Load Testing Concepts

- What is the difference between Load, Stress, Spike, and Soak testing?
- What are throughput and latency in load testing?
- What metrics do you monitor during a load test?

1) What is the difference between Load, Stress, Spike, and Soak testing?

Load Testing

A performance testing method called load testing is used to gauge how well an application performs under a predetermined and regulated workload. In order to assess reaction time, system performance, and general stability, the number of virtual users is gradually increased up to the predetermined limit. Making sure the application runs smoothly under typical operational conditions is the aim.

Example:
Let's say the program must accommodate 750 concurrent users according to the Non-Functional Requirements (NFR). In order to verify that the application satisfies the necessary requirements, the system's response time and performance metrics are tracked as traffic is progressively increased up to 750 users during load testing.

Stress Testing

Stress testing is a performance testing method that applies a user load that is more than the anticipated capacity in order to assess an application's robustness. In order to see how robust and dependable the system is under harsh circumstances, the load is progressively increased beyond the typical usage limit.

Stress testing's primary goal is to evaluate how well an application manages high demand and how it responds when pushed beyond its predetermined limitations, rather than necessarily crashing it. The program may continue to operate correctly or fail during this testing, and both results offer insightful information about how the system behaves..

Example:

Assume that the Non-Functional Requirements (NFR) call for 800 concurrent users to be supported. In order to assess the application's scalability, reliability, and capacity to handle more traffic than anticipated, the user load is increased above 800 users during stress testing.

Difference between Load, Stress,

Load Testing	**Stress Testing**
To verify application performance under expected user load	To evaluate system behavior beyond normal capacity
Equal to or less than the expected number of users	Greater than the expected number of users
Response time, throughput, and efficiency	Stability, reliability, and failure behavior
Ensure the application meets performance requirements.	Identify limits and assess robustness under extreme conditions
System is expected to function normally	System may slow down or fail
Before release to validate normal performance	After load testing to analyze extreme scenarios

Spike Testing

Spike testing is a kind of performance testing that assesses how an application responds to an abrupt, sharp rise in user load for a brief amount of time. In this testing, there is little to no ramp-up time, and the load is applied almost instantly. Similarly, the load is removed quickly, resulting in a rapid ramp-down.

Spike testing is used to confirm whether the system can withstand unforeseen spikes in traffic and continue to function dependably without crashing or experiencing significant degradation..

Example:
Applications that encounter abrupt spikes in traffic, such those that purchase tickets online for big events (like a match between Bangladesh and India), the launch of new products, or massive seasonal sales like Black Friday, might benefit greatly from spike testing.

Soak Testing

Soak testing, sometimes referred to as endurance testing, is a kind of performance testing that assesses an application's stability and dependability over a long time span. In order to find problems like memory leaks, resource exhaustion, or performance degradation that could happen over time, the system is continuously subjected to a normal or expected user load during this testing.

Soak testing's primary goal is to make sure the program can withstand prolonged use without malfunctioning and keeps up its reliable performance over extended periods of time.

Example:
Soak testing, for example, entails running the system with a load of 600 concurrent users for several hours or days in order to confirm long-term stability and effective resource usage.

● What are throughput and latency in load testing?

Throughput

The total number of requests or transactions that a server can successfully process in a given amount of time is referred to as throughput. This duration can be expressed in hours, minutes, or seconds. Requests per second (RPS) or transactions per second (TPS) are two common ways to quantify throughput in load testing. A higher throughput shows that the system can handle more requests effectively when under load.

Latency

The time it takes for a request to get from the client to the server and for the client to receive the first byte of the answer is known as latency. It shows how long it takes for the client and server to communicate. Latency merely measures the network and transmission delay; it does not account for the time the server actually spends processing the request.

- What metrics do you monitor during a load test?

During a load test, a number of performance metrics are tracked to assess how the application reacts to anticipated user demand. These measures guarantee overall system reliability and assist in locating performance bottlenecks.

Response Time – Measures how quickly the application replies to user requests.

Throughput – Represents the volume of requests or transactions handled within a given time frame.

Concurrent Users – Indicates how many users are actively using the system at the same time.

Error Percentage – Tracks failed requests to evaluate application stability under load.

Time to First Byte (TTFB) – Calculates the delay between sending a request and receiving the first byte of the response.

CPU Consumption – Monitors processor usage to detect overutilization or performance constraints.

Memory Consumption – Observes RAM usage to identify memory leaks or inefficient allocation.

Network Performance – Measures data transfer rates and network latency during the test.

Disk Operations (I/O) – Evaluates read and write activity to uncover storage-related bottlenecks.