**Summary and Reflections Report**

Donald Thibodeaux

CS-320-T4514 Software Test Automation& QA

Angel Cross

4/15/2022

Summary and Reflections Report

**Summary:**

My testing approach was to test the software requirements by testing functional requirements. When testing we covered what the end user would experience when entering data. Also, some non – functional requirements such as the reliability of the software. We test to see if task can be added and deleted as well as length of task id, task description and task name. Junit tests provide code coverage percentage which measures the code quality and stability of your code. This makes sure that all implementation methods are covered. I know that the junit test code coverage is good when the score is 85% or higher. I ensured my test were technology sound by testing the code implantation to make sure everything is in working order. For example, making sure to test for name lengths, and adding and removing task. I also added good to ensure there were no null values using if statements to ensure efficiency. @Test

@DisplayName("Task description cannot have more than 50 characters")

void testTaskDescriptionWithMoreTHanfiftyCharacters() {

Task task = new Task("do the tasak and do the task","do the task");

if(task.getTaskDescription().length() > 50) {

fail("TaskName Has more than 50 characters");

}}

public void setTaskDescription(String taskDescription) {

if(taskDescription == null || taskDescription.isEmpty()) {

this.taskDescription ="NULL";

}else if(taskDescription.length() >50) {

this.taskDescription = taskDescription.substring(0,50);

}else {

this.taskDescription = taskDescription;

}

For my milestones I have employed static analysis which is where I tested my code for defects. I tested to see if the code would react like it was written. I tested for input as well as compiler errors. I tested for id and description length as well as making sure that the field was not empty. The code also had to be tested for update options and delete options. For the milestones I also used peer review for my code. I have only used unit testing at this point. I have not used integration testing, regression testing, security testing, functional testing, and compliance testing at this point. I believe unit testing is essential to good code. I have seen at times from research that they are people that work backwards as writing test first then writing the code according to the test. I believe testing is a good way to find hole or errors in code. I believe the best practice for projects is not to wait for testing to be passed on to quality, but the developer should also utilize testing to check his or her code.

**Reflections:**

For my project I utilized static and dynamic testing find errors and test functionality. I tested for input values and code coverage. Static testing is used to find errors without using code, in my case I used the snhu help center to have my code reviewed to see if any errors exist. For dynamic testing I tested the execution of my code. This involved testing input functions for length, adding, and deleting names, addresses, descriptions, task, and appointments, updating descriptions names, task names and appointments. I also tested code for code coverage. I did not cover integration, system, and acceptance testing during my project. Integration testing is performed when two or more tested units are combined into a larger structure. The test is often done on both the interfaces between the components and the larger structure being constructed, if its quality property cannot be assessed from its components. System testing tends to affirm the end-to-end quality of the entire system. System test is often based on the functional/requirement specification of the system. Non-functional quality attributes, such as reliability, security, and maintainability, are also checked. Acceptance testing is done when the completed system is handed over from the developers to the customers or users. The purpose of acceptance testing is rather to give confidence that the system is working than to find errors. The value of these test is to make sure code that is functional and to avoid releasing a product that has Escapements and Defects. Without testing throughout the life cycle product development code with serious problems can be released. Sometimes functionality can suffer, and no major problems can arise besides a recall but other times the software not being functional can cause loss of life, lawsuits, and loss of massive income. These tests can possibly avoid this major risk no code is going to be 100% error free, but test can point out major issues in software. The mindset I adopted during this project is to test and code. I tried to write small amounts of code then test the code. I never realized how important testing is, you may have spent large amounts of time on code and think that everything is fine but once you start testing you realize the code needs to be refactored. I was cautious not to go overboard with code and just focus on meeting the requirements for the program. It was important to appreciate the complexity and interrelationships of the code because when testing for certain items I had to make sure the object was null and also there was a max number of characters that could be entered. When testing code, it is important to put bias aside and let the test do their job. I used junit testing to fix errors in my code. I put my bias aside by knowing the fact that no matter who wrote the code it can always be better and one way to improve code is to test your code and trust the test. I can understand how some developers can be bias of their code that they worked hard on. When you put in some much time and effort in a project it can become more than just a project it becomes your baby, and you will be protective of your baby(code).

Being a software engineer is an important job and requires a commitment to creating quality code. You not only have an obligation to the end user but to yourself to make sure your code reacts the way that it was designed. There have been many instances of corrupted code being released and cost millions in damages and time. Sometime code can also have worst consequences such as death. It is important to plan ahead for testing, the testing process is important and should be given the appropriate respect. Cutting corners should not be an option when writing and testing code. Technical debt should be avoided as you do not want to speed extra money and time to rework software because you cut corners the first time. I will avoid technical debt by taken time make sure I turn in quality code. I will also test code as I go and not leave testing until the end of a project. I believe the agile methodology is the best way to develop code because of the focus on testing throughout the lifecycle.

**Reference**

Software testing techniques. GeeksforGeeks. (2021, March 1). Retrieved April 16, 2022, from https://www.geeksforgeeks.org/software-testing-techniques/