Al Castle

CS 320 (T3692)

**Project Two**

*Summary*

My whole approach when making the Grand Strand Systems was to make sure that all of the requirements that we were asked to do for the Contact Service, the Task Service and the Appointment Service objects. Some of these included things like the Contact Service to have requirements on the length of names and making sure that the users will be able to update IDs within the program. Another one of the main requirements came from the Appointment Service which required that “the appointment object shall have a required appointment Date field. The appointmentDate field cannot be in the past.” Finally, we had some requirements from the Task file that included “the task service shall be able to add tasks with a unique ID.”

To ensure that all the various features were met, we were asked to use Junit testing within eclipse. These tests help us find any errors or bugs that are located within a program and help the efficiency of a project by quickly finding them. We used primarily Junit testing in the first milestone we were asked to create a program for contacting the system, the requirements asked us to set up different methods to add update and delete contacts. The second milestone we were asked to test data structures and in the final milestone, the one we did in module (5), we were asked to test the functions that could only confirm the data that we gave it. To make sure that my code is efficient, for example the code that I have pictured below was a test to see if there is 50 characters’ then pass, if not fail. This test should satisfy the description requirement. What would make this efficient is that it takes from the description mutator in my task class and since we know that the part of code will check if the description is less than 50 characters’, the test will check in description and see if it passes or fails without using an unnecessary amount of code.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated with medium confidence

Attached below is the picture of my unit testing, this picture represents the percentage of the test covered. As you can see, I got an 80% or more coverage through each of the tests, which is what I believe we were looking for in the project guidelines. I’m not sure if 100 percent coverage is standard and if most companies will be strict when looking at how much coverage the test runs. What I liked about these tests were that you can go to the specific line of testing, and it will tell you what might be wrong with it. There are some missed instructions in all but my ContactTests file.

A screenshot of a computer

Description automatically generated

*Reflection*

Throughout this course we learned that static testing “test software without executing it.” This type of testing is supposed to find error in the code that will help make corrections easier and cheaper. Static testing is also used to find and remove any errors that may in the documentation of the project. This is important because if there are any errors in documentation, it will save a project time and money if these tests are done correctly. This isn’t something we really did as far as testing documentation, we obviously had to make sure that the names were as follows when testing our different classes, but we did do some Black Box (also known as specification based) techniques come mainly from test cases “directly from the specification or form some other kind of model of that the system should do.” We were given a set of instructions that each class needed to have and then we implemented tests form those classes.

This project was a step up from other projects that we have done in the past courses. The main thing I would get held up on is making sure that the proper imports were added and also making sure that the name of the Classes were properly inputted in the tests. This sounds like an easy thing to make sure of, but I got held up on this early in the project. It is important to appreciate of the interrelationships of the code because each class object that we worked with had specific requirements to be tested such as the length of the characters.

Bias can play a factor in pretty much everything and it is no different when reviewing your own code. If you are proud of your work but there are holes in it that could make it better, it is always good to put any ego aside and do what is best for the project. If that means that you have to go back and fix something, then you should do that. Someone that is testing their own code might not put it through strenuous testing such as when looking at how the app actually preforms on someone’s hardware. As we seen in the reading throughout the course, even if a program runs, it could run bad once implemented on someone’s computer or phone.

It is also important for a software developer to make sure that they write code that is readable, scalable, and reliable. The fewer problems that a program has, the better secure it will be, and this is important for a project like we worked on because users will be putting in their names and any other information such as an email address. As we seen in the module 7 discussion there are many ways that software can bug and cause catastrophic failure to a company and making sure that code is overseen by a supervisor, or a team can make sure that failures like that don’t happen. If we look at the contact service, we can see that each contact has a unique ID, and we do this to make sure that it can’t be interchanged with one another to avoid any errors. I learned a lot about testing and how to write code that you can change. More often than not you will have errors and things that you need to go back and change, and testing is a good way to do this.

**Sources**

* *Previous work done in modules 3-6*
* *Rajkumar, et al. “Static Testing and Dynamic Testing: What Are the Differences.” Software Testing Help, 3 Jan. 2022, https://www.softwaretestinghelp.com/static-testing-and-dynamic-testing-difference/.*