What is Error, Defect, Bug and failure?

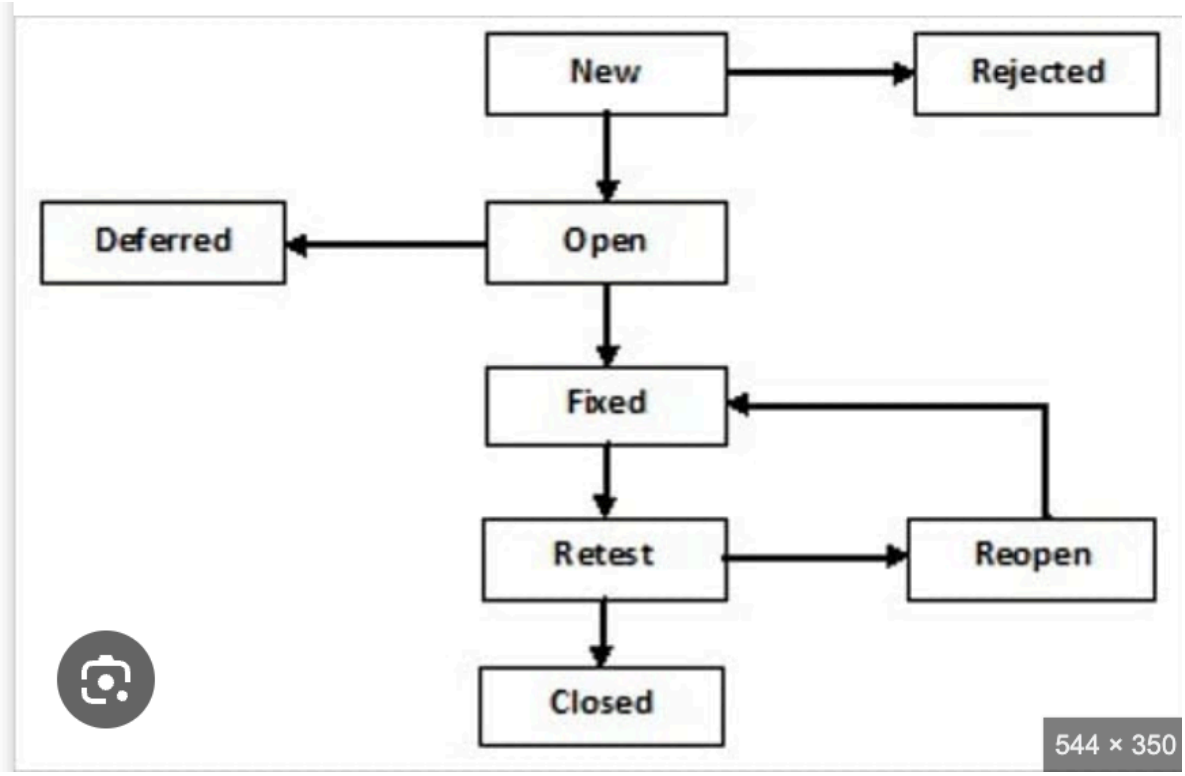
- Error: A mistake in coding is error.
- Defect: An error found by tester is defect.
- Bug: Defect accepted by development team is bug.
- Failure: Build does not meet the requirements is failure.

Difference between Priority and Severity

Priority	Severity
If you are raising any bug for any application, how soon you want the developer to fix that bug is called priority.	The impact of Defect /bug on the customer business workflow is known as Severity.
Typically determined by QA testers or developers based on the technical impact.	Typically determined by project managers, product owners, or business stakeholders
Severity affects how much a defect disrupts the system's core functionality.	Priority affects the order in which defects are fixed, considering business and user needs.

What is Bug Life Cycle?

The duration or timespan between the first time defect found and the time when defect closed, rejected, differed or postponed is called defect life cycle.



New:The defect is newly created and reported by the tester. It has not yet been reviewed or acknowledged.

Assigned:he defect is reviewed by the Test Lead or Project Manager, and it is assigned to the developer or responsible person for resolution.

Open:The defect has been acknowledged by the developer, and they are actively working on fixing it.

Fixed:The developer has fixed the defect by modifying the code or making necessary changes to the system.

Pending Retest: developer gives the system for retesting.

Retest:The defect fix is verified by the tester in the QA environment.

Verified:

What is priority?

If you are raising any bug for any application, how soon you want the developer to fix that bug is called priority. It indicates the urgency of addressing the bug, and is often determined by the business, project deadlines, or release schedules. High-priority defects are typically fixed sooner, while low-priority defects may be addressed later.

High Priority: The defect needs to be fixed immediately or in the next release because it affects the product's core functionality or user experience, and needs to be resolved urgently.

Medium Priority: The defect should be fixed, but it can wait for the next release or for a minor update. It does not severely impact functionality, but it’s still important to address.

Low Priority: The defect does not need immediate attention and can be fixed later. These bugs typically have minimal impact on the overall functionality or are considered cosmetic in nature.

What is severity?

The impact of Defect /bug on the customer business workflow is known as Severity.It describes how critical or serious the defect is in relation to the system's behavior and user experience. A higher severity typically

means that the bug causes significant issues or failures, while a lower severity means that the bug has a relatively minor effect.

High Severity: The defect causes a major failure or critical issue (e.g., a system crash, loss of data, or an issue that prevents a key feature from working). It may prevent the user from using the software or cause it to behave incorrectly in important situations.

Medium Severity: The defect causes a less critical issue but still impacts functionality, possibly affecting some features or part of the system, but the software can still be used with workarounds.

Low Severity: The defect is minor and does not affect the overall system's functionality. It could be a cosmetic issue, such as UI misalignment or a minor visual glitch that does not impact the user experience significantly.

Bug categories are…

Defects can be categorized into different types basing on the core issues they address.

Functionality Defects:Defects directly related to functionalities.

Performance Defects:Software doesn’t meet the expected performance requirements.

User Interface Defects:Difficult to operate for the users. Not user friendly.

Security Defects:Software doesn’t protect the user’s data privacy.

Compatibility Defects:Software does not work correctly on different hardware and software configuration.

Advantage of Bugzila .

Difference between priority and severity

priority	severity
Refers to the impact of the defect on the system or functionality.	Refers to how soon the defect needs to be fixed based on business needs

priority	severity
Typically determined by testers or developers.	Typically determined by project managers, product owners, or stakeholders.
Affects the ability to conduct further testing.	May not impact testing but affects when the issue is resolved.
Focuses on the technical impact of the defect.	Focuses on the business or delivery aspect of the defect.

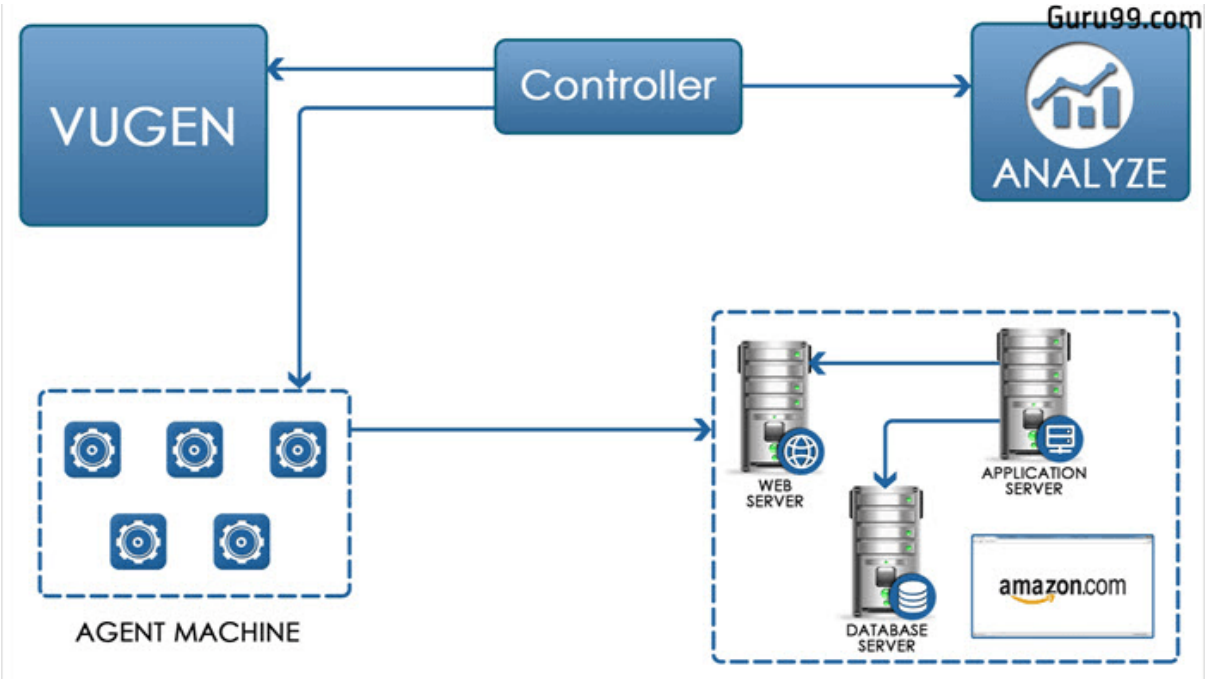
Which components have you used in Load Runner?

The main components of Load Runner, a performance testing tool, are:

Load Generator: Generates load on the application by following scripts. It simulates multiple virtual users to create realistic user scenarios.

Controller: Controls, launches, and sequences instances of Load Generator. It defines scenarios, allocates resources, and monitors test execution.

Analysis: Assembles logs from various load generators and formats reports for visualization of run result data and monitoring data. It creates performance reports in various formats, such as HTML, Excel, Word, PDF, or custom formats.



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

LoadRunner, you can set the number of virtual users (Vusers) in a few ways:

Add a Vuser group:

In the Run Dashboard, select More > Add Group. Then, enter the number of Vusers to run.

Edit a Vuser group:

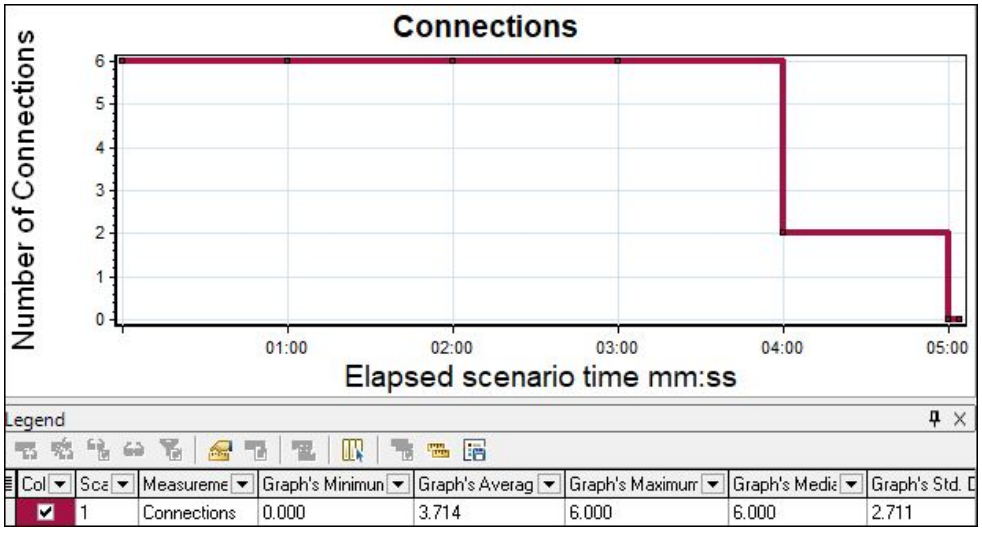
In the Run Dashboard, open the Groups tab, hover over the group name, and select Edit Group. Then, edit the number of Vusers assigned to the group.

Distribute Vusers by number:

In the Performance Test Designer window, select a group in the Groups grid. Then, enter the number of Vusers to allocate to that group in the Vusers column.

Define a schedule:

Define the number of Vusers when you define the test schedule.



What is Correlation?

Correlation in Load Runner is a process that captures dynamic values returned by a server and passes them to subsequent requests. This is useful when a test script has dynamic values or generates them during execution. Correlation ensures that the test script executes without errors.

Automatic correlation
Load Runner can automatically perform correlation.
Manual correlation
Users can manually add correlations that the automatic engine missed, or identify all correlations themselves.

What is the process for developing a Vuser Script?

Step 1- Record the Vuser Script.
Step 2- Playback and improve the recorded vuser script.
Step 3- Define and test the different run-time parameters.
Step 4- Use the script in a LoadRunner scenario.

How Load Runner interacts with the application?

Load Runner interacts with the application under test by simulating virtual users (Vusers) that generate load and perform actions on the application.

It helps test the application's performance under various conditions, such as heavy traffic or peak usage, to identify potential bottlenecks and areas of improvement.

1. Recording Vuser Scripts
2. Creating Virtual Users (Vusers)
3. Running Load Tests
4. Protocol-Specific Communication
5. Monitoring Performance
6. Analysis and Reporting
7. Error Handling and Recovery

how many VUsers are required for load testing?

The numbers of virtual users needed for load testing depends on several factors, including the test script, the number of test engine instances, and desired load.

Test script: The number of VUsers in the test script.

Test engine instances: The number of test engine instances.

Desired load: The number of visits per hour, the average session duration, and the desired number of VUsers.

What is the relationship between Response Time and Throughput?

Response Time: - The time it takes for a system to process a request and return a result.

Throughput: - The number of requests a system can handle per unit of time.

The correlation between response time and throughput is especially clear when requests are sequential. In this case, longer response times mean lower throughput, which indicates that the system can't handle many transactions.

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.