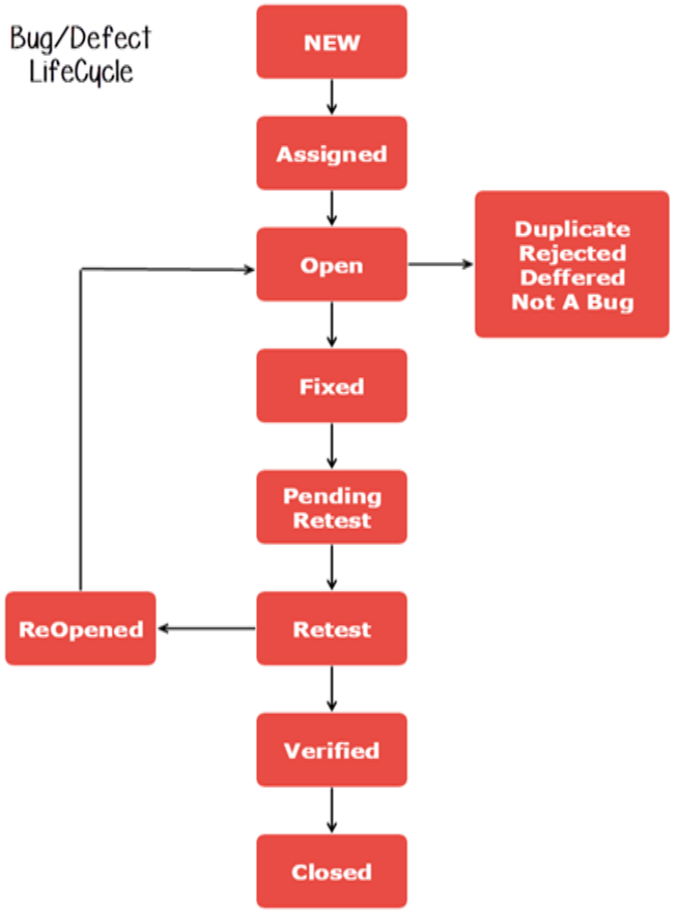
Software Testing Assignment

Module: -4

(1) What is Bug Life Cycle?

- A computer bug is an error, flaw, mistake, failure or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes.
- When a bug is discovered, it goes through several states and eventually reaches one of the terminal states, where it becomes inactive and closed.
- Error made by people in either a program's source code or its design.
- The process by which the defect moves through the life cycle is depicted next slide.



(2) What is priority?

- Priority is term that defines how fast we need to fix a defect.
- Priority is a parameter to decide the order in which defects should be fixed.
- Priority relates to the scheduling of defects to resolve them in software.
- The value of priority is subjective.
- The value of priority changes from time to time.
- The product manager basically decides a defects priority level.

(3) What is severity?

- Severity is a term that denotes how severely a defect can affect the functionality of the software.
- Severity is basically a parameter that denotes the total impact of a given defect on any software.
- Severity relates to the standards of quality.
- The value of severity is objective.
- The value of severity changes continually from time to time.

(4) Which components have you used in Load Runner?

- **Virtual User Generator (VuGen):** This component is used to record user actions on a web or mobile application and generate scripts that simulate user behaviour during performance testing.
- **Controller:** The Controller component is used to design and execute performance tests. It allows users to define scenarios, specify the number of virtual users, and distribute the load across multiple machines.
- **Load Generators:** Load Generators are responsible for generating the load on the system under test. They execute the scripts created in VuGen and simulate the behaviour of multiple virtual users accessing the application simultaneously.

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

1. **Open Load Runner Controller**: Launch Load Runner and open the Controller tool.
2. **Create a Scenario**: In the Controller, you need to create a scenario. A scenario defines the Vusers, scripts, load generators, and other settings for your performance test.
3. **Define Vusers**: Once you have created a scenario, you can define the number of Vusers you want to simulate. Typically, this is done in the "Vuser Groups" section of the scenario.
4. **Set Vuser Properties**: After defining the number of Vusers, you can further customize their behaviour by setting various properties such as script, load distribution, and pacing.
5. **Start the Test**: Once you have configured the scenario with the desired number of Vusers and other settings, you can start the test execution.
6. **Monitor Performance**: During the test execution, you can monitor the performance metrics such as response times, throughput, and server resources to analyse the behaviour of your application under load.
7. **Analyse Results**: After the test completes, you can analyse the results to identify any performance bottlenecks or issues.

(6) What is Correlation?

- Correlation is a process used in performance testing, particularly in scenarios involving web applications, to handle dynamic data that changes with each transaction. When a user interacts with a web application, certain data values, such as session IDs, transaction IDs, or unique identifiers, are generated dynamically by the server and included in subsequent requests. Correlation is the process of identifying these dynamic values and replacing them with parameters or variables so that the script can be replayed accurately.

(7) What is the process for developing a Vuser Script?

There is a simplified step-by-step process for developing a Vuser script:

- **Record the Script**: Use the tool's recording feature to capture user interactions with the application. Simply navigate through the application as a user would while the tool records the actions.
- **Enhance the Script**: Review the recorded script and make any necessary adjustments for accuracy. This might include adding think time between actions, handling dynamic data, and inserting validation points.

- **Parameterization**: Identify any dynamic values in the script (e.g., usernames, session IDs) and replace them with parameters. This allows for flexibility in testing with different data sets.
- **Correlation**: If the script contains dynamic data that changes with each transaction, perform correlation to replace these values with parameters or variables.
- **Validation**: Add validation points to the script to verify that the application responds correctly to user actions. This ensures that the script accurately simulates user behaviour.
- **Customization (Optional)**: Depending on your testing requirements, you may need to customize the script further. This could include adding transactions, simulating different user scenarios, or configuring the script for specific test conditions.
- **Test Execution**: Execute the script using the tool's testing capabilities. This simulates multiple virtual users interacting with the application simultaneously to generate load.
- **Results Analysis**: Analyse the test results to identify performance issues, errors, and areas for improvement. Use the performance metrics collected during the test to assess the application's performance under load.
- **Iterative Improvement**: Based on the results analysis, make any necessary adjustments to the script or test scenario to address issues or optimize performance. Script development is often an iterative process, so be prepared to refine the script as needed.

(8) How Load Runner interacts with the application?

- Load Runner interacts with the application under test in a manner that closely mirrors real user behaviour while rigorously evaluating the application's performance under varying load conditions.
- Initially, Load Runner captures user interactions with the application through script recording, acting as an intermediary between the client and server to intercept HTTP/S requests and responses.
- Following recording, Load Runner enhances the script by incorporating realistic elements such as think time to mimic user delays and parameterization to handle dynamic data.

- Additionally, Load Runner's correlation capabilities enable the identification and replacement of dynamic values, ensuring script accuracy during execution. During script execution, Load Runner simulates multiple virtual users executing the script concurrently, orchestrated by its Controller component.
- This load generation process provides insights into the application's scalability and performance under different load scenarios. Load Runner's comprehensive monitoring capabilities track various performance metrics, including response times, throughput, and server resource utilization, allowing testers to pinpoint performance bottlenecks and areas for improvement.
- Ultimately, Load Runner's results analysis tools enable testers to derive actionable insights from test results, facilitating informed decisions to optimize the application's performance and enhance user experience.
- Through these interactions, Load Runner serves as a robust platform for load testing, ensuring the reliability and efficiency of applications in real-world usage scenarios.

(9) How many Vusers are required for load testing?

- The number of Vusers required for load testing depends on several factors, including the goals of the test, the complexity of the application, the expected user load, and the available resources. There's no one-size-fits-all answer to this question, as it varies from application to application and depends on the specific testing objectives.
- Example like:

  Time taken for a single VUser to complete the operations = [a] + [b] + [c] = 10 second in 1 hour, the single users can simulate (60*60)/10 = 360 users transactions To simulate 25,000 users transaction = 25,000/360 ~ 70 VUsers To simulate a load test for this scenario for one hour, 105 VUsers (14 + 21 + 70) are required.

(10) What is the relationship between Response Time and Throughput?

- Response time and throughput are both important metrics used to evaluate the performance of systems, especially in computing and networking contexts. They are related but measure different aspects of system performance.

1. **Response Time**: Response time, also known as latency, refers to the time taken for a system to respond to a request or perform a task. It typically includes the time spent processing the request as well as any time spent

waiting in queues or in transit. In computing, response time is often measured in milliseconds or seconds.

2. **Throughput**: Throughput, on the other hand, refers to the rate at which a system can process tasks or requests over a given period of time. It is often measured in terms of the number of tasks completed or data transferred per unit of time, such as requests per second or bytes per second.

- 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 decreases (i.e., the system becomes more responsive), throughput often increases because the system can process more tasks in a given time period.

- **Trade-off:** However, achieving low response times may require allocating more resources or optimizing system components, which can impact throughput. Conversely, maximizing throughput may lead to higher response times due to increased contention for resources or higher processing loads.

- **Optimization:** System designers often need to strike a balance between response time and throughput based on the specific requirements of the application or system. For example, in real-time systems, minimizing response time may be critical, even if it means sacrificing some throughput. In contrast, in batch processing systems, maximizing throughput may be a higher priority, even if it means slightly higher response times.