Dominic Aguirre

CS320

12/11/2022

Project Two

With this assignment I tried my best to make sure that the application would work given as many inputs as possible. While also making sure that all the runs pass without mistake. Luckily, I was able to succeed at this goal thought testing on all three features. When running each Junit, all the run’s pass meaning that the testing is successful.

I think my software was very good in the end and worked efficiently and effectively. One of the example of this can be found in the TaskService,Java, to add a new task I made sure that the task didn’t already exist before adding the new task with the relevant information.

Graphical user interface, text, application, email

Description automatically generated

I minimized the amount of code needed and it came out very efficiently with no unneeded hassle.

Software testing techniques that I user were testing to make sure the classes I created, this can be seen in the Appointment testing the best. I make sure to thoroughly check each part of the appointment from the description being null to the description being too long. All while making sure that it passes each test to ensure that the program works properly.

A software testing technique I didn’t use would be white box testing, white box testing has the tester testing the code of the software rather than the program itself. This often points to finding security holes, expected outputs, functioning loops and so on. This can be useful, however, for making sure your program is secure and very efficient by making sure only what’s needed is there and working.

When working with large projects that can be thousands of lines of codes its important that you keep a level head and ensure that you are focusing on one task at a time. When working on this project and more specifically the testing, I tried to make sure I was focusing on the testing itself in the moment rather than all the other issues that could come up. Making sure not to test everything all at once is another way I tried to make sure that I kept from overcomplicating things.

When testing, I try to make sure I tested as many different ways that I could to make sure that I could eliminate bias as much as possible. Bias can come from writing inputs the same way as you intended it to be input but not all users might do it the same way as you.

Graphical user interface, text, application

Description automatically generated

When testing the appointments, I made sure that I checked all aspects of it several different ways under different circumstances to ensure that it worked as intended.

Comments are essential for any good software engineering. Comments allow others to understand your code without having to read every single line of code in the software. When you move onto another project someone else might oversee maintaining and updating the existing code and if you don’t use comments, it can be very difficult for someone new to come in and understand what your intentions were for different lines of code. With this its important to also make sure that you make everything to standard rather than cut corners because while it may work now it might hinder future versions of the software depending on how you might have cut a corner. You want to ensure that you code is the best it can be so that the software runs and functions the best it can. This can all help lower technical debt by ensuring that more time won’t need to be spent later on something that could have been done right the first time. Technical debt often occurs when the first iteration is sloppier as it takes longer to go back to fix these mistakes depending on how far along you are. Core systems could rely on the lazy code making it so lots has to be re-written costing more time.