Cody Smith

10/19/2025

CS-320

Project 2

When implementing the unit tests for this project, I made sure to refer to the program requirements every step of the way. By consistently checking to make sure I was testing for exactly what the product needed to include, it was very simple to make sure that my tests covered everything that was required. With over 80% coverage with the Junit coverage checker, I am confident that my code tests cover the majority of the product code. To ensure my code was technically sound I implemented the checks for the various input requirements in the setters for the variable itself, that way any time the data is changed, it first runs the check to ensure the input is valid. A screen shot of a computer code

AI-generated content may be incorrect.

To ensure my code is efficient I made sure to only include checks for the requirements outlined in the project requirements, and nothing more, in order to ensure we only had the code that was necessary to meet the requirements.

Software testing techniques I used in this project include black-box oriented testing, meaning I had a strong focus on testing what the system does, not how it does it. I treat each class or method as a sealed unit, feeding it inputs and checking that the outputs match the expectations. The practical uses and implications of this black-box testing is that is ensures the software behaves according to user and business requirements, not developer assumptions. These tests also provides a user centric testing, where we are not relying on how the code functions, and just testing that it does indeed function, it is in a way simulating the user experience, thus ensuring a positive user experience.

Some testing that I did not implement include testing to make sure that a value can be assigned to a variable that is the exact character limit in place. While I did check if it will not assign a value that is over the character limit, I never tested for the exact amount. I also did not check to make sure that the unique ID’s that needed to be created, were actually unique, or what would happen if you tried to make an object with an existing ID.

The mindset I made sure to implement during this project was a thorough one, as I wanted to make sure that I was hitting all of the requirements of the project, and testing to make sure those requirements were met. In order to be that thorough, I kept the requirements page where I could see it as I worked on this project, and referred back to it often. With this constant double checking the requirements, it helped me make sure that the project variables all we constructed properly and the tests double checked that.

One way that I could have helped to better limit the bias of my testing was if I did more testing for situations where the code wont work, instead of just testing that it did work. I did have a bit of an assumption that the code was going to work properly, which I can see limited the scope of the testing.