



# INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X

Impact factor: 4.295

(Volume 4, Issue 2)

Available online at: [www.ijariit.com](http://www.ijariit.com)

## Criteria to consider for a successful performance testing tool selection

Rama R Anem

[rama.anem@gmail.com](mailto:rama.anem@gmail.com)

SunPower Corporation, Austin, TX

### ABSTRACT

*To inform our audience about the criteria for a successful performance testing tool selection for testing performance on any software stack.*

*The right test tool enables the productivity, optimization, and scale required. It provides the opportunity to find best test tool and evaluate its services faster, and establish successful implementation using it.*

**Keywords:** Performance testing, Load testing, Performance testing tool.

---

### 1. MAIN BODY

#### What is performance testing

Performance testing is an aspect of non-functional testing, goal of which is to collect information about system stability and responsiveness under various workload. Performance testing provides insight on scalability, reliability and resource usage of the system under test.

#### Why performance testing done

Performance testing consists of multiple subtypes of tests, which help to determine if system is capable to handle desired load and helps to understand what the bottlenecks are. This type of tests helps to understand what improvements should be done to code or configuration before going live.

#### Importance of performance testing

Performance tests are necessary to have properties of the system that cannot be achieved or proven by functional testing, such as:

- better performance
- increased Stability
- proven Scalability

#### What are the parameters to consider while selecting a performance testing tool

Various parameters needed to be considered, here is the list below

- **Basic requirements**

List contains what are the basic parameters to look at first:

1. **Platform Compatibility** -The tool should be able to function well on various platforms

2. **GUI friendly** – Graphical user interface is a type of user interface that allows users to interact with ease. Tool should be capable to interact with GUI of the application.
3. **Recording GUI Scenarios** – tool should provide ability to manually record GUI scenarios and convert them to test scripts
4. **Command Line Support** – Command line interface is means of interacting with a computer program where the user issues commands to the program in the form of successive lines of text (command lines).  
Tool should be able to issue and accept commands through Command line interface which is useful if tool will be used to do automated tests on Jenkins or other build management tool
5. **Ease of Learning** – Tool should be easy to learn
6. **Web Protocol Support** – Web protocols are TCP/IP stands for Transmission Control Protocol/Internet Protocol. The Internet Protocol part of the standard refers to the addressing of data message packets. UDP, HTTP and FTP are the additional protocols which have different functions and purposes that ultimately work together to provide assorted capabilities through World Wide Web. Tool should support all the Web Protocols
7. **Reporting Capability** – Tool should have Reporting Capability

- **Flexibility requirements:**

1. **Functional API testing support** – An application programming interface, or API, works to connect an application to the web and to other APIs. API testing is testing the APIs and the integrations they enable work in the most optimal manner. Functional API testing should be done using the tool
2. **Test Scripting/Assertion Language** - The Script assertion runs a ~~groovy~~ script to perform custom checks on the message. You can verify the message content, headers, properties and other components. Test Scripting can be done through the tool
3. **Mock web service creation** – Service Mocking or simulation is the practice of creating a facsimile environment that works similar to the environment you're facsimileing. Tool should give ability to create mock web services with advanced behavior.

- **Maintainability**

Easy to Maintain test set: adding new scenarios, building test scenarios based on existing ones, bulk edit of test parameters.

- **Community**

Active community generates positive feedback loop and as result lots of manuals, support forums and further development of the tool. How to determine if tool has active and supportive community? You need to looks for:

1. **Stackoverflow most searched** – Tool should be searched mostly in Stack overflow
2. **Stackoverflow most votes** – Most votes for the tool in the Stack overflow.
3. **Market Demand** – The aggregate of the demands of all potential customers for a specific product over a specific period in a specific market.
4. **Support from Forums/Community** – Tool should have Support from Forums/Community where you can share your questions or proudly present your solution to a problem.

- **Requirements related to Load**

Tool should be able to generate required load. What is testers' PC against 48 stack of multiple 48 core servers? Tool should provide ability to generate sufficient load on server. To ensure this, pay attention to following properties of the tool:

1. **Scalability** – Scalability is the capability of a system, network or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth. If tool is scalable, it becomes handy for generating high load that cannot be done by one instance of the tool
2. **Best throughput** – Throughput is the maximum rate of production or the maximum rate at which something can be processed. Tool should give the Best Throughput
3. **KB/sec Processed** – Processing rate is very important for scalability testing of the application, especially emulating UI interaction which would require download of UI code and API call execution.
4. **Best response time** – Response time is the total amount of time it takes to respond to a request for service. Tool should give the Best Response time under maximal load to generate higher possible load

Below is the example where the parameters considered

	Artillery	Comments	SOAP UI FREE	Comments	SOAP UI PAID	Comments	jmeter	Comments	Validated(Y/N)
<b>Platform Compatibility</b>	Windows, Linux, (May be Mac)		Windows, mac, unix	Start with free version and anytime move to a paid version seamlessly.	Windows, mac, unix		Windows, mac, unix		N
<b>GUI friendly</b>	No	Planning to develop UI in react.	Yes		Yes		Yes		Y
<b>Command Line Support</b>	YES		YES				YES		Y
<b>Ease of Learning</b>	Easy to moderate	Light weight test package. Test case flow written in JSON or YAML. Object/Data captured as JSONPath and XPath, REGEX and HEADERS	Moderate to steep.	Heavy Java/groovy package s. Test Case flow needs to be written in groovy. No Javascript support in free version.	Moderate to steep.		Moderate to steep.	Heavy Java/groovy package s. Test Case flow needs to be written in groovy, bsh and javascript.	Y
					Lot of out of box features like JSON assertions,html reports, full fledge support for api using groovy and javascript		Easy for basic scenarios. Knowledge on bash shell , groovy and javascript scripting required to create custom assertions and function		

							al test flow.		
<b>Web Protocol Support</b>	HTTP, HTTPS, WebSockets		HTTP, HTTPS	Not supporting all but few with the help of 3rd party plugins	HTTP, HTTPS,		http, https,	support for ◦Web ◦SOAP ◦FTP ◦Database via JDBC ◦LDAP ◦Message-oriented middleware (MOM) via JMS ◦Mail - SMTP(S), POP3(S) and IMAP(S) ◦Native commands or shell scripts ◦TCP	N
<b>Support from Forums /Community</b>	New Open Source Community		SOAPUI OPENSOURCE COMMUNITY		Commercial support		Twitter-jmeter Mailing List		Y
<b>Reporting Capability</b>	JSON, HTML		NO	No out of box reporting available for free version. Though one may hack using groovy scripting and using available testrunn	YES		Yes		Y

				er and context object.					
<b>Market Demand</b>	Infancy stage with couple companies using it. Community is of around 120 people. Just around a year old.		Established in 2005 with a well support community of 70000 users				Established well and in marker since 1998		N
<b>Functional API testing support</b>	NO		YES		YES		YES		Y
							Reference		
<b>Test Scripting/Assertion Language</b>	JSON, XML		XML Assertion types only (caveat - JSON available if content response type is JSON) - Groovy scripting works for basic SOAP UI objects testrunner, context, log etc. (Need to use JSONSupporter for JSON		- Javascript and groovy, - JSON + XML Assertion types		java and groovy, junit functional tests can be integrated in jmeter + javascript		Y

			<i>custom assertio n syntax)</i>						
<b>Mock web service creation</b>	<i>No</i>		<i>YES</i>		<i>YES</i>		<i>No</i>		<i>N</i>
<b>Maintainability</b>	<i>Can be maintained as a suite of JSON and YML files can be build and run using batch or shell script.U ser defined framewo rk.</i>		<i>maintain s as a suite of test cases.</i>	<i>Need to investig ate the design patterns used for scaling the api testing</i>			<i>maintain s as a suite of test cases.</i>	<i>Need to investig ate the design patterns used for scaling the api testing</i>	<i>y</i>
<b>Scalability</b>	<i>Promisi ng but not tried and tested extensiv ely</i>		<i>YES</i>	<i>Need to investig ate the design patterns used for scaling the api testing</i>	<i>Yes</i>		<i>Yes</i>	<i>Need to investig ate the design patterns used for scaling the api testing</i>	<i>N</i>
<b>Stackoverflow most searched</b>	<i>2 results</i>		<i>9193 results( includes function al soapui testing)</i>				<i>15,853 results</i>		<i>y</i>
<b>Stackoverflow most votes</b>	<i>1</i>		<i>74</i>				<i>534</i>		<i>y</i>

<b>Best response time</b>	N/A		WINS				AVERAGE		y
<b>Best throughput</b>	N/A		AVERAGE				WINS	Reference	y
<b>KB/sec Processed</b>	N/A		AVERAGE				WINS		y
<b>Recording GUI Scenarios</b>	N/A		check on this Not out of the box	NOT Scenario's though the free version records traffic			YES	We can record functional scenarios using http proxy in workbench and save it under recording controller. Later, we can perform load on this Record sampler to get the performance for each page.	N

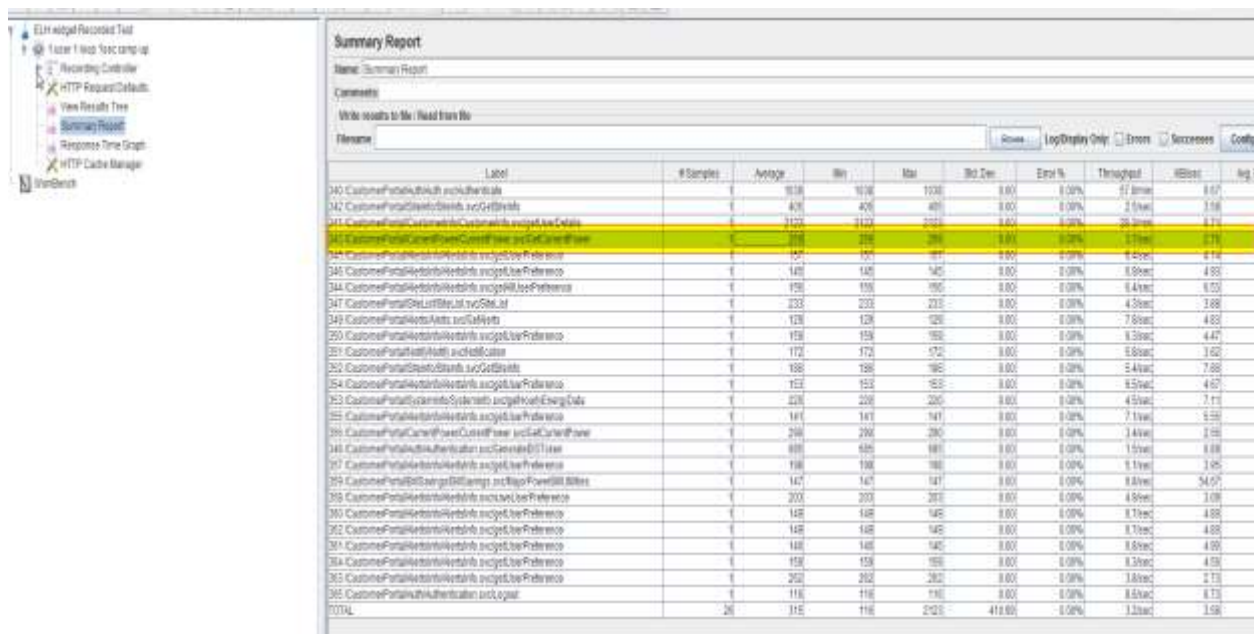
Use Case 1 – Measure the performance of a specific Widget/API response time and throughput when the application user logs in with username/password through GUI.

Real Chrome browser response time for single user - **859ms**

Name	Status	Type	Initiator	Size	Time	Time
Authenticate	200	xhr	affine.mcu.js	485 B	27.33 s	
getUserDetails?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	1.3 KB	4.04 s	
GetDeviceInfo?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	1.2 KB	1.05 s	
getDeviceInfo?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	823 B	431 ms	
getUserPreference?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	445 B	472 ms	
SiteInfo?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	708 B	1.43 s	
GetDeviceInfo?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	418 B	1.01 s	
getUserPreference?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	527 B	1.15 s	
Notification?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	418 B	1.17 s	
en-us.json?ts=1471976118655	200	xhr	affine.mcu.js	4.5 KB	285 ms	
getUserPreference?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	562 B	481 ms	
GetDeviceInfo?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	1.2 KB	705 ms	
GetCurrentPowerInfo=6134a6c8-a62b...	200	xhr	affine.mcu.js	542 B	436 ms	
getHourlyEnergyData?token=...	200	xhr	affine.mcu.js	1.7 KB	3.42 s	
getUserPreference?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	562 B	614 ms	
getUserPreference?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	562 B	763 ms	
MajorPowerSwitchDelay?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	7.8 KB	1.88 s	
GetDeviceInfo?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	1.2 KB	280 ms	
GetCurrentPowerInfo=6134a6c8-a62b...	200	xhr	affine.mcu.js	542 B	229 ms	
GetDeviceInfo?id=6134a6c8-a62b...	200	xhr	affine.mcu.js	1.2 KB	528 ms	
GetCurrentPowerInfo=6134a6c8-a62b...	200	xhr	affine.mcu.js	542 B	859 ms	

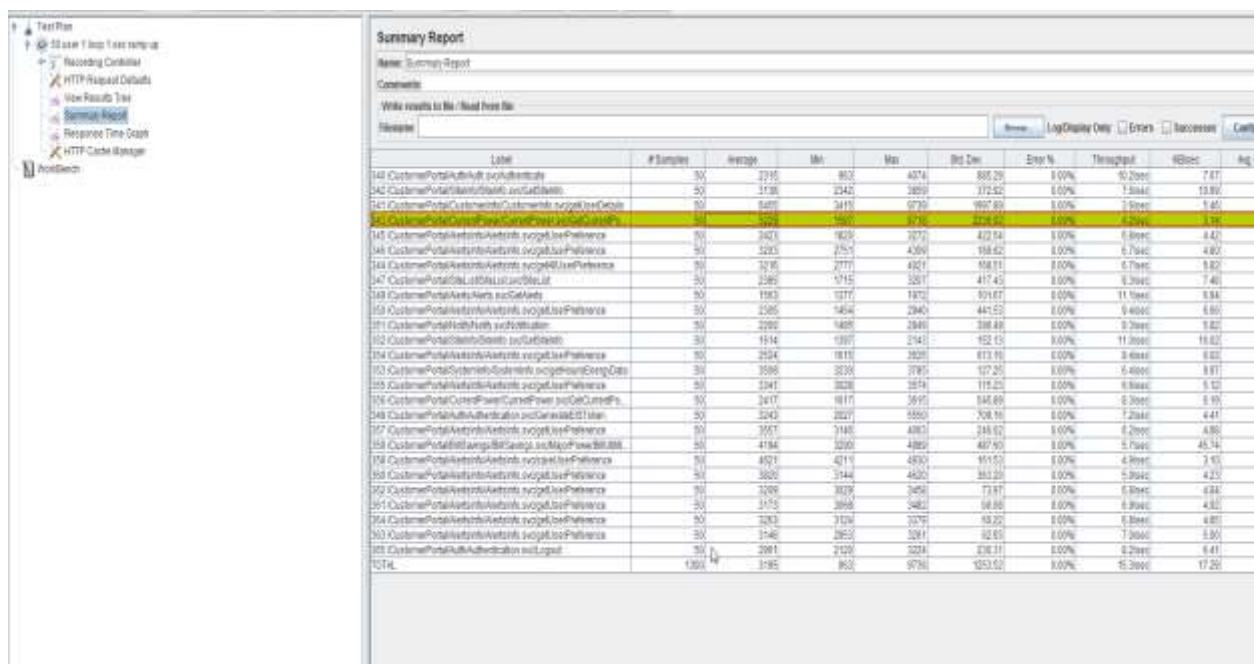
✚ Using Jmeter to record the browser scenario to measure the api performance

- 1) Follow STEPS TO RECORD A BROWSER SCENARIO USING HTTP PROXY to record and create a test for different thread groups.
  - 2) Used below thread group configuration in this use case.
    - 1 user 1 loop 1 sec ramp up
    - 50 user 1 loop 1 sec ramp up
    - 50 user 10 loop 1 sec ramp up
  - 3) Performance results for each of this configuration is as below
- 1 user 1 loop 1 sec ramp up
- response time - **269ms**
  - throughput – **3.7/sec**



Label	# Samples	Average	Min	Max	Std Dev	Error %	Throughput	KB/sec	Avg B
340 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
342 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
344 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
346 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
348 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
350 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
352 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
354 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
356 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
358 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
360 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
362 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
364 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
366 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
368 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
370 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
372 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
374 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
376 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
378 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
380 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
382 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
384 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
386 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
388 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
390 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
392 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
394 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
396 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
398 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
400 CustomerPortalAuthentication	1	838	838	1030	0.00	0.00%	3.7/sec	8.67	1.67
TOTAL	20	315	116	210	418.00	0.00%	3.7/sec	3.58	

- 50 user 1 loop 1 sec ramp up
- response time - **5225ms**
  - throughput – **4.2/sec**



Label	# Samples	Average	Min	Max	Std Dev	Error %	Throughput	KB/sec	Avg B
340 CustomerPortalAuthentication	50	3395	863	4074	885.26	0.00%	10.29/sec	7.87	
342 CustomerPortalAuthentication	50	3138	2342	3899	172.52	0.00%	1.80/sec	13.89	
344 CustomerPortalAuthentication	50	6409	3419	9799	9907.89	0.00%	3.09/sec	1.45	
346 CustomerPortalAuthentication	50	3998	1960	8736	2226.50	0.00%	4.20/sec	3.98	
348 CustomerPortalAuthentication	50	2403	1629	3273	472.54	0.00%	6.89/sec	4.47	
350 CustomerPortalAuthentication	50	2203	2757	4269	198.62	0.00%	6.79/sec	4.80	
352 CustomerPortalAuthentication	50	3236	2777	4821	558.11	0.00%	6.79/sec	5.82	
354 CustomerPortalAuthentication	50	3285	1715	5257	417.45	0.00%	6.29/sec	7.40	
356 CustomerPortalAuthentication	50	1983	1377	3472	931.67	0.00%	11.70/sec	6.84	
358 CustomerPortalAuthentication	50	2585	1454	3945	441.53	0.00%	9.40/sec	6.86	
360 CustomerPortalAuthentication	50	2200	1489	3349	398.48	0.00%	9.79/sec	8.82	
362 CustomerPortalAuthentication	50	1914	1397	3243	162.15	0.00%	11.39/sec	16.82	
364 CustomerPortalAuthentication	50	2934	1811	3929	813.16	0.00%	3.49/sec	9.83	
366 CustomerPortalAuthentication	50	3596	3039	5785	127.25	0.00%	6.40/sec	9.81	
368 CustomerPortalAuthentication	50	3241	3828	5574	119.23	0.00%	6.86/sec	5.12	
370 CustomerPortalAuthentication	50	3477	1817	5545	545.69	0.00%	6.39/sec	9.19	
372 CustomerPortalAuthentication	50	3243	2827	5550	708.16	0.00%	7.20/sec	4.41	
374 CustomerPortalAuthentication	50	3557	3148	4983	348.62	0.00%	6.29/sec	4.86	
376 CustomerPortalAuthentication	50	4184	3200	4989	467.90	0.00%	5.79/sec	46.74	
378 CustomerPortalAuthentication	50	4521	4211	4830	151.53	0.00%	4.89/sec	7.15	
380 CustomerPortalAuthentication	50	3820	3144	4420	383.23	0.00%	5.99/sec	4.23	
382 CustomerPortalAuthentication	50	3206	3679	3456	73.81	0.00%	6.89/sec	4.44	
384 CustomerPortalAuthentication	50	3173	3098	3482	16.46	0.00%	6.89/sec	4.82	
386 CustomerPortalAuthentication	50	3283	3124	3579	159.25	0.00%	6.89/sec	4.80	
388 CustomerPortalAuthentication	50	3140	2853	3381	162.65	0.00%	7.30/sec	5.80	
390 CustomerPortalAuthentication	50	2981	2129	3229	836.11	0.00%	6.29/sec	6.41	
TOTAL	1300	3185	863	9796	1053.62	0.00%	16.39/sec	17.20	

- 50 user 10 loop 1 sec ramp up – This should generate 500 api requests internally but it stops and hangs at 250 api requests with total requests > 3000. May be 3000 requests is the threshold point for 1 machine with 8 GB RAM
- response time - **7261ms**



- throughput – **16.3/sec**

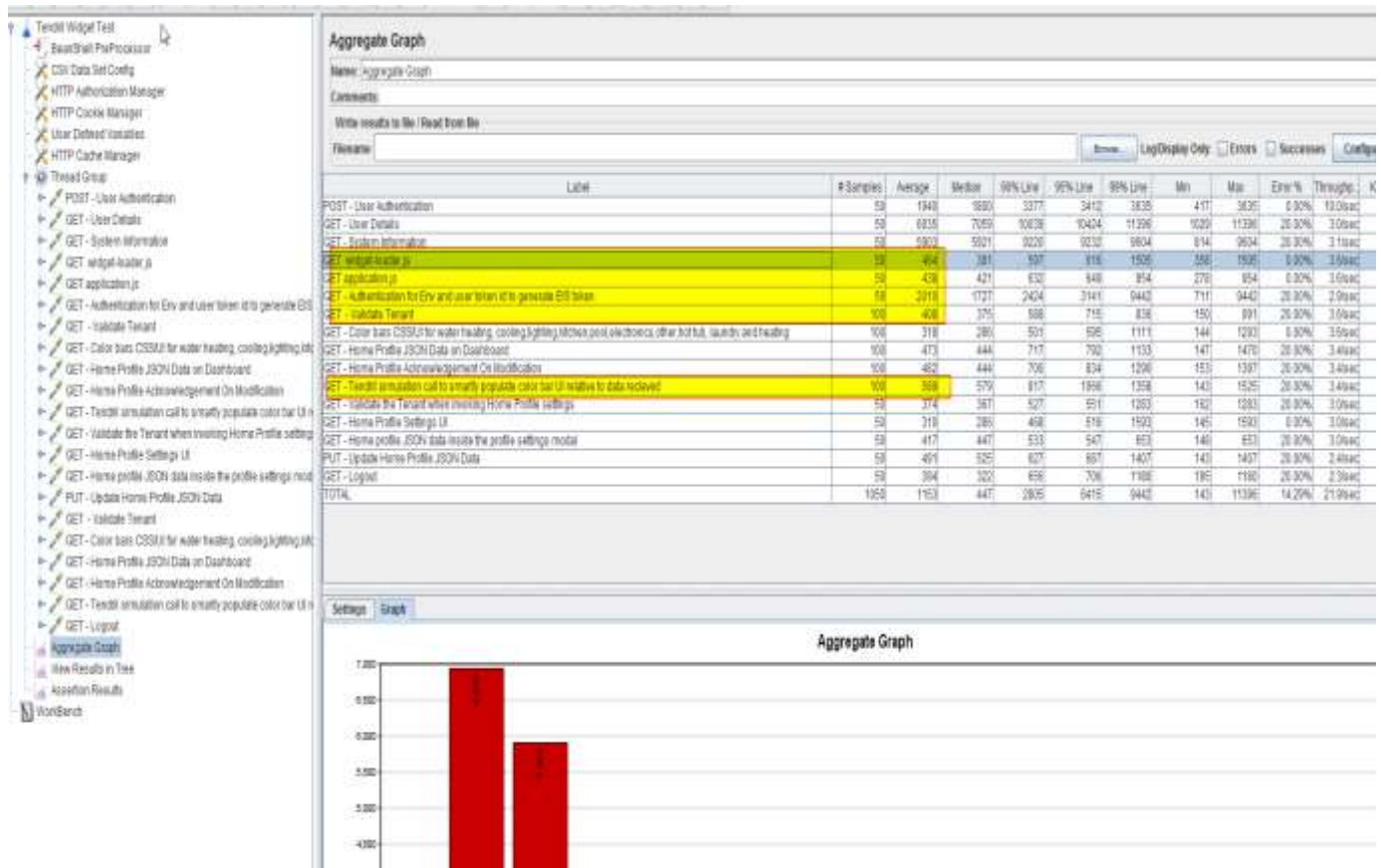
Label	Average	# Samples	Min	Max	Std. Dev.	Err. %	Throughput	Errors
242 CustomerFeedbacksServiceTest	659.1	250	52.76	1508.1	278.14	0.00%	16.3/sec	0.00
243 CustomerFeedbacksServiceTest	381.1	250	16.81	1601.0	278.14	0.00%	16.3/sec	0.00
244 CustomerFeedbacksServiceTest	217.0	250	2.98	1601.0	278.14	0.00%	16.3/sec	0.00
245 CustomerFeedbacksServiceTest	169.1	250	2.98	1601.0	278.14	0.00%	16.3/sec	0.00
246 CustomerFeedbacksServiceTest	138.1	250	2.98	1601.0	278.14	0.00%	16.3/sec	0.00
247 CustomerFeedbacksServiceTest	108.1	250	2.98	1601.0	278.14	0.00%	16.3/sec	0.00
248 CustomerFeedbacksServiceTest	78.1	250	2.98	1601.0	278.14	0.00%	16.3/sec	0.00
249 CustomerFeedbacksServiceTest	48.1	250	2.98	1601.0	278.14	0.00%	16.3/sec	0.00
250 CustomerFeedbacksServiceTest	18.1	250	2.98	1601.0	278.14	0.00%	16.3/sec	0.00
TOTAL	659.1	6250	52.76	1508.1	278.14	0.00%	16.3/sec	0.00

Use Case 2 – Measure the performance of API response time and throughput using direct api call without using GUI.

- Api end point
  - How to get the API with headers and auth tokens
- Go to web application on Chrome
  - Login with user name/ password
  - Chrome settings -> More tools -> Developer tools -> Network
  - User will see all the api service calls as one highlighted in yellow. Right click and copy the curl cmd for the service call. It will look something like below.
- ```
curl "endpoint?id=d80951aa-82af-4b8a-959d-05e2e20944a9" -H "Accept-Encoding: gzip, deflate, sdch, br" -H "Accept-Language: en-US,en;q=0.8" -H "User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36" -H "Accept: application/json, text/plain, */*" -H "Referer: web application" -H "Cookie: AWSELB=57318FD51833457EFBBF47BB858C674A73F7561C16C492D7438459646BBE685EADCAAFEAA228523D5636AFAAC42B6290EDC6337551E5BB09677D4A4EC0E586CD81874076CA; liveagent_oref=; liveagent_sid=4a0caed9-6537-4d24-aaed-d0afb2331304; liveagent_vc=2; liveagent_ptid=4a0caed9-6537-4d24-aaed-d0afb2331304; AMCV_44B127E253DB49AD0A490D4E""40AdobeOrg=793872103""7CMCIDTS""7C17037""7CMCMID""7C80575008065281569984992602016157553249""7CMCAAMLH-1472236220""7C9""7CMCAAMB-1472581066""7CNRX38WO0n5BH8Th-nqAG_A""7CMCAID""7CNONE; _gat=1; _ga=GA1.2.1576135503.1470682384" -H "Connection: keep-alive" --compressed
```

- Postman tool - **Initial call it takes 659ms and subsequent requests is 381ms.**
- Jmeter tool to test individual end point api call to measure performance
- You can also parameterize the username and password in the request parameters.

- 1 user 1 loop 1 sec ramp up
  - response time - **485ms**
  - throughput – **2.1/sec**
- 50 user 1 loop 1 sec ramp up
  - response time - **1018ms**
  - throughput – **22.1/sec**
- 50 user 10 loop 1 sec ramp up
  - response time - **1557ms**
  - throughput – **29.6/sec**



## 2. CONCLUSION

Following above discussed criteria helps select best performance testing software/ tool for your requirement. And by selecting appropriate tool improves the productivity, optimization and scaling.

## 3. REFERENCES

- [1] <https://jmeter.apache.org/>
- [2] <https://www.soapui.org/load-testing/reference/loadtest-window.html>
- [3] <https://artillery.io/docs/getting-started/>