**Jmeter Performance Testing – Hands-on**

**Performance testing –** It is a testing to ensure software applications will perform well under the particular workload. It is necessary when we have a company with own server especially running during deals.

Eg..if we want to do performance testing for a product..

Steps:

1.Write a script to buy a product

2. Do load testing – 1000 users at a time

3.Do stress testing - 10K users for continuous 5 hrs

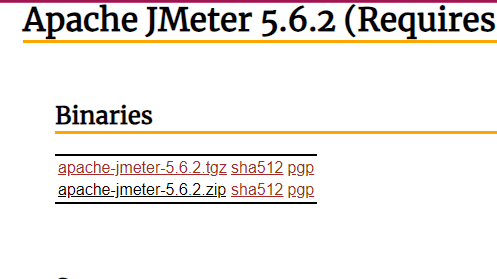
This test can run with headerless.It measures the behaviour of the application through metrics under workload.

**Jmeter:**

It is an open source software. It is built using java, so that it supports all operating systems (cross platform support). Scripting is not essential, but it is done using record and playback.

**Jmeter Installation:**

Download Zip file for windows.



Download java for execution.

Click this jar fileA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Test plan – For planning the test to be executed.

Only one project can be run. Workbench is the practise area for storing components

**Record and playback:**

When the record is clicked, test is created automatically in jmeter, while performing actions in browser.

Right click test plan -> Add - > Non test elements - >Https test script recorder

A screenshot of a computer

Description automatically generated

Http tab is opened.

A screenshot of a computer

Description automatically generated

Port represents that the jmeter will record the actions that are performed only in that number.

Target Controller – it is the location where the recorded tests are stored.

To test , we need request and response. Gif and banners are unwanted actions during running and makes the script tedious. Hence it is removed using URL patterns to exclude field.

A screenshot of a computer

Description automatically generated

Linking Jmeter with firefox browser.

Open firefox -> Click setting - >click network settings - >choose Manual proxy configuration.

Provide the exact port number mentioned in Jmeter.

A screenshot of a computer

Description automatically generated

Click ok.

For HTTPS , we need certificate to trust.

In firefox, go to settings -> click certificate in search box -> click view certificate - > click import

A screenshot of a certificate manager

Description automatically generated

Select the security certificate from the bin folder.

A screenshot of a computer

Description automatically generated

Now the certification is added.

A screenshot of a software security device

Description automatically generated

Click start and perform actions in Firefox and click stop

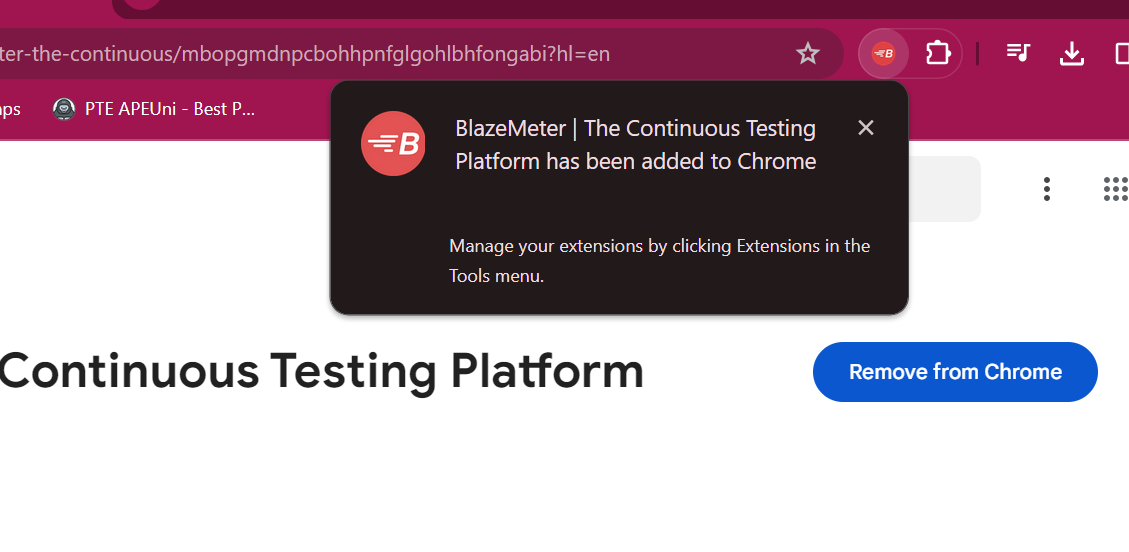
A screenshot of a computer

Description automatically generated

Scripts are automatically created.

**Blazemeter:**

Adding blazemeter recorder as an extension to the chrome for load testing.



Successfully logged in to the blazemeter.

This pluging automatically records the action and transfer the created test (.JMX files) to the jmeter.

**Start recording**

A screenshot of a computer

Description automatically generated

Click save

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

In blazemeter, we won’t get additional unnecessary scripts.

Click file – open = to import the newly created script to jmeter from downloads

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Thread Group:**

It is a set of users executing the same scenario.

Start and stop the recording of booking flight in blazedemo.com

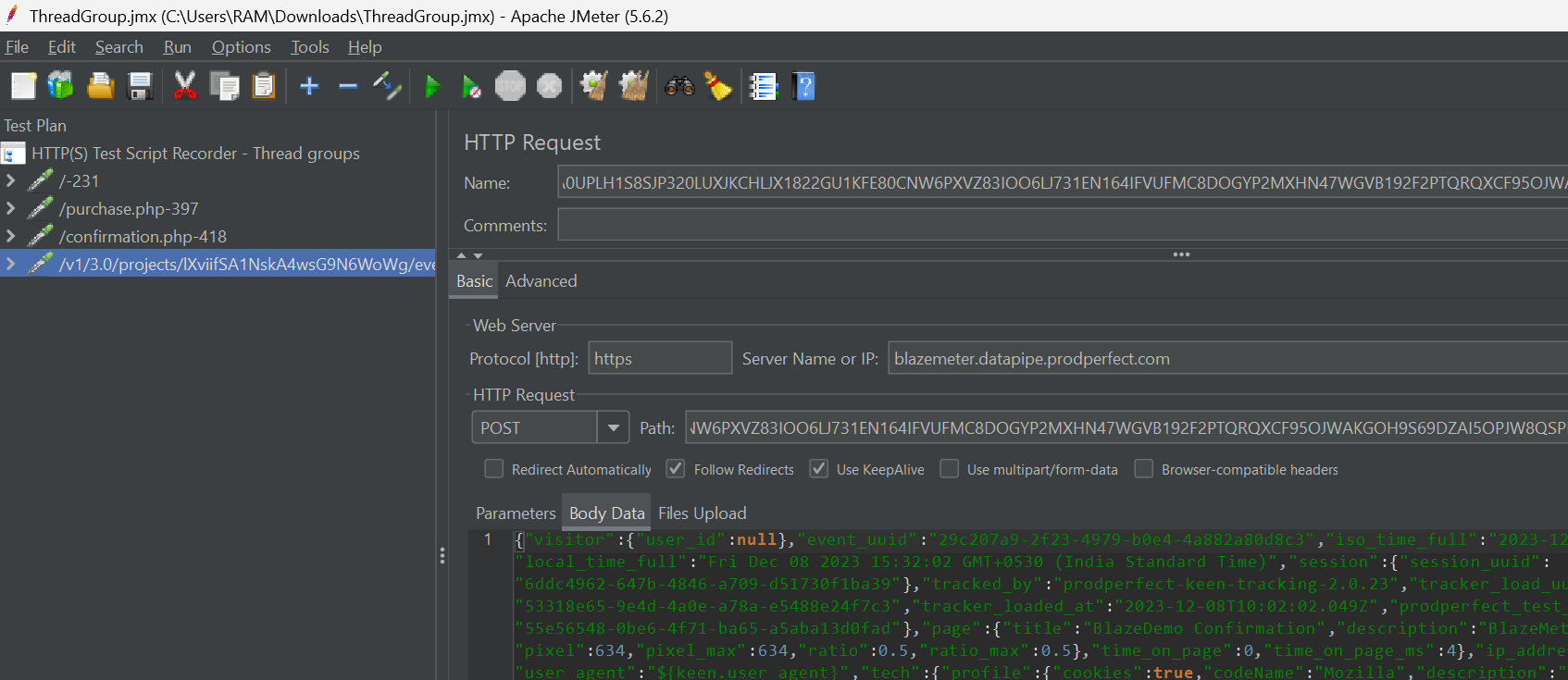
A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

Remove the unwanted files and save the project.



Adding thread group:

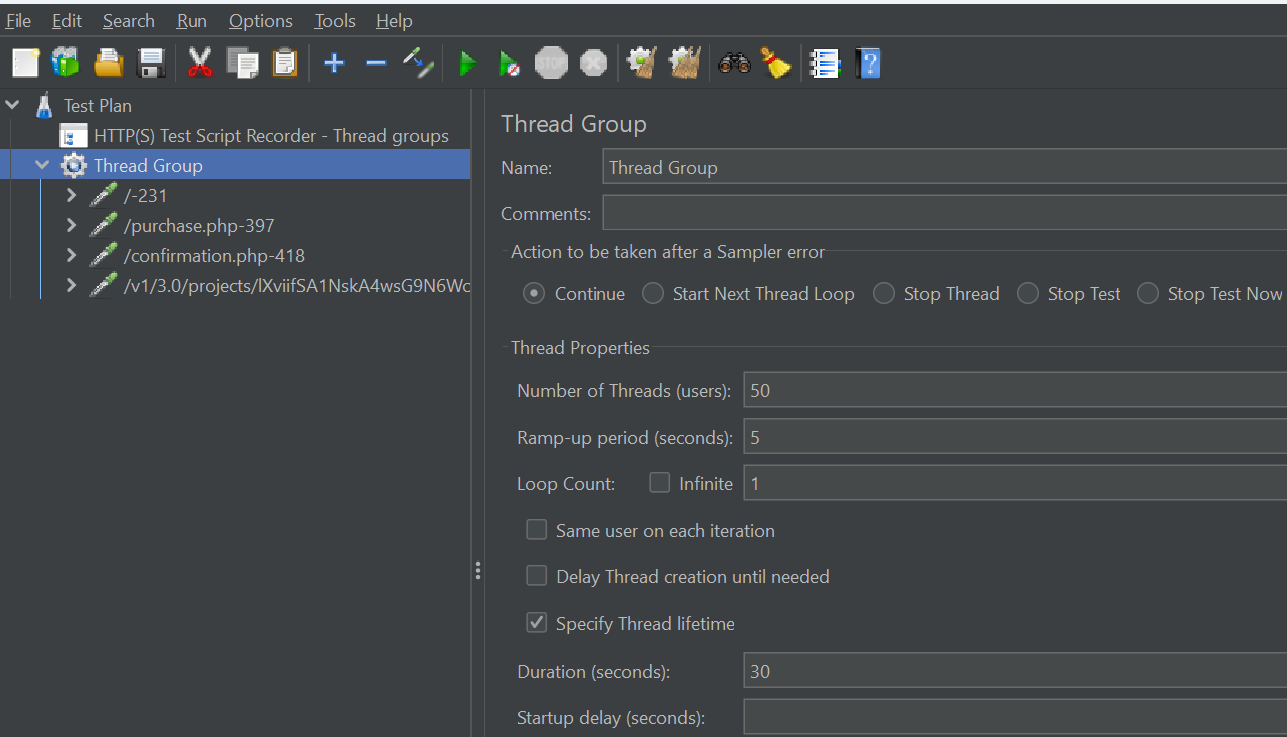
A screenshot of a computer

Description automatically generated

Ramp up period – All the users wont be starting on the exact time.For every second, other set of users will be joing.

Choose Loop forever. And give the time as required.

For creating loop test with more users, we use thread group. So cut the required scripts and add it in thread group.



**Listener:**

It will track and listen “how the jmeter is working”. It shows the results, graphs and load parameters of the project.

Add listener - > view results tree

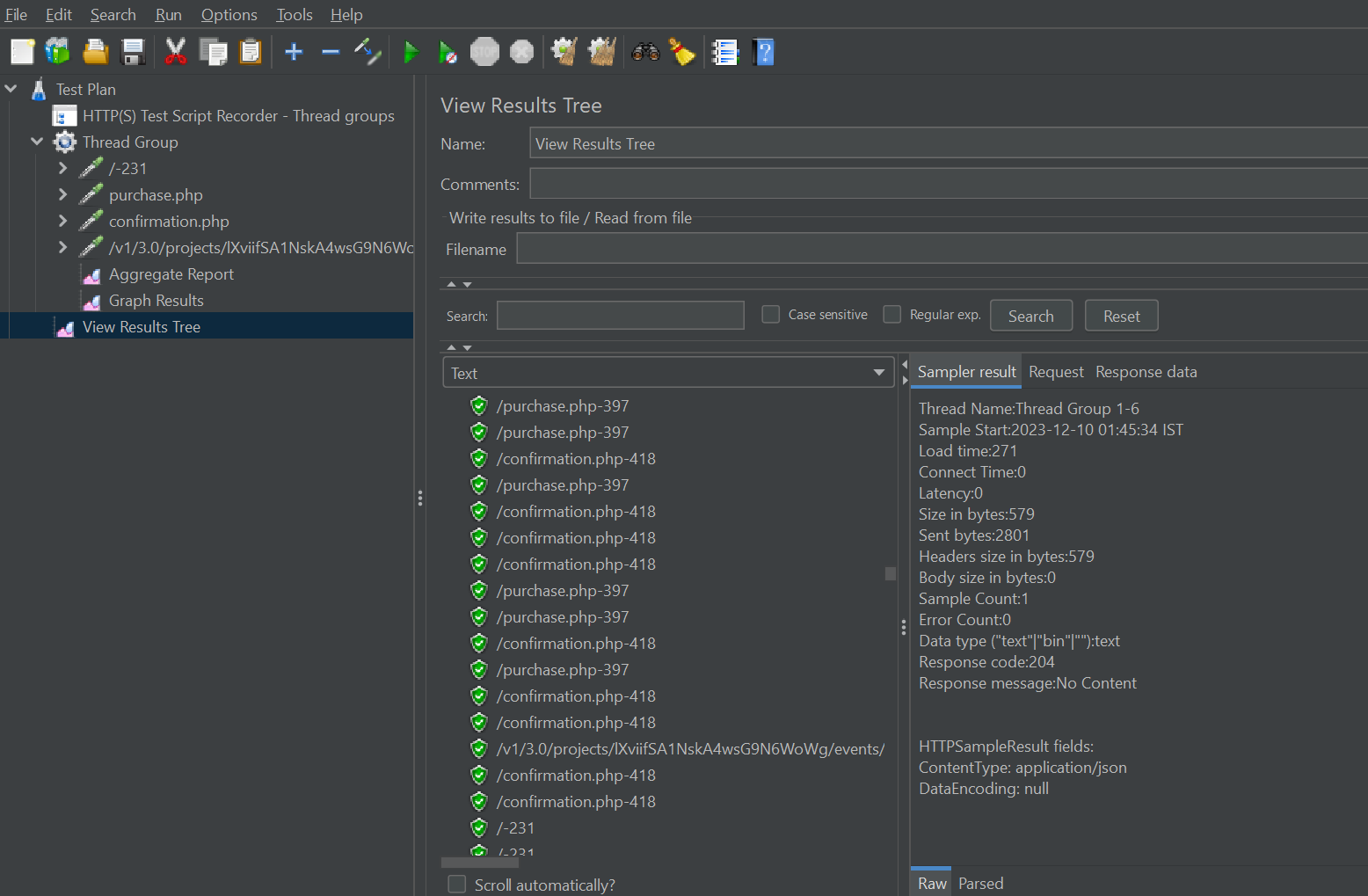
Add listener - > Aggregate report

Add listener -> Graph results

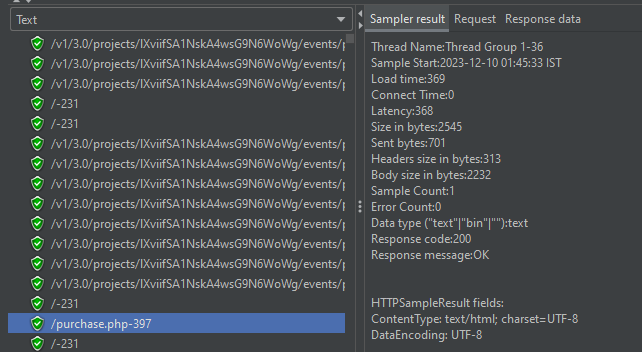


Now run the test and check the results in following metrics.

1.**View results tree**

****

Everything runs behind the screen.It is in headless mode ie..it worn’t run in UI.



Check for response message and response code in Sampler Result.

Tips: Check workload with single user and then work with multiple users.

Challenges in using Jmeter:

While running, the result tree is pass but the response body fails.

2.Aggregation Report:

A screenshot of a computer

Description automatically generated

Sample: Number of users hit the specific request

Average: it is the average time taken by all the samples to execute that request.

Min: the shortest time taken for the specific request to execute.

Max : the worst case. The maximum time taken for the specific request to execute.

Error= Fail percentage

Throughput: This is the time taken by the server in handling and processing the request. It is the number of requests processed per second by the server.

90 % of line = 90 % of samples took no more than this time

Similarly for 95% and 99% of line, it won’t take more than the mentioned time.

Deviation:

A screenshot of a computer

Description automatically generated

If the line is straight, if refers there is no deviation.

Throughtput:

A screenshot of a computer

Description automatically generated

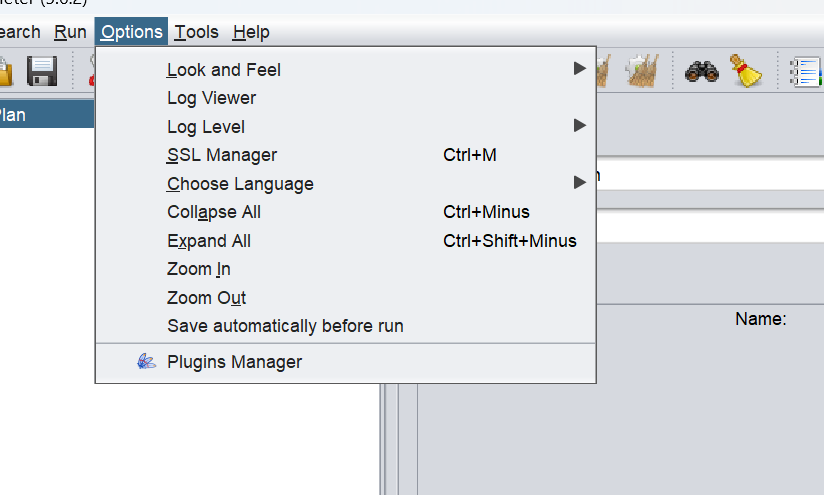
For good performance, throughput should increase and deviation should decrease. Deviation should be below average.

A screenshot of a computer

Description automatically generated

Download Jmeter plugins manager JAR file and store it in Jmeter - > bin - >ext

Now reopen the Jmeter software for this open to be added.



Download custom thread groups.

1.Adding concurrency thread group:

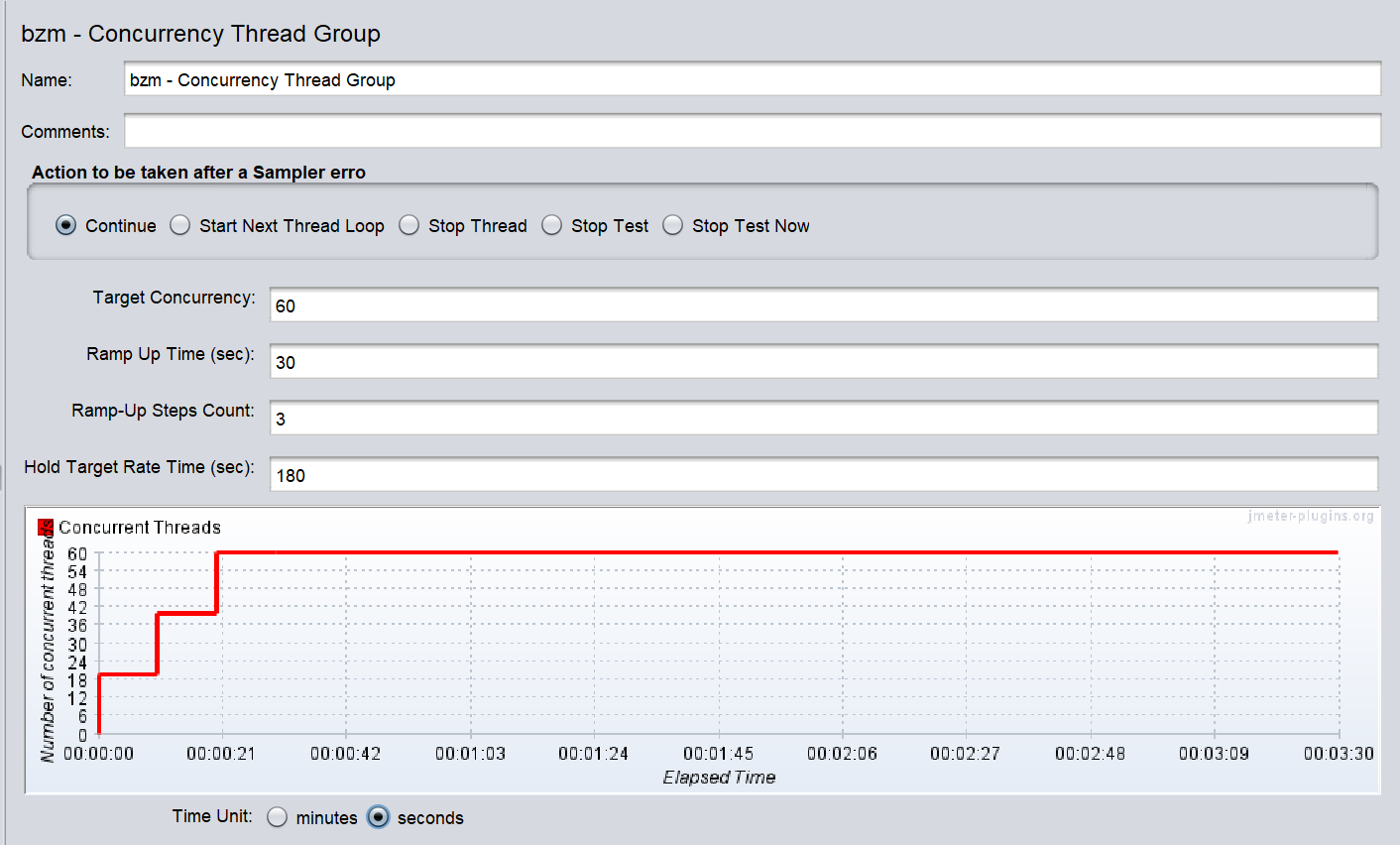
A screenshot of a computer

Description automatically generated

Setting Parameters for concurrency thread Group

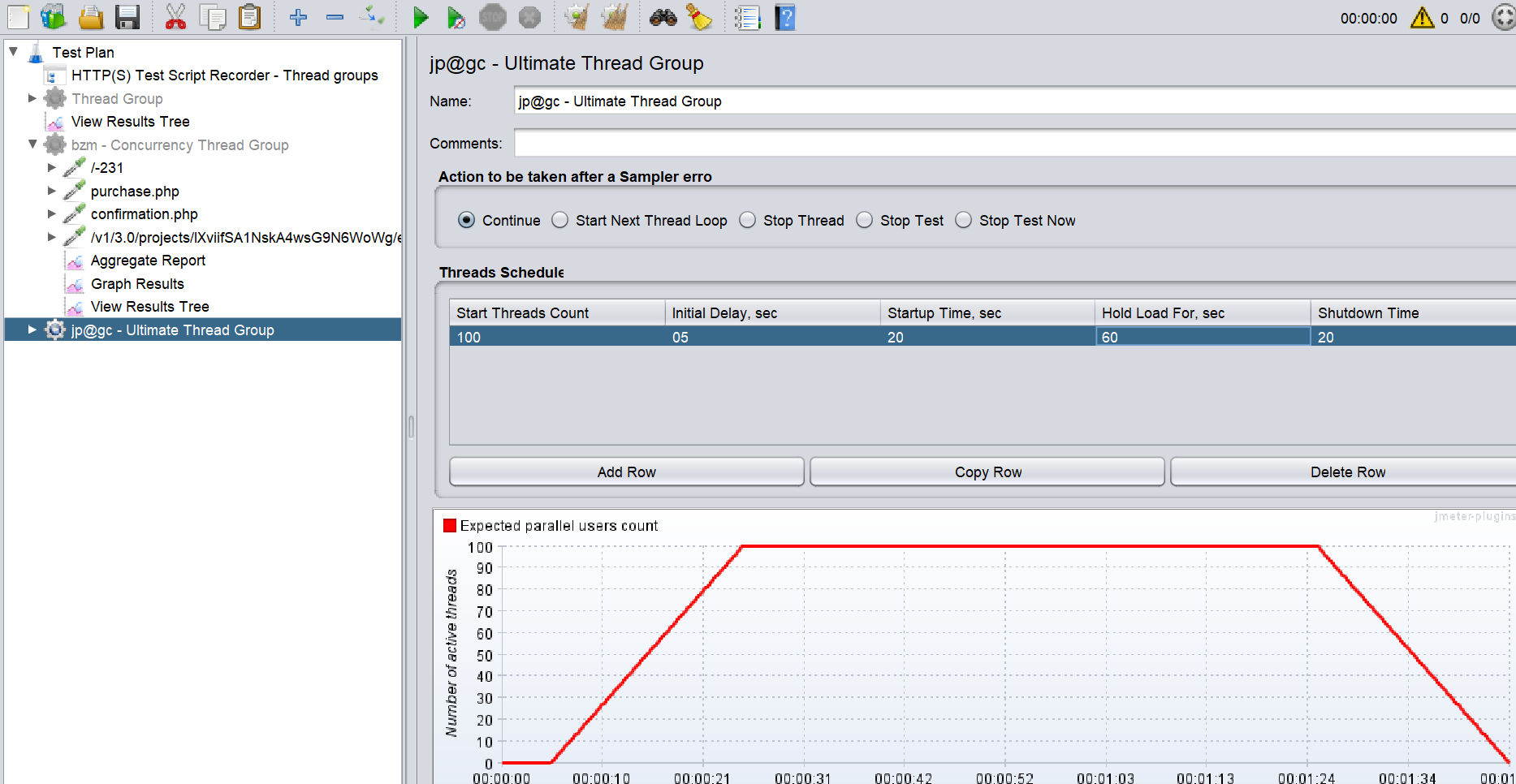
A screenshot of a computer

Description automatically generated



2.Ultimate Thread Group

This is based on the realtime sale deal. No all users will start exactly. So some delay time is produced. Ramp up time is produced , so that the probability of more user arrive from that particular time. And the shut down time is produced for sudden locking of the server.



**HTTP cookie manger:**

Log in to the demo UI link : <https://the-internet.herokuapp.com/login>

Start the https test recorder and do the login workflow.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Remove the extra scripts and do the clean up.

A screenshot of a computer

Description automatically generated

Create thread group and add a listener to it.

A screenshot of a computer

Description automatically generated

Here, though the test passes, the **response body data is not as expected**.

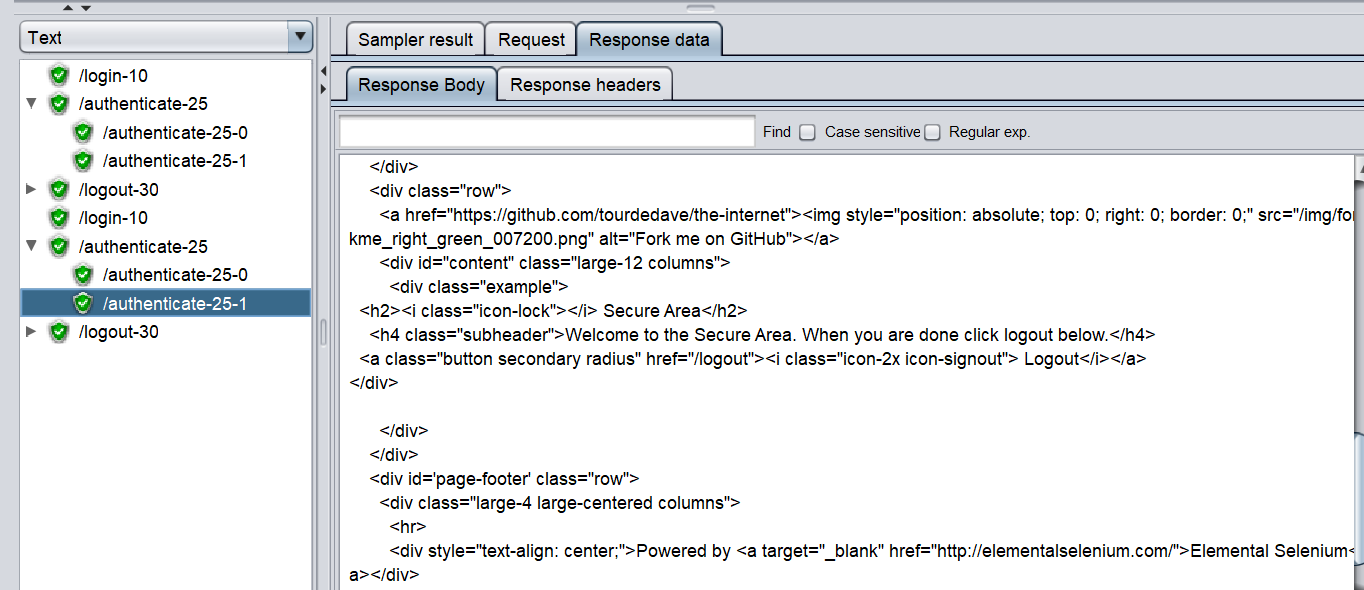
Session cookies are stored by the browser. For eg, we login to our facebook profile only once and when we reopen the facebook, we don’t need to login again. The request are sent along with session cookies to easily open our profile without logging in. Cookies are expired as previously defined.

Add http cookie manager and verify that the response for first login contains Cookies. It is sent along with the request for further run.

A screenshot of a computer

Description automatically generated

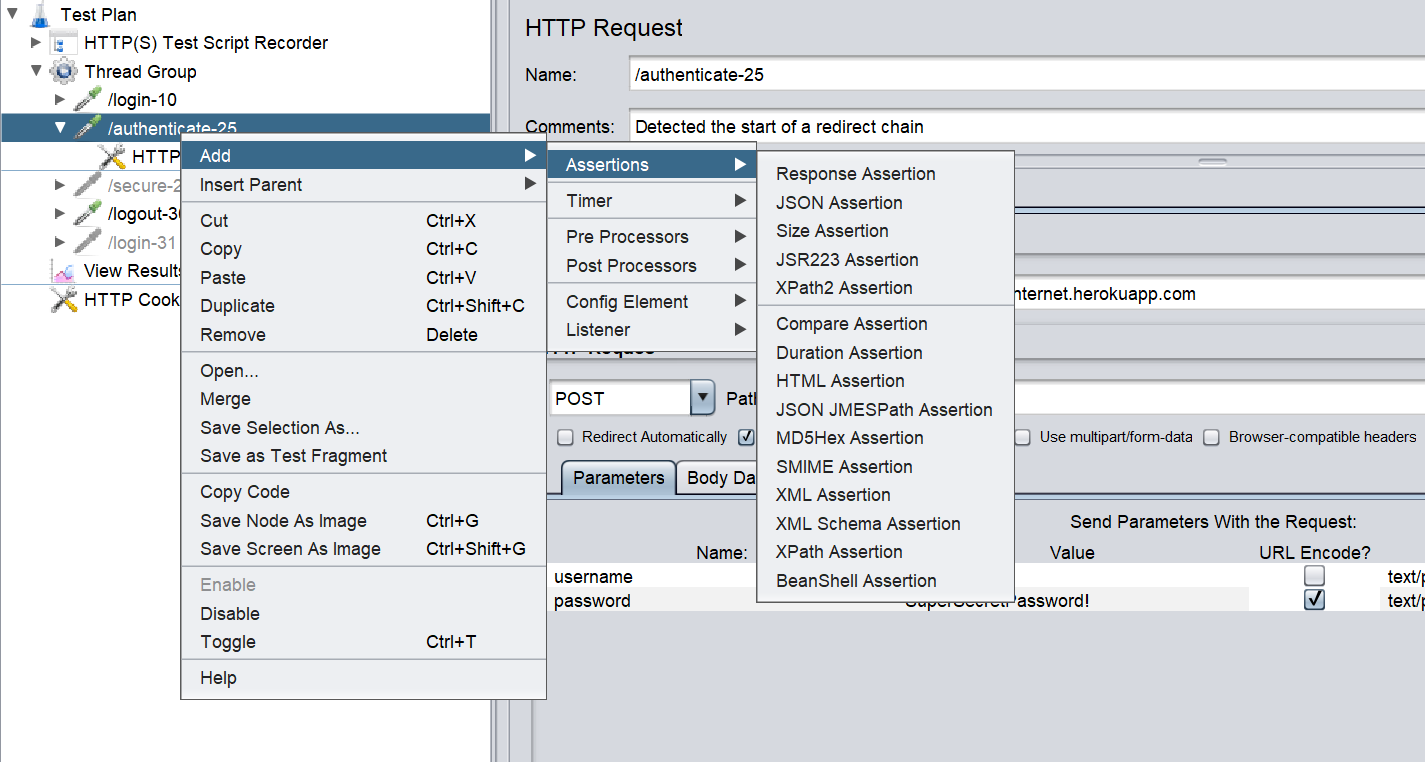
When rerunning the scripts, the session cookies are added to the request and now works fine.



**Adding Assertions**

**1.Response Assertions:**

We are going to apply response assertions on Authenticate samples.



Add the required message to check whether the given text is present or not

A screenshot of a computer

Description automatically generated

Add the response code to response assertion:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

All the test are passed.

Test is failed when cookie manager is disabled.

A screenshot of a computer

Description automatically generated

To view the assertion results -> add assertion results

A screenshot of a computer

Description automatically generated

Results are viewed while running

A screenshot of a computer

Description automatically generated

A screenshot of a computer

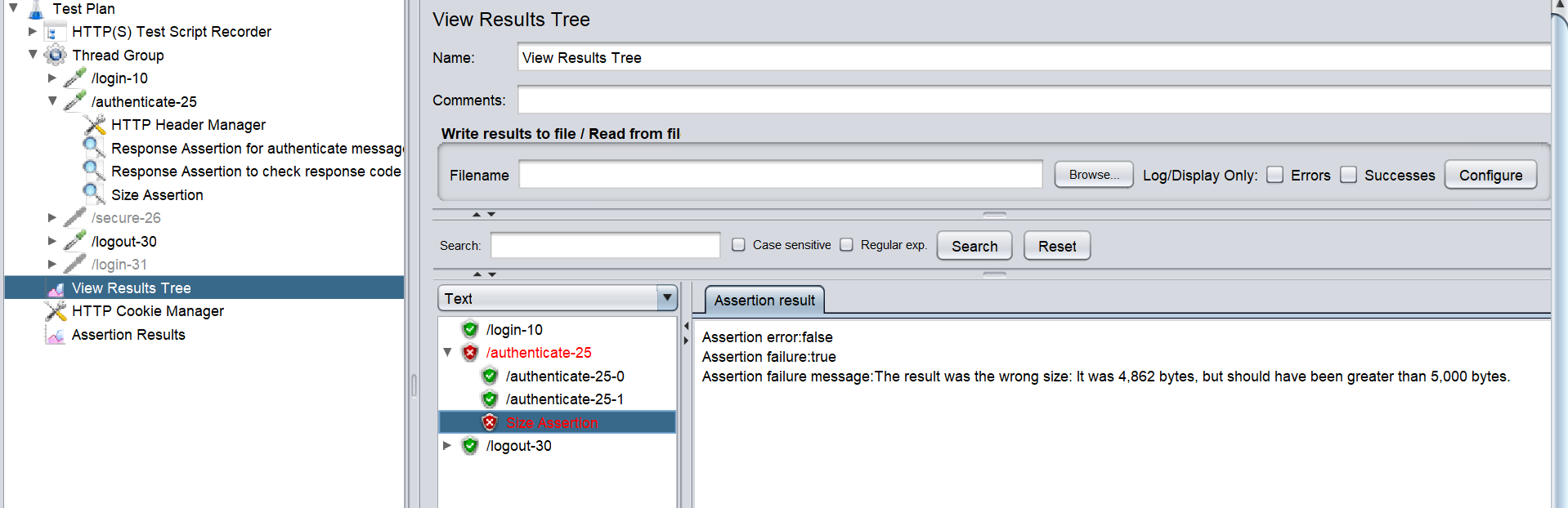
Description automatically generated

We can check this assertion with main and subsamples accordingly.

We have pattern matching results tab like: contains, equals, and , or..

2. **Size Assertion:**

When website is fully loaded by more users, some of the elements in that page is not loaded. The throughput is reduced. Hence , we should note that the “size in bytes” should not decrease on heavy load.



Now the size in bytes has been provided to check for maximum capability of loading the page

A screenshot of a computer

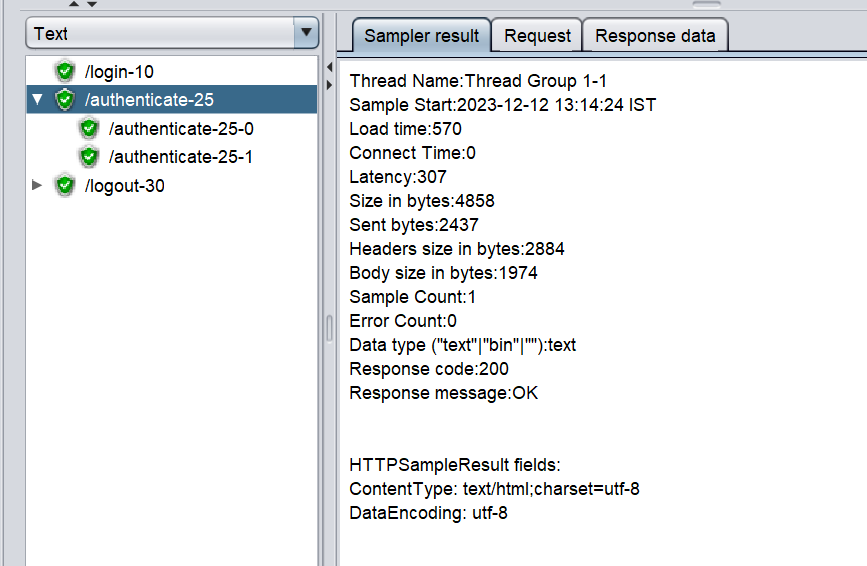
Description automatically generated

A screenshot of a computer

Description automatically generated

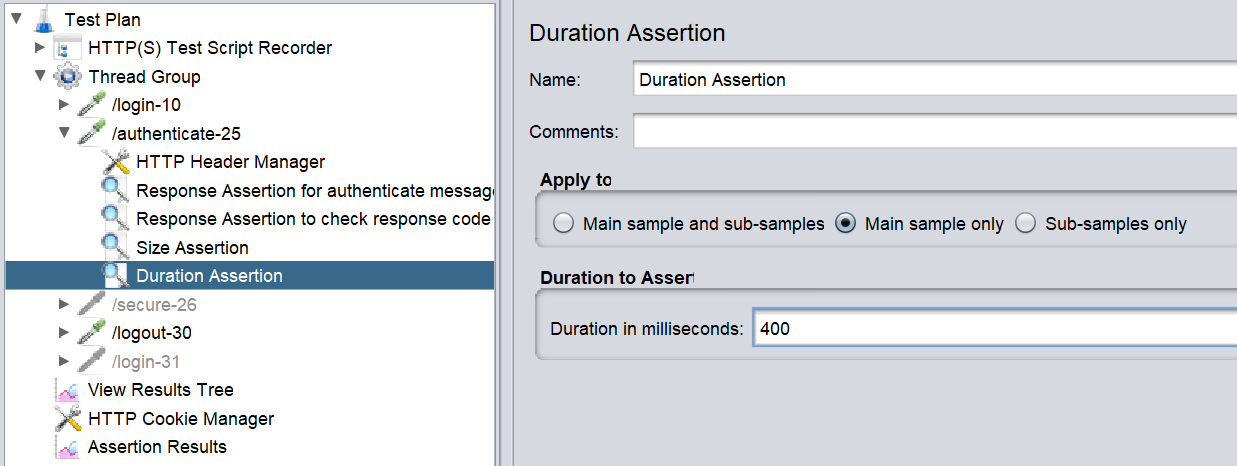
**3. Duration Assertions:**

It shows the time difference between the request sent and the response we got. Hence the application should aim for shorter time. It can be noted from the load time in sampler result.



Here load time is 570 milliseconds.

Let us try with 400 milliseconds, the test should fail.



A screenshot of a computer

Description automatically generated

Let us try with 700 milliseconds, the test should pass

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Jmeter controller for modulewise metrics:**

Download webtour sample application and load the bat file for demonstration.

Controller is also know as container which is used to store a set of requests. Sampler is also called request.

Real time flight booking Scenario:

1. Login – store in login controller
2. Search flight, give all the details and select flight = 3 request = store in flight selection controller/container
3. Give passenger details, payment transaction and confirm booking = Reservation controller.

Enter <http://localhost:1080/WebTours/index.htm> in the address bar.



User recording controller in HTTPs test script recorder and add a recording controller to thread group.

A screenshot of a computer

Description automatically generated

Create a recording controller and transactional controller

Now run the test and make a copy of the recording controller to the transactional controller.

A screenshot of a computer

Description automatically generated

Create another transactional controller for storing flight search script.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Add listeners like view results tree and aggregate report

A screenshot of a computer

Description automatically generated

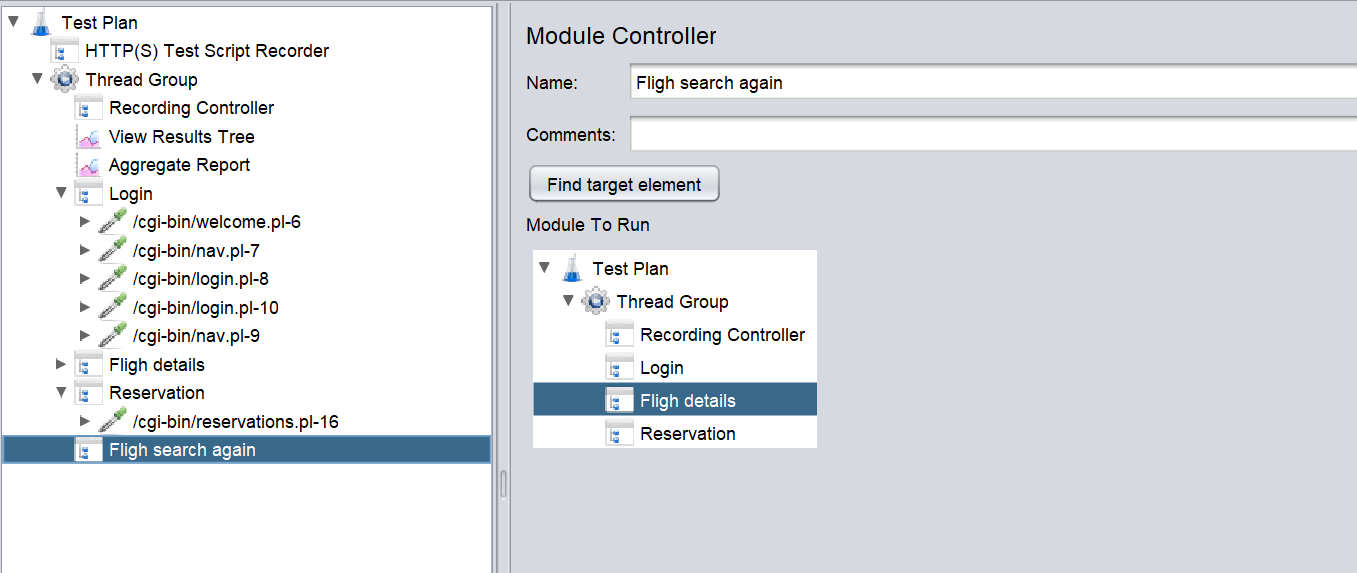
All the test are passed

A computer screen shot of a computer

Description automatically generated

**Module controller**  - it is used to run in module level. For eg.. we can do login after sign off.

Here flight details are given again using module controller



This can be seen in number of samples run

A screenshot of a computer

Description automatically generated

**Interleave controller:** One sampler per iteration is executed. It is from top to bottom.

**Runtime controller:** This controller holds the server for the give time. It is used to give more load in a specific time.

**Random controller:** picks any sampler randomly.

**If controller :** conditions are given in javascript.

Variables are stored in testplan

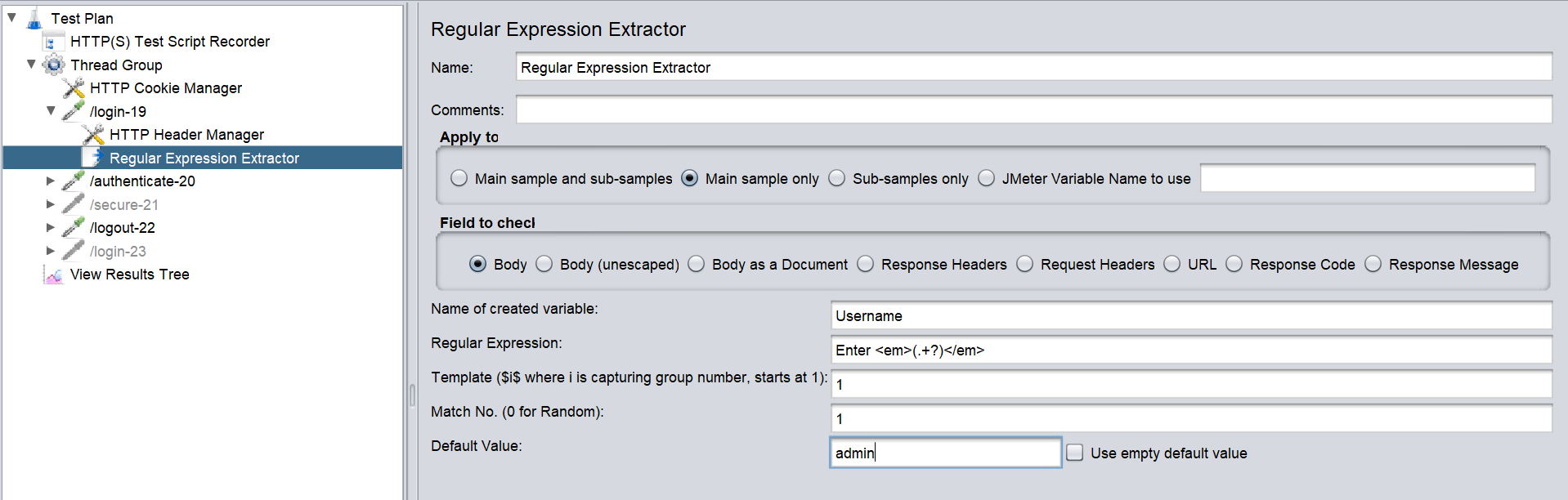
**Timers:**

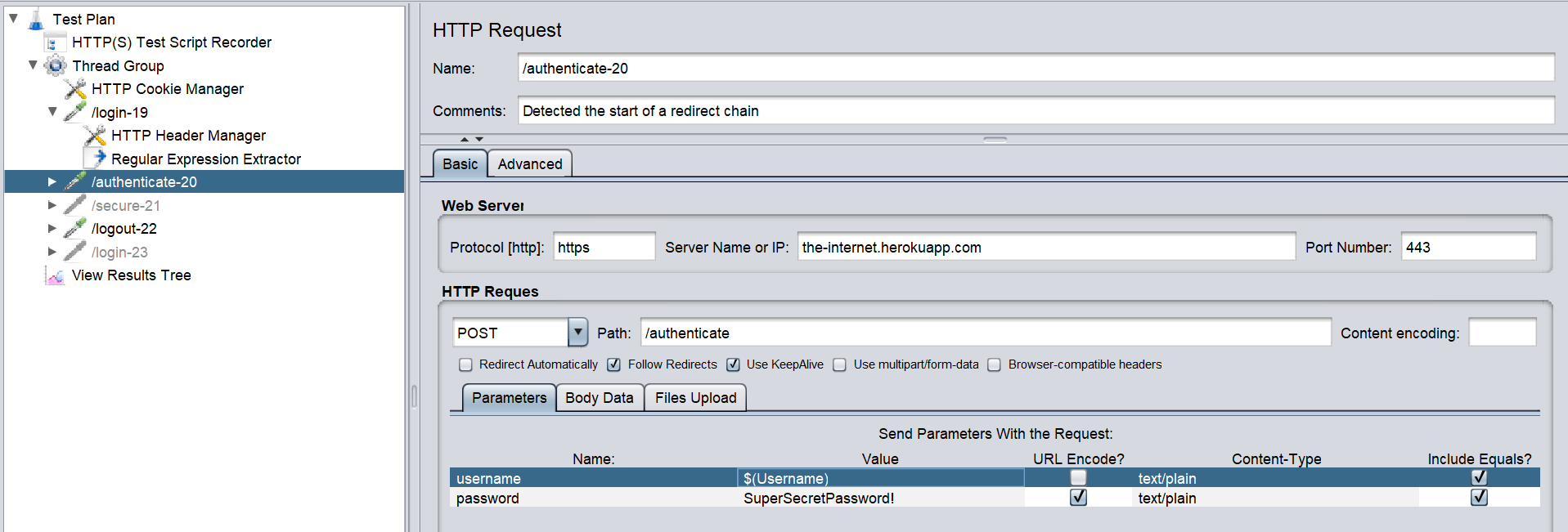
In real case, we won’t give timer between each runtime of user. This is done by timers.

**Using Regular Expressions Extractor:**

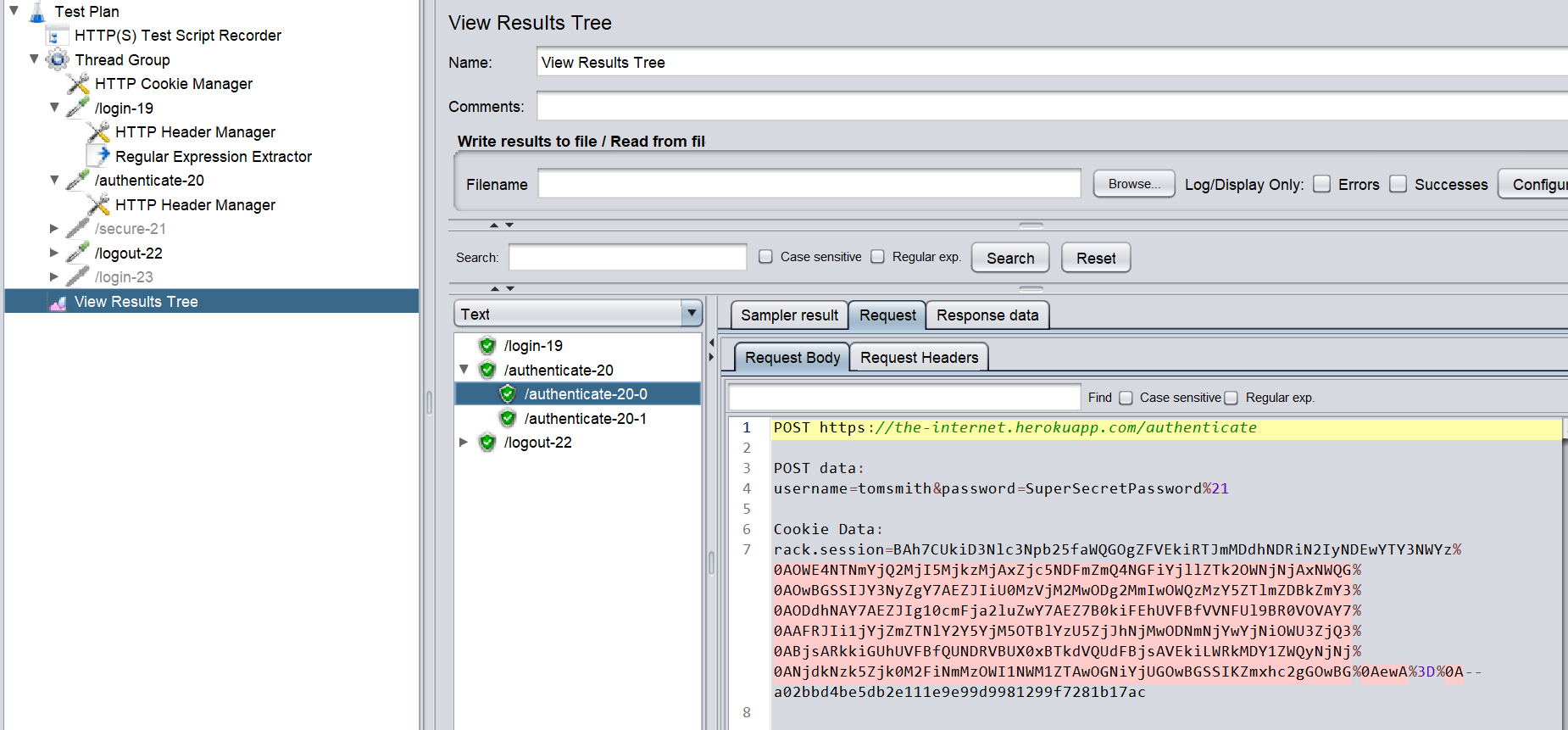
When the username is dynamically changed, we can use it by creating a variable and storing the values dynamically.

Code: .+? =any value with that match





Give it as ${Username} in parameter to get the dynamic variable.



Test is passed with dynamic username.

With single extractor we can grab multiple group name using debug sampler.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

We can pick the bulk data with csv data config file.

**Beanshell scripting:** It is used as a preprocessor steps for doing scripting and give conditional statestment for required output.

**Handling dynamic Responses:**

Our own scripts have been created for the whole flight booking.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Challenges faced**: User Session value changes dynamically for every login. Hence it should be made adapt for every login. Hence create a request before login.

Finding user session value in the response

A screenshot of a computer program

Description automatically generated

Query parameters are added along with path

A screenshot of a computer

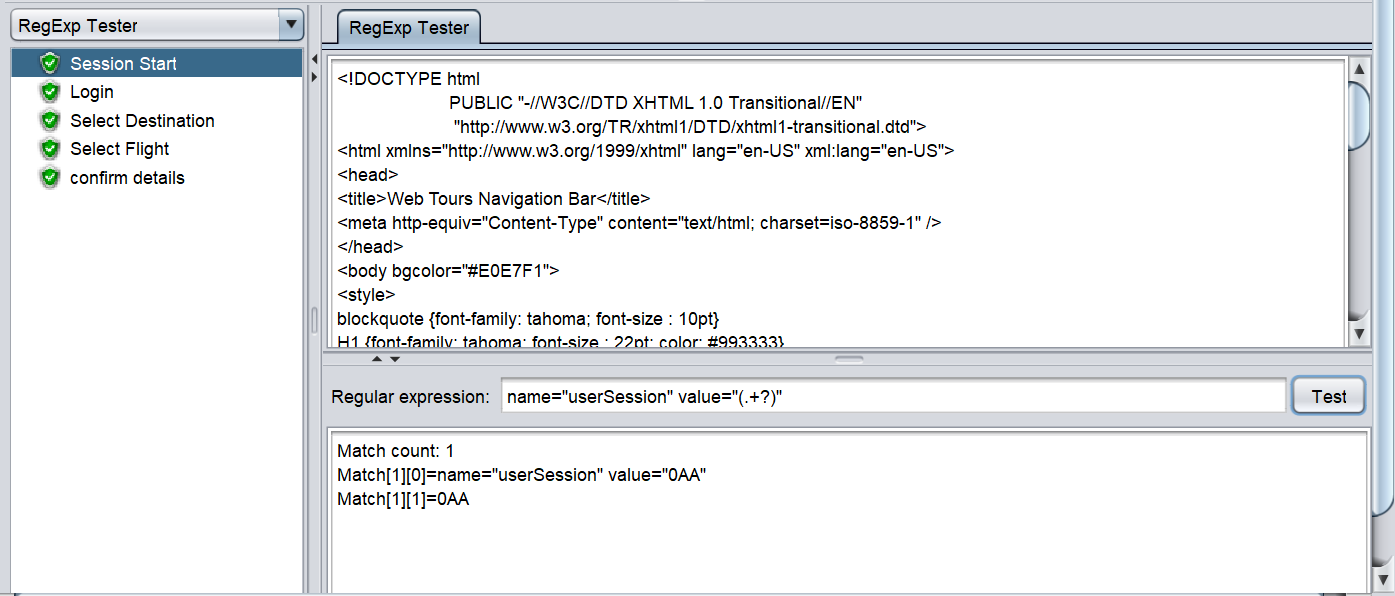
Description automatically generated

Session value is OAA

A black text on a white background

Description automatically generated

Grabbing value using Regular expression tester.



Adding regular expression in the post processor of Session Start

A screenshot of a computer

Description automatically generated

Logged in successfully.

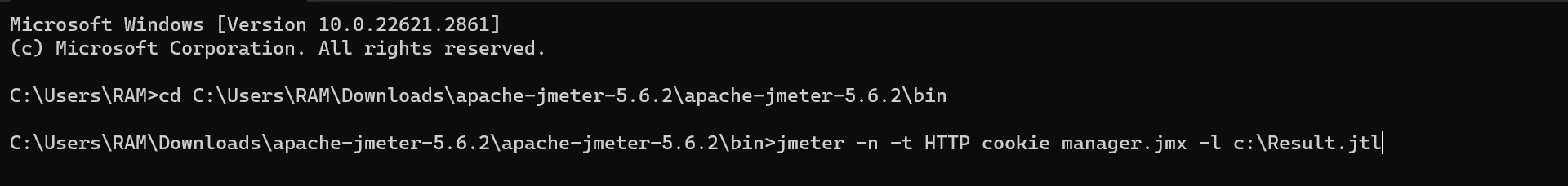
**Correlation :** It is extracting the dynamic variable in the previous http request and making it available for all the load test.

**Running JMeter in Non GUI mode:**

-n = it represents the jmeter to run in Non GUI mode.

-t = it represents the testplan name.

-l = for reporting results



Upload the file in blazemeter.com and view the performance result.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Results:

A screenshot of a computer

Description automatically generated

**Notes:** It is not recommended to use many users in single machine because heavy load uses more memory space ,cpu capacity and at certain point the test produces wrong results because of no ram space.

**Jmeter Distributed Service:**

The master machine distributes the scripts to server machines and then serves the required site.

Select and start the slave jmeter slave server. Note the IP address of slave and update in master jmeter properties. Now restart the jmeter master machine and run the selected nodes.

Using team viewer, we can view master and slave jmeter separately.

**Notes:**

Two connected systems should have similar Jmeter version, sub network, same OS, same directory tree.

**Monitoring Server Performance:**

Previously we monitored web application performance. Here we monitor the server performance.

**Client server monitoring:**

When we hit the server hosted by the owner, the server performance for the client is measured. Hence yourkit java profiler is used for monitoring **server performance.**

Your kit activation key: **YEVALJ9S0NS1TJAU1MV718**

A screenshot of a computer

Description automatically generated

Attaching with Jmeter

A screenshot of a computer

Description automatically generated

Heap memory/dynaminc memory: Once the java job is done, the memory is released.

Non heap memory/ Statis memory: once the java job is done, the memory is not released.

A screenshot of a computer

Description automatically generated

CPU processing view while the thread is running in apache Jmeter.

Snapshot and the generations are captured.

A screenshot of a computer

Description automatically generated

Both client and server machine’s web application performance and server performance should be measured.