CS-320-T2824 Software Test Automation & QA

Project Two

Eric Wallace

Dec. 11, 2022

My approach to project one was closely aligned with software requirements. I double checked to make sure that each requirement was addressed and completed. One thing I did differently in project one that I did not do in any of the previous modules is I created a utility class which contains functions to check the length of variables, validity of dates, generate a valid and invalid ID and date. This ensured that tests were standardized across all different modules.

Prior to this course I had very little experience with JUnit testing, all of my testing consisted of writing code and then testing that code by executing the code to make sure it ran and if it did I had considered it good to go. I had not given much thought into code or unit testing and honestly did not understand what testing truly was, I had assumptions of what I thought unit testing was and I was wrong. I now understand the power and need for unit testing and in the future, I will use unit testing on projects until I get a job on a development team that has designated testers. Software testers are a vital piece of a development team, I had studied the role of a software testing in Agile but didn’t go into details on software testing.

I know the effectiveness of my testing because my code coverage was 100% on all tests, 100% on all classes. On the Appointment module, for the Appointment and AppointmentService classes I had 85%- and 90%-method coverage along with 88%- and 87%-line coverage respectively. For the Contact module, on the Contact and ContactService classes I had 90%- and 92%-method coverage and 96%- and 83%-line coverage. Lastly, the Task module, the Task and TaskService classes had 85%- and 90%- method coverage and 93%- and 78%- lines coverage. Early on the lower numbers on coverage of methods and lines in the individual modules concerned me but after watching a video we were introduced in a previous week I was put at ease by finding out that there will not be 100% coverage on the individual classes.

As for the efficiency of my code, since I really have no experience with testing, I’m not sure what would be considered acceptable or efficient. Complete testing takes 323 milliseconds, to me that seems efficient, but it may be wrong.

The techniques I used for software testing in project one was I first determined the requirements laid out in the rubrics for project one, next I planned how I would carry out the tests and finally I put my plan into use. To accomplish this, I totally rewrote the code for every class in the project, I standardized each of the classes with every other. Every method within a class was in the same exact order as every other class which made it easier when my tests failed, for example, if a class failed on a certain method that meant that the other classes would also fail. So, when it came to correcting a failing test, I would change one test once it passed, I could then change it in the subsequent tests and then they would pass.

The use of functions to check variables that they were not null and did not violate length restrictions. This cut down on the amount of code I had to right, it ensured tests were the same across all tests. Another technique which I outlined at the beginning of this report that I employed was using a function to generate the IDs and dates for each test. I was able to designate the use of a valid ID when needed and then an invalid ID at other times. All of this would not be possible with the various methods used in the classes to accomplish the requirements. For every class there was a insert, update and remove methods and this is on top of all the mutator and accessor methods which were heavily used in the testing.

One thing I did in rewriting the code was each of the tests were standardized but I also utilized the use of the AssertAll assertion. Using the AssertAll assertion allowed me to minimize the amount of code needed to get 100% code coverage. In using the AssertAll assertion I was able to write the valid, invalid, and null tests all in one instead of having to break them up into different tests. Through AssertEqual and AssertThrows I was able to accomplish 100% code coverage which I believe was successful. I won’t say that I did not make mistakes and I did not miss anything because I almost undoubtedly did miss something, and I know I made mistakes.

In conclusion, coming into project one I tried my very best to come into the project with the mindset that I could not measure successful tests based on whether my code compiled and ran successfully. To prevent this, I not once compiled or ran my code, I did this, so I did not form a bias against testing. I was afraid if I had used tactics employed in the past I would not see the importance of software testing and how it plays an important role in the development process. If I had done that it would have reinforced my prior behavior to incorrectly develop software and make common mistakes. I can honestly say this course has taught me a lot about the development process and the importance of software testing in the profession of software engineering. Before I was unaware of the use of testing in the profession and will forever change my approach to software development and how I write code.