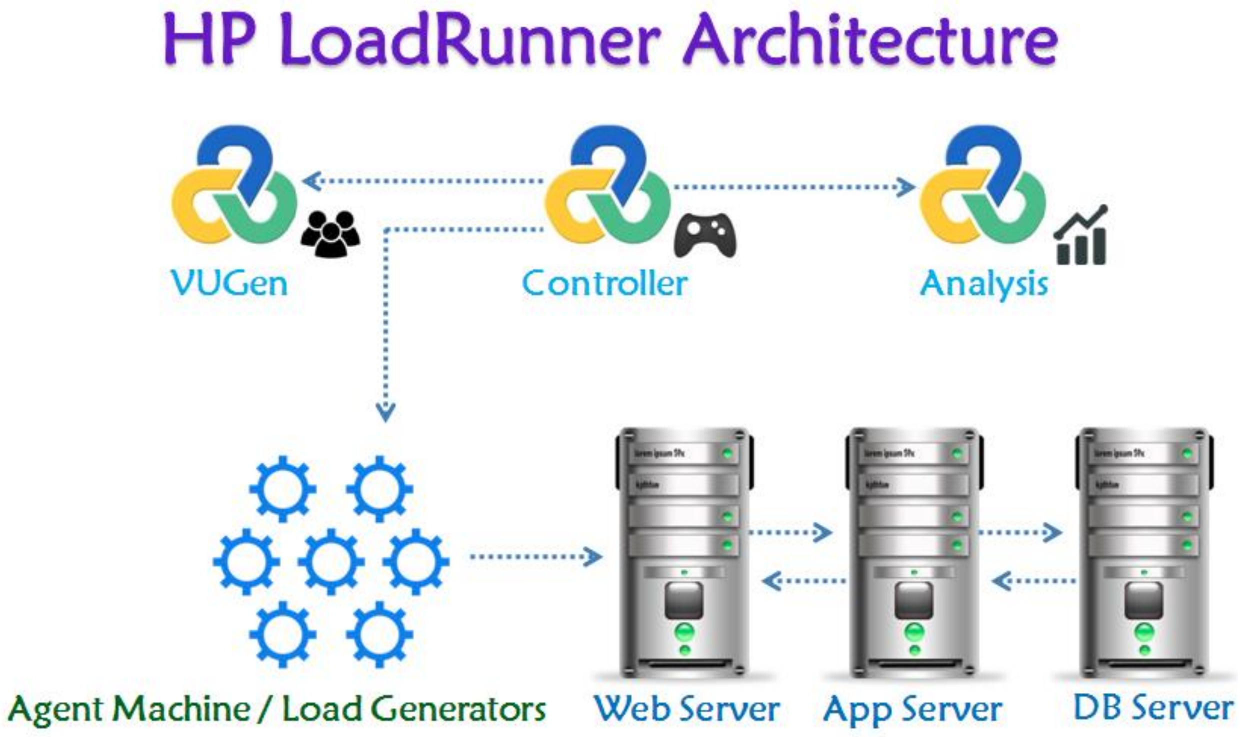
Automation Core Testing (Load Runner Up and Selenium IDE)

Which components have you used in Load Runner?

LoadRunner is a performance testing tool that uses several components to simulate user activity, measure system performance, and analyze results. Some of the key components used in LoadRunner include:

1. Virtual User (Vuser) scripts
2. Controller
3. Load Generators
4. Analysis

These components work together to simulate user activity, measure performance, and analyze results.



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

In LoadRunner, we can set the number of Vusers in the following ways:

- **Controller:** In the Controller, go to "Scenario" > "Vusers" and set the "Number of Vusers" field.
- **Runtime Settings:** In the Vuser script, go to "Runtime Settings" > "General" and set the "Number of Vusers" field.
- **Command Line:** Use the command line option "-n" followed by the number of Vusers, for example: "lr -n 10".

Note: The number of Vusers can also be controlled through the LoadRunner API and through external automation tools.

What is Correlation?

correlation is the process of capturing and replacing dynamic values in a script with unique values for each virtual user (Vuser). This ensures that each Vuser interacts with the application independently, simulating real-user behavior.

- Handle session IDs, tokens, and other unique identifiers
- Simulate unique user inputs, like usernames and passwords
- Capture and reuse dynamic data, such as timestamps or order numbers

By correlating dynamic values, you ensure that your load test accurately reflects real-world usage and avoids errors caused by duplicate or hardcoded values.

What is the process for developing a Vuser Script?

The process for developing a Vuser script in LoadRunner is:

- **Record:** Capture user interactions with the application using LoadRunner's recording tool.
- **Analyze:** Review the recorded script, identify dynamic values, and determine what needs to be correlated.
- **Parameterize:** Replace dynamic values with parameters to enable unique data for each Vuser.
- **Correlate:** Capture and replace dynamic values with correlated data.
- **Script Enhancement:** Add think time, loops, conditional statements, and other logic to simulate realistic user behavior.
- **Verification:** Validate the script's functionality and accuracy.
- **Debug:** Test and debug the script to ensure it runs smoothly.
- **Finalize:** Prepare the script for load testing by setting runtime settings and configuring logging.

This process helps create a robust Vuser script that accurately simulates real-user interactions with the application.

When to used Usablity Testing?

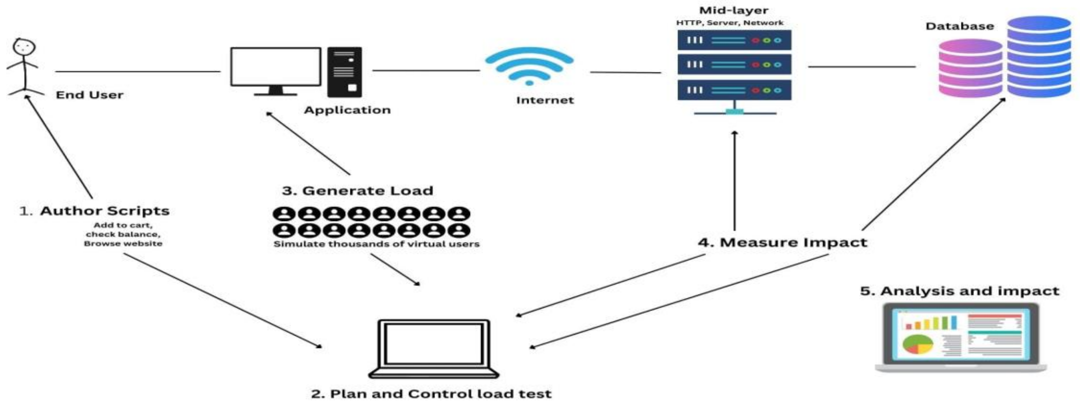
Usability testing helps ensure a positive user experience, identify areas for improvement, and inform design decisions throughout the product development lifecycle.

How Load Runner interacts with the application?

LoadRunner uses software agents called "protocols" to communicate with the application. These protocols mimic real-user interactions by sending requests to the application's servers and receiving responses. The protocols used depend on the application's architecture and technology stack.

Here's how it works:

1. **application Selection:** LoadRunner selects the appropriate protocol (e.g., HTTP, Web Services, Citrix) based on the application's technology.
2. **Script Recording:** LoadRunner records user interactions with the application using the selected protocol, capturing requests and responses.
3. **Script Replay:** During load testing, LoadRunner replays the recorded script, sending requests to the application's servers using the same protocol.
4. **Request/Response:** The application processes the requests and sends responses back to LoadRunner, which analyzes the responses to verify functionality and performance.
5. **Data Correlation:** LoadRunner correlates dynamic data, such as session IDs or tokens, to ensure each virtual user interacts with the application independently.
6. **Generate Load:** LoadRunner generates a large volume of virtual users, each executing the script and interacting with the application, simulating real-world usage.



How many VUsers are required for load testing?

The number of VUsers (Virtual Users) required for load testing depends on several factors, including:

1. Expected user load: Estimate the number of real users who will be using the application simultaneously.

2. Application complexity: More complex applications may require more VUsers to simulate realistic usage.

3. Test goals: Identify what you want to achieve with load testing (e.g., peak performance, stress testing).

4. Hardware resources: Ensure the load testing environment can handle the desired number of VUsers.

Here are some general guidelines:

- **Low-load testing:** 10-50 VUsers (e.g., small applications, development testing)
- **Medium-load testing (Stress):** 50-200 VUsers (e.g., medium-sized applications, performance testing)
- **High-load testing (Scalability):** 200-1,000 VUsers (e.g., large applications, stress testing)
- **Extreme-load testing (flood):** 1,000+ VUsers (e.g., very large applications, extreme stress testing)

Remember, the key is to simulate realistic user behavior and gradually increase the load to measure the application's performance and identify bottlenecks.

What is the relationship between Response Time and Throughput?

Response Time and Throughput are two related but distinct performance metrics:

- Response Time: The time it takes for an application to respond to a user request or action. It measures how long a user must wait for a response.

- Throughput: The number of user requests or actions an application can handle within a given time period (e.g., transactions per second).

The relationship between Response Time and Throughput is:

- Inverse correlation: As Throughput increases (more requests handled), Response Time typically decreases (faster responses).
- Trade-off: Improving one metric can impact the other. For example, optimizing for faster Response Times might reduce Throughput, and vice versa.
- Balance: Aim for a balance between Response Time and Throughput to ensure a good user experience and efficient system performance.
- In load testing, analyzing both Response Time and Throughput helps identify performance bottlenecks and optimize application performance.