SOFTWARE TESTINGS

1. **Manual Testing:**

- Manual testing is the process of manually executing test cases without using any automation tools. Testers play the role of end users and verify that the software behaves as expected. The goal is to identify bugs, errors, or missing requirements.

Why Manual Testing?

- Automation can't cover everything — like UI/UX issues, usability testing, and exploratory testing.
- Some applications (especially early-stage or rapidly changing ones) are not stable enough for automation.
- Cost-effective for short-term projects.
- Good for ad-hoc or exploratory testing, where formal scripts aren't available.

Phases/Steps in Manual Testing

1. Requirement Analysis
    - Understand what the application is supposed to do.
2. Test Planning
    - Define the testing strategy, resources, schedule, and deliverables.
3. Test Case Design
    - Write detailed test cases describing inputs, actions, and expected outcomes.
4. Environment Setup
    - Prepare hardware, software, network, and tools needed.
5. Test Execution
    - Execute test cases manually and record the results.
6. Defect Reporting
    - If a bug is found, report it using tools like Jira, Bugzilla, etc.

7. Retesting and Regression Testing

- After fixes, re-test to verify the defect is fixed and check other areas are unaffected.

8. Test Closure

- Prepare a test closure report summarizing testing activities and results.

When to Prefer Manual Testing?

- When the project is short-term and doesn't require repeated execution.
- When exploratory, usability, or ad-hoc testing is needed.
- When requirements are frequently changing.
- During initial stages of development.

2. Automation Testing

Automation testing is the process of using specialized tools and scripts to execute tests automatically, compare actual outcomes with expected outcomes, and report results — without human intervention after test script creation.

In simple words: Automation = Using software to test software.

Why Automation Testing?

- Faster execution of tests compared to manual.
- Repeatable — good for regression, load, and performance testing.
- More reliable — no human errors in repetitive tasks.
- Saves time and cost in the long run.
- Enables continuous testing in CI/CD pipelines (DevOps).

3.Unit Testing

- Unit testing is a type of software testing where individual units/components of a software are tested in isolation to ensure they are working correctly.
- A "unit" is usually the smallest piece of code — like a function, method, or class.

- The main goal is to validate that each unit of the software performs as designed, independently of other parts.

Why Unit Testing?

- Detects issues early in the development cycle.
- Makes code easier to change and refactor.
- Ensures better code quality.
- Helps document the code — tests act like examples.
- Supports Agile development by enabling frequent changes with confidence.

Characteristics of a Good Unit Test

- Fast — Should run quickly.
- Isolated — Should not depend on databases, files, or network.
- Repeatable — Same result every time.
- Self-validating — Should have clear pass/fail output.
- Readable — Easy to understand.

Unit Testing Process

1. Write a test case for a specific method or function.
2. Provide inputs and define expected outputs.
3. Run the test to check if actual output matches expected output.
4. Fix the code if the test fails and rerun until it passes

4.Integration Testing

- Integration testing is a type of software testing where individual modules (units) are combined and tested as a group to check how they interact with each other.
- The goal is to detect errors in the interaction between integrated units, not within the units themselves.

In short:

- Unit Testing tests parts individually.
- Integration Testing tests parts together.

Why Integration Testing?

- Units/modules may work fine individually, but fail when combined.
- Helps to identify issues like:
- Incorrect data passing
- Interface mismatches
- Broken APIs
- Data flow problems
- Ensures that modules/components work properly together.

When is Integration Testing Done?

- After unit testing is completed.
- Before system testing begins.

5.Performance Testing

Performance testing is a type of software testing that evaluates the speed, scalability, stability, and responsiveness of an application under a particular workload.

In simple terms:

> How fast, how stable, and how well does the system perform when many users use it?

Why is Performance Testing Important?

- Detect bottlenecks (slow database queries, memory leaks, etc.)
- Ensure the system can handle expected traffic.
- Improve user experience — slow apps frustrate users.
- Validate SLAs (Service Level Agreements) and performance goals.
- Avoid system crashes during peak usage (like shopping sales, IPO launches, etc.).

Main Goals of Performance Testing

- Speed — Is the application fast enough?

- Scalability — Can it handle growth (more users, more data)?
- Stability — Does it behave consistently under varying loads?