Elizabeth Robles

CS-320 Software Test Automation& QA

Southern New Hampshire University

December 5th, 2022

Reflections Report

Throughout this course I have tried my absolute best to align to the software requirements in the contact, task, and appointment services for the mobile application for our customer. In the contact service assignment, I followed all requirements for the Contact Class by including a contact ID string, a first and last name, a phone string field, and an address field. Then, in the Contact Service Class I made sure to only let the user include a unique string field, allow them to delete existing contacts, and update contact fields based on the contact ID. In the task service assignment, specifically the Task Class, I included a task ID, a string filed for the name, and a string field for a description. Then, in the Task Service Class I made the task ID unique, that way, repeating IDs could not be recreated, I allowed the user to delete tasks based on a specific ID and update the task field such as names and descriptions, also based off the unique ID. Lastly, in the appointment service assignment, categorically the Appointment Class, I again created an appointment ID, a date field using java.util.Date, and a description string describing the reason for appointment. In the Appointment Service class, I made it possible for the user to add appointments using a unique appointment ID that has not been used before, and delete appointments, also based off the ID. So overall, I did cover all software requirements needed.

I know that the overall quality of my JUnit tests were done well based off the fact that the overall coverage percentage for all services was 86%, with the appointment service covering 88.1%, the contact service covering 96.6%, and the task services covering 100% of the code. When running JUnit tests, it is important to aim for at least 80% coverage, and that was accomplished last week in project one. With coverage, it is also essential to write code that is technically sound. That can easily be done by using proper naming conventions for Java in all classes, packages, methods, and variables. It can also be done by commenting on your code to show the steps you took in fulfilling the requirements, it also helps other developers down the line to understand the methods you took to accomplish those tasks. An example of this was in the Contact Service Class for the update contact field,

//update contact fields

**public** **boolean** updateContactField(String contactId, String firstName, String lastName, String phone, String address)

I then proceeded to write code to allow the user to update the contact field per instructions. I made certain that my code was efficient by writing the least lines of code possible while meeting the requirements for the mobile app. I also included an ArrayList for all the services, **private** ArrayList<Contact> contactList;, **private** ArrayList<Task> taskList;, and **private** ArrayList<Appointment> appointmentList; to hold the data from all services, this ensures that the program runs smoothly using an in-memory data structure.

For this project, I used a dynamic testing technique. As I wrote the code for each of the classes, I proceeded to work on the tests for them as well. For example, after writing the code for the Appointment.java file, I immediately got to work on the AppointmentTest.java file to test the code I had just written. Using this dynamic technique makes sure that the software you are building works properly throughout the entire process. Within dynamic testing there is also functional testing that is used to verify that all requirements listed have been met, this can be done through JUnit tests, integration, system, and accepting testing, which was done during project one. Another software testing technique that I did not use for this project was static testing. Static testing is the process in which requirements are gathered before the project even begins, this is done by arranging walkthroughs, an analysis, interviews with users and stakeholders, inspections, and specifying requirements before the development process starts. This could be a beneficial option for Grand Strand Systems in the future as it can help us save money and catch errors early on.

Software development projects and situations will require different testing techniques based on the needs of the client. For this client, it was not necessary for us to do any static testing as it was a straightforward project, that is why unit testing worked so well, its also a cheap testing option that gives you results quickly. If, for example, we were to take on a software project for the US Military, we would likely need to spend hundreds of hours on static testing, and then take on a test-early-and-often approach for the remainder of the project. That way, our client knows we are delivering high quality, reliable, and secure software.

Throughout this project, I took considerable care to avoid as much danger as I possibly could. I was able to avoid danger by creating the JUnit test cases for all classes that I created. Its important to appreciate the complexity and interrelationships of the code I was testing because all the services offered something different to the user with different types of functionalities, however, they are all somewhat alike and can work together in an efficient way to allow the user to add contacts, create appointments and tasks all under one application.

As the developer, I can understand that there is a bias when it comes to testing your own code, one might often be so confident in their coding abilities, that when it comes to testing their own code, they believe extensive testing is not necessary, however, developers are only human, and they often make mistakes, so thinking as a tester and continuously looking for bugs is a smart way to catch any errors you might have missed otherwise. Lastly, as a software engineering professional, it is important to not cut corners when it comes to writing or testing code. To avoid technical debt as a practitioner in this field, I will emphasize the importance of delivering high quality software to the company and the client. A fast delivery of software is not always a good thing, often times, it just means the software is full of bugs and will need to be reworked on later with costly fixes. My best recommendation to avoid technical debt is to take on an agile methodology; with this methodology we can break up the project into several phases and allows us to work closely with the client, and deliver working software at each stage.

References

*What is dynamic testing? types, Techniques & Example*. Guru99. (2022, November 5). Retrieved December 5, 2022, from https://www.guru99.com/dynamic-testing.html