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

**Q.1 Which componants have you used inLoad Runner?**

In LoadRunner, several components are used to simulate virtual users and measure system performance. The key components of LoadRunner are:

1. **Virtual User Generator (VuGen)**: This is the component used to create and record scripts that simulate the actions of real users interacting with the system. You can record user interactions with a variety of protocols, such as HTTP, Web Services, FTP, and more. After recording, scripts can be enhanced and parameterized to simulate different user behaviors.
2. **Controller**: The Controller is the component used to manage and orchestrate performance tests. It allows you to configure test scenarios, schedule the execution of virtual users (VUs), and monitor the execution of the test. You can also set up load generators and configure real-time monitoring during the test.
3. **Load Generators (LG)**: These are machines that execute the virtual user scripts created by VuGen. The load generators simulate the user load by generating traffic according to the scripts and scenarios defined in the Controller. Multiple load generators can be used for scaling the load.
4. **Analysis**: After a test is executed, the Analysis component is used to interpret the results and generate performance reports. The data collected during the test is analyzed to identify bottlenecks, response time issues, throughput, and other performance metrics. Reports can be customized to help with performance tuning and troubleshooting.
5. **SiteScope**: SiteScope is an optional monitoring tool that can be used in LoadRunner to monitor the performance of servers and infrastructure in real time. It provides system-level metrics such as CPU usage, memory, disk space, and network performance, which can be correlated with LoadRunner test results to identify performance issues.
6. **Agent**: The LoadRunner agent is a process that runs on the Load Generator machines to execute the scripts. The agent collects data during the test and reports it back to the Controller.

These components work together to create a full performance testing environment. The combination of VuGen for scripting, Controller for test orchestration, Load Generators for load execution, and Analysis for result interpretation enables comprehensive load testing and performance validation.

**Q.2 How can you set the number of Vusers in load runner?**

In LoadRunner, you can set the number of Virtual Users (Vusers) in the **Controller** component when configuring a test scenario. Here’s how you can set the number of Vusers:

1. **Open Load Runner Controller**:
   * Launch the **Controller** application where you will define the test scenario.
2. **Create a New Scenario**:
   * Select **New Scenario** or open an existing one.
   * If creating a new scenario, choose the test type (e.g., **Manual Scenario**, **Goal-Oriented Scenario**).
3. **Define the Number of Virtual Users (Vusers)**:
   * In the **Scenario** view, click on the **Vusers** tab.
   * You’ll see a section where you can define the number of Vusers. Here, you can specify the total number of virtual users (Vusers) that will be simulated in the test.

For **Manual Scenario**:

* + - Under **Vusers per Group**, you can specify the number of Vusers for each user group (a user group simulates a particular user behavior in the test). You can define the number of Vusers for each script and adjust the number based on the test requirements.

For **Goal-Oriented Scenario**:

In this case, you can define a **target goal** (such as response time or throughput) and then load the desired number of Vusers to meet that goal. The system will adjust the number of Vusers automatically during the test execution to meet the set goal.

1. **Distribute Vusers across Load Generators**:
   * You can also specify how to distribute the Vusers across different load generators. For example, if you have multiple machines (Load Generators), you can allocate a certain number of Vusers to each machine to scale the load as per the test requirements.
   * Under **Load Generators**, select the machine and specify the number of Vusers each machine will handle.
2. **Adjust Test Duration and Ramp-up**:
   * You can also define how the Vusers will be ramped up during the test. In the **Run Time Settings**, set the **Vuser start time** to specify how quickly the Vusers will be created (ramp-up period).
3. **Save and Start the Test**:
   * After configuring the number of Vusers, save your scenario.
   * Once everything is set up, click on **Start** to execute the test with the specified number of Vusers.

**Q.3 What is Correlation?**

**Correlation** in LoadRunner refers to the process of handling dynamic values (like session IDs, tokens, or timestamps) that change with each virtual user or each execution of a script during a load test. These values are typically returned by the server in response to a request and are required in subsequent requests to maintain the continuity and accuracy of the user session.

Without correlation, the script would fail because static values (like hard-coded session IDs or dynamic tokens) would be used, which would not match the ones generated during test execution. Correlation ensures that the values are dynamically captured during the script execution and used correctly in subsequent requests.

**Types of Correlation:**

1. **Automatic Correlation**:
   * LoadRunner can automatically detect and correlate dynamic values using its **Correlation Wizard**. This tool scans the response data for dynamic content and generates the appropriate correlation functions in the script.
   * You can enable this by choosing **Correlation** in the VuGen script and running the automatic correlation feature. LoadRunner will automatically insert correlation functions like web\_reg\_save\_param() or lr\_save\_param() to capture and use dynamic values.
2. **Manual Correlation**:
   * In some cases, automatic correlation might not detect all dynamic values, or it may incorrectly correlate values. In such cases, you can manually correlate the dynamic values.
   * Manual correlation involves identifying dynamic values in the server response and writing custom correlation code to capture those values. For example web\_reg\_save\_param("sessionID", "LB=SessionID=", "RB=</session>", LAST);
   * Here, web\_reg\_save\_param() is a function used to capture the session ID from the response and store it in a parameter (in this case, sessionID). This value can then be used in subsequent requests.

**Steps for Correlation:**

1. **Identify Dynamic Data**:
   * During the recording of the script, carefully review the server response to identify dynamic values that are likely to change with each test run (e.g., session IDs, user tokens, timestamps).
2. **Capture the Dynamic Data**:
   * Use correlation functions (either automatic or manual) to capture the dynamic values from the response and save them as parameters.
3. **Use Correlated Data**:
   * Once the dynamic value is captured and stored in a parameter, use that parameter in subsequent requests where that dynamic value is required. For example:

web\_submit\_data("submit", "Action={URL}/submit", "SessionID={sessionID}", LAST);

1. **Verify Correlation**:
   * After implementing correlation, run the script in **Replay Mode** to ensure that the dynamic values are correctly captured and used, and that the script functions as expected.

**Importance of Correlation:**

* **Ensures Accuracy**: Correlation helps to simulate real user behavior by dynamically handling values that change with each session.
* **Avoids Script Failure**: Without correlation, tests would fail because static values would no longer match what the server expects.
* **Scalability**: It allows the script to handle a large number of virtual users with different dynamic data values in each request.

In summary, correlation is an essential technique in LoadRunner for simulating realistic user behavior by dynamically handling changing values in the server responses during performance tests.

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

The process for developing a **Vuser Script** in LoadRunner involves several steps, from planning and recording user actions to enhancing and parameterizing the script to simulate realistic user behavior. Below is a step-by-step guide for developing a Vuser script in LoadRunner:

### Step-by-Step Process for Developing a Vuser Script:

#### 1. ****Identify the Test Requirements****

* Before creating a script, understand the **test objectives** (e.g., load testing, stress testing) and the **user scenario** (e.g., number of users, actions, and data flow).
* Determine which protocols (HTTP, FTP, Web Services, etc.) are being used in the application you are testing.

#### 2. ****Record the User Actions Using VuGen (Virtual User Generator)****

* Open **VuGen** (Virtual User Generator) to begin the script development process.
* Choose the **protocol** (e.g., HTTP, Web Services, etc.) based on the application you are testing.
* Click **Record** to start capturing the actions of a real user interacting with the application. For example, you may record a user logging in, browsing, submitting forms, or logging out.
* **During recording**, perform the tasks manually as a real user would do. VuGen captures each action, such as clicking buttons, entering data, and navigating between pages, and automatically generates a script that mimics these actions.
* After recording, stop the recording process. VuGen will generate a **script** in the scripting language (typically C or JavaScript) corresponding to the protocol you selected.

#### 3. ****Enhance the Script****

* **Script Validation**: After recording, replay the script to check that it works correctly and mimics the user flow.
* **Parameterization**: Replace hardcoded values (e.g., login credentials, item IDs) with parameters to simulate a variety of users with different data. For example:

c

CopyEdit

web\_submit\_data("submit", "Action={URL}/submit", "username={username}", LAST);

* **Correlation**: Dynamic values (such as session IDs, tokens) must be captured and handled through **correlation** to ensure that the correct values are used in subsequent requests.
  + Use the **Correlation Wizard** (automatic) or manually write correlation functions (web\_reg\_save\_param) to capture dynamic values returned by the server.
* **Error Handling**: Add error handling in the script to ensure the script continues to run smoothly, even when unexpected conditions occur.
* **Think Time**: Add **think time** (delays between user actions) to simulate real user behavior, making the test more realistic.

lr\_think\_time(5); // Pause for 5 seconds between actions

#### 4. ****Parameterization****

* **Data-Driven Testing**: Use different data sets for multiple users by parameterizing values like user IDs, passwords, or transaction amounts. This simulates multiple users performing the same actions with different data.
* LoadRunner provides a feature called **Parameter List** to associate dynamic data with parameters in the script.
* Example of parameterizing a username:

web\_submit\_data("login", "username={username}", "password={password}", LAST);

#### 5. ****Use Transaction and Custom Functions****

* **Transactions**: Use **transaction functions** to measure the response times of specific actions. For example, wrapping actions in lr\_start\_transaction and lr\_end\_transaction functions helps track how long a specific action takes.

lr\_start\_transaction ("login");

// Login actions

lr\_end\_transaction("login", LR\_AUTO);

* **Custom Functions**: You can also write custom functions to perform complex tasks or calculations, and reuse them across different scripts.

#### 6. ****Validate and Test the Script****

* **Replay** the script to ensure it works as expected, with all the dynamic values handled properly and the actions being executed as intended.
* Check for errors or failures during the replay. If there are issues, debug the script to resolve them (e.g., incorrect correlation, missing parameters, etc.).
* Adjust the script as necessary to handle different scenarios or edge cases.

#### 7. ****Parameterize Load and Run Time Settings****

* **Configure Load**: In **LoadRunner Controller**, set up the number of **Virtual Users (Vusers)**, **runtime settings**, and the **duration** of the test.
* Adjust the **think time**, **ramp-up period**, and **test duration** to simulate realistic user load and behavior.

#### 8. ****Execute the Test****

* Once you have validated and finalized the script, you can use **LoadRunner Controller** to create a test scenario and execute the script with multiple virtual users (Vusers).
* In the Controller, you will specify the load pattern, the number of Vusers, and the test duration. The Controller will manage the Vuser execution on multiple Load Generators, simulating real user traffic on the application.

#### 9. ****Monitor the Test Execution****

* During the test, monitor the performance of the application using the **Controller's real-time monitoring** capabilities.
* Track metrics such as **response time**, **throughput**, **error rates**, and **resource utilization** to evaluate how the system is handling the load.

#### 10. ****Analyze the Results****

* After completing the test, analyze the results using **LoadRunner Analysis**.
* Identify performance bottlenecks, issues with response time, system failures, or resource limitations.

### Summary of Steps:

1. **Identify Test Requirements**.
2. **Record User Actions** using VuGen.
3. **Enhance the Script** with correlation, parameterization, and error handling.
4. **Validate and Test** the script by replaying.
5. **Configure Load** and runtime settings in the Controller.
6. **Execute the Test** with multiple Vusers.
7. **Monitor the Test** in real time.
8. **Analyze the Results** for performance insights.

**Q.5 How LoadRunner interacts with the application?**

LoadRunner interacts with the application by simulating virtual users (Vusers) that generate load on the system. It does this by sending requests to the application and receiving responses, just like real users would do. The process of interaction varies depending on the protocol being used for testing (e.g., HTTP, Web Services, FTP), but the overall flow remains consistent.

Here’s a detailed explanation of how LoadRunner interacts with the application:

**1. Recording User Actions**

* **Virtual User Generator (VuGen)** records the actions performed by a real user interacting with the application. These actions are typically recorded as a series of requests (e.g., HTTP requests, database queries, etc.) that simulate user behavior, such as logging in, browsing, or submitting data.
* During recording, VuGen captures these actions and generates a script that mimics the behavior of the user, which will later be played back to simulate virtual users interacting with the application.

**2. Virtual User (Vuser) Simulation**

* Once the script is recorded and enhanced (e.g., by parameterizing values, adding correlation, or incorporating think time), LoadRunner uses **Virtual Users (Vusers)** to simulate multiple users interacting with the application.
* Each Vuser sends requests to the application, one by one, or in parallel, depending on the test configuration. The Vusers behave like real users but are automated and controlled by LoadRunner.

**3. Interaction with the Application**

LoadRunner interacts with the application in several ways, based on the protocol chosen and the type of requests being made:

* **HTTP/HTTPS Requests (for Web applications)**:
  + LoadRunner mimics HTTP requests to interact with the application. For example, when a Vuser accesses a website, LoadRunner sends HTTP GET requests to retrieve web pages, and POST requests to submit forms or data.
  + It interacts with the application in the same way a browser would, sending requests to the web server and receiving responses (e.g., HTML, images, JavaScript).
* **Web Services (SOAP, REST)**:
  + For web services testing, LoadRunner simulates requests such as SOAP or RESTful API calls. It sends data to the server, and the server responds with data, which can be validated for correctness and performance.
  + LoadRunner can simulate various API operations like GET, POST, PUT, DELETE, etc.
* **Databases (e.g., SQL Server, Oracle)**:
  + LoadRunner can also simulate database transactions (e.g., SQL queries) by interacting with the database server. It sends SQL commands, such as SELECT, INSERT, UPDATE, DELETE, and receives responses.
* **FTP**:
  + LoadRunner can simulate file transfers via FTP (File Transfer Protocol). It sends commands like **PUT**, **GET**, **DELETE**, and handles file transfers to and from the FTP server.
* **Other Protocols**:
  + LoadRunner supports a wide range of protocols like **JDBC**, **Citrix**, **SAP**, **Oracle**, and others, depending on the application type. Each protocol defines how LoadRunner interacts with the application, whether it's sending commands to a database, interacting with an SAP server, or managing remote desktop sessions via Citrix.

**4. Controller and Load Generators**

* **Controller**:
  + The LoadRunner **Controller** is responsible for orchestrating the entire load test. It controls the execution of Vusers by managing the number of virtual users, their behavior, and how they interact with the application. The Controller communicates with the **Load Generators** to send Vuser scripts and monitor the test.
* **Load Generators**:
  + Load Generators are the machines that actually execute the Vuser scripts. They simulate the Vuser interactions with the application (sending requests, receiving responses, etc.). The more load generators you have, the more Vusers you can simulate.
  + Load Generators can interact with the application by sending requests to the application servers based on the Vuser scripts. For example, if you're testing a web application, the Load Generator will send HTTP requests as defined in the script.

**5. Request/Response Flow**

* **Request**:
  + A Vuser sends a request to the application (for example, an HTTP GET or POST request). The request includes specific information, such as the URL, headers, parameters, and any session-related data (like cookies or tokens).
* **Response**:
  + The application processes the request and sends back a response, which can include HTML content, JSON data, or other resources. The response may also contain dynamic content (e.g., session tokens, transaction IDs) that need to be correlated in the script.
* LoadRunner captures the response and checks the status, such as whether it was successful or if there were any errors. It can also record response times and throughput to evaluate performance.

**6. Handling Dynamic Data (Correlation)**

* Many applications generate dynamic data in responses (e.g., session IDs, tokens) that need to be captured and used in subsequent requests. This dynamic data is handled through **correlation**.
  + Load Runner can automatically correlate dynamic data (using **Correlation Wizard**) or it can be manually handled by the tester using correlation functions like web\_reg\_save\_param() or lr\_save\_param().
* For example, when a Vuser logs into a website, the server might return a session ID in the response. The script needs to capture this session ID and use it in future requests to maintain the user's session.

**7. Think Time and User Behavior Simulation**

* LoadRunner adds **think time** between actions to simulate realistic user behavior. Think time is the pause between actions to represent real user delays as they navigate the application.
* It helps simulate user interaction patterns like reading a page, filling out a form, or waiting for a page to load.

**8. Real-Time Monitoring**

* During the test execution, LoadRunner can monitor the application’s performance by tracking various metrics (e.g., response time, throughput, resource usage).
* The Controller provides real-time feedback on the status of the Vusers and allows you to monitor the application's behavior while the load test is running.

**9. Data Collection and Analysis**

* After the test execution, LoadRunner collects the results and generates performance metrics.
* **Analysis** component in LoadRunner is used to interpret the collected data, such as response times, throughput, error rates, and system resource utilization.
* The analysis helps identify performance bottlenecks, latency, or areas that need optimization.

**Summary of Interaction Flow:**

1. **Recording**: LoadRunner records user actions (requests and responses).
2. **Scripting**: The recorded actions are translated into a Vuser script.
3. **Virtual Users**: Multiple Vusers simulate real user traffic by executing the script.
4. **Requests/Responses**: Vusers send requests to the application and receive responses.
5. **Correlation**: Dynamic data (like session IDs) is captured and used in subsequent requests.
6. **Load Generation**: Load Generators execute the scripts and simulate the load on the application.
7. **Monitoring**: Real-time metrics (response time, resource usage) are monitored during the test.
8. **Analysis**: After the test, performance data is analyzed to identify issues.

**Q.6 What is the relation between response time and Throughput?**

The relationship between **response time** and **throughput** is fundamental in performance testing and reflects how the system behaves under load. These two metrics are related but represent different aspects of application performance. Here’s a breakdown of each and how they interact:

**1. Response Time**

* **Definition**: Response time is the amount of time it takes for the application to respond to a request from a virtual user (Vuser). It’s measured from the time the request is sent until the response is received. It reflects the **delay** experienced by the user when interacting with the system.
* **Key Point**: Response time includes both the time taken by the system to process the request and any delays in the network or infrastructure.
* **Units**: Typically measured in **milliseconds (ms)** or **seconds**.
* **Components of Response Time**:
  + **Server Processing Time**: Time taken by the server to process the request.
  + **Network Time**: Time for the request to travel over the network and the response to return to the client.
  + **User's Think Time**: Sometimes considered as part of user behavior, it’s the time between interactions of a user (e.g., pauses between page loads).

**2. Throughput**

* **Definition**: Throughput is the amount of data sent or received by the system per unit of time. It represents the **volume of data** transferred between the client and server, or the **number of transactions** processed in a given time period.
* **Key Point**: Throughput measures the system’s **capacity** to handle requests, often expressed in terms of **requests per second** (RPS) or **bytes per second** (BPS).
* **Units**: Typically measured in **requests per second (RPS)** or **bytes per second (BPS)**.

**Relationship Between Response Time and Throughput**

1. **Inverse Relationship at High Load**:
   * **When the load increases (i.e., more Vusers or requests are simulated)**, throughput generally increases because the system is processing more requests over time.
   * However, as the system gets more load, **response time tends to increase**. This is because the server is processing more requests, and resources become more strained, causing delays in processing each request.
   * For example:
     + At low load, response time might be low, and throughput could be moderate.
     + At higher load, throughput increases (more data is transferred), but response time increases as the system becomes more stressed and is unable to process requests quickly.
2. **Optimal Throughput vs. Acceptable Response Time**:
   * **Throughput** is an indicator of how well the system can handle load, while **response time** shows how quickly the system responds under that load.
   * There is an **optimal point** where you want the maximum throughput with **acceptable response time**. However, there is a point where increasing throughput further will result in a significant increase in response time, which may indicate that the system is overloaded.
   * For example:
     + If you push too many virtual users (Vusers) or transactions on the system, throughput might continue to increase initially, but at some point, the system cannot keep up, and response time sharply increases.
     + This is a sign of **resource saturation** or a **bottleneck** in the system.
3. **Throughput as a Key Driver of Response Time**:
   * Higher throughput means that more requests are being processed by the server. If the server is not scaled or optimized to handle this increased traffic, the response time for individual requests will rise.
   * If a system has **good throughput** but poor response time, it may be processing many requests, but not efficiently. This could be due to poor server performance, slow database queries, or inefficient code.
4. **Impact of Latency**:
   * **Latency** (network delay) can affect both response time and throughput. Increased latency can slow down the overall response time and can also reduce the throughput (as it will take longer for each request/response to complete). For example, with high network latency, even though the system can process many requests per second, the network delays slow down how fast those requests can be made.

**Graphical Relationship**

In performance testing, you might see a graph where:

* **X-axis**: Represents the **load** (number of Vusers or requests per second).
* **Y-axis (1)**: Represents **response time** (typically in ms).
* **Y-axis (2)**: Represents **throughput** (typically in RPS or BPS).

As you increase load:

* **Throughput** increases gradually, reflecting the system’s ability to handle more requests.
* **Response time** increases after a certain point, indicating that the system is becoming stressed or approaching its limit.

**Summary:**

* **Response time**: How long it takes for the application to respond to a request.
* **Throughput**: The volume of data or number of requests handled by the system per unit of time.
* At higher loads, **throughput** increases but **response time** typically increases as well due to system strain.
* There is a balance to be found between maintaining high throughput and keeping response time within acceptable limits.
* Too high a load can push the system beyond its capacity, resulting in poor performance (increased response times and potential system failure).