Adam Conger

CS 320

Project 2

Grand Strand System Summary and Reflection

It is essential to develop software to certain standards or requirements. If this is not done, then the application could have functionality that serves little purpose. For our project with Grand Strand Systems, requirements were addressed for each module component. For instance, many of the fields of data we need in our project were not allowed to be null. To satisfy this requirement, we coded JUnit testing exception lines to trigger an error if the field is null. Lines 32-36 of my TaskTest.java file address this example:

void testTaskIdNull() {

Assertions.*assertThrows*(IllegalArgumentException.class, () -> {

new Task(null, "Johnathon Jones Esq.", "This is a description that is fifty characters lng");

The action that’s happening here shows an assertion checking for the ID field and if it has a value or not. The other fields are the sample data strings that satisfy the requirements of our other value fields in Task.java (ID, Name, Description).

Overall, the test coverage quality of my Java files was good and could have been better if I knew more methods of JUnit testing. For example, I was not able to get the method for adding and removing and instance of the various ServiceTest.java files. I think despite that adversity my code for those files, for instance TaskServiceTest.java was efficient and technically sound:

*@Test*

void testAddTask() {

Task task = new Task("1234567890", "Johnathon Jones Esq.", "This is a description that is fifty characters lng");

TaskService addTask = TaskService.add(task);

*assertNotNull*(addTask);

Although it isn’t so clear in this document, the above method only requires three lines to create an instance of the Task object, and test that it contains data. A less efficient way would be to initialize Task objects that only initialized one field at a time. It would consume more memory and be at risk for security vulnerabilities.

In reflecting on the project approach, there were different software techniques available to use for this project. Ultimately, Boundary Value Analysis was primarily used. For most of our data requirements, we were looking at string values that had character limits while also not able to be null. Therefore the boundaries would be null or the particular max value for the strings (one assignment had the description be no more than 50 characters). The only other testing technique I think the project could benefit from is Decision Table testing, as we would want to make sure our application only proceeds once we have a unique ID filled in that is not null.

It is important to adopt an appropriate mindset when testing as a software developer. First, be cautious in that your application test coverage may be zero for the first few tries. Software development is an iterative approach, so in my case I always made sure I had at least one functioning method, even if it was one line, before I went onto the next test case. I tried to remove bias from testing my own code but I can admit that it may not always be effective. Especially when you may be testing a larger code base, it could get tedious to look through one’s own code base (‘yes, I remember looking at this method 300 times now’). Evaluating another code base will give the developer a sense of responsibility to do an adequate job. Ultimately, if a developer is not disciplined in all of this, then the software is at stake. Secure software is too easily taken for granted, and having a disciplined, critical approach to creating secure tests and using industry-standard testing techniques and cases will make one a great developer.