Project Two

Software Test Automation & QA

4/17/2021

Brandon Lombard

Throughout Project One, I was required to write and test a mobile application for a customer that allows the user to add, update, and delete contact, task, and appointment objects within the application. I was able to properly align my project with the software requirements by conducting complete tests and fully following the customer’s requirements. I used both Static and Dynamic Testing to ensure this program was fully functional. Static Testing is conducted without running code, while Dynamic Testing involves running the code for the test. This project required that the ID, firstName, lastName, and phone Strings should not be longer than ten characters and they should not be null. Additionally, the contact ID String should not be updatable. I was able to ensure this works by testing the user input and if they was more than ten characters, then ensured they couldn’t have a null value. For example, I entered contactID as “1010101010” to see if it allowed the value, and it passed. Then, I tried “112233445566778899” as a too long contact ID and it failed because the user should not be allowed to enter that long of a contact ID. My Junit tests ran at a 90% pass rate showing that they were effective.

During this project, I ensured that my program was both technically sound and efficient. An example of how my program was both technically sound and efficient can be look at through Task.java. I made my code have proper spacing and was free of syntax errors. Additionally, by having many comments I am able to keep efficient and easy to read code for myself and if a peer reviews my code. While developing code on eclipse, I also pay close attention to potential errors that show up on debugging. Additionally, I used software development best practices such as writing logical and simple code, using comments, etc. The piece of code below is an example of how my code was technically sound and efficient (taken from Task.java).

**//Mutator for task Id**

**//Check if ID is less than 10 chars and not null**

**public void setTaskId(String taskId) {**

**if ((taskId.length() > 10 || taskId == null)) {**

**throw new IllegalArgumentException("Invalid");**

**}**

Software testing is a hugely important part of the software development lifecycle. Testing allows the developers to understand what needs to be fixed before the final product is released to the public. There are a multitude of software testing techniques that can be used during a project, however I used only a couple of these techniques for the 6-1 Project One Submission. One of the software testing techniques that I used for this project was “Decision Table Testing” where inputs from the user are tested against the project. For example, the character ID could not be longer than ten characters in Project One, so if the user created an ID with eleven or more characters, then the program would output an error and tell the user that the input exceeds the allowed character limit. Another software testing technique that I used was “Regression Testing”, which is meant to save developers time during their testing. This form of testing ensures that new changes made to the code will not negatively impact the old codes functionality. While using this method, I would retest code after I made a few changes to ensure it was still working at the level I needed it to.

There are more software testing techniques that I did not directly use in this project, to include end-to-end tests, smoke testing, exploratory testing, etc. End-to-end testing involves “replicating a user behavior with the software in a complete application environment” (Pittet). This method, however, can be costly and time consuming and in larger projects: hard to maintain. Smoke testing, on the other hand, test the basic functionality of the program you are developing. They are quick to execute and ensure that the major features of the program are working as wanted. Finally, exploratory testing is where test cases are not created in advance, so testers check the code manually. Usually, this type of test shouldn’t last more than two hours, and it is made to test a specific area of the program. Additionally, it can be costly to perform and should be done by a team.

While acting as a software tester in this project, I tried to maintain simplicity while also conducting thorough tests to ensure my program was functioning properly. I tried to maintain a decent amount of caution when testing my code, because I am not experienced with testing enough until I took this class. I continuously tested my program to ensure functionality while ensuring there were not any major bugs in the code. I tried to maintain a determined and organized mindset for this project. I believe organization of code is particularly important with proper indentations, code spacings, and comments to describe what is happening in that piece of code. The test cases I wrote for this project were based around the expected input value(s) from the user. This includes where the user was to enter an ID with a length not more than ten digits. By testing a value larger than ten digits, I was able to ensure the user couldn’t input larger values.

I tried to limit my bias in my review of the code by anticipating potential bugs and issues that may come up. I did not look at it as a project I developed, I instead treated it as if I were testing another classmate’s code. It is important to not have bias when testing code because that could allow you to make mistakes or oversee bugs and other issues. If I were developing a personal project, I would have another Computer Science student take a look at my work and test it while giving me their opinions. I believe another set of eyes can be the most useful method of ensuring a fully functioning project. It is especially important to be disciplined in my commitment to quality as a software engineering professional. Overall, the user is the most important regarding the project. The team is developing the software for the end-user, so it only makes sense to be disciplined to having a quality software program. Said software could impact lives and could cost the company money if there are too many bugs or issues. Additionally, the program should have a high level of security so as to not be tampered with my hackers. Both writing and testing code are equally important, so developers should never cut corners at all throughout the software development lifecycle. I plan to avoid technical debt as a practitioner in this field by always staying focused on my development goals and deadlines, and not cutting corners writing or testing code. Additionally, I will try to maintain outstanding communication between myself and my team members to ensure a polished final product within the set forward deadlines.

**Reference:**

Pittet, S. (n.d.). *The different types of testing in Software*. Atlassian. https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing.