---

1. What is Performance Testing? What is the Use of it?

Performance Testing is a non-functional software testing technique that determines how the stability, speed, scalability, and responsiveness of an application hold up under a given workload. Unlike functional testing, which checks "if" a feature works, performance testing checks "how well" it works. It ensures the software does not crash or lag when thousands of users access it simultaneously.

Uses and Objectives of Performance Testing:

1. **Validating Response Time:**
   - **Use:** To measure the time it takes for the system to respond to a user request (e.g., loading a webpage, processing a payment).
   - **Goal:** To ensure critical transactions happen within an acceptable threshold (e.g., under 2 seconds).
2. **Determining Scalability:**
   - **Use:** To figure out the maximum user load the software application can handle.
   - **Goal:** To help plan for future growth. For example, knowing that the current server setup handles 10,000 users allows IT to plan infrastructure upgrades before reaching 20,000 users.
3. **Checking Stability Under Load (Stress Testing):**
   - **Use:** To observe how the system behaves under extreme conditions or peak traffic (e.g., during a flash sale).
   - **Goal:** To identify the "breaking point" of the application—where it crashes or throws errors—and ensure it recovers gracefully.
4. **Identifying Bottlenecks:**
   - **Use:** To pinpoint specific areas causing delays, such as slow database queries, memory leaks, or network limitations.
   - **Goal:** To allow developers to optimize code or database indexing to improve overall efficiency.
5. **Reliability (Endurance Testing):**
   - **Use:** To run the system at a significant load for an extended period (e.g., 24-48 hours).
   - **Goal:** To detect memory leaks or performance degradation that only occurs over time.

---

2. Illustrate Selenium Tool Suite in Detail 🧰

Selenium is not a single tool but a suite of software tools, each catering to different testing needs of an organization. It is the industry standard for open-source automated testing of web applications.

The suite consists of four main components:

1. Selenium IDE (Integrated Development Environment)

- **Description:** A browser extension (originally Firefox, now Chrome/Edge) that allows for "Record and Playback."
- **Usage:** Used for rapid prototyping of test scripts. A tester hits record, navigates the site, and the IDE saves the actions.
- **Limitation:** It is not suitable for complex test cases requiring loops, conditional logic, or database connectivity.

2. Selenium RC (Remote Control) - *Legacy*

- **Description:** The first tool to allow writing tests in multiple programming languages (Java, Python, etc.).
- **Architecture:** It relied on a "JavaScript Injection" technique. A proxy server (Selenium Server) had to be launched to trick the browser into believing the test script was served from the same domain as the web app.
- **Status:** It has been officially deprecated and removed in favor of WebDriver.

3. Selenium WebDriver

- **Description:** The successor to Selenium RC and the core of modern Selenium.
- **Architecture:** Unlike RC, WebDriver does not use a proxy. It communicates **directly** with the browser's native capabilities using a browser-specific driver (e.g., ChromeDriver,

GeckoDriver).
- **Advantage:** It is faster, more stable, and supports complex user interactions (like drag-and-drop) more accurately.

4. Selenium Grid

- **Description:** A tool used for parallel execution.
- **Usage:** It allows running tests on different machines against different browsers and operating systems simultaneously.
- **Goal:** To significantly reduce the time required to run a large test suite.

---

3. How Would You Explain Selenium WebDriver? Explain it. 🚀

Selenium WebDriver is an open-source web automation framework that allows you to execute your tests against different browsers, not just Firefox (unlike the original IDE). It is an Application Programming Interface (API) that creates a direct communication channel between your test code and the web browser.

Detailed Explanation:

1. **Architecture (The "Handshake"):**
   - **Client Library:** You write your test script in a language like Java, Python, C#, or Ruby.
   - **JSON Wire Protocol (or W3C Standard):** Your code commands are converted into a JSON format and sent via HTTP requests.
   - **Browser Driver:** Each browser has a specific executable (e.g., chromedriver.exe, geckodriver.exe). This driver receives the JSON command and translates it into an internal action the browser understands.
   - **Real Browser:** The browser executes the command (e.g., clicks a button) and sends the response back through the driver to your code.
2. **Key Features:**
   - **Multi-Browser Support:** Supports Chrome, Firefox, Safari, Edge, and Internet Explorer.

- **Multi-Language Support:** Tests can be written in Java, Python, C#, Ruby, JavaScript, etc.
- **Speed:** Since it speaks directly to the browser engine without an intermediary proxy server (like Selenium RC), it is significantly faster.
- **Handling Dynamic Elements:** It provides robust strategies (Locators like XPath, CSS Selectors, ID) to find elements on dynamic web pages.

---

4. Explain Robotic Process Automation (RPA) in Detail

🤖

Robotic Process Automation (RPA) is a technology that uses software robots ("bots") to emulate human actions interacting with digital systems and software. Unlike traditional automation which requires backend integration (APIs), RPA works on the **User Interface (UI)** layer, just like a human employee.

Key Characteristics:

1. **Rule-Based Tasks:** RPA is best suited for repetitive, rule-based tasks that do not require creative judgment. If the process follows a strict "If-Then" logic, a bot can do it.
2. **Non-Invasive:** RPA sits on top of existing IT infrastructure. It doesn't require changing the underlying database or legacy systems. It simply logs in, clicks, copies, and pastes data between applications.
3. **Cross-Application:** A single RPA bot can switch between applications seamlessly. For example, it can copy data from an email, paste it into an Excel sheet, log into an SAP ERP system, and enter that data into a form.

Common Use Cases:

- **Data Entry:** Transferring data from paper forms or emails into CRM systems.
- **Invoice Processing:** Reading invoices (using OCR), extracting amounts/dates, and entering them into accounting software.
- **HR Onboarding:** Creating email accounts, generating ID badges, and granting system access for new employees automatically.

Popular Tools:

- **UiPath:** Known for its user-friendly drag-and-drop interface.
- **Blue Prism:** Focused on enterprise-grade security and scalability.
- **Automation Anywhere:** A cloud-native RPA platform.

---

5. How to Choose Automation Testing Tools? Explain it. 🛠️

Selecting the right automation tool is critical for the success of the testing project. The wrong choice can lead to wasted budget and shelf-ware.

Factors to Consider:

1. **Project Requirements & Technology Stack:**
   - Does the tool support the application's technology? (e.g., If you are testing a desktop app, Selenium won't work; you need UFT or WinAppDriver. If testing a mobile app, you need Appium).
2. **Budget (Cost):**
   - **Open Source:** Tools like Selenium and Appium are free but require higher technical skills to set up and maintain.
   - **Commercial:** Tools like UFT (QTP) or TestComplete are expensive but come with vendor support and easier interfaces.
3. **Team Skills:**
   - Does the team know how to code? If your testers are non-programmers, a "Codeless" or "Record & Playback" tool (like Katalon Studio) is better. If they are developers, a code-heavy library like Selenium or Cypress is preferable.
4. **Reporting Capabilities:**
   - Does the tool provide detailed, easy-to-read reports? Good reporting helps identify the root cause of failures quickly with screenshots and logs.
5. **CI/CD Integration:**
   - Can the tool easily integrate with DevOps tools like Jenkins, Azure DevOps, or GitLab? This is essential for continuous testing.
6. **Support and Community:**
   - For open-source tools, a large active community (like Selenium's) is vital for troubleshooting. For commercial tools, dedicated vendor support is key.

---

6. Can You Explain Selenium Grid in Detail? 🌐

Selenium Grid is a smart proxy server that allows tests to be routed to remote browser instances running on different machines. Its primary purpose is **parallel execution**.

Architecture (Hub and Node Model):

1. **The Hub:**
   - The central point where you load your tests.
   - There is only one Hub in a Grid.
   - It receives the test request (e.g., "Run this test on Windows 10 using Chrome").
   - It searches its registry for a registered Node that matches these requirements.
2. **The Nodes:**
   - These are the worker machines that are registered to the Hub.
   - There can be many Nodes.
   - Each Node can have a different operating system (Windows, Linux, macOS) and different browser versions.
   - The Node executes the instructions sent by the Hub.

Benefits of Selenium Grid:

- **Time Saving:** Instead of running 100 tests sequentially (which might take 10 hours), you can run them in parallel across 10 machines (taking only 1 hour).
- **Cross-Browser Testing:** You can verify that your application works correctly on Chrome on Windows, Safari on Mac, and Firefox on Linux simultaneously.

---

7. Construct Different Automated Testing Process 🔄

The Automated Testing Process involves a series of phases to ensure automation is implemented effectively.

1. **Test Tool Selection:**
   - Decide on the tool based on budget and technology (as discussed in Q5).
2. **Define Scope of Automation:**
   - Not everything should be automated. Select test cases that are critical, repetitive, time-consuming, and stable. Avoid automating tests for features that change frequently.
3. **Planning, Design, and Strategy:**
   - Decide on the automation framework (Data-Driven, Keyword-Driven, or Hybrid).
   - Plan the test environment setup and schedule.
4. **Test Development (Scripting):**
   - Write the actual test scripts. Create common functions (e.g., login(), logout()) to reuse code.
   - Prepare test data (input files).
5. **Test Execution:**
   - Run the scripts. This can be done locally or triggered automatically via a CI/CD server (Jenkins) or Selenium Grid.
6. **Maintenance:**
   - As the application UI changes, test scripts will break. Maintenance involves updating locators and logic to match the new application version.

---

8. Differentiate Between Manual Testing and Automation Testing VS

Feature	Manual Testing	Automation Testing
1. Definition	Testing software by human hand without using any automated tools.	Testing software using specialized tools and scripts to execute test cases.
2. Speed	Slower, as it depends on human speed. Humans tire and slow down.	Significantly faster. Computers can execute thousands of steps per minute without fatigue.
3. Reliability & Accuracy	Lower. Humans are prone to errors, especially when	Higher. Scripts perform the exact same operation every

	repeating the same tedious task.	time without error (unless the script is buggy).
4. Cost (ROI)	Lower initial cost (no tools/scripting time), but higher long-term cost for regression.	Higher initial cost (tools/scripting), but high ROI in the long run for large regression suites.
5. Suitability	Best for Exploratory, Usability, and Ad-hoc testing where human intuition is needed.	Best for Regression, Load, and Performance testing where repetition is key.
6. User Interface	Can provide feedback on the "Look and Feel" (UI/UX) of the application.	Cannot check "Look and Feel" (e.g., if a color looks "good"). It only checks functional attributes.
7. Programming Knowledge	No programming knowledge is required.	Programming knowledge is usually required (unless using codeless tools).
8. Parallel Execution	Impossible for a single tester to do multiple tests at once.	Easy to execute multiple tests simultaneously on different machines (Parallel testing).

---

9. What is Automation Testing? Explain Process with Neat Diagram and Example. 🤖

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports.

The Process (Summary):

1. **Test Tool Selection**
2. **Scope Definition**
3. **Planning/Design**
4. **Development**
5. **Execution**
6. **Maintenance**

(Note: See diagram tag in Question 7 for visual representation)

Example Scenario: Login Page Testing

- **Manual Approach:** A human opens the browser, types www.site.com, clicks "Login", types "User1", types "Pass123", clicks "Submit", and looks at the screen to see if "Welcome User1" appears.

- **Automation Approach:**
  - A script is written:

```
Python
driver.get("www.site.com")
driver.find_element(By.ID, "user").send_keys("User1")
driver.find_element(By.ID, "pass").send_keys("Pass123")
driver.find_element(By.ID, "submit").click()
assert "Welcome User1" in driver.page_source
```

  - When run, the computer opens the browser and performs these actions in seconds, reporting "PASS" or "FAIL" automatically.

---

10. Describe Apache JMeter 📉

Apache JMeter is an open-source, Java-based software designed to load test functional behavior and measure performance.

i) Aim / Purpose

- **Load Testing:** To simulate a heavy load on a server, group of servers, network, or object to test its strength or to analyze overall performance under different load types.
- **Performance Analysis:** To determine how the system performs in terms of responsiveness and stability under a particular workload.[1]

ii) Working[2]

- JMeter acts like a group of users sending requests to a target server.[3]
- **Thread Group:** You define a "Thread Group" where each thread represents a virtual user. If you set 100 threads, JMeter creates 100 vir[4]tual users.[5]
- **Samplers:** These are the actual requests (e.g., HTTP Request, FTP Request).[6]
- **Listeners:** Thes[7]e capture the server's response and visualize the data (Graphs, Tables) to show response times, error rates, and throughput.

iii) Advantages

1. **Open Source:** It is completely free to use.
2. **Platform Independent:** Being 100% Java, it runs on Windows, Linux, and Mac.
3. **Protocol Support:** Supports HTTP, HTTPS, FTP, Database (JDBC), Mail (SMTP/POP3), and more.
4. **Extensible:** Highly customizable with plugins.

---

11. What is Selenium RC? Explain in Detail 📻

Selenium Remote Control (RC), also known as Selenium 1, was the first automated web testing tool that allowed users to write test scripts in a programming language of their choice (Java, C#, PHP, Python, etc.) rather than just the recorded scripts of Selenium IDE.

How it Worked (The Proxy Injection):

- **The Problem:** Browsers have a security restriction called the **Same Origin Policy**, which prevents JavaScript code from one domain (your test script) from accessing elements on a different domain (the website you are testing).
- **The Solution:** Selenium RC introduced the **Selenium Server**. This server acts as an HTTP proxy. It tricks the browser into believing that the test script and the web application are coming from the same domain.
- **Process:**
  1. The test script sends a command to the Selenium Server.
  2. The Selenium Server injects a JavaScript program (called **Selenium Core**) into the browser.
  3. Selenium Core executes the command on the web page.
  4. The browser returns the result to the Server, which sends it back to your script.

Why it is Deprecated:

- **Complex Architecture:** Running the standalone Selenium Server was cumbersome.
- **Slower Execution:** The extra layer of JavaScript injection slowed down tests compared to WebDriver's direct communication.
- **Redundancy:** Selenium WebDriver (Selenium 2.0) merged with RC to provide a superior, simpler, and faster architecture, rendering RC obsolete.