Software Test Automation Project Two

**Testing approaches utilized for each feature.**

For each of the three features, I was tasked with completing I utilized an automated unit testing approach. “Automated approaches commonly use a testing framework to develop test cases. These frameworks are also set to flag and report any failed test cases while also providing a summary of test cases” (2019). When coding these features, I tried my best to follow the software requirements that were outlined for each segment. For example, the contact service feature required that the contact service class be able to add and delete contacts per contact ID. For this requirement, I ensured that I created the appropriate Getter and Setter variables so that my program could set and add a contact ID and also get the Contact ID that needed to be deleted. For example,

public String getContactID() {

return contactID; }

public void setContactID(String contactID) {

this.contactID = contactID; }

**Quality of the JUnit tests.**

The overall quality of my JUnit tests could be a lot better. I had a hard time when creating these features and struggled a lot just to need the software requirements and test that they were working as needed. As such, I neglected to test for many failed test cases along the way.

**Experience writing the JUnit tests.**

So far my experience with JUnit tests has been interesting to say the least. I appreciate the opportunity that I had to be able to work with JUnit tests and I now understand how valuable it can be during the testing phase. I do however need to get more practice coding JUnit tests before I will be comfortable writing them with an acceptable coverage percentage. When it came to making my code technically sound, I had to ensure that my code was easy to understand and well-tested. As such, I tried to code as simply and clearly as possible while meeting all the project requirements. I then created the JUnit tests to ensure that my testing scenario passed my unit tests. For example.

@Test

void idNameTooLong() {

assertThrows(IllegalArgumentException.class, () ->{

new Contact("111111111111", "firstName", "lastName", "123456789", "1234 Rock Hill Road"); }); }

@Test

void firstNameTooLong() {

assertThrows(IllegalArgumentException.class, () ->{

new Contact("1", "firstNameLong", "lastName", "123456789", "1234 Rock Hill Road"); }); }

**Ensuring code efficiency.**

I ensured my code was efficient by combing through and making such that I removed any unnecessary and unessential operations. I also turned to various other resources such as the student academic center, YouTube, and Stack Overflow whenever I became stuck and couldn’t continue. These resources played a huge role in aiding the efficiency of my code.

**Reflection**

For each of the three features that I created, I utilized the testing approach of White Box Testing. “White Box Testing is a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. White box testing in software engineering is based on the inner workings of an application and revolves around internal testing” (Hamilton, 2022). This testing approach was best suited as I to code all three features and as such knew had access to the code and knew its inner workings.

I also utilized dynamic testing in the form of JUnit tests. “Dynamic testing is a type of software testing that requires the execution of code. The benefit of testing executable code is that QA analysts can get a look at how the software performs while running in a real-world situation” (Boog, 2022).

**Software testing techniques not used for the project.**

One software testing technique that I did not use for this project was static testing. Static testing is a cost-effective testing technique that is usually done at the beginning of the development process and is done before the code is run. It is used to locate any errors and bugs in the code before they can become major problems later on for the development team. For the assigned features I did not use the static testing technique, I waited until my code was completed and then I ran unit tests on small sections of the code.

**Practical uses and implications.**

White Box testing is used to ensure that the code is not vulnerable to any known security tests and exploits. Dynamic testing is typically used to ensure that the system or program is working as intended. Lastly, static testing is used to detect any bugs or errors in code early in the development cycle.

**Mindset**

Coming into this course I wasn’t sure what to expect, however now that I have finished this course I feel as though it has given me a whole new mindset when it comes to testing code. I didn’t realize the amount of work and thought processes that went into testing projects and ensuring that they were working properly. When taking on the role of a software tester I had to be extremely cautious and meticulous with my code. I had to write, run, edit and rewrite my code to ensure that all the requirements were tested for and passed. For example, my JUnit test had to test to ensure that an exception was thrown if the contact ID was null. At first, I forgot to replace the contact ID to test to see if my code would throw an error therefore the JUnit test failed. I had to go over my code identify the problem and replace the contact ID with null.

@Test

void idIsNull() {

assertThrows(IllegalArgumentException.class, () ->{

new Contact(null , "firstName", "lastName", "123456789", "1234 Rock Hill Road"); }); }

Appreciating the importance of the complexity and interrelationships of my code allowed me to take the time to sit down and understand what I needed my code to do and how I needed to go about getting code to run.

**Assess the ways bias was limited in code.**

For me, limiting my bias was not hard to do. I came into the assignment knowing that I had no previous knowledge of using JUnit and as such, I knew that there would be mistakes made along the way. As such, I did my best to remain open and willing to accept help and criticism. I sought help from SNHU’s academic tutors and they provided some great insight about my code that I then took and applied to the work I had already done. I think in general there will also be a concern of bias from someone looking at a developer coding and testing his own work, but I believe that as long as you are following best coding practices and remaining open to review and critique that should not be a big issue.

**Importance of being disciplined in commitment to quality.**

When writing or testing code it is always important to follow the codes of ethics and professionalism. We are preparing to go into the workforce and if we do not begin to instill these qualities as a developer it will start to show in our work and could possibly affect our chances of getting and keeping a job. Cutting corners, in the long run, will do nothing but leave you liable and at risk of technical debt. The best ways to avoid technical debt are to practice writing optimal code, creating proper documentation, and code testing.

**References**

* Hamilton, T. (2022, November 19). *White box testing – what is, techniques, example & types*. Guru99. Retrieved January 21, 2023, from https://www.guru99.com/white-box-testing.html
* Contributor, T. T. (2019, August 15). *What is unit testing? definition from whatis.com*. Software Quality. Retrieved January 21, 2023, from https://www.techtarget.com/searchsoftwarequality/definition/unit-testing
* The QA Lead, & Boog, J. (2022, December 10). *9 types of software testing in software engineering*. The QA Lead. Retrieved January 21, 2023, from https://theqalead.com/test-management/types-of-software-testing/