**Introduction to Performance Testing**

**Performance Testing**

Performance testing is a type of **non-functional testing** that evaluates how a system behaves under different loads. It ensures that software applications perform well under expected and peak traffic conditions.

**Objectives of Performance Testing**

1. **Assess System Speed** – Measures response times for various user interactions.
2. **Ensure Scalability** – Determines if the system can handle an increasing number of users.
3. **Check Stability** – Ensures the system does not crash under load.
4. **Identify Bottlenecks** – Finds performance issues such as slow database queries or memory leaks.

**Types of Performance Testing**

1. **Load Testing** – Measures system behavior under normal and peak loads.
2. **Stress Testing** – Tests the system beyond its capacity to check stability.
3. **Scalability Testing** – Verifies the system’s ability to scale up/down with increased workload.
4. **Endurance Testing** – Evaluates system performance over a long period.
5. **Spike Testing** – Observes how the system handles sudden spikes in traffic.

**Example: Performance Testing for an E-commerce Website**

**Scenario:**

A company wants to ensure that its website can handle **1,000 users per second** without delays.

**Tool Used: Apache JMeter**

**Test Plan:**

1. Simulate **1,000 concurrent users** browsing and purchasing items.
2. Measure response time for checkout transactions.
3. Identify database and server slowdowns.

**JMeter Test Script (Basic Example)**

<ThreadGroup numThreads="1000" rampUp="10">

<Sampler>

<HTTPSampler domain="ecommerce.com" path="/checkout" method="POST"/>

</Sampler>

<Listener>

<SummaryReport/>

</Listener>

</ThreadGroup>

**Expected Outcome:**

* Response time should be **below 2 seconds**.
* Server CPU and memory usage should remain stable.
* No **timeouts or crashes** should occur.

**Conclusion**

Performance testing is essential to ensure **speed, scalability, and stability** in applications. Tools like **JMeter, Gatling, and LoadRunner** help identify performance bottlenecks before deployment.