Name : Muhammad Shahzeb

Roll # : Sp22-bse-073

Assignment : Software Concepts Assignment

Submitted to : Miss Mehwish Sabir

Date : 7/5 /2023

Testing Methodologies

White Box Testing:

White Box Testing also referred to as Clear Box or Structural testing, White Box testing involves the tester having complete knowledge of the underlying organization, programming, and coding of the system under test. The tester has access to the algorithms, design specifications, and source code. Making ensuring that the internal parts and logic work properly is the aim. Statement coverage, branch coverage, path coverage, and code reviews are examples of white box testing methodologies.

Example: Let's say you are testing a login procedure. White Box testing allows you to examine the login module's source code, determine whether all potential code paths are covered, and confirm that the login credentials are properly validated against the database.

Black Box Testing:

In this method, the tester does not have access to the internal workings of the system being tested. Without taking into account the internal code or design, the tester focuses simply on the inputs and anticipated results. This approach places a focus on testing the system from the viewpoint of the end user. Equivalence partitioning, boundary value analysis, and use case testing are methods applied in black box testing.

Example: Think of yourself as a web application tester. Black Box testing involves interacting with the user interface of the application, providing different inputs, and then watching the results. Without knowing how the program is internally implemented, you would test scenarios like providing valid and invalid data, looking for error messages, and confirming that the application acts as expected.

Grey Box Testing:

The Grey Box testing methodology combines the White Box and Black Box testing approaches. The internal organization and implementation of the system are only partially known to testers. They might have access to specific design documents, database schemas, or only have rudimentary programming experience. Grey Box testing aids in locating and focusing on particular regions that need testing while taking the system into account.

Example: Consider the following scenario: You are evaluating an online store. You are familiar with the application's architecture and database layout when performing Grey Box testing. Without having complete access to the underlying code, you can run tests based on that understanding, such as determining whether the user's cart data is correctly recorded in the database.

Conclusion:

In conclusion, the degree of access and knowledge that testers have about the internal workings of the system varies between White Box, Black Box, and Grey Box testing approaches. Black Box tests from the standpoint of the end user, White Box focuses on the internal structure, and Grey Box integrates both methods.

Question 2: Difference between Unit, integration, system and regression testing explain with examples…?

Unit Testing:

Testing individual units, such as functions, methods, or classes, one at a time, is known as unit testing. The goal is to confirm that each unit performs according to specification and behaves as intended. Developers often carry it out at the coding stage. Unit tests make debugging simpler and aid in the early detection of issues.

Example: Consider creating a calculator application as an example. Writing tests for each mathematical operation (such as addition and subtraction) during unit testing will help to guarantee that the correct results are produced for a variety of inputs. Each unit test would concentrate on a particular function and independently verify its behavior.

Integration Testing:

Testing the interactions and communications between various system modules or components is known as integration testing. It seeks to find any flaws that may appear during integration of these components and make sure they function properly. At many levels, such as module integration, API integration, or system integration, integration testing can be done.

Example: Think about a website that has a front end and a back end. You would test the interactions between the frontend and the backend during integration testing. This entails determining if the frontend correctly submits queries to the backend API and whether the backend returns the anticipated data in response. It guarantees that the parts work together effortlessly.

System Testing:

System testing is done on a fully integrated system to determine whether or not it complies with the requirements. It confirms that every aspect of the system operates as expected and up to user expectations. After integration testing, system testing is carried out, covering end-to-end scenarios such as user interfaces, data handling, and communications with external systems.

Example: Consider the following scenario: You created an internet retail platform. The entire flow, including user registration, product search, adding goods to the cart, the checkout process, and making sure purchases are handled properly, would be tested as part of the system testing process. It makes sure the system works together as a whole.

Regression Testing:

Regression testing is done to make sure that recent software upgrades or improvements don't introduce new flaws or break functionality that was previously working. Retesting the current features is necessary to confirm that they are still correct after modifications have been made. Regression testing supports long-term program stability and quality maintenance.

Consider the following scenario: After fixing an issue in your calculator application, regression testing is now necessary. To make sure that the bug patch did not create any new problems or disrupt any previously functioning activities, you would rerun all the current unit tests, integration tests, and system tests.

Conclusion:

Regression testing makes sure that changes or bug fixes do not negatively affect the functionality already present. Unit testing, on the other hand, focuses on testing individual units of code. Integration testing confirms the interaction between various components. In order to guarantee the overall quality and dependability of software systems, each sort of testing is quite important.