, the system could later have problems with an empty variable. An issue that could have been avoided if the tester hadn’t cut corners.