SESSION UNDERSTANDING DOCUMENT ON PERFORMANCE TESTING AND JMETER TOOL:

Performance Testing

Definition: Performance testing is a type of non-functional testing aimed at determining the responsiveness, stability, scalability, and speed of a software application under a particular workload. The primary goal is to identify and eliminate performance bottlenecks in the software.

Objectives:

- Ensure the software application performs well under expected workloads.
- Identify and fix performance issues before the software is deployed to production.
- Verify the software's reliability and scalability.
- Provide insights into how the application behaves under stress and peak load conditions.

Types of Performance Testing

Load Testing:

Purpose: To assess how the system behaves under expected load conditions.

Description: Involves simulating a typical number of users performing a set of activities to determine if the application can handle the expected load.

Stress Testing:

Purpose: To evaluate the system's behavior under extreme load conditions.

Description: Involves pushing the system beyond its normal operational capacity to identify breaking points and performance bottlenecks.

Spike Testing:

Purpose: To determine the system's ability to handle sudden, large spikes in load.

Description: Simulates a sudden increase in the number of users or transactions to test how the system responds to abrupt changes in load.

Endurance Testing (Soak Testing):

Purpose: To verify the system's stability and performance over an extended period.

Description: Involves running the system at a normal load for an extended duration to identify potential memory leaks, performance degradation, and other issues that might arise over time.

Scalability Testing:

Purpose: To determine how well the system scales with increasing load.

Description: Involves gradually increasing the load on the system to understand its capacity and identify the maximum load it can handle before performance becomes unacceptable.

Volume Testing:

Purpose: To assess the system's ability to handle a large volume of data.

Description: Involves testing the system's performance by populating the database with a large amount of data and examining its response and behavior.

Common Performance Testing Tools

Apache JMeter: An open-source tool used for load and performance testing.

LoadRunner: A commercial performance testing tool from Micro Focus.

Gatling: An open-source tool designed for high-performance load testing.

NeoLoad: A performance and load testing tool for web and mobile applications.

BlazeMeter: A cloud-based performance testing platform compatible with JMeter.

Understanding and executing performance testing effectively ensures that software applications meet performance expectations and provide a good user experience under various conditions.

Apache JMeter

Apache JMeter is a popular open-source tool designed for performance testing and load testing of web applications. Developed by the Apache Software Foundation, it is widely used by developers and testers to measure the performance of web applications and services, simulate heavy loads, and analyze the results. Here's an overview of JMeter:

Key Features

1. **Open-Source**: JMeter is free and open-source, which makes it accessible to everyone and allows for community contributions.

2. **Platform Independent**: JMeter is written in Java, making it platform-independent. It can run on any operating system that supports Java, such as Windows, Linux, and MacOS.
3. **GUI and CLI Modes**: JMeter offers a graphical user interface (GUI) for designing test plans and viewing results, as well as a command-line interface (CLI) for running tests in non-GUI mode, which is useful for automation and continuous integration.
4. **Protocol Support**: JMeter supports a wide range of protocols and technologies, including:
    - HTTP, HTTPS (web)
    - SOAP/REST (web services)
    - FTP
    - JDBC (database)
    - JMS (Java Messaging Service)
    - TCP
    - LDAP
5. **Recording Capabilities**: JMeter includes a proxy server to record user actions on web applications and create test scripts automatically.
6. **Extensibility**: Users can extend JMeter's functionality by writing custom plugins and samplers, using third-party plugins, or integrating it with other tools.
7. **Test Plan Creation**: JMeter uses a hierarchical structure to create test plans. Components include:

    - **Thread Groups**: Represent a group of users or virtual users.
    - **Samplers**: Define the type of request to send (e.g., HTTP request).
    - **Listeners**: Collect and display results of the test execution.
    - **Config Elements**: Provide default values and variables.
    - **Assertions**: Validate responses.
    - **Timers**: Introduce delays between requests.

8. **Distributed Testing**: JMeter supports distributed testing, allowing users to run tests across multiple machines to simulate a large number of users.
9. **Result Analysis**: JMeter provides various listeners to visualize results, including graphs, tables, and log files. Users can analyze response times, throughput, error rates, and other performance metrics.
10. **Integration with CI/CD**: JMeter can be integrated with continuous integration/continuous deployment (CI/CD) pipelines using tools like Jenkins, enabling automated performance testing as part of the development process.

Typical Use Cases

- **Load Testing**: Simulating a large number of users to evaluate how a web application performs under load.

- **Stress Testing**: Determining the maximum capacity of an application by pushing it beyond its normal operational limits.
- **Performance Testing**: Measuring response times, throughput, and resource utilization of an application under various conditions.
- **Functional Testing**: Validating the functionality of web applications and APIs.
- **Regression Testing**: Ensuring that new code changes do not negatively impact the performance of existing functionality.

To get started with JMeter, follow these steps:

1. **Download and Install**: Download the latest version of JMeter from the Apache JMeter website and install it.
2. **Launch JMeter**: Run the jmeter.bat file (Windows) or jmeter script (Linux/MacOS) to start the GUI.
3. **Create a Test Plan**: Use the GUI to create a test plan by adding thread groups, samplers, listeners, and other components.
4. **Configure and Run Tests**: Configure the test parameters and run the test plan. Monitor the results using the listeners.
5. **Analyze Results**: Analyze the collected data to identify performance bottlenecks and areas for improvement.

JMeter is a powerful tool for performance testing, offering flexibility, extensibility, and a rich set of features to help developers and testers ensure the reliability and scalability of their web applications.