Nishesh Shukla
101994823
CSCI 3308
Elizabeth Boese
HW3: Unit Testing

**Unit Testing Code:**

Avatar: Lion
ID: 44CFAD
Link: http://54.153.4.164/kata/edit/44CFAD?avatar=lion

**Essay:** *Unit, Integration, Functional, and Acceptance Testing Strategies*

*1.*

In any type of software (application, program, etc) testing plays a main role in the success of the software. There are many ways to test software, but from business perspective, the least time consuming/less cost of testing would be the ideal testing strategy. Unit, Integration, Functional, and Acceptance Testing are all great strategies.

Unit testing is a process in which the smallest testable parts of an application are being tested (also called units) individually and independently examined for operation and functionality. The main goal of it is take the smallest piece of the entire code, excluding it from the rest, and then determine whether it behaves exactly as intended/expected. If each unit satisfy this test, then combined it theoretically should work. It allows the elimination of dependencies on other modules during testing, also allows testers to test a single unit without having to wait for the entire application to be completed. It is a lot easier and cost efficient to identify the problem and have it fixed immediately that after the entire application is fulfilled. It also leads into enhancing the code architecture/structure. A major downside for this type of testing could be that it could get redundant and very time consuming having to each particular unit of the entire application.

Integration testing is a process in which the integration or interfaces between components, interactions of different parts of the system are being tested. It is a test that in commonly conducted right after two different components of a system/application are merged. There are three types of integration testing, Big Bang, Top-Down, and Bottom-up testing. Big bang tests every component/module after integrating them simultaneously and then evaluating it as a whole. By doing so it makes sure that each of the components function as a whole. Although, if there is a failure it is difficult to trace as every component was merged simultaneously. Top-Down testing tests everything from Top to bottom following the architectural structure of the program. It is beneficial as it is very consistent due to the fact that it is performed in an environment that is very close to real time usage of the application/software. However, it too is tested at the very end of the creation of the program, hence, if there is an issue it will be difficult to trace. Bottom-up testing is the opposite of Top-Down testing; one great benefit though, is that the testing can be performed while development (making sure the customer specifications are being met as well). A great testing strategy to run when more than one component in software is being created and merged.

Functional testing is a Quality Assurance process involving a type of black box testing; it bases its test cases on the specifications of the component under test. This

Nishesh Shukla
101994823
CSCI 3308
Elizabeth Boese
HW3: Unit Testing
allows the internal logic of the software to be tested. It is a test that requires giving an input and testing the output, works like a compiler after write a piece of code. This type of testing doesn't prioritize how the process occurs but the results of the process. It also verifies the program by checking it against the customer specifications/design documents. A few benefits of such a test could be that it simulates an actual real time usage and doesn't make any structure assumptions. On the other hand, it has a high potential of missing logic errors in the software and can become redundant.

Lastly, Acceptance Testing is more oriented around the user/customer testing. After the conduction of Unit/Functional/Integration tests, this test takes place. The main goal and focus for this test is to establish confidence in the system and to focus on functionality, validating it fit to use. There are 3 types of Acceptance Testing, Operational, Contract, and Compliance. Operational (also known as production acceptance test) focuses on verifying whether the system met the requirements for operation. Contract focuses on performing a test against the contract's criteria, mainly done in custom-made software. Compliance (Regulation acceptance test) focuses on validating the software by making affirming it is meeting all the legal, government, and safety regulations.

Overall Unit, Integration, Functional, and Acceptance Testing are all very crucial to the success of a particular software/application and crucial for a company to use these strategies. Each of them play a different role in testing the overall product, even though it might seem redundant, it is a necessary method/precaution for the success of the software.

*2.*

A company that has many unit tests covering nearly all of code base versus a company that doesn't practice writing test codes has a significant advantage. The reason why a test code must be written is so that the company can verify that the program/software is efficiently operational according to the needs of the customer. If a test code isn't written, then the company is putting all of its faith in the assumption that the software meets the requirements/criteria of the customer. The company without the test code also faces a great deal of loosing the customer, getting into legal issues, and loosing reputation. I, personally, would feel more comfortable with a company that takes their time to write test cases covering almost every single base of the software before releasing it to the public/customer than a company that gambles on the success of the software and releasing it to the customer/public without testing it.