Performance testing is a type of software testing that is used to evaluate the performance of a system or application. It is a non-functional testing technique that focuses on measuring and analyzing the system's responsiveness, stability, scalability, and resource utilization under a variety of workloads.

Goals of performance testing

The goals of performance testing are to:

- Identify and fix performance bottlenecks
- Ensure that the system meets performance requirements
- Measure the impact of changes to the system
- Predict the system's performance under different workloads
- Improve the overall user experience

Types of performance testing

There are many different types of performance testing, each of which focuses on a different aspect of system performance. Some of the most common types of performance testing include:

- Load testing: This type of testing is used to determine how the system performs under increasing loads.
- Stress testing: This type of testing is used to determine how the system performs under extreme loads.
- Endurance testing: This type of testing is used to determine how the system performs over long periods of time.
- Scalability testing: This type of testing is used to determine how the system scales to handle increasing workloads.
- Capacity testing: This type of testing is used to determine how much load the system can handle before it breaks.

Performance testing tools

There are many different performance testing tools available, both commercial and open-source. Some of the most popular performance testing tools include:

- Jmeter: An open-source performance testing tool that can be used to load test, stress test, and endurance test a variety of systems.
- LoadRunner: A commercial performance testing tool from Hewlett Packard Enterprise.
- WebLoad: A commercial performance testing tool from Radware.
- BlazeMeter: A cloud-based performance testing tool from Perforce.

Performance testing process

The performance testing process typically consists of the following steps:

1. Define performance goals: The first step is to define the performance goals for the system. What are the acceptable response times? What is the maximum number of concurrent users that the system should be able to handle?
2. Identify performance metrics: Once the performance goals have been defined, the next step is to identify the performance metrics that will be used to measure the system's performance. These metrics may include response times, throughput, error rates, and resource utilization.
3. Design performance tests: The next step is to design performance tests that will measure the performance of the system against the defined performance metrics. The performance tests should be designed to simulate the expected workload that the system will experience in production.
4. Execute performance tests: The next step is to execute the performance tests. The performance tests should be executed in a controlled environment to ensure that the results are accurate.
5. Analyze performance results: The next step is to analyze the performance results. The analysis should identify any performance

bottlenecks and determine whether the system meets the defined performance goals.

6. **Report performance results: The final step is to report the performance results to stakeholders. The report should include the performance metrics, the performance goals, and any recommendations for improvement.**

Load Testing

Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously. The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.

This testing usually identifies –

- The maximum operating capacity of an application
- Determine whether the current infrastructure is sufficient to run the application
- Sustainability of application with respect to peak user load
- Number of concurrent users that an application can support, and scalability to allow more users to access it.

It is a type of non-functional testing. In Software Engineering, Load testing is commonly used for the Client/Server, Web-based applications – both Intranet and Internet.

Need of Load Testing:

Some extremely popular sites have suffered serious downtimes when they get massive traffic volumes. E-commerce websites invest heavily in advertising campaigns, but not in Load Testing to

ensure optimal system performance, when that marketing brings in traffic.
Consider the following examples

- Popular toy store Toysrus.com, could not handle the increased traffic generated by their advertising campaign resulting in loss of both marketing dollars, and potential toy sales.
- An Airline website was not able to handle 10000+ users during a festival offer.
- Encyclopedia Britannica declared free access to their online database as a promotional offer. They were not able to keep up with the onslaught of traffic for weeks.

Many sites suffer delayed load times when they encounter heavy traffic. Few Facts –

- Most users click away after 8 seconds' delay in loading a page
- $ 4.4 Billion Lost annually due to poor performance

Why Load Testing?

- Load testing gives confidence in the system & its reliability and performance.
- Load Testing helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment.
- Load testing gives excellent protection against poor performance and accommodates complementary strategies for performance management and monitoring of a production environment.

Goals of Load Testing:

Loading testing identifies the following problems before moving the application to market or Production:

- Response time for each transaction
- Performance of System components under various loads
- Performance of Database components under different loads
- Network delay between the client and the server

- Software design issues
- Server configuration issues like a Web server, application server, database server etc.
- Hardware limitation issues like CPU maximization, memory limitations, network bottleneck, etc.

Load testing will determine whether the system needs to be fine-tuned or modification of hardware and software is required to improve performance. To effectively conduct load testing, you can utilize various performance testing tools that are available to help you identify areas for improvement.

Smoke testing and regression testing are two important types of software testing that are used to ensure the quality of software applications.

Smoke testing is a shallow level of testing that is performed to verify that the basic functionality of an application is working correctly. It is typically performed early in the development lifecycle, before any other testing is done. Smoke tests are designed to quickly identify any major problems with the application, so that they can be fixed before they cause more serious issues.

Regression testing is a more in-depth type of testing that is performed to ensure that changes to an application have not introduced any new bugs. It is typically performed after any changes have been made to the application, to ensure that the changes have not broken any existing functionality. Regression tests are designed to cover all areas of the application, including areas that have not been changed.

Here is an example of how smoke testing and regression testing can be used together:

Suppose a software developer is making a change to the login page of an application. The developer would first perform a smoke test on the login page to ensure that the basic functionality is working correctly, such as being able to enter a username and password and log in to the application. If the smoke test fails, the developer would fix the problem and then re-run the smoke test until it passes.

Once the smoke test passes, the developer would then perform a regression test on the login page, as well as on other parts of the application that might have been affected by the change. The regression test would cover all of the functionality of the login page, as well as other areas of the application that are related to the login process. If the regression test fails, the developer would fix the problem and then re-run the regression test until it passes.

By using smoke testing and regression testing together, software developers can ensure that their applications are of high quality and that changes to the application are made in a safe and reliable way.