Project 2

CS 320 Software Automation & GA

Secally Barbosa

Southern New Hampshire University

June 20th, 2023

Throughout this course, I learned a lot about why and how unit testing approaches are crucial to development. I tried to stick with the same habits and practices throughout the entire process of contact, task, and appointment modules. Each module would include a list of requirements, which then led me to making my own written down list in sections to know which direction to go in piece by piece. I also used rubrics as a list a client would give me. Creating a Junit test for each portion would create reassurance that the methods and practices used thoroughly the development were correct and executed correctly. I knew these JUnits were correct because they would be green. I would also keep running the tests to check for Syntax errors or other potential issues.

My experience with Junit tests grew throughout the course, and grew with depending on them to make sure the code was always sound and up to par. Using this technique on each application allowed me to be able to make updates, add anything, or even delete things. I learned that testing is essential and makes the code more successful in the end. The Junit testing was very dependable. A few times throughout these developments, it caught errors that then lead me to making corrections and enhancing my code. For example, using white box testing lets me test said developments using what I have learned and what I know about the code itself.

I used other techniques as well, like black box testing. Black box testing would be used when requirements are specified, and they must be met. These specifications would only be the structure and design of the system externally. Also, white box testing is only used when the creator/tester has knowledge of the system and how it works. This would be used in terms of the design of the code specifically, not the external structure like black box testing. Using all these forms of testing let me ensure that the code was going to meet all requirements. Using many different types of testing allows the code to be looked at in many ways, also ensuring that all different kinds of vulnerabilities/errors can be found prior to “completing” a development.

Regression testing in the type that would be done to make sure that the code that was created would change the current standing of the functionality in the original project/product. This would also find different types of bugs or new ways to make the code more functional.

I employed caution to a certain extent. I was cautious in terms of making sure all requirements were met and all code was sound, but also was a little less cautious in terms of coding practices. I feel that especially now, with not much experience in coding, I like to explore and be open with how the code can be completed. Doing this lets me learn how not to do certain things, and how to properly do other things. I feel that making mistakes throughout the creating process is necessary and makes the end goal easier. In terms of bias, I don’t think I really understand how it could interfere with the creation of a code, so I did some research. After doing some research, I see that developers sometimes set a timer when looking over codes, or they only test certain parts of the codes to prevent bias.

Being disciplined leads to being a reliable code. While building, I ran into many issues, and did not get perfect grades per submission. Getting the feedback and being disciplined enough to retry the code and to spend more time on fixing it to make it meet the requirements was discipline. It is important not to cut corners because then it can lead to further issues, and you just wouldn’t be meeting the clients’ needs. I plan to avoid technical debt as a practitioner in the field by being as organized as possible and making sure that time is used wisely.

References

Livezey, S. (n.d.). *The software engineering disciplines*. Disciplines. https://www.softwareengineeringauthority.com/index.php/disciplines

*6 examples of bias for action for software engineers*. Create the perfect Self-Review. (n.d.). https://getworkrecognized.com/blog/bias-for-action-examples-software-engineer