CS 320 Project Two

Ryan Helferich

Southern New Hampshire University

CS 320

Two major pieces to the software development cycle in an application consist of testing and quality assurance for the program. While there are various types of testing for applications depending on the task at hand, in this application we use white box-based testing and functional based testing for the tasks, appointments and contacts features of the application. The white box testing dives directly into the source code with immediate tests to check for the data entering the software while also checking for the outcome’s success in validity, whereas our functional testing identified and handled the functionality of the project to ensure it operates correctly.

The application consisted of different tasks with a single goal in mind, to develop a new mobile application for customers to handle contacts, tasks, and appointments. One essential technique used in the development of this project was to understand the requirements for the project. This consisted of handling and organizing each variation of an appointment, contact or tasks to ensure it operates as expected. The requirements process consisted of listing out each specific detail of the individual class before code was created, allowing the perspective of an initial prototype of the class and its expectations.

An additional technique used in the application was to whiteboard and layout how the individual classes and tests will be working together to ensure the project works correctly. Although this is closely related to understanding the requirements, it’s one thing to analyze requirements in comparison to laying out a basic design that fulfills those given requirements. Another used was organizational based testing of each class. This technique consisted of breaking down each case and exception that can occur in the classes, while creating a test case in the contrary to find any faults in the code.

After writing the Contact, Tasks and Service classes for the program, I began writing the test for each class to reflect the software requirements for the class. I began by making a note of what each class is required to have and allow, while keeping the test classes in line with the main classes. The task and task service class begin the setting methods with an exception case checking that the (for example) “SetName” method would only be setting a name that is shorter than 20 characters and not empty. We checked if the name was null and counted the length of the name string. If everything passed, we would continue and set the name. This same method was used in every setter and effectively utilized the exceptions throw handler which would also not allow an invalid value to be entered. The theory behind this is that we will check for the worst case possible and ensure it does not occur, before changing any values at all.

The Junit test for each class was set to work hand in hand with the exceptions used in the setters of the original classes being tested. Each test will throw a value that is not meeting requirements to try to get the original classes to break and allow an invalid name, description, or id into the object. These Junit tests can be considered entirely effective and valid based on the percentage of the test passing at 100%. If there was a change in the requirements for the program, we would also need to change the values within the classes and tests to maintain functionality. Ensuring the code was technically sound by repeating the test with various values was essential to determine which test did not meet the requirements. In the setup method for the Task Test (line 15) I created a structured format that was replicable to the Task Service Test (line 19) to ensure each value was being tested accurately the same, and in the correct order. By organizing the setup in the same format across both tests, we can ensure that if we were to add a new value to the task test, we would be able to not see it clearly in the task service test. To ensure the code was written efficiently, we can refer to the idea of using the exceptions as a first case detection when setting the name, description, and id. I initially begin to check for the requirements, then carry on this the rest of the code. This prevents issues with attempting to check illegal cases later in the software and increases performance as the program does not have to make additional decisions if the first exception is thrown.

The functional portion of our testing technique was solely based on the concept of whether the code was functioning how it should be per the requirements of the project. After analyzing the initial test used to determine if the class values were being handled with exceptions and tests, we could then use functional testing to track the outputs of each class created to ensure everything was held together. This process was used by creating classes manually through tests with various pieces of information being thrown at it, simulating the usage of the application by a consumer. By doing so, were able to see each objects information from its class upon creation, aligning it to the requirements the client or consumer would expect.

The integration portion of the testing we used was to determine how the classes are integrating with each other. I was able to generate a few tests separately from the application to track each classes core values and determine if they were being handle the same across each one. The specific ID information, descriptions and naming systems within the classes were nearly replicated from one another, ensuring the application would be able to integrate each class with the same information correction.

The white box, functional, integration and analysis testing techniques were solely used in this project as they prove to be an essential foundation for a basic application to handle possible exceptions and faults that may arise. Although there are additional tests that can be run on this application, the need for it was limited as this application had three main features that worked close together with similar values. The tests used were specific in identifying the information being provided by the consumer or user, in relation to the requirements and limitations of the project itself. Practical usage of these testing techniques within other projects may vary depending on what type or forms of data are being handled, as well as the potential security importance of the application. For example, banking applications or medical systems must be tested in every way possible as they could essentially destroy a person’s life through the act of a single mistake or error. The level of the project and time frame given determines the outcome of how tests can be committed. One software testing technique that was not used explicitly as much as it should be the use of multiple test cases for each individual requirement. The current test cases apply and work for the requirements of the project. These tests allow the application to work as required per the class but do not conform to every single possibility or exception that could occur within the code. Extensive testing in this case would be to handle exposed inputs that could cause an error or glitch down the road, but within the given time frame and requirements, these further tests were pushed off.

After completing this project, it was essential to reflect on the entire design process and mentality used as a software engineer/tester. Being able to analyze exact requirements before handling this project was a huge step that most developers don’t experience, especially when the projects are sources of side knowledge of income. Considering this project was for a ‘company’ that required specific features to be made, the mindset of the project entirely was changed to accompany the application as a solution rather than a test. On a larger scale, bias within or between software engineers can vary depending on how one person might think the data of the application is being handled properly or incorrectly. Given that I was the only person working on this application, another software engineer may determine that the tests on the classes could go deeper in terms of handling any potential security issue that would occur (like potential injections directly into the classes via user input), leading to vulnerabilities. Discipline within programming must be consistent to produce a quality product for the consumer. One person’s way of organizing code can vary to another, by following standard ethics and organizational standards in the field, this can be avoided. In the field and future products, I remain on the concept that the code must not only be consistent with the consumers needs but also reflect a well-structured application that is safe for any user to access.