Performance tests

→Performance is concerned with achieving response times, throughput, and resource utilization levels that meet the performance objectives for the project or product.

- Performance testing: Determining or validating the speed, scalability, and stability characteristics of the system or application under test.
- Load testing: Determining or validating performance characteristics of the system or application under test when subjected to workloads and load volumes anticipated during production operations.

- Stress testing: Determining or validating performance characteristics of the system or application under stressful conditions, such as limited memory, insufficient disk space, or server failure.

- With the help of performance testing, a product is assessed to determine its readiness for release, evaluated against performance criteria, compared to other systems, and identify sources of performance problems.

Basic activities in performance testing

1. Identify the test environment: During this stage, test environments(physical and production), tools and resources available to the test team are identified.
2. Identify Performance Acceptance Criteria: response time, throughput, and resource utilization goals and constraints should be identified.

→In general, response time is a user concern, throughput is a business concern, and resource utilization is a system concern.

3. Plan and Design Tests: Identify key scenarios, define test data, and establish metrics to be collected.

4. Configure the Test Environment: Prepare the test environment, tools, and resources necessary to execute each strategy as features and components become available for test. Ensure that the test environment is instrumented for resource monitoring as necessary.

5. Execute the Test: Run and monitor your tests. Validate the tests, test data, and results collection. Execute validated tests for analysis while monitoring the test and the test environment.

6. Analyze Results, Report, and Retest: Consolidate and share results data. Analyze the data both individually and as a cross-functional team. Reprioritize the remaining tests and re-execute them as needed. When all of the metric values are within accepted limits, none of the set thresholds have been violated, you have finished testing that particular scenario on that particular configuration.

Questions performance test answers

- Is system capacity meeting business volume under both normal and peak load conditions?
- Is a component meeting expectations?
- Is the observed performance issue caused by a particular component?
- Will performance be consistent over time?
- Is there external interference that was not accounted for?

- How many users can the application handle before undesirable behavior occurs when the application is subjected to a particular workload?
- How much data can my database/file server handle?
- Are the network components adequate?
- What happens if the production load exceeds the anticipated load?
- What kinds of failures should we plan for?
- What indicators should we look for in order to intervene prior to failure?

- Is the application fast enough to satisfy end users?
- Is the business able to process and utilize data collected by the application before that data becomes outdated? (For example, end-of-month reports are due within 24 hours of the close of business on the last day of the month, but it takes the application 48 hours to process the data.)
- Is the application capable of presenting the most current information to its users?
- Is a Web Service responding within the maximum expected response time before an error is thrown?

Software Testing Metrics

- A successful software testing is evaluated by software testing metrics that give information about the effectiveness, efficiency, and progress of the testing.
- In general, with the help of testing metrics, in addition to determining the efficiency and effectiveness of testing process, it is possible to make better decisions for further testing process.
- Moreover, test metrics give an insight about what type of improvement is required to achieve quality software.

- A metric defines an extent to which a given component, process or system satisfies certain characteristics quantitatively.
- Based on the measurements they focus on, there are three types of test metrics.

Process metrics: Are related with the SDLC.

Product metrics: Are connected with the quality of the end product.

Project metrics: Refers metrics related with project team and tools involved in the testing.

- Base metrics: refer raw information collected about requirements, test cases and defects. This include total test cases prepared, total requirements, total test cases executed, passed, failed, total number of critical, high, medium and low severity defects.
- Calculated (derivative) metrics: These are metrics that give information about the effectiveness of a given testing. It is possible to imply how far a given test succeed with the help of such metrics.

1. Defect Density: It gives information about whether a software is ready for release or not. It is the ratio of the total number of defects found in a software during a specific period of time divided by the size of that particular module.

Defect Density = Defect Count/Size of the Release/Module

2. Defect Leakage: If any defects are left undetected by the team and are found by the user, it is known as defect leakage or bug leakage.

Defect Leakage = (Total Number of Defects Found in UAT/ Total Number of Defects Found Before UAT) x 100

3. Defect Removal Efficiency: Measures the ability of the development team to remove various defects from the software, prior to its release or implementation. It is measured per test type and is used as an indirect measure for software quality and performance.

DRE = Number of defects resolved by the development team/ (Total number of defects at the moment of measurement)

4. Defect Category: measured during the process of the software development life cycle (SDLC).

Insight about software's usability, performance, functionality, stability and reliability can be found.

Defect Category = Defects belonging to a particular category/ Total number of defects.

5. Defect Severity Index: It is the degree of impact a defect has on the development of an operation or a component of a software application being tested. Defect severity index (DSI) offers an insight into the quality of the product under test and helps gauge the quality of the test team's efforts. Additionally, with the assistance of this metric, the team can evaluate the degree of negative impact on the quality as well as the performance of the software.

Defect Severity Index (DSI) = Sum of (Defect * Severity Level) / Total number of defects

6. Test Case Effectiveness: The objective of this metric is to measure the efficiency of test cases that are executed by the team of testers during every testing phase. It helps in determining the quality of the test cases.

Test Case Effectiveness = (Number of defects detected / Number of test cases executed) x 100

7. **Test Coverage:** Test coverage is another important metric that defines the extent to which the software product's complete functionality is covered. It indicates the completion of testing activities and can be used as criteria for concluding testing. It can be measured by implementing the following formula:

Test Coverage = Number of detected faults/number of predicted defects.

8. Test Design Coverage: Similar to test coverage, test design coverage measures the percentage of test cases coverage against the number of requirements. This metric helps evaluate the functional coverage of test case designed and improves the test coverage. This is mainly calculated by the team during the stage of test design and is measured in percentage.

Test Design Coverage = (Total number of requirements mapped to test cases / Total number of requirements) x 100

9. **Test Execution Coverage:** measured during test execution, it determines the coverage of testing with the help of total number of test cases executed as well as the number of test cases left pending.

Test Execution Coverage = (Total number of executed test cases or scripts / Total number of test cases or scripts planned to be executed) x 100

10. Test Tracking & Efficiency: Test efficiency is an important component that needs to be evaluated thoroughly. It is a quality attribute of the testing team that is measured to ensure all testing activities are carried out in an efficient manner.

- **Passed Test Cases Coverage:** It measures the percentage of passed test cases.

(Number of passed tests / Total number of tests executed) x 100

- **Failed Test Case Coverage:** It measures the percentage of all the failed test cases.

(Number of failed tests / Total number of test cases failed) x 100

- **Test Cases Blocked:** Determines the percentage of test cases blocked, during the software testing process.

(Number of blocked tests / Total number of tests executed) x 100

- **Fixed Defects Percentage:** With the assistance of this metric, the team is able to identify the percentage of defects fixed.

(Defects fixed / Total number of defects reported) x 100

- **Accepted Defects Percentage:** The focus here is to define the total number of defects accepted by the development team. These are also measured in percentage.

(Defects accepted as valid / Total defect reported) x 100

- **Defects Rejected Percentage:** Another important metric considered under test track and efficiency is the percentage of defects rejected by the development team.

(Number of defects rejected by the development team / total defects reported) x 100

- **Defects Deferred Percentage:** It determines the percentage of defects deferred by the team for future releases.

(Defects deferred for future releases / Total defects reported) x 100

- **Critical Defects Percentage:** Measures the percentage of critical defects in the software.

(Critical defects / Total defects reported) x 100

- **Average Time Taken to Rectify Defects:** With the assistance of this formula, the team members are able to determine the average time taken by the development and testing team to rectify the defects.

(Total time taken for bug fixes / Number of bugs)

Test Effectiveness: the percentage of the difference between the total number of defects found by the software testing and the number of defects found in the software.

Test Effectiveness (TEF) = (Total number of defects injected + Total number of defects found / Total number of defects escaped) x 100

Metrics Lifecycle

1. **Analysis**: at this stage the metrics are identified and defined.

2.Communicate: the identified metrics are required to be communicated and with all test team members and also the team members should be educated about data points to be collected required for the test metric.
3.**Evaluation:** Data should be collected, verified and validated and lastly values for a metric is calculated.
4.**Reports :** a report with clear conclusion is prepared, shared to developers, stakeholders and testing team. Also feedback is received to improve testing in the future.