Performance testing for a REST API based software

An **API** or Application Programming Interface is a set of programming instructions for accessing a web-based software application.In other words, a set of commands used by an individual program to communicate with one another directly and use each other's functions to get information.

Rest stands for Representational State Transfer. It is an architectural style and an approach for communication used in the development of Web Services. REST has become a logical choice for building APIs. It enables users to connect and interact with cloud services efficiently.

API testing (Application Programming Interface Testing) is a software testing type which focuses on the determination if the developed APIs meet expectations regarding the functionality, reliability, performance, and security of the application. When we test an API, we deal with the stuff under the covers -- JSON, XML, REST, and web services.

There are mainly 4 methods involve in API Testing like GET, POST, Delete, and PUT.

1. **GET**- The GET method is used to extract information from the given server using a given URI. While using GET request, it should only extract data and should have no other effect on the data.
2. **POST**- A POST request is used to create a new entity. It can also be used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
3. **PUT**- Create a new entity or update an existing one.
4. **DELETE**- Removes all current representations of the target resource given by a URI.

Performance Testing is crucial to determine that the web application under test will satisfy **high load** requirements. It can be used to analyze overall server performance under heavy load.

Certain tools for such performance testing are:

- **Apigee**: Apigee is a cross-cloud API testing tool, allowing users to measure and test API performance, supports and build API using other editors like Swagger. Identify performance issues by tracking API traffic, error rates, and response time
- **JMeter**: JMeter (open source) is widely used for functional API testing although it is actually created for load testing. It can be used for both static as well as dynamic resources performance testing
- **Swagger**: Swagger is an API testing tool that allows users to start their functional, security, and performance testing right from the Open API Specifications.

And many more…

JMeter

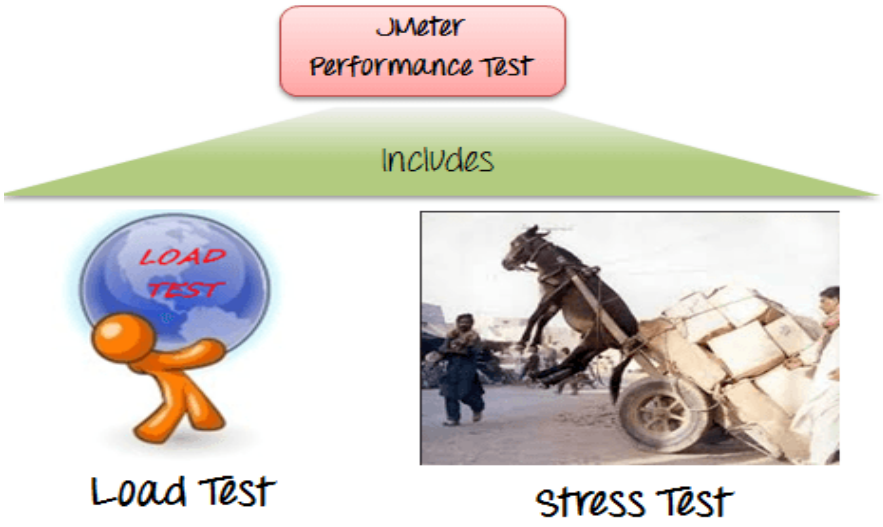
Apache JMeter is open-source software that is popular for performance testing. This tool is designed to load test functional behavior and measure performance. JMeter (open source) is widely used for functional API testing although it is actually created for load testing.

JMeter is a 100% Java application and should run correctly on any system that has Java installed correctly. It does require Java version 8.

Apache JMeter testing tool offers following benefit in Performance Testing:

- JMeter can be used to test the performance of both static resources such as JavaScript and HTML, as well as dynamic resources, such as JSP, Servlets, and AJAX.
- JMeter can discover maximum number of concurrent users that your website can handle
- JMeter provides a variety of graphical analyses of performance reports.

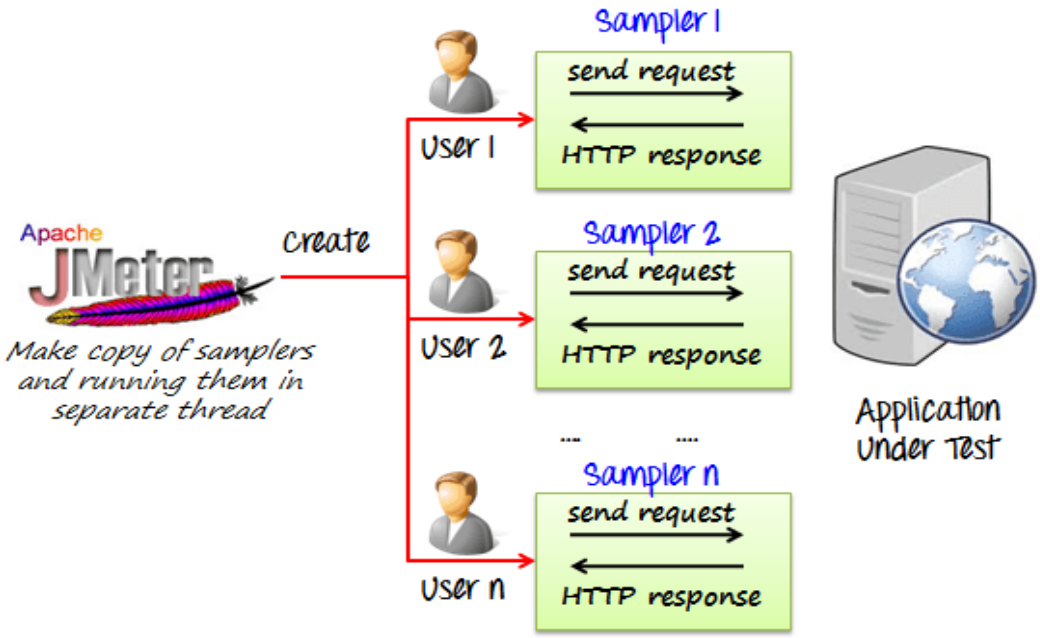
JMeter Performance Testing includes:



Load Testing: Modeling the expected usage by simulating multiple user access the Web services concurrently.

Stress Testing: Every web server has a maximum load capacity. When the load goes beyond the limit, the web server starts responding slowly and produce errors. The purpose of the Stress Testing is to find the maximum load the web server can handle.

The figure below shows how JMeter load Testing simulates the heavy load:

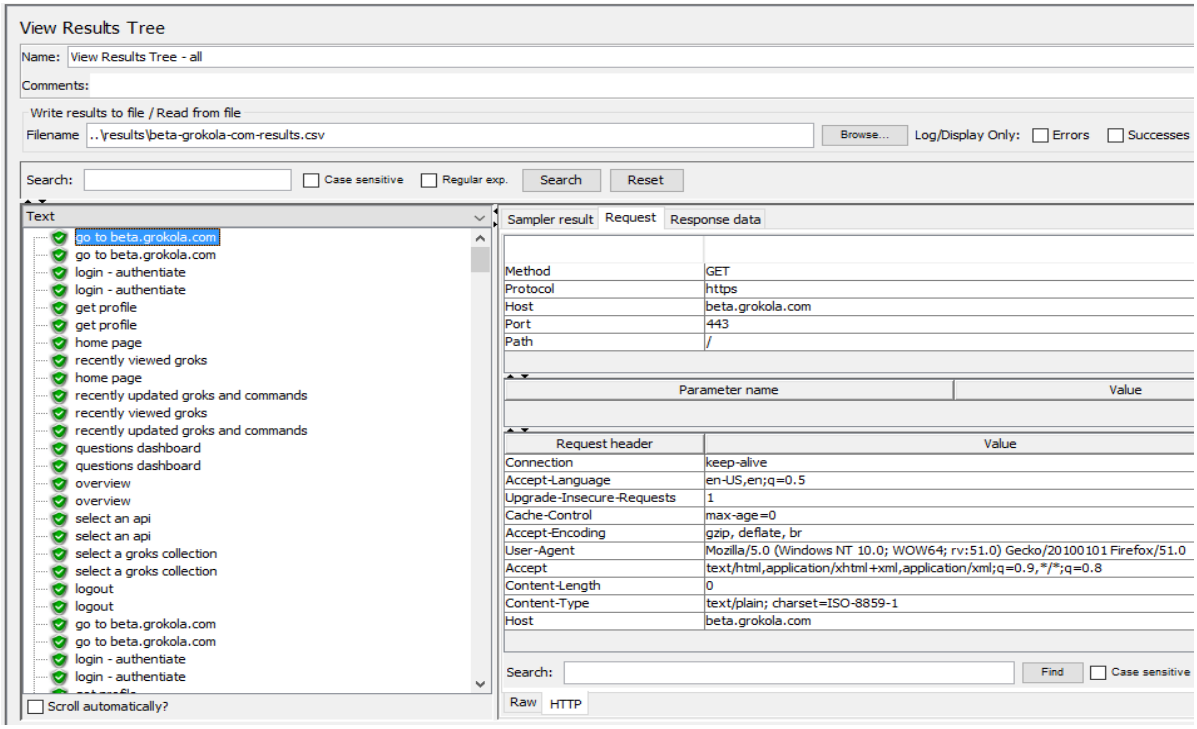


Report generation format

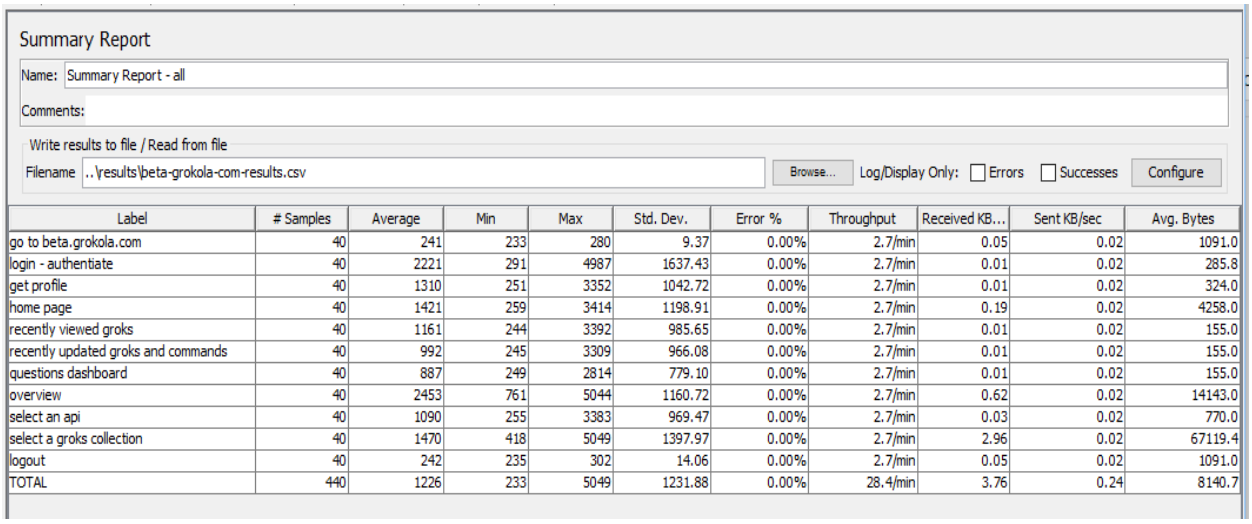
Listeners provide access to the information JMeter gathers about the test cases while JMeter runs.

- The **Graph Results listener** plots the response times on a graph.

- The **View Results Tree Listener** shows details of sampler requests and responses, and can display basic HTML and XML representations of them.



- The **Summary Report** creates a table row for each differently named request in your test.



Summary Report

Name: Summary Report - all

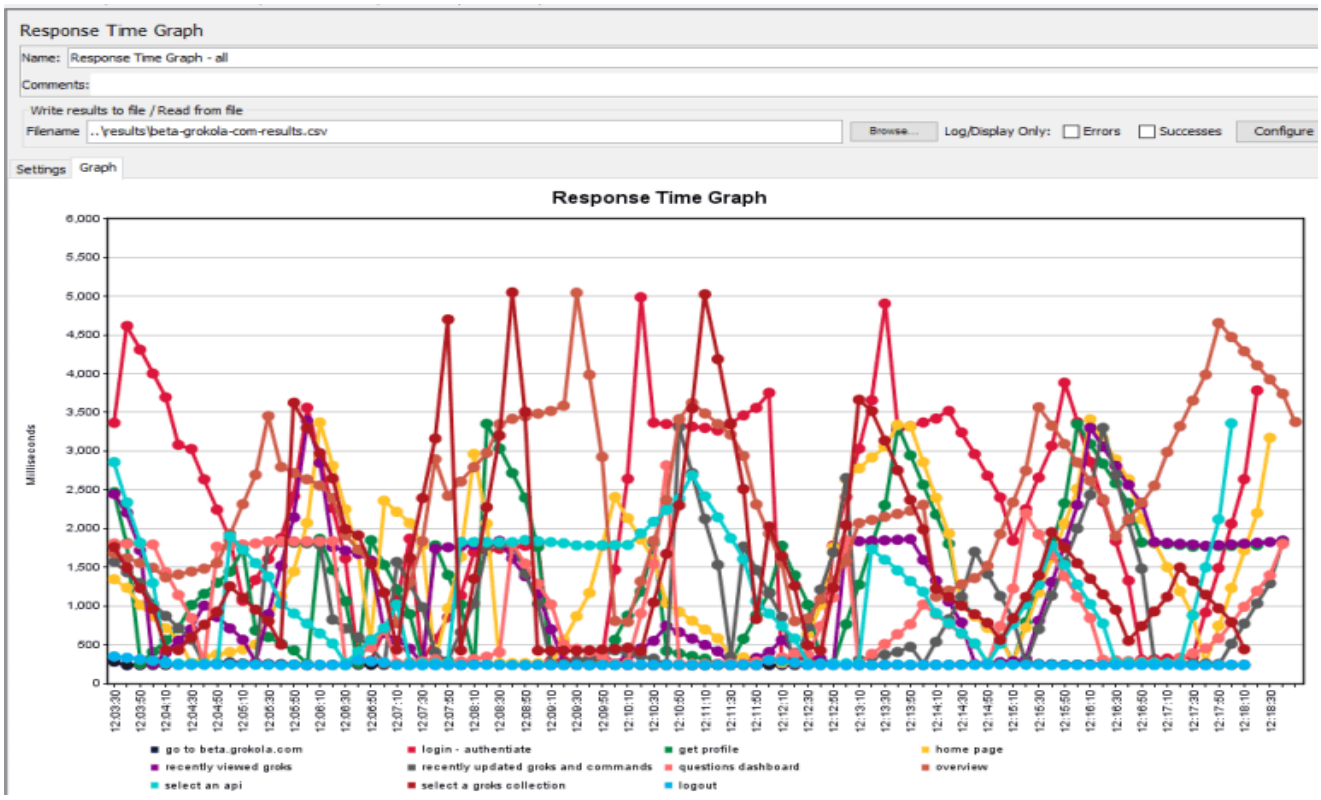
Comments:

Write results to file / Read from file

Filename ..\results\beta-grokola-com-results.csv Browse... Log/Display Only: Errors Successes Configure

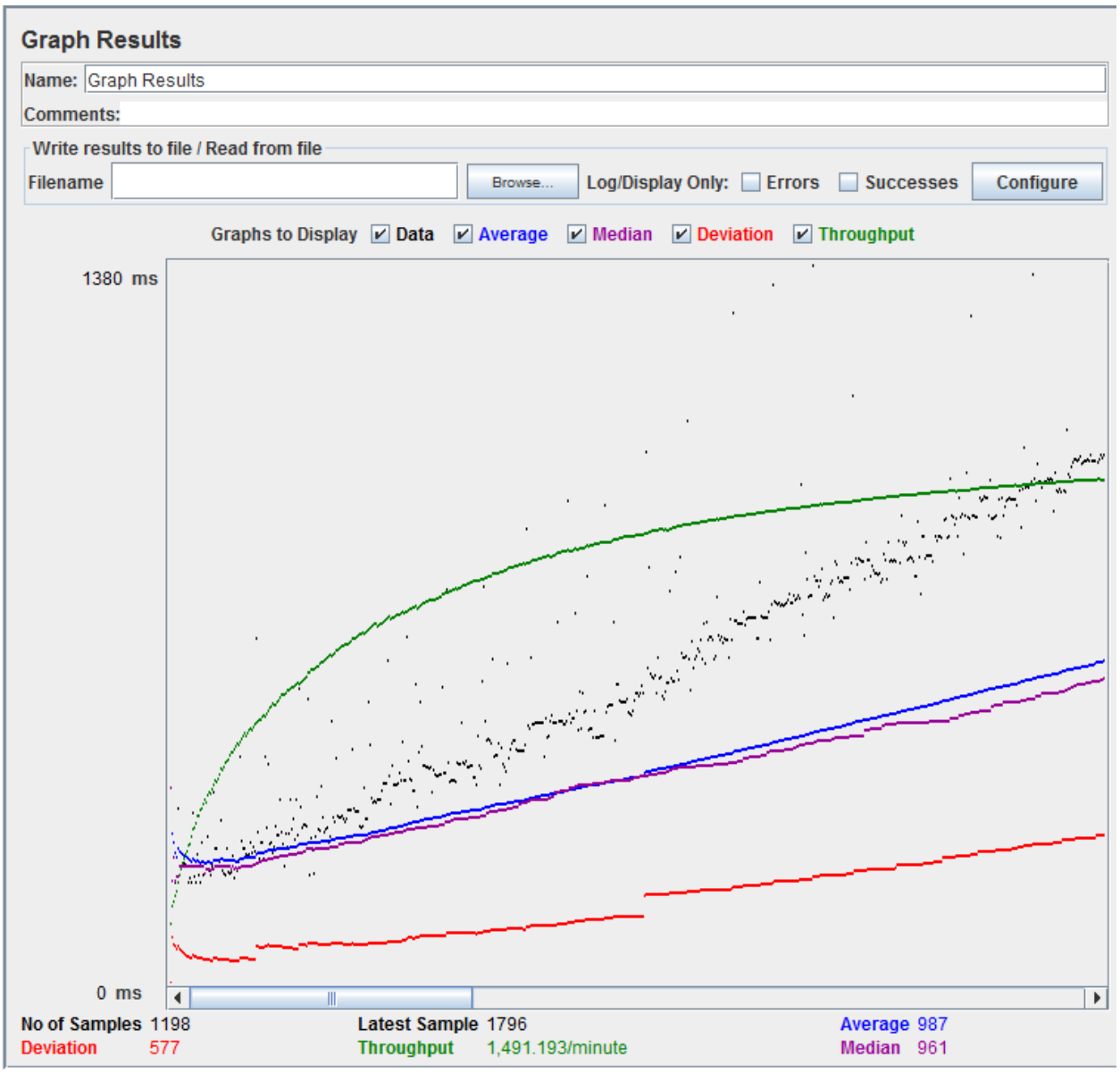
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
go to beta.grokola.com	40	241	233	280	9.37	0.00%	2.7/min	0.05	0.02	1091.0
login - authentiate	40	2221	291	4987	1637.43	0.00%	2.7/min	0.01	0.02	285.8
get profile	40	1310	251	3352	1042.72	0.00%	2.7/min	0.01	0.02	324.0
home page	40	1421	259	3414	1198.91	0.00%	2.7/min	0.19	0.02	4258.0
recently viewed groks	40	1161	244	3392	985.65	0.00%	2.7/min	0.01	0.02	155.0
recently updated groks and commands	40	992	245	3309	966.08	0.00%	2.7/min	0.01	0.02	155.0
questions dashboard	40	887	249	2814	779.10	0.00%	2.7/min	0.01	0.02	155.0
overview	40	2453	761	5044	1160.72	0.00%	2.7/min	0.62	0.02	14143.0
select an api	40	1090	255	3383	969.47	0.00%	2.7/min	0.03	0.02	770.0
select a groks collection	40	1470	418	5049	1397.97	0.00%	2.7/min	2.96	0.02	67119.4
logout	40	242	235	302	14.06	0.00%	2.7/min	0.05	0.02	1091.0
TOTAL	440	1226	233	5049	1231.88	0.00%	28.4/min	3.76	0.24	8140.7

- The **Response Time Graph** draws a line chart showing the evolution of response time during the test, for each labeled request.



Additionally, listeners can direct the data to a file for later use. Every listener in JMeter provides a field to indicate the file to store data to. There is also a Configuration button which can be used to choose which fields to save, and whether to use CSV or XML format.

For Example: The picture below presents a graph of a test plan, where we simulated 100 users who accessed on website www.google.com.



At the bottom of the picture, there are the following statistics, represented in colors:

- Black: The total number of current samples sent.
- Blue: The current average of all samples sent.
- Red: The current standard deviation.
- Green: Throughput rate that represents the number of requests per minute the server handled

To analyze the performance of the web server under test, you should focus on 2 parameters

1. **Throughput**
2. **Deviation**

The Throughput is the most important parameter. It represents the ability of the server to handle a heavy load. The higher the Throughput is, the better is the server performance.

In this test, the throughput of Google server is 1,491.193/minute. It means Google server can handle 1,491.193 requests per minute. This value is quite high so we can conclude that Google server has good performance

The **deviation** is shown in red - it indicates the deviation from the average. The **smaller** the **better**.

Evaluation of 3 critical parameters

[1]. Speed

The rise of the smartphone and easy connectivity to the internet has made access to information easier with most of them accessing websites and web applications from their smartphones.

As a performance engineer, I would like to run my load test on different network speeds like 1 mbps, 4 mpbs, 16 mbps and so on to ensure the app performance.

Check the testing environment network bandwidth (where the test to be conducted) before simulating the speed. Because JMeter can only simulate the bandwidth equal to or less than the testing environment network bandwidth

Why it is needed?

Day by day, mobile traffic over desktop traffic is growing, there is a need to focus on mobile users for performance testing. hence, it is very important to watch how good we are at mobile else it could be lose to business. Mobile web traffic comprises users with different connection speeds. So, it is very important to perform load testing on mobile apps with different connection speeds.

[2]. Scalability

Scalability testing is a type of testing done for the websites which are subject to heavy loads. Here the term 'load' refers to the number of users using the website. Scalability testing is performed to check whether the website which is developed is capable enough to handle the large loads. It is a type of performance testing, where the performance of the website is measured under the various load conditions.

Scalability testing helps in analyzing the speed, effectiveness, reliability, scalability, and interoperability of the system, computer, network, program or application. The other most important use of scalability testing is to check whether the server is crashing if the load increases. Hence scalability test is performed to determine the various load conditions (low, medium and heavy loads). Also, it analyzes the time taken by the website to load under the various load conditions.

Which Tasks Performed In The Scalability Testing?

- The response time of the website.
- Hits, request and transactions per second.
- Various loads (number of users)- low medium or heavy conditions.
- Web server (Request or Response per second).
- Data sent and received.
- Impact of data storage in the database due to increase in load.
- Memory (CPU) usage.
- Time is taken to load the webpage or website

Challenges During The Scalability Testing:

- Sometimes, the functional errors are not identified using the scalability testing.
- Ideally, scalability testing should be performed on the realistic scenarios or the production environment. But due to budget constraints, the client may not be able to provide the environment.
- A good knowledge of the system is required to analyze the result reports.
- Sometimes it is difficult to guess the stress to be applied.

Different Phases/Lifecycle of Scalability Testing:

Mainly there are four phases or stages of scalability testing-

#1) STRUCTURE VALIDATION STAGE– In this phase, mainly the transactions in the application is verified. In other words, the scalability of the application structure is verified.

#2) BENCHMARKING STAGE– At this phase, a set of metrics is developed or the benchmark is set up in order to test the scalability of the application.

#3) REGRESSION TESTING STAGE– This type of testing is performed if there is any change request arrives. The application should be stable enough and should give the same response even if any change is done. The application should be stable and should not degrade in scalability.

#4) FINE CONTROL STAGE-All the units of the application are integrated in order to test the complete scalability and workload capability of the software.

[3]. Stability

STABILITY TESTING measures the ability of a software product to continue to function, over long period of time, exercising its full range of use, without failing or causing failure. The purpose of Stability Testing is to check if the application will crash at any point in time.

Stability Testing is done to check the efficiency of a developed product beyond normal operational capacity, often to a breakpoint. **There is greater significance** is on error handling, software reliability, robustness and scalability of a product under heavy load rather than checking the system behavior under normal circumstances. Stability testing assesses stability problems. This testing is primarily intended to stress the software component to the maximum. It is a Non-Functional Technique.

Stability testing is also referred to as a Load or endurance testing.

Why do Stability Testing

This kind of testing helps users to understand the ways the system will work in real-life situations.

Hence, Stability Testing allows you to check,

- Provide confidence in the stability of your system under test.
- Ensure that your system can handle large programs.
- Monitor the effectiveness of your system.
- Test system stability under stress.

It plays an important role in product development as it is used to determine the limitations of a software product under test before it is released or the areas of more improvement before the product goes live or at Production.

How to do Stability Testing

- To determine the scope and objective of the testing, we must ensure that the Application Server(s) do not crash during the Load Test executions.
- To determine the Business issues, verify the system performance and load as per end user perspective.
- To assign the different Responsibilities and Roles like -Creating Test plan, Test Case design, Test case review, Test execution, etc.
- To ensure the Test deliverables within the specified time
- To ensure proper Load Testing tools and experience team is present for the same.
- To measure the risk and cost involves in the testing. This will determine the cost of each execution in terms of CPU utilization and memory.
- Determine the Defect tracking and reporting and there proper mapping with the requirements.

Test Case for Stability Testing for CPU Performance

- To verify the Upper limit of the system.
- How system crashes or recovers.
- A total number of transactions completed per request.
- Whether or not transaction response stays steady or increases over time.
- How the system behaves under heavy load.
- Its response and behavior under heavy load.

Test Reports for Stability Testing

Several statistics are gathered and measured during test executions; these numbers are analyzed in order to generate a report and to identify possible performance problems.

Examples of statistics collected under test are:

- **Transaction Response Times**: The average time is taken to perform transactions during the test. This statistic will evaluate whether the performance of the server is within the acceptable minimum and maximum transaction performance time periods defined for the system.

  This information will evaluate the time taken in processing the request by the web server and sent to the application server, which in most of the cases will make a request to a database server.
- **Hits Per Second**: The number of hits made on the server by users. This statistics benefits to determine the number of load users generate, with respect to a number of hits.
- **Throughput:** The amount of throughput on the Web server during the test which is measured in bytes. Throughput means the amount of data that the users received from the server at any given time. This statistic helps to evaluate the amount of load that users generate.
- **Transaction per second:** These are the total number of completed transactions (both successful and failed) performed during a test. This statistic helps to check the actual transaction load on the system.
- **CPU:** CPU percentage utilization spent during a test.
- **Memory:** Memory usage during a test.
- **Disk:** utilization of disk spaces spent during a test.