**Module-4 Automation Core Testing (Load Runner Up and Selenium)**

Q1. Which components have you used in Load Runner?

* Load Runner is a performance testing tool commonly used for load testing, stress testing, and performance monitoring of web and mobile applications. It typically includes several components or modules, such as:
* VuGen (Virtual User Generator): Used to create and record scripts that simulate user interactions with the application under test. These scripts define the actions that virtual users will perform during load testing.
* Controller: Manages and orchestrates the execution of load tests by coordinating multiple virtual users and distributing the load across various load generators.
* Analysis: Provides tools for analyzing and interpreting the results of load tests. It generates reports and graphs that help identify performance bottlenecks, trends, and areas for optimization.

Q2. How can you set the number of Vusers in Load Runner?

* In LoadRunner, the number of Virtual Users (Vusers) can be set using the Controller module. Here's how you can set the number of Vusers:

1. Open Controller: Launch the LoadRunner Controller application.

2. Create a Scenario: If you haven't already done so, create a new scenario or open an existing one.

3. Define Vusers: In the Scenario pane, locate the Vusers section.

4. Set Number of Vusers : Enter the desired number of Vusers in the appropriate field or use the slider to adjust the number.

5. Distribute Vusers: If you have multiple load generators, you can distribute the Vusers across them by specifying the number of Vusers per load generator.

6.Save Changes: Once you have set the desired number of Vusers, save your changes to the scenario.

7. Run Scenario: After configuring the scenario, you can run it to start the load test with the specified number of Vusers.

* By following these steps, you can easily set the number of Vusers in LoadRunner and conduct load tests to simulate various levels of user activity on your application.

Q3. What is Correlation?

* Correlation ensures that the virtual users (Vusers) in a load test simulate real users accurately by dynamically handling data that changes from one interaction to another. It involves identifying and capturing dynamic data from server responses, defining rules to extract and store this data, and then inserting it into subsequent requests sent by the Vusers during the load test.

Q4. What is the process for developing a Vuser Script?

* Developing a Vuser script in LoadRunner involves several steps to capture and simulate user interactions with an application. Here's a general process for developing a Vuser script:

1. \*\*Recording\*\*:

- Use LoadRunner's Virtual User Generator (VuGen) to record user actions on the application under test.

- Start recording the script while navigating through the application as a typical user would.

- Perform actions such as logging in, browsing pages, submitting forms, and interacting with elements.

2. \*\*Enhancing the Script\*\*:

- Review the recorded script and enhance it as needed.

- Add transactions to measure response times for specific actions.

- Parameterize dynamic values (e.g., usernames, session IDs) to make the script reusable and realistic.

- Add comments and annotations to document script logic and actions.

3. \*\*Correlation\*\*:

- Identify dynamic data in server responses that need to be correlated.

- Use LoadRunner's correlation tools to automatically or manually correlate dynamic values.

- Update the script to include correlated values to maintain session integrity.

4. \*\*Parameterization\*\*:

- Replace hard-coded values with parameters to simulate multiple users with different data.

- Define parameterization rules to generate data dynamically during test execution (e.g., usernames, passwords, input data).

5. \*\*Validation\*\*:

- Validate the script by replaying it to ensure that all recorded actions are executed correctly.

- Check for errors, missing correlations, or parameterization issues.

6. \*\*Script Execution Settings\*\*:

- Configure runtime settings for the Vuser script, including load distribution, iteration settings, and logging options.

- Define any think time settings to simulate realistic user pacing between actions.

7. \*\*Script Debugging\*\*:

- Debug the script if necessary to address any runtime errors or unexpected behavior.

- Use debugging tools in VuGen to step through the script and identify issues.

8. \*\*Execution and Analysis\*\*:

- Execute the script in the LoadRunner Controller to simulate user load.

- Monitor and analyze test results to identify performance bottlenecks, errors, or issues.

- Use LoadRunner Analysis tools to generate reports and graphs for performance metrics.

9. \*\*Iterative Refinement\*\*:

- Iterate on the script based on test results and performance analysis.

- Fine-tune the script parameters, correlations, and logic to achieve desired performance objectives.

Q5. How Load Runner interacts with the application?

LoadRunner interacts with the application under test in a manner that simulates real user behavior while generating load on the system. Here's how LoadRunner interacts with the application:

1. \*\*Recording User Actions\*\*: LoadRunner captures user interactions with the application by recording HTTP or HTTPS requests sent from the client (browser) to the server. This recording process is typically done using LoadRunner's Virtual User Generator (VuGen) and involves initiating a recording session, navigating through the application, and performing various actions such as logging in, browsing pages, and submitting forms.

2. \*\*Script Generation\*\*: During the recording process, LoadRunner generates a Vuser script based on the user actions captured. This script contains instructions that simulate user behavior, including sending HTTP requests, handling responses, and performing validations.

3. \*\*Parameterization and Correlation\*\*: LoadRunner allows for parameterization and correlation of dynamic data in the script. Parameterization involves replacing hard-coded values with parameters to simulate multiple users with different data, while correlation involves extracting dynamic data from server responses and using it in subsequent requests to maintain session integrity.

4. \*\*Execution\*\*: LoadRunner executes the Vuser script in the LoadRunner Controller, which simulates multiple virtual users (Vusers) accessing the application concurrently. Each Vuser executes the script independently, sending requests to the application server and receiving responses.

5. \*\*Load Generation\*\*: The LoadRunner Controller distributes the load generated by Vusers across multiple load generators. Load generators are machines or instances responsible for generating user load by executing the Vuser scripts.

6. \*\*Monitoring and Analysis\*\*: LoadRunner monitors various performance metrics such as response times, throughput, and resource utilization during test execution. It collects data from the application servers, load generators, and network infrastructure to analyze the performance of the application under different load conditions.

7. \*\*Reporting\*\*: LoadRunner provides reporting and analysis tools to generate detailed reports and graphs based on the collected performance data. These reports help identify performance bottlenecks, scalability issues, and areas for optimization in the application.

Overall, LoadRunner interacts with the application by simulating user behavior, generating load, monitoring performance, and analyzing results to assess the performance and scalability of the application under various load conditions.

Q6. What is the relationship between Response Time and Throughput?

Response time and throughput are two key performance metrics that are closely related in the context of load testing and performance testing:

1. \*\*Response Time\*\*: Response time, also known as latency, is the amount of time it takes for the system to respond to a user request. It measures the time elapsed from when a user sends a request to the system until the system sends back a response. Response time is typically measured in milliseconds (ms) or seconds (s).

2. \*\*Throughput\*\*: Throughput is the number of transactions processed by the system within a certain period of time. It represents the system's capacity to handle a certain volume of requests or transactions concurrently. Throughput is typically measured in transactions per second (TPS) or requests per second (RPS).

The relationship between response time and throughput can be summarized as follows:

- \*\*Inverse Relationship\*\*: In general, there is an inverse relationship between response time and throughput. As response time increases, throughput tends to decrease, and vice versa. This relationship is often referred to as the "response time-throughput trade-off."

- \*\*Impact of Load\*\*: When the system is under heavy load or high concurrency, response time tends to increase as the system struggles to process incoming requests efficiently. As a result, throughput may decrease as the system becomes saturated and is unable to handle additional requests within a given time frame.

- \*\*Optimization\*\*: Improving system performance and scalability can help optimize the relationship between response time and throughput. By optimizing system resources, reducing bottlenecks, and improving algorithms and architecture, it's possible to achieve better response times and higher throughput simultaneously.

Q7. What is Automation Testing?

Automation testing is the process of using specialized software tools and scripts to perform tests on software applications automatically, without the need for manual intervention. It involves the creation and execution of test scripts or test cases that simulate user interactions with the application, validate functionality, and verify expected outcomes.

Q8. Which Are the Browsers Supported by Selenium Ide?

* Google Chrome: Selenium IDE can be installed as a browser extension in Google Chrome. It provides a user-friendly interface for recording, editing, and running Selenium test scripts directly within the browser.
* Mozilla Firefox: Selenium IDE is available as a browser extension for Mozilla Firefox. Users can install it from the Firefox Add-ons marketplace and use it to create and execute Selenium test scripts in Firefox

Q9. What are the benefits of Automation Testing?

Automation testing offers several benefits that can significantly improve the efficiency, reliability, and effectiveness of the software testing process. Some of the key benefits of automation testing include:

1. Faster Testing: Automation testing enables faster execution of test cases compared to manual testing. Automated tests can be run overnight or in parallel across multiple environments, allowing for quicker feedback on the quality of the software.

2. Reusability: Test scripts and automation frameworks can be reused across multiple test cycles, projects, or applications. Once created, automated tests can be easily modified, extended, and maintained, saving time and effort in the long run.

3. Improved Accuracy: Automated tests execute predefined actions and verifications consistently, eliminating the potential for human errors associated with manual testing. This improves the accuracy and reliability of test results.

4. Increased Test Coverage: Automation testing enables comprehensive test coverage by executing a large number of test cases, scenarios, and data permutations. It ensures that all critical functionality and edge cases are tested thoroughly, leading to better software quality.

5. Regression Testing : Automation testing is particularly useful for regression testing, where tests are repeated to ensure that new changes or updates to the software do not introduce defects or regressions. Automated regression tests help catch issues early and prevent them from reaching production.

6. Scalability: Automation testing can easily scale to handle large test suites, complex test scenarios, and diverse environments. It allows teams to run tests in parallel across different configurations, platforms, and devices, enabling comprehensive test coverage.

7. Continuous Integration and Delivery (CI/CD): Automation testing integrates seamlessly with CI/CD pipelines, where tests are automatically triggered as part of the software development process. This ensures that code changes are thoroughly tested and validated before being deployed to production, leading to faster and more reliable releases.

8. Cost Savings: While there may be upfront costs associated with setting up automation frameworks and writing test scripts, automation testing ultimately leads to cost savings over time. By reducing the need for manual testing efforts and increasing testing efficiency, automation testing helps organizations deliver high-quality software more efficiently and cost-effectively.

Overall, automation testing offers numerous benefits that contribute to improved software quality, faster time-to-market, and greater confidence in the reliability of the software product.

Q10. What are the advantages of Selenium?

Automation testing offers several benefits that can significantly improve the efficiency, reliability, and effectiveness of the software testing process. Some of the key benefits of automation testing include:

1. Faster Testing: Automation testing enables faster execution of test cases compared to manual testing. Automated tests can be run overnight or in parallel across multiple environments, allowing for quicker feedback on the quality of the software.

2. Reusability: Test scripts and automation frameworks can be reused across multiple test cycles, projects, or applications. Once created, automated tests can be easily modified, extended, and maintained, saving time and effort in the long run.

3. Improved Accuracy: Automated tests execute predefined actions and verifications consistently, eliminating the potential for human errors associated with manual testing. This improves the accuracy and reliability of test results.

4. Increased Test Coverage: Automation testing enables comprehensive test coverage by executing a large number of test cases, scenarios, and data permutations. It ensures that all critical functionality and edge cases are tested thoroughly, leading to better software quality.

5. Regression Testing: Automation testing is particularly useful for regression testing, where tests are repeated to ensure that new changes or updates to the software do not introduce defects or regressions. Automated regression tests help catch issues early and prevent them from reaching production.

6. Scalability: Automation testing can easily scale to handle large test suites, complex test scenarios, and diverse environments. It allows teams to run tests in parallel across different configurations, platforms, and devices, enabling comprehensive test coverage.

7. Continuous Integration and Delivery (CI/CD): Automation testing integrates seamlessly with CI/CD pipelines, where tests are automatically triggered as part of the software development process. This ensures that code changes are thoroughly tested and validated before being deployed to production, leading to faster and more reliable releases.

8. Cost Savings: While there may be upfront costs associated with setting up automation frameworks and writing test scripts, automation testing ultimately leads to cost savings over time. By reducing the need for manual testing efforts and increasing testing efficiency, automation testing helps organizations deliver high-quality software more efficiently and cost-effectively.

Overall, automation testing offers numerous benefits that contribute to improved software quality, faster time-to-market, and greater confidence in the reliability of the software product.

Q11. Why testers should opt for Selenium and not QTP?

Testers may opt for Selenium over QTP (now known as UFT, Unified Functional Testing) due to several reasons:

1. Open-Source: Selenium is open-source, making it freely available and accessible to all testers regardless of budget constraints.

2. Cross-Browser and Cross-Platform Compatibility: Selenium supports testing across various browsers and operating systems, providing consistent testing experiences across different environments.

3. Language Support: Selenium supports multiple programming languages, allowing testers to choose the language they are most comfortable with or that best fits their project requirements.

4. Community Support and Resources: Selenium has a large and active community of developers and testers who contribute to its development and provide support through forums, blogs, and online resources.

5. Flexibility and Extensibility: Selenium's architecture allows for flexibility and extensibility, enabling users to customize and extend its functionality to suit their specific testing needs.

Overall, Selenium's open-source nature, cross-browser compatibility, language support, community support, and flexibility make it a preferred choice for many testers over QTP/UFT.

Q12. Why testers should opt for Selenium and not QTP?

Testers may opt for Selenium over QTP (now known as UFT, Unified Functional Testing) due to several reasons:

1. \*\*Open-Source\*\*: Selenium is open-source, making it freely available and accessible to all testers regardless of budget constraints.

2. \*\*Cross-Browser and Cross-Platform Compatibility\*\*: Selenium supports testing across various browsers and operating systems, providing consistent testing experiences across different environments.

3. \*\*Language Support\*\*: Selenium supports multiple programming languages, allowing testers to choose the language they are most comfortable with or that best fits their project requirements.

4. \*\*Community Support and Resources\*\*: Selenium has a large and active community of developers and testers who contribute to its development and provide support through forums, blogs, and online resources.

5. \*\*Flexibility and Extensibility\*\*: Selenium's architecture allows for flexibility and extensibility, enabling users to customize and extend its functionality to suit their specific testing needs.

Overall, Selenium's open-source nature, cross-browser compatibility, language support, community support, and flexibility make it a preferred choice for many testers over QTP/UFT.