PERFORMANCE TESTING:

Performance testing is a non-functional software testing technique that evaluates the speed, responsiveness, stability, reliability, scalability, and responsiveness of an application under a given workload.

While doing performance testing on the application, we will concentrate on the various factors like **Response time, Load, and Stability** of the application.

Response time: Response time is the time taken by the server to respond to the client's request.

Load: Here, Load means that when **N-number** of users using the application simultaneously or sending the request to the server at a time.

Stability: For the stability factor, we can say that, when N-number of users using the application simultaneously for a particular time.

ATTRIBUTES OF PERFORMANCE TESTING:

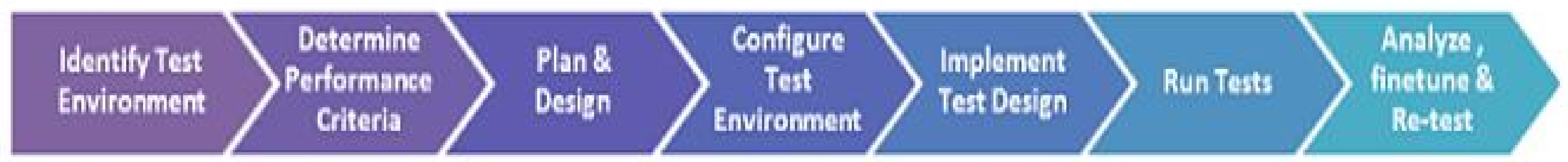
Scalability: Performance testing examines how well a system can handle increased workload and user traffic. It helps determine if the system can scale effectively to accommodate growing demands without a significant drop in performance.

Load Testing: This type of performance testing involves subjecting the system to varying levels of simulated workload to assess its behavior and performance under different levels of concurrent users or transaction.

Stress Testing: Stress testing evaluates how the system behaves when pushed beyond its normal operating capacity. It aims to identify the breaking points and measure the system's ability to recover after handling excessive load.

Endurance Testing: Also known as soak testing, this attribute involves evaluating the system's performance over an extended period. It helps uncover memory leaks, resource exhaustion, and other performance issues that may occur during prolonged usage.

Response Time: Performance testing measures the time it takes for the system to respond to user requests under different loads. It helps determine if the response time remains acceptable as the workload increases.


Identify Test Environment
Determine Performance Criteria
Plan & Design
Configure Test Environment
Implement Test Design
Run Tests
Analyze , finetune & Re-test

Step 1) Identify Your Testing Environment:

Know your physical test environment, production environment and what testing tools are available. Understand details of the hardware, software and network configurations used during testing before you begin the testing process. It will help testers create more efficient tests. It will also help identify possible challenges that testers may encounter during the performance testing procedures.

Step 2) Identify the Performance Acceptance Criteria

This includes goals and constraints for throughput, response times and resource allocation. It is also necessary to identify project success criteria outside of these goals and constraints. Testers should be empowered to set performance criteria and goals because often the project specifications will not include a wide enough variety of performance benchmarks. Sometimes there may be none at all. When possible finding a similar application to compare to is a good way to set performance goals.

Step 3) Plan & Design Performance Tests

Determine how usage is likely to vary amongst end users and identify key scenarios to test for all possible use cases. It is necessary to simulate a variety of end users, plan performance test data and outline what metrics will be gathered.

Step 4) Configuring the Test Environment

Prepare the testing environment before execution. Also, arrange tools and other resources.

Step 5) Implement Test Design

Create the performance tests according to your test design.

Step 6) Run the Tests

Execute and monitor the tests.

Step 7) Analyze, Tune and Retest

Consolidate, analyze and share test results. Then fine tune and test again to see if there is an improvement or decrease in performance. Since improvements generally grow smaller with each retest, stop when bottlenecking is caused by the CPU. Then you may have the consider option of increasing CPU power.