Matthew Marsh

CS-320 Project 2

August 16th, 2024

**Describe your unit testing approach for each of the three features.**

* **To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.**

My approach was very close to the requirements that the client had requested. I had

missed a couple of lines of code that did not meet the requirements. I missed the

requirement for the phone number for the number of digits. I had missed a couple of the

signature requirements for the TaskService Class and for the ContactService Class.

* **Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were effective based on the coverage percentage?**

When running my Junit tests, they had come back that they were done and that I had met

the requirements for what I had coded the tests to do. I was not aware until after the

grading that I had made a few mistakes on the coding that caused certain test to not

perform to expectations. When I ran the tests, it had stated that I was at 100% coverage.

**Describe your experience writing the JUnit tests.**

* **How did you ensure that your code was technically sound? Cite specific lines of code**

**from your tests to illustrate**.

When writing my code, I ensured that the variables were written withing the requirements that the client had asked for were being met. When coding the variables, it helps to have those to help speed up the process of the tests in the case that you need to be able to update the code if the requirements need changed. Portion of the coding is below:

@Test

void addContactTest() {

Contact contact = new Contact("1", "firstName", "lastName", "1234567890", "address");

Contact contact2 = new Contact("1", "firstName", "lastName", "1234567890", "address");

//adds contact by unique id

assertTrue(cServ.addContact(contact).get("1").equals(contact));

//does not add contact by non-unique id

assertEquals(contact, cServ.addContact(contact2).get("1"));

}

@Test

void deleteContactTest() {

Contact contact = new Contact("1", "firstName", "lastName", "1234567890", "address");

cServ.addContact(contact);

//deletes contact by id

assertEquals(null, cServ.deleteContact("1").get("1"));

* **How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.**

I ensured that my code was efficient by doing repeated tests, to make sure that the code was free of errors and running correctly as it was designed to do. Portion of that code is below:

@Test

void deleteAppointmentTest() {

Appointment appointment = new Appointment("1", date, "description");

aServ.addAppointment(appointment);

//deletes appointment by id

assertEquals(null, aServ.deleteAppointment("1").get("1"));

}

@Test

void unmodifiableTest() {

Appointment appointment = new Appointment("1", date, "description");

//initialize exception objects from service methods

Exception addException = assertThrows(UnsupportedOperationException.class, () -> {

aServ.addAppointment(appointment).clear();

});

Exception deleteException = assertThrows(UnsupportedOperationException.class, () -> {

aServ.deleteAppointment("1").clear();

});

//throws UnsupportedOperationException when trying to modify returned unmodifiableMap

assertTrue(addException.toString().contains("UnsupportedOperationException"));

assertTrue(deleteException.toString().contains("UnsupportedOperationException"));

**Reflection**

* **Testing Techniques**

**What were the software testing techniques that you employed in this project?**

**Describe their characteristics using specific details.**

I used Junit testing techniques to test my project. This was done to ensure that all the

requirements were met that were listed by the client. Junit is a good way of testing

because it breaks down the project in separate parts to ensure that they are functioning as

it was intended to be.

* **What are the other software testing techniques that you did not use for this project?**

**Describe their characteristics using specific details.**

I did not used integration, which tests how the different components work together to

ensure they function cohesively as intended for the system. I did not use regression

testing, which is used to test the system after you have made changed to ensure that no

new bugs have been introduced into the system. And I did not use system testing, which

tests the complete system to ensure that the requirements have been met.

* **For each of the techniques you discussed, explain the practical uses and implications**

**for different software development projects and situations.**

Unit testing’s practical use is to ensure that the software is functioning at its base units as

expected. It also helps to ensure that code is working how you designed it to work.

Integration is good for finding defects in the system when two or more modules are

integrated together, which aids in finding defects sooner. This helps to save time and

money. Regression testing is used to ensure that any changes to the code do not cause

new defects in the already existing functions, which helps to ensure that the software is

working as a cohesive unit. System testing is used to ensure that the software is meeting

the requirements that have been laid out. With the other types of testing, system testing is

done from the angle of the end user, which helps to evaluate the experience and make the

software more user-friendly.

**Mindset**

* **Assess the mindset that you adopted working on this project. In acting as a software**

**tester, to what extent did you employ caution? Why was it important to appreciate**

**the complexity and interrelationships of the code you were testing? Provide specific**

**examples to illustrate your claims.**

In this test the requirements were straightforward and simple. Complexity can play a

huge role when you are doing more testing. If you try to over complicate the testing

process, it can lead to over testing. You also run the risk of introducing a bug from over

testing the system.

* **Assess the ways you tried to limit bias in your review of the code. On the software**

**developer side, can you imagine that bias would be a concern if you were**

**responsible for testing your own code? Provide specific examples to illustrate your**

**claims.**

Creating the code for this project was simple, even though I made a few mistakes in

the process. I think that testing your own code is not the best option. Some of us don’t

like to admit that we didn’t make a mistake, even when it is in neon lights, flashing at us,

that there is an error. We tell ourselves that everything is ok and brush it off, not

realizing that we may have just made a huge error.

* **Finally, evaluate the importance of being disciplined in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.**

Being disciplined in your commitment to quality is not necessarily a bad thing. Many people see it as being a control freak, which is not the case. It is called having pride in what you do and how it is done. Not cutting corners is very important, especially in the field of coding. When you start cutting corners, you can possibly leave major gaps in the system that make it vulnerable to attacks. To avoid technical debt, you must ensure that your code is written well and free of any errors or shortcuts. You don’t want to hand a “finished” project to a client and leave them vulnerable, because you did not want to put the time or effort to ensure that it either met or exceeded their expectations. This puts the company and your reputation at risk and is not worth it in the long run.