Module-5

1.Which components have you used in Load Runner?

Ans: 1. **Virtual User Generator (VuGen)**:

* + VuGen is used to record user actions on client applications or websites and generate scripts that simulate user behavior.

1. **Controller**:
   * The Controller is used to manage and control the execution of load tests. It orchestrates multiple virtual users to simulate real-world user behavior.
2. **Load Generator**:
   * Load Generators are used to simulate virtual users (VUsers) and generate load on the system under test. They execute the scripts created in VuGen and generate the load as specified in the test scenario.
3. **Analysis**:
   * The Analysis component is used to analyze and interpret the results of the load tests. It provides graphs, reports, and statistics to evaluate system performance, identify bottlenecks, and make recommendations for improvements.
4. **Monitoring**:
   * LoadRunner can monitor various system resources such as CPU usage, memory usage, network performance, etc., during the load test execution. This helps in correlating system behavior with performance metrics.
5. **Protocols**:
   * LoadRunner supports multiple protocols (such as HTTP, HTTPS, Web Services, Citrix, etc.) which define how virtual users interact with the application under test. Different protocols are used based on the nature of the application being tested.

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

Ans: In LoadRunner, the number of Virtual Users (VUsers) can be set using the LoadRunner Controller, which is the component responsible for managing and executing load tests.

3.What is Correlation?

Ans: In the context of performance testing and scripting, **correlation** refers to the process of capturing dynamic values from server responses and replacing them with unique parameters in subsequent requests. This is crucial in load testing tools like LoadRunner, JMeter, and others where scripts are used to simulate user interactions with web applications or services.

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

Ans: Developing a VUser script in LoadRunner involves several systematic steps to accurately simulate user interactions with an application or system under test. Here's a structured process for developing a VUser script:

**1. Understanding Requirements and Scenarios**

* **Requirement Analysis**: Gather detailed requirements from stakeholders regarding the application behavior to be tested, user scenarios, and expected performance metrics.
* **Scenario Definition**: Define the specific user scenarios that need to be simulated (e.g., login, search, purchase, logout) based on typical user behavior or specific business processes.

**2. Recording the Script**

* **Launch VuGen**: Start Virtual User Generator (VuGen), which is part of LoadRunner, and select the appropriate protocol (e.g., HTTP/HTTPS, Web Services, Citrix) based on the application under test.
* **Record Script**: Use VuGen to record the interactions with the application. During recording, perform the user actions that constitute the defined scenarios (e.g., navigating through pages, filling forms, submitting requests).

**3. Enhancing and Parameterizing the Script**

* **Handling Dynamic Data**: Identify and handle dynamic values such as session IDs, timestamps, and other parameters that need to be correlated. Use automatic or manual correlation techniques to replace these dynamic values with parameters.
* **Parameterization**: Replace hard-coded values in the script (e.g., usernames, passwords, search terms) with parameters. Parameterization allows varying inputs for different VUsers during test execution, making the simulation more realistic.
* **Adding Transactions and Timers**: Insert transactions to measure response times for specific actions or business processes. Add think time (delays) between actions to simulate realistic user pacing.

**4. Script Debugging and Validation**

* **Playback and Debugging**: Playback the script to ensure it accurately reproduces the recorded user actions without errors. Debug and fix any scripting errors or issues with correlation, parameterization, or logic.
* **Validation**: Validate that the script behaves as expected by reviewing server responses, ensuring proper handling of dynamic data, and verifying the correctness of parameterized values.

**5. Script Customization and Validation**

* **Customization**: Customize the script as needed to simulate specific scenarios or edge cases required by the performance testing objectives.
* **Validation**: Validate the customized script to ensure it accurately reflects the desired user behavior and interactions with the application under varying load conditions.

**6. Execution and Performance Monitoring**

* **Configuring LoadRunner Controller**: Use LoadRunner Controller to configure the test scenario, specifying the number of VUsers, load distribution, and ramp-up/ramp-down periods.
* **Executing the Test**: Run the load test to simulate multiple VUsers executing the script simultaneously. Monitor system resources, transaction response times, and other performance metrics during test execution.

**7. Analyzing Results**

* **Using LoadRunner Analysis**: After test execution, analyze the results using LoadRunner Analysis. Review graphs, reports, and statistics to identify performance bottlenecks, scalability issues, and areas for optimization.
* **Performance Tuning**: Based on analysis results, fine-tune the application or system configuration to improve performance, scalability, and reliability.

**8. Documentation and Reporting**

* **Documentation**: Document the VUser script, including details of scenarios, parameterization, correlations, and any customizations made.
* **Reporting**: Prepare comprehensive performance test reports summarizing test objectives, methodologies, results, findings, and recommendations for stakeholders.

5.How Load Runner interacts with the application?

Ans: LoadRunner interacts with the application under test in a manner that simulates real user behaviour while capturing performance metrics. Here’s how LoadRunner typically interacts with an application:

### 1. ****Recording User Interactions****

* **Protocol Selection**: LoadRunner supports various protocols such as HTTP/HTTPS, Web Services, JDBC, Citrix, etc. The protocol is chosen based on how the application communicates with clients.
* **Script Recording**: Using Virtual User Generator (VuGen), LoadRunner records interactions between the client (where VuGen runs) and the application. This includes HTTP requests, form submissions, clicks on UI elements, and other actions performed by the user.
* **Capturing Dynamic Data**: LoadRunner automatically captures dynamic data (like session IDs, cookies, tokens) during recording. It allows for correlation of these values to ensure scripts can handle unique data per VUser.

### 2. ****Script Execution****

* **Parameterization**: Scripts are parameterized to replace recorded values (like usernames, passwords, search terms) with variables. This allows for dynamic input and avoids script redundancy.
* **Think Time**: LoadRunner inserts think time between actions to simulate realistic user pacing. This is configurable to reflect typical user behavior.
* **Handling Dynamic Data**: Scripts handle dynamic data using correlation techniques. This involves identifying dynamic values in server responses and replacing them with parameters that vary across VUsers.

### 3. ****Load Generation and Distribution****

* **Load Configuration**: In LoadRunner Controller, scenarios are defined with the desired number of Virtual Users (VUsers) and other test parameters (e.g., ramp-up, duration).
* **Distribution of Load**: LoadRunner distributes VUsers across load generators, simulating simultaneous user activity from multiple locations or devices. This helps in evaluating how the application handles concurrent requests.

### 4. ****Monitoring and Metrics Collection****

* **Real-time Monitoring**: During test execution, LoadRunner monitors various metrics like response times, throughput, CPU usage, memory usage, and network performance.
* **Transaction Monitoring**: LoadRunner tracks transactions defined in scripts to measure the performance of specific user actions or business processes.

### 5. ****Analysis and Reporting****

* **Results Analysis**: After test completion, LoadRunner Analysis tool is used to analyze performance metrics collected during the test. This includes generating graphs, reports, and statistics.
* **Identifying Bottlenecks**: Analysis helps identify performance bottlenecks such as slow response times, high resource utilization, or scalability issues under load.

### 6. ****Integration and Collaboration****

* **Integration with CI/CD**: LoadRunner can integrate with CI/CD pipelines to automate performance testing as part of continuous integration and deployment processes.
* **Collaboration**: Results and reports from LoadRunner are shared with stakeholders to provide insights into application performance and recommendations for optimization.

6.How many VUsers are required for load testing?

Ans: Determining the number of Virtual Users (VUsers) required for load testing depends on several factors related to your specific testing objectives, application characteristics, and performance goals. Here are some considerations to help you determine the appropriate number of VUsers:

**1. Application Usage Profile**

* **User Base**: Understand the typical number of concurrent users expected to use the application during peak times. This could be based on historical data, user projections, or business requirements.
* **Scenarios**: Identify key user scenarios or business processes that need to be simulated during load testing. Different scenarios may require different numbers of VUsers.

**2. Performance Goals**

* **Response Time**: Determine acceptable response times for critical transactions or user actions. The number of VUsers should be adjusted to identify when response times degrade under load.
* **Throughput**: Define the expected throughput (transactions per second) that the application should handle without performance degradation.

**3. Scalability Testing**

* **Scalability Goals**: Test the application's scalability by gradually increasing the number of VUsers to determine at what point the application starts to experience performance issues or reaches its maximum capacity.

**4. Infrastructure and Environment**

* **Hardware and Network Configuration**: Consider the capacity and configuration of the servers, databases, network infrastructure, and load generators available for the test.
* **Test Environment**: Ensure the test environment mirrors the production environment as closely as possible in terms of configuration and resources.

**5. Testing Constraints**

* **Time and Resources**: Balance the number of VUsers with the available time and resources for conducting the load test. Consider constraints such as testing window, budget, and tool licensing limitations.

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

### Ans: Relationship between Response Time and Throughput

1. **Inverse Relationship**: In general, Response Time and Throughput have an inverse relationship. This means that as Throughput increases (more requests being processed per second), Response Time tends to increase.
   * **Reasoning**: When the system is handling a higher number of requests simultaneously (high Throughput), resources like CPU, memory, and network bandwidth may become more constrained. This can lead to increased queuing delays or processing times, resulting in higher Response Times for individual requests.
2. **Load Impact**: Under increasing load (higher Throughput), Response Time can spike as resources become saturated or contention for shared resources (such as database connections or CPU cycles) increases.
3. **Capacity Planning**: Understanding this relationship is crucial for capacity planning and performance tuning:
   * **Optimization**: Balancing system resources and optimizing configurations can help maintain acceptable Response Times even as Throughput increases.
   * **Threshold Identification**: Performance testing helps identify the maximum Throughput at which Response Time remains within acceptable limits. Beyond this point, Response Time may degrade rapidly, indicating a performance bottleneck.

8.What is the difference between hits/second and requests/second?

### Ans: Key Differences

* **Scope**: Hits/second includes all types of server accesses or interactions, while requests/second specifically counts HTTP requests for individual resources.
* **Granularity**: Hits/second can provide a broader view of overall server traffic, including all types of resource accesses. Requests/second focuses more narrowly on the number of distinct HTTP requests processed.
* **Application**: Hits/second is often used in web server logs and analytics tools to track general server usage and popularity of different resources. Requests/second is particularly useful for assessing the server's capacity to handle specific types of HTTP requests efficiently.

9.What is Automation Testing?

Ans: Automation testing refers to the use of specialized software tools and frameworks to automate the execution of tests on software applications or systems. It involves writing scripts or using automation tools to perform repetitive and often complex test cases automatically, rather than manually executing them.

10.Which Are The Browsers Supported By Selenium Ide?

Ans: Selenium IDE supports the following browsers:

1. **Mozilla Firefox**:
   * Selenium IDE primarily operates as a Firefox extension. It integrates directly into the Firefox browser, allowing testers to record and playback tests directly within the browser.
2. **Google Chrome**:
   * Selenium IDE is available as an extension for Google Chrome as well. Users can install Selenium IDE from the Chrome Web Store to perform test automation directly in Chrome.
3. **Microsoft Edge**:
   * Selenium IDE can also be used with Microsoft Edge. It is available as an extension in the Microsoft Edge Add-ons store, enabling testers to automate tests within the Edge browser.
4. **Other Browsers**:
   * While Selenium IDE has extensions for Firefox, Chrome, and Edge, support for other browsers may vary. As of the latest updates, these are the primary browsers officially supported by Selenium IDE.

11.What are the benefits of Automation Testing?

Ans: Here are some key benefits of automation testing:

1. **Improved Efficiency**:
   * **Faster Execution**: Automated tests can run much faster than manual tests, especially when dealing with repetitive test cases or large volumes of data.
   * **Parallel Execution**: Automation allows tests to be executed simultaneously on multiple machines or environments, accelerating the testing process.
   * **Continuous Integration/Continuous Deployment (CI/CD)**: Integrating automated tests into CI/CD pipelines enables frequent and rapid feedback on code changes, facilitating faster release cycles.
2. **Increased Test Coverage**:
   * **Comprehensive Testing**: Automation allows for the execution of a large number of test cases across different browsers, devices, and operating systems, ensuring broader test coverage.
   * **Regression Testing**: Automated tests can quickly and reliably verify that new code changes haven’t introduced unintended side effects or regression issues in existing functionality.
3. **Consistency and Accuracy**:
   * **Reduced Human Error**: Automated tests execute predefined steps precisely every time they run, reducing the risk of human error inherent in manual testing.
   * **Repeatability**: Tests can be rerun consistently under the same conditions to validate software behaviour and ensure consistent results.
4. **Cost-Effectiveness**:
   * **Resource Optimization**: While initial setup and maintenance of automated tests may require investment, automated testing ultimately reduces the need for manual testing resources over time.
   * **Early Bug Detection**: Automated tests help identify bugs and issues early in the development lifecycle when they are less costly and time-consuming to fix.
5. **Facilitates Continuous Testing**:
   * **Integration with DevOps**: Automation supports the principles of DevOps by providing continuous feedback on software quality throughout the development process.
   * **Continuous Improvement**: Automated tests can be integrated with monitoring tools to gather performance metrics and identify areas for improvement continuously.
6. **Supports Agile Practices**:
   * **Iterative Testing**: Automation enables frequent execution of tests in short development cycles typical of Agile methodologies, ensuring that software increments meet quality standards.
   * **Quick Feedback**: Automated tests provide rapid feedback to developers, enabling them to iterate quickly and address issues promptly.
7. **Enhances Test Reusability and Maintainability**:
   * **Modular Test Design**: Automation frameworks allow for modular and reusable test components, reducing duplication and maintenance efforts.
   * **Script Maintenance**: Changes in the application can be accommodated more efficiently through updates to automated test scripts, ensuring they remain aligned with application changes.

12.What are the advantages of Selenium?

Ans: Selenium is a widely used open-source toolset for automating web browsers across different platforms. It provides several advantages that make it popular among developers and testers for automating web applications:

1. **Open Source and Community Support**:
   * Selenium is open-source, freely available, and supported by a large community of developers and testers worldwide. This community actively contributes to its development, shares knowledge, and provides support through forums, blogs, and online resources.
2. **Browser Compatibility**:
   * Selenium supports multiple browsers including Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, and Opera. This cross-browser compatibility allows tests to be executed across different browsers without modifying the test scripts extensively.
3. **Platform Independence**:
   * Selenium supports various operating systems (Windows, macOS, Linux) and programming languages (Java, C#, Python, Ruby, JavaScript), making it highly versatile and suitable for diverse development environments.
4. **Flexibility and Extensibility**:
   * Selenium offers flexibility in terms of test scripting. It supports multiple programming languages and frameworks (like TestNG, JUnit, NUnit) for writing test scripts, allowing teams to choose tools and technologies that best fit their needs.
   * Selenium can also be extended with additional functionalities through plugins or integrations with other tools and frameworks, enhancing its capabilities for specific testing requirements.
5. **Supports Parallel Testing**:
   * Selenium Grid enables parallel test execution on multiple machines or virtual machines, allowing tests to run concurrently across different browsers, operating systems, and devices. This improves testing efficiency and reduces execution time significantly.
6. **Rich Set of Tools and APIs**:
   * Selenium offers a suite of tools:
     + **Selenium IDE**: A record-and-playback tool for creating simple scripts.
     + **Selenium WebDriver**: A powerful API for programmatically interacting with web browsers, enabling more complex and sophisticated test scenarios.
     + **Selenium Grid**: Facilitates parallel test execution across multiple browsers and platforms.
   * These tools cater to different aspects of automated testing, from rapid prototyping with Selenium IDE to scalable and distributed testing with Selenium Grid.
7. **Integration with CI/CD Pipelines**:
   * Selenium integrates seamlessly with Continuous Integration (CI) and Continuous Deployment (CD) pipelines. It can be integrated with popular CI/CD tools like Jenkins, Bamboo, and TeamCity to automate test execution as part of the build and deployment process.
8. **Wide Adoption and Industry Standard**:
   * Selenium is widely adopted across industries and is considered a de facto standard for web application testing. Its robustness, flexibility, and extensive community support make it a preferred choice for automated testing by organizations of all sizes.

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

Ans: Testers may opt for Selenium over QTP (QuickTest Professional, now known as Micro Focus UFT) for several reasons, depending on their specific needs, preferences, and the context of their testing environment. Here are some key factors that might influence this decision:

### 1. ****Open-Source vs. Commercial Tool****

* **Selenium**: Selenium is an open-source toolset, freely available for use. It appeals to organizations looking to minimize tooling costs and leverage community-driven support and development. It provides flexibility in terms of customization and integration with various development and testing frameworks.
* **QTP/UFT**: QTP/UFT is a commercial tool offered by Micro Focus. While it provides comprehensive features and support, it involves licensing costs that might be prohibitive for smaller teams or projects with budget constraints.

### 2. ****Cross-Browser and Platform Compatibility****

* **Selenium**: Selenium supports multiple browsers (Chrome, Firefox, Edge, Safari, Opera) and operating systems (Windows, macOS, Linux). It allows testers to write scripts once and execute them across different browsers and platforms without significant modifications.
* **QTP/UFT**: QTP/UFT historically focused more on Windows-based applications and browsers, although recent versions have expanded support. However, Selenium’s broad cross-browser compatibility and platform independence remain advantageous for testing web applications.

### 3. ****Programming Language Flexibility****

* **Selenium**: Selenium WebDriver supports multiple programming languages such as Java, C#, Python, Ruby, JavaScript (Node.js), and more. Testers can choose the language they are most comfortable with or align with their project’s tech stack.
* **QTP/UFT**: QTP/UFT primarily uses VBScript for scripting, which may limit flexibility for teams accustomed to other programming languages or those working in environments where specific languages are preferred or mandated.

### 4. ****Community Support and Ecosystem****

* **Selenium**: Selenium has a large and active open-source community. It offers extensive documentation, forums, user groups, and third-party plugins and integrations. The community contributes to ongoing development, bug fixes, and support resources.
* **QTP/UFT**: QTP/UFT also has a supportive community and vendor-provided support, but the size and openness of the community may not match the scale and collaborative nature of Selenium’s community-driven ecosystem.

### 5. ****Integration with CI/CD Pipelines****

* **Selenium**: Selenium integrates seamlessly with Continuous Integration (CI) and Continuous Deployment (CD) pipelines. It supports integration with popular CI/CD tools like Jenkins, Bamboo, TeamCity, etc., enabling automated execution of tests as part of the development workflow.
* **QTP/UFT**: QTP/UFT provides integration with CI/CD pipelines as well, but Selenium’s open-source nature and extensive integration capabilities may offer more flexibility and customization options.

### 6. ****Skill Set and Learning Curve****

* **Selenium**: Selenium WebDriver requires knowledge of programming languages and frameworks, which can be advantageous for testers who prefer coding and want to build deeper technical skills. It aligns well with the trend towards upskilling in automation and DevOps practices.
* **QTP/UFT**: QTP/UFT traditionally offered a more GUI-driven approach with VBScript, which may be easier for testers without strong programming backgrounds. However, Selenium’s WebDriver API and scripting capabilities appeal to testers looking for more control and customization.