CS320 | 7-2 PROJECT TWO

DEONNE LUDWIG

06/18/2023

**SUMMARY**

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

Used the software requirements as an outline for testing. All three class components (Contact, Task, and Appointment) of the software requirements included a unique ID for that service.

**- Each object required a unique ID String that could not be longer than 10 characters and could not be null or updatable.** Achieved by setting object ID to final and setting the conditional checks where (objectID == null || objectID.length() > 10) invoked an error.

**- Each object also required various String fields that could not be longer than a set number of characters and could not be null.** Achieved by setting the conditional checks where (objectRequirement == null || objectRequirement.length() > xx) invoked an error.

**- One requirement unique to the Appointment class was a required appointment Date field that could not be in the past and could not be null.** Achieved by importing java.util.Date and use of before(new Date()) for past dates.

All three class components (Contact, Task, and Appointment) of the software requirements included the

**- requested ability to add, delete and update with a unique ID for that service.**

Achieved by using UUID and enabling adding, deleting, and updating per ID the variables to an array list.

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

By running the JUnit and Coverage testing to see what errors were discovered.

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer program

Description automatically generated with low confidenceContactService TaskService AppointmentService

A screenshot of a computer

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer program

Description automatically generated with low confidenceA screenshot of a computer program

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer program

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with low confidence

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

I tried to keep it as simple and concise as possible while still achieving the desired outcome. Using comments and trying to keep it very readable and condensing within coding standards like an example of declaring variables together: private String taskID, taskName, taskDescription;

**REFLECTION**

— **What were the software testing techniques that you employed for each of the milestones? Describe their characteristics using specific details.**

We have used Junit testing and configured assertions to validate the output of the code paths. As an example, to test the requirements of an “appointment description,” it could be set up as if (appointmentDescription == null || appointmentDescription.length() > 50) and would throw an exception if those criteria were not met. Then to test that output we set the values outside of those perimeters.

— **What are the other software testing techniques that you did not use for the milestones? Describe their characteristics using specific details. For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.**

Some other testing techniques that we did not use would include Acceptance testing where the applications would be required to be running while testing and check to make sure it meets the business requirements. Also, another example might be Performance testing where our system would be tested on how it performs under a workload and help to identify things like bottlenecks and peak traffic stability.

— **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.**

It seems extra caution should be employed when testing your own code to not overlook a vulnerability and/or create an unintentional bias. There becomes a direct symbiosis between the code and the testing code as they become intertwined to support each step of even the most complex code and the desired outcomes. An example of a direct relationship is if your project presents “The contact object shall have a required firstName String field that cannot be longer than 10 characters. The firstName field shall not be null.”, tests would be created for each parameter to test if the name were too long or if the name field was null as the preliminary tests and could get more involved and complex.

**— 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.**

Bias is definitely a concern in general for coding but especially if conducting testing on your own code. Assumptions and invisibility are only a couple that can be easily and unintentionally introduced. Tried to standardize the testing across the different classes of the project to avoid any biases for each individual component but also allow for the uniqueness of certain requirements of examples like unique IDs, and appointment dates. With limited intensive details about the project, it would be even more possible for extensive biases to exist in more areas.

**— 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.**

As a developer in the field, it seems taking the additional fractional time in the preventative measures of the development would far outweigh the cost and technical debt created by cutting corners. Hopefully over time, one would develop the coding skills that could produce not only complete and stable code but testing to support it in an expedited manner. Once the initial code and testing is developed the extra steps taken to test just a little further becomes minimal in comparison to the time and cost of a post-production mistake. Not only would I want to represent my company and myself well with quality deliverables, but it is also only good business to try to deliver the best products attainable to the client.