Assignment – 02

SOFTWARE TESTING STYLES

1.UNIT TESTING: Unit testing is a software testing method where individual parts of a program, called units, are tested in isolation. A unit is usually the smallest testable piece of code, like a function, method, or class. The goal of unit testing is to make sure that each part of the program works exactly as expected.Developers usually write unit tests as they build the code, often using a testing framework like JUnit (for Java), PyTest (for Python), or Jest (for JavaScript). Each unit test is designed to check a specific function with various inputs to confirm that it produces the correct output. If a unit test fails, it indicates that the code being tested has a bug.One major benefit of unit testing is that it catches problems early. Since tests are run frequently during development, developers can quickly spot and fix mistakes before they grow into bigger issues. Unit tests also make it safer to change or refactor code later because they serve as a safety net—if something breaks, the tests will reveal it immediately.Unit testing is a core part of Test-Driven Development (TDD), where developers write tests before writing the actual code. This approach ensures the code is written to meet specific requirements right from the start.Good unit tests are fast, repeatable, and independent. They should not depend on external systems like databases or APIs; if needed, mock objects or stubs are used to simulate those dependencies.

2. INTEGRATION TESTING: Integration testing is a type of software testing where individual units, modules, or components of a system are combined and tested together. The main goal is to check if these different parts work correctly when integrated and to identify any issues in their interactions.

Integration testing ensures that data flows properly between modules and that combined functionality works as expected. Problems often arise when separate

units, which work well individually, fail when put together due to interface mismatches, data handling errors, or communication problems.

There are several approaches to integration testing:

*.Big Bang Integration: All components are combined at once and tested together. It’s fast but harder to debug because if something goes wrong, it’s tough to find where the problem is.

*.Top-Down Integration: Testing starts from the top modules and moves downward. It often uses "stubs" (fake modules) to simulate lower parts that aren’t ready yet.

*.Bottom-Up Integration: Testing begins with the lower-level modules first, moving upward. It uses "drivers" (temporary programs) to simulate higher-level modules.

*.Sandwich/Hybrid Integration: Combines both top-down and bottom-up methods for more flexibility.

Integration testing can be done manually or with automation tools. It's especially important in complex systems where multiple teams develop different parts independently.

3.BLACK BOX TESTING: Black Box Testing is a software testing method where the internal structure, design, or code of the program is not known to the tester. Instead, the focus is only on inputs and outputs: testers provide inputs and observe the outputs to see if the software behaves as expected. It's called a "black box" because the internal workings are hidden, like looking at a sealed box—you can see what goes in and what comes out, but not what happens inside.This type of testing is mainly used to validate the functionality of the system against the specified requirements. Testers do not concern themselves with how the software processes the input, only with whether it produces the correct output. As a result, black box testing can be performed by both QA testers and end-users, not just developers.

There are different techniques used in black box testing, such as

*.Equivalence Partitioning: Dividing input data into valid and invalid partitions and testing one example from each.

*.Boundary Value Analysis: Focusing on the edges of input ranges.

*.Decision Table Testing: Using tables to show different combinations of inputs and their corresponding outputs.

*.State Transition Testing: Checking the system's behavior when it changes from one state to another.

Common types of black box testing include functional testing, system testing, acceptance testing, and sometimes even regression testing.

4.WHITE BOX TESTING: White Box Testing is a software testing technique where the tester has full knowledge of the internal structure, design, and code of the application being tested. Unlike black box testing, which focuses only on inputs and outputs, white box testing allows the tester to look "inside" the system and understand how it works.In white box testing, the tester examines the logic, control flow, and data flow of the program to design test cases. This ensures that all possible paths, loops, conditions, and statements are properly tested. The main goal is to verify the internal operations of an application and to identify hidden errors or bugs that may not be obvious from the outside.

Common techniques in white box testing include:

*.Statement Coverage: Ensuring every line of code is executed at least once.

*.Branch Coverage: Ensuring each possible branch (true/false decisions) is tested.

*.Path Coverage: Testing all possible paths through the code.

*.Loop Testing: Checking that loops function correctly under different conditions.

Developers often perform white box testing during unit testing, but testers with programming knowledge can also carry it out at integration and system levels. Tools like JUnit (for Java) and NUnit (for .NET) help automate white box testing.

5.REGRESSION TESTING: Regression testing is a type of software testing used to make sure that recent code changes haven't negatively affected existing features. When developers add new features, fix bugs, or make updates, there's always a risk that old parts of the software might break. Regression testing helps catch those issues early.The main goal is to confirm that the software still behaves as expected. This is done by re-running existing test cases that were previously successful. If any test fails during regression testing, it signals that the recent changes caused a problem.

There are two main ways to do regression testing:

1.Manual Regression Testing: Testers manually re-check key functionalities.

2.Automated Regression Testing: Scripts automatically run large sets of tests, which saves time and effort, especially for big projects.

Typically, regression testing happens after functional testing and before the software is released. Teams often run it regularly, especially in Agile development, where frequent updates are made.

Key benefits of regression testing:

*.Ensures stability after updates

*.Builds confidence in software quality

*.Helps catch hidden bugs early

6.LOAD TESTING: Load testing is a type of performance testing that checks how a system behaves under expected or peak user loads. The goal is to ensure that the application can handle real-world usage without performance issues like slow responses, crashes, or downtime. In simple terms, it's like putting a system under pressure to see if it can perform well when many users are using

it at the same time.During a load test, a specific number of virtual users are simulated to interact with the application. These users perform typical tasks, such as logging in, searching, or making transactions. Testers gradually increase the load until they reach the desired level which could be the normal expected traffic or even higher to find the breaking point.

Load testing helps identify several key things: response times, server throughput, system resource usage (like CPU and memory), and any bottlenecks or weak points. If issues are found, developers can optimize the system before it goes live, saving time, money, and customer frustration.It’s important to design realistic load tests based on actual user behavior and traffic patterns. Tools like Apache JMeter, LoadRunner, or Locust are often used to automate the process and collect detailed performance data.