Project Two

Shawn Whittaker

Southern New Hampshire University

There are various approaches and techniques employed to test an application’s behavior and compare it to the expected behavior. These strategies are known as software testing techniques or methods, which can include functional and non-functional testing, white and black box testing, and unit up to system wide testing.

Functional testing is used to verify each function and method of the software operates within the required specifications. Every function of the system is tested, using inputs and verifying the outputs to the expected results. Functional testing checks for User Interface, APIs, databases, security, and client/server applications. Non-functional testing checks aspects such as performance, usability, and reliability of a system. They do not check the coding to ensure function, they are used to ensure the program can meet the prescribed specifications.

Similar to functional testing, we have white box testing. White box testing checks the internal functions of the system and is often performed during the functional tests. These include testing code statements, branches, paths, and conditions, to ensure the program logic is sound. Black box testing examines the behavior of the software, but from an external end-user point of view. This means black box testers do not require programming knowledge, as their main goal is to test the software behavior to ensure things cannot go wrong, whereas white box testing ensures the program acts correctly. The distinction between acting correct and not being able to act incorrect is the main difference between black and white box testing.

Unit testing is the smallest fragment of working code that can be tested. This usually includes a single working method, to ensure piece by piece the program can run correctly. From there we have integration testing, which tests each method we start combining them together to ensure each function can work properly when added to more methods and not the new methods do not break the current functioning methods, which is also known as regression testing – making sure each method added does not break the current function of the program. From here we have system testing, which is a black box testing method used to evaluate the completely integrated system and ensures it meets specified requirements.

For each of the milestones, we have implemented JUnit testing, which is a java Unit test able to automatically check the functions of each of our methods to ensure function ability. As mentioned above, the unit test verifies the smallest logical code fragments or methods piece by piece, ensuring the code operates as expected and does not interfere with other fragments as they join into the system.

For our Contact Service assignment, we created both a Contact class and a Contact Service class. This enables us to define the contact ID number, first and last name, phone number and address in the Contact class, and then the ability to add, update, and delete the defined contact using the Contact Service class. In our JUnit testing, we designed a Contact test, as well as the Contact Service Test. The Contact test asserts a value into each method to test the contact ID number, the first and last name, phone number and address. The Contact Service test examines the addition, updating, and deleting features in the Contact Service test one method at a time, to verify contact strings can be manipulated correctly. The Contact test covers 88% of the code, and the Contact Service test covers 93.5% of the code.

In our Task assignment, we created both Task and Task Service classes. In the Task class, we defined task ID number, name, and a description of that task. The Task Service class allows us to make a new task, update, or delete a task. Our JUnit tests for the Task class manually inserts a task ID number, first name, last name, and a description string. The Task Service test verifies the methods for adding, updating, and removing a Task are valid. Both the Task and Task Service tests also check if any value is too long or null, as many hackers try to brute force their way into the code using these inputs. The Task test covers 84% of the code, and the Task Service test covers 93% of the code.

Similar to our first two assignments, our Appointment Service assignment is comprised of two classes, the Appointment and Appointment Service, as well as two JUnit test cases, the Appointment test, and the Appointment Service test. The Appointment class defines an appointment ID number, appointment date, and the description of that appointment. The Appointment Service class allows us the ability to create, update, and remove an appointment, but also gives us the function to search for an appointment. The Appointment test case asserts values into the Appointment ID number, the appointment date, and the description of that appointment. The Appointment Service test case verifies the service class has the ability to add, create, remove, and search for an appointment, and validates these methods. Both test cases check for valid string length and null inputs. The Appointment test covers 86% of the code, and the Appointment Service test covers 96% of the code.

Caution was employed in the addition of checking for valid string length and null inputs, as these are often exploited by hackers. A bias a software developer often has the trying to prove their code runs correctly. This often leads to overlooking the faults when the code may not work. Developers prove the code runs correctly, but QA testers must ensure the code cannot run incorrectly. This distinction makes a big difference in the quality and rigor tests should validate code. Many software developers look to cut corners as they write hundreds of thousands of lines of code. It is important to be disciplined to not cut corners, copy and pasting from Stack Overflow may get you the desired effect you are looking for, but that does not mean your code will not run incorrectly. Again, running correctly is very different from not being able to run incorrectly. The inability to run incorrectly means no bugs or glitches can be found, and the software is safe and secure.

JUnit Tutorial For Beginners. (2021). Software Testing Help. https://www.softwaretestinghelp.com/junit-tutorial/