Southern New Hampshire University

CS 320 Software Test Automation

Module 4 Journal

George M Harrison Jr.

In this week and last weeks modules, the Task Service and Contact Services tests, were short sections of code was required for a program by a client to meet specifications for both classes to be written. Once the code was written, test case scripts were then written for each of the classes. The approach taken to ensure the code was aligned with software requirements was tedious but, I decided to write the error checking code to ensure that all character requirements were met. For example, in the Contact.java requirements the firstName could be no greater than 10 characters. This was written as such in the code to ensure this specification was met:

Graphical user interface, text

Description automatically generated

When running the J Unit test on the Contact Service and Tast service code. Junit test and Coverage Test were conducted to measure how much of the code ran during the unit test. As the completed test indicates for the ContactTest.java class file, the JUnit test was complete quickly (0.113 seconds) with both file run, no error or failures. This would indicate that code is intact and error free however, this J Unit tests do not cover incorrect implementations. Unfortunately, the code coverage was not very high for the coverage test. In order to tests to be considered to be meeting the requirements of the client it has to pass certain productions and functionality tests but, in the coverage, test we need at least 80% to be considered acceptable. This code still requires some work.

Graphical user interface, text, application

Description automatically generated

The strategy used to in the code testing and writing was to for hypothetical data entries. Initially, the tests on involved those conditions which were proper and correctly entered. These would be tested against the assertions. Here the data fields are tested and given data that is with in the specifications.

Text

Description automatically generated with low confidence

Next, data is tested again. This time we ensure that all the field are not within specifications and attempt to forced them to fail. We want it to fail therefore, we set them to too long for the required field. This is how Unit Testing ensures technically sound and efficient code.

Graphical user interface, text, application, Word

Description automatically generated