Matthew Allen

Professor Omar Toledo Lopez

Software Test Automation

02/28/2024

Project 2

My testing approach was guided by the given requirements. For Example, the requirements stated that Contact class should be able to get and set the first and last name, phone number, and address. When setting these fields, they should not be null and should not exceed a specific length. First, I tested that you could get each value and that it held the expected value. Next, I tested that each setter would throw an exception if the maximum length was exceeded. With the ContactService class, I tested to make sure a new Contact object could be added to the list of Contacts and that it held the proper values. These were the principals I used for each of the other classes. Use the requirements to identify what tests need to be written.

To ensure that my Junit tests were effective on the basis of coverage percentage, I tested every method. Each method was tested with input that was intended to work correctly and input that should throw an error. When updating one of the objects in the object list, I checked that you could correctly update the object and that an object would throw if given improper data.

To ensure that my code was technically sound, I followed good coding practices. In each service class, the methods will check the arguments before they are sent to the called method in the Contact, Task, or Appointment classes. Also, the only fields that are public are the ones that will need to be accessed by an outside class, otherwise they are private.

In this project, I employed dynamic and static testing. Static testing is a way of checking for software defects without executing the code. It is done early on in the software development lifecycle and improves the quality of the software. Static testing can be done by reviewing the codebase and supporting documents, or automatically by using tools that analyze code. Dynamic testing in software is a way of checking for software defects by running the code and checking the behavior and output. It is used to check the software’s functionality and see how it behaves when given different inputs and conditions. For these projects, I did not use integration testing. Integration testing is used to test cross-class functionality, where unit tests are used to test the functionality of individual methods.

Static testing and dynamic testing are different in many ways. In static testing, the code is never run. The code is analyzed to improve the quality of its readability and functionality. In dynamic testing, the code is run to ensure the output is what is expected when given different inputs. They are both important because they can help improve the quality of the code and ensure

To get into the mindset of a software tester, I considered the requirements and tried to push the limits what the code should accept, and not accept. To do this, I gave edge case arguments and invalid arguments to the different methods, as well as tested the valid inputs. I can imagine that there is potential for bias when testing your own code. You would be able to write tests that cover the bare minimum requirements for code coverage, and give input that would pass or fail as you wanted. Cutting corners can lead to software quality taking a huge hit. Technical debt will accumulate, leading to maintenance challenges and increased defect rates in the future. It is important to stay on top of tests, code coverage, and meeting requirements.