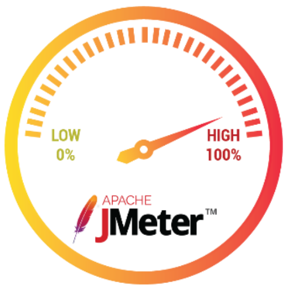


Performance Testing using Apache JMeter

Software Requirements Specification (SRS) Document

1. Introduction

1.1 Purpose

The purpose of this document is to define the scope, objectives, and requirements for performance testing of the application using Apache JMeter. This document outlines the test scenarios, success criteria, and performance benchmarks to ensure the system meets the required performance standards.

1.2 Scope

The performance testing will evaluate the responsiveness, stability, scalability, and reliability of the application under various load conditions. The tests will be conducted using JMeter to simulate user loads and analyse system behaviour under different conditions such as load, stress, endurance, and spike testing.

2. Performance Testing Objectives

- **Load Testing: Validate the application's ability to handle the expected number of concurrent users and transactions.**
- **Stress Testing: Identify the application's breaking point and measure how it recovers from high traffic.**
- **Stability Testing : Checking the performance of an application by applying the load for a particular duration of time is known as Stability Testing.**
- **Scalability Testing: Determine if the application can scale up or down efficiently with increasing or decreasing load.**

3. Test Strategy

3.1 Tools & Technologies

- **Performance Testing Tool: Apache JMeter**
- **Monitoring Tools: New Relic, Grafana, JConsole, or server logs**
- **Environment: Staging/Pre-production environment**
- **Test Data: Simulated user data for realistic load testing**

3.2 Performance Metrics

- **Response Time (Average & 90th percentile)**

- **Throughput (requests per second)**
- **Error Rate (% of failed requests)**
- **CPU & Memory Utilization**
- **Database Query Performance**

3.3 Assumptions & Constraints

- **Test execution will be in a controlled environment.**
- **Test data will be generated using scripts or preloaded datasets.**
- **Network latency and third-party integrations are not in scope.**

4. Information Architecture:

4.1 Add Thead Group

Thread Group - <users to perform action> Thread Group is an element that controls the number of threads (basic users), the ramp-up period, and the loop count of the requests. These include:

- Number of threads: 50
- Ramp-Up Period(ins): 5
- Loop Count: 3

The above configuration means that each user will hit the target server 3 times(loop count-3) and in total there will be 50 users(number of threads) hitting the target server with a time difference between two consecutive user requests of 5s(ramp-up period). Ramp-up period basically means if user 1 puts request at 1.00:00 PM then user 2 will wait for 5s and put his request at 1:00:05 PM. Similarly, user 3 will wait for 5s after user 2 hit then puts his request, and so on.

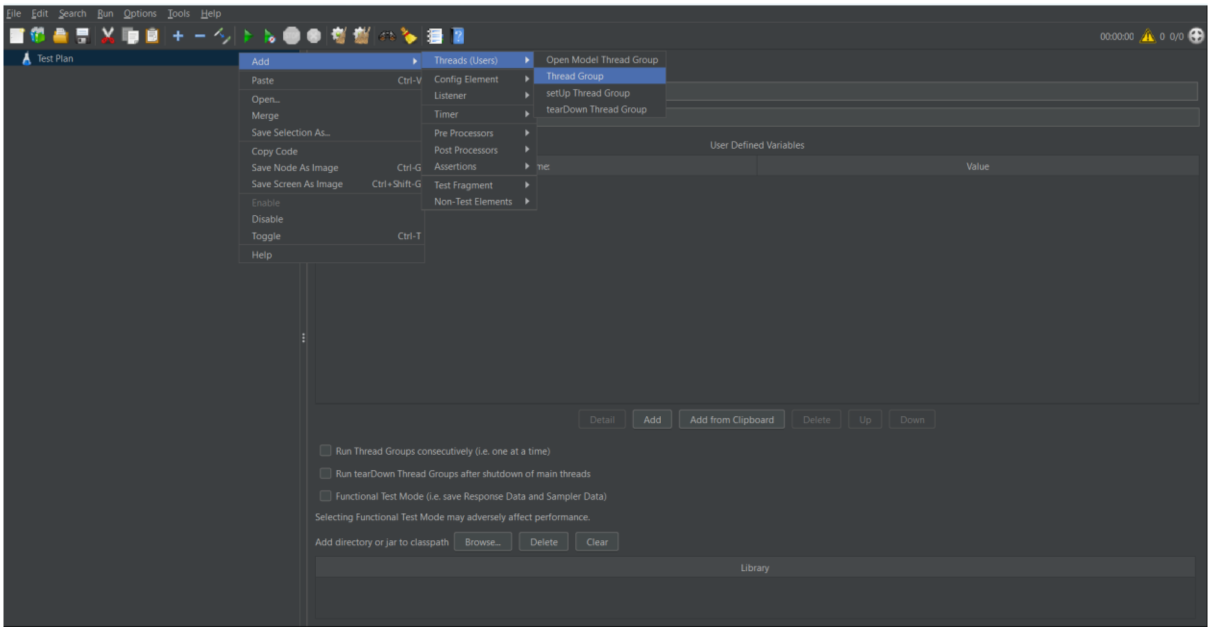
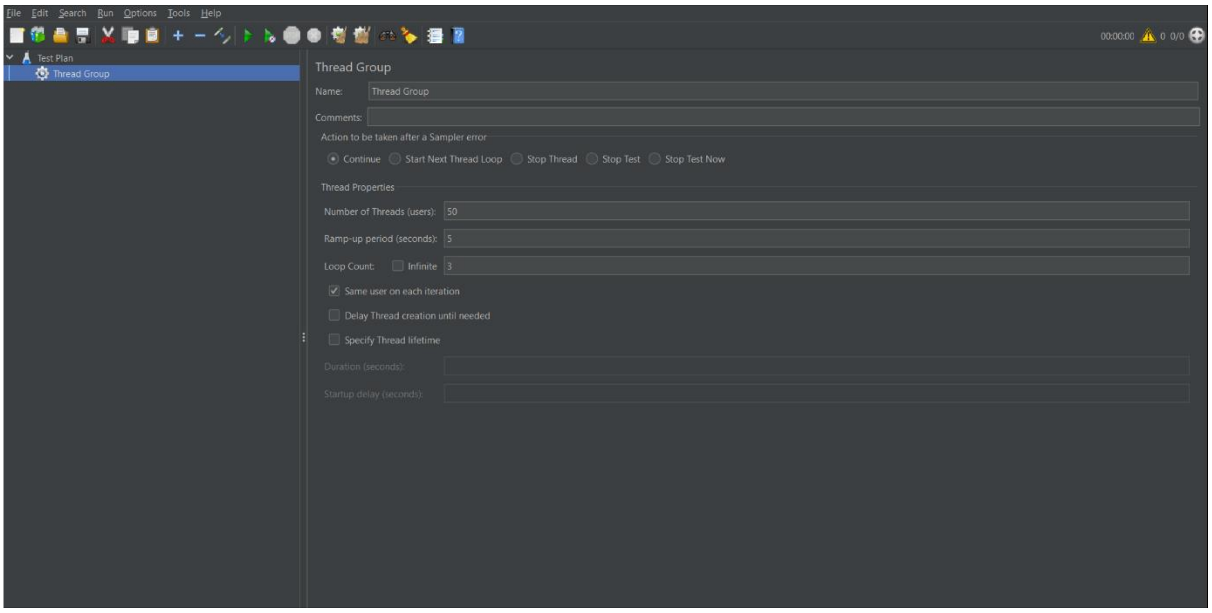


Figure 1



Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 50

Ramp-up period (seconds): 5

Loop Count: Infinite 3

[✓] Same user on each iteration

Delay Thread creation until needed

Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

Figure 2

4.2 Default HTTP Request -Controllers - Samplers - <Action to perform>

Path - Add -> Config Element -> HTTP Request Defaults.

◦ Protocol by default is “https”. If target is “http” then specify explicitly ; else leave it as “https”

◦ This Default HTTP request will capture the default settings for our HTTP requests that our users will be making onto the target server. This default setting will be used by another sampler(HTTP request) for its defaults.

◦ We will set web server/IP as www.google.com

HTTP Request -

◦ Path - Add -> Sampler -> HTTP Request

◦ This can be considered a further extension of default HTTP requests.

◦ Suppose we want to go to the Calendar page of google

◦ We will enter the path as “calendar”

◦ JMeter will append “calendar” with server-name of Default HTTP Request = https://www.google.com/calendar

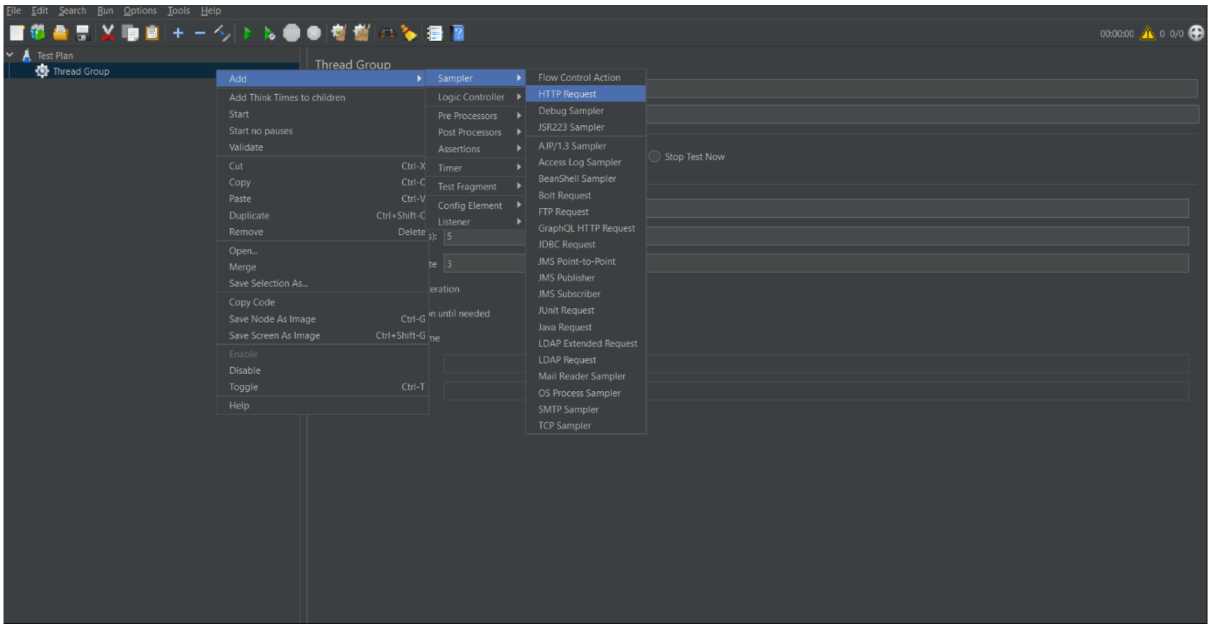


Figure 3

4.3 Listeners

Listeners - <Post Action listen to outcome and show> Listener basically provides different reports that the tester can view. It will provide more information on the outcome of the actions performed and will help the tester to analyse the performance results in depth. It can be set different reports to measure the performance.

- Path - Add -> Listener -> Graph Results
- OR Path- Add → Listener -> Summary Report

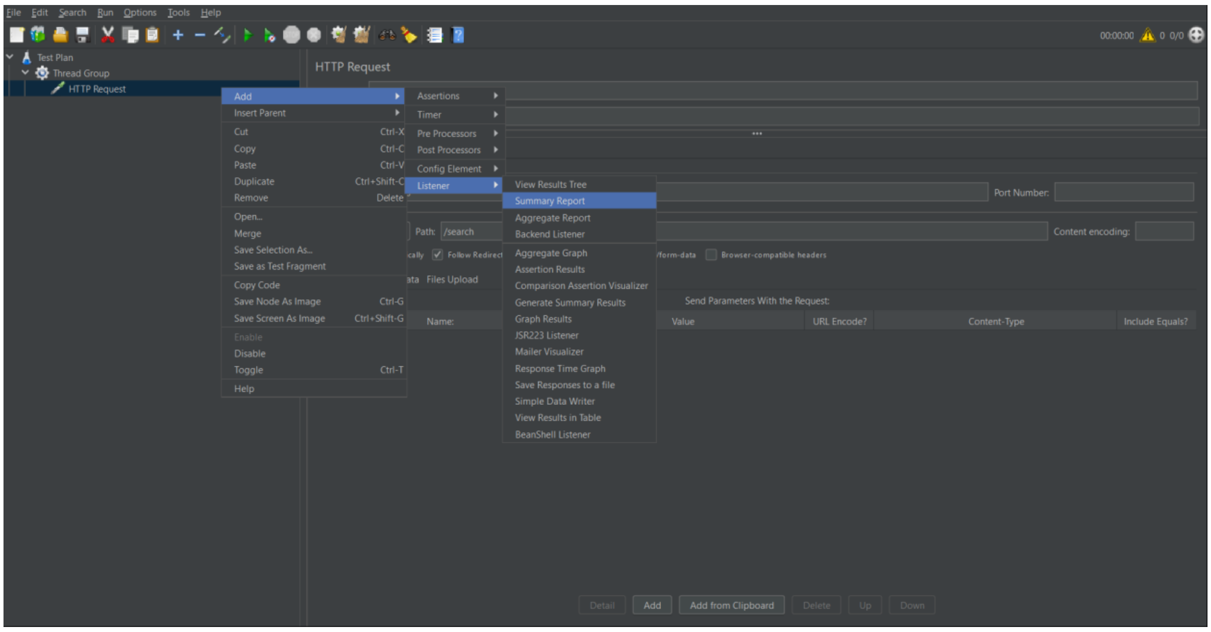


Figure 4

4.4 JMeter Timers

JMeter is built to send many requests at once without pausing between each request of each user. In that case, it could overwhelm the target server by making too many requests in a short amount of time. Server overloading happens if I send so many thousands of requests to the target web server in a few seconds. Now comes the concept of the timer in JMeter to delay each request that a thread (user) makes. This overloading problem of the target server can be resolved by Timer. Real-time also users take time to reach the website and make the clicks by analyzing the website, etc. So using a timer will mimic the real-time behavior of the user.

Types Such as :

1-Constant Throughput Timer

JMeter Constant Throughput Timer is used as an element to achieve a goal like achieving the desired throughput (Total Number of Requests). This timer tries to maintain a constant throughput throughout the test and achieve the target.

2- Gaussian Random Timer

This timer is used to generate and add the random delay before the execution of a sampler. This timer is based on the Normal or Gaussian Distribution Function.

3- Uniform Random Timer : Uniform Random Timer basically delays each user request for a random amount of time The inputs include:

- Name = name of the timer
- Random Delay Maximum = maximum number in milliseconds to delay (RANDOM)
- Constant Delay Offset (milliseconds) = Additional value in ms .

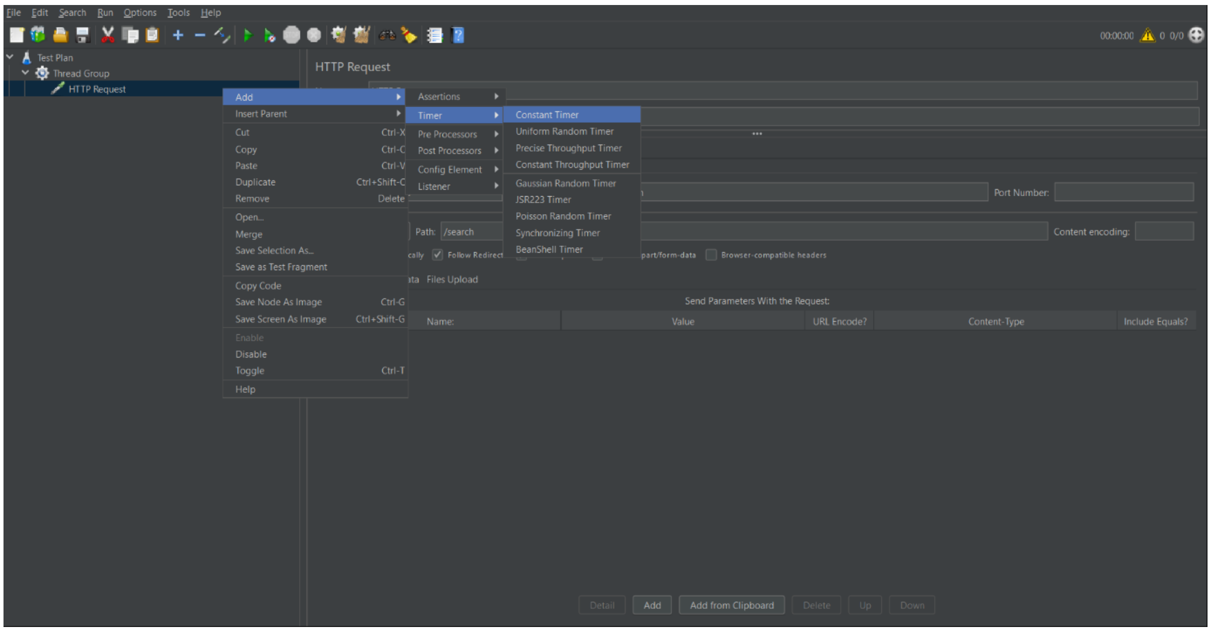


Figure 5

4.5 JMeter Assertions : Performance testing falls under the non-functional testing category. But we need to also get the correct functional response. We have the option to validate the response from an incoming server and then based on that a warning message can be shown if the validation fails.

Response Assertion :is used to validate the presence of a particular string (pattern) or some fields like response code, response message, etc. The pattern string could be a number, word, statement, etc. which could be present either in the server response or in the JMeter request.

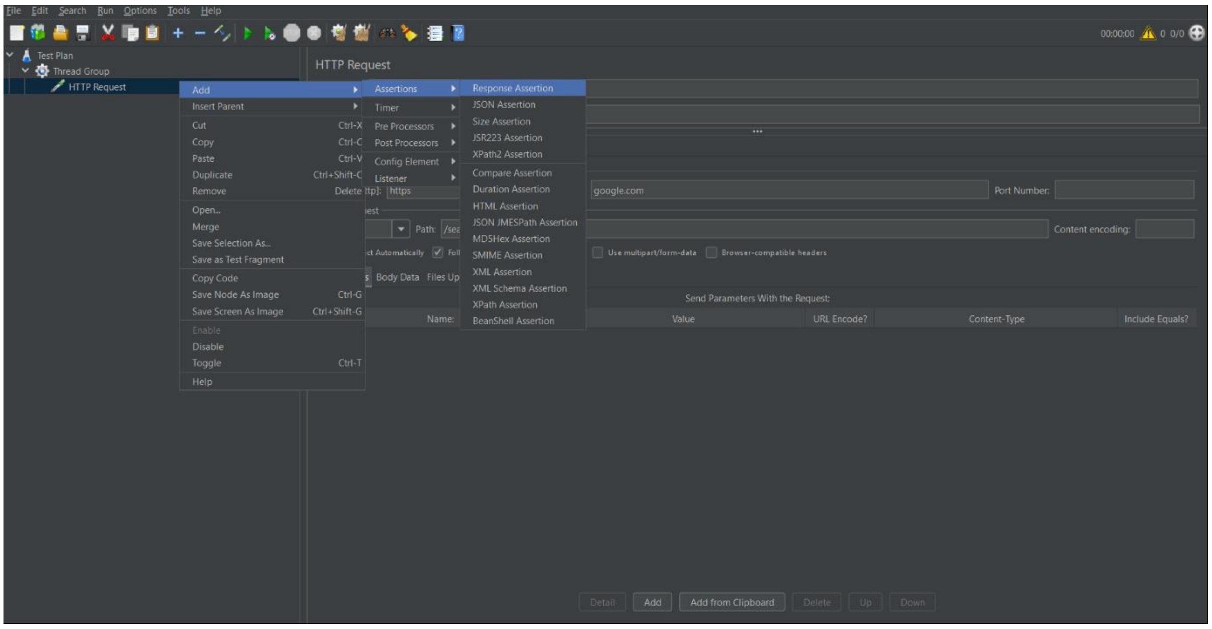
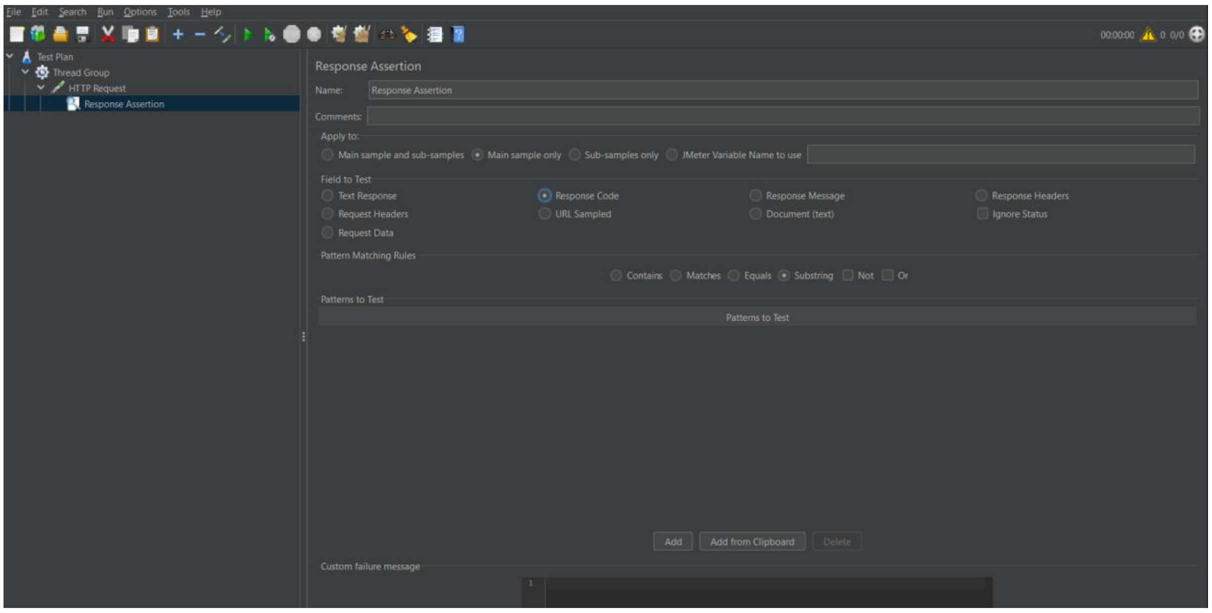


Figure 6



Response Assertion

Name: Response Assertion

Comments:

Apply to: Main sample and sub-samples [✓ Main sample only] Sub-samples only JMeter Variable Name to use

Field to Test: Text Response [✓ Response Code] Response Message Response Headers Request Headers URL Sampled Document (text) Ignore Status Request Data

Pattern Matching Rules: Contains Matches Equals [✓ Substring] Not Or

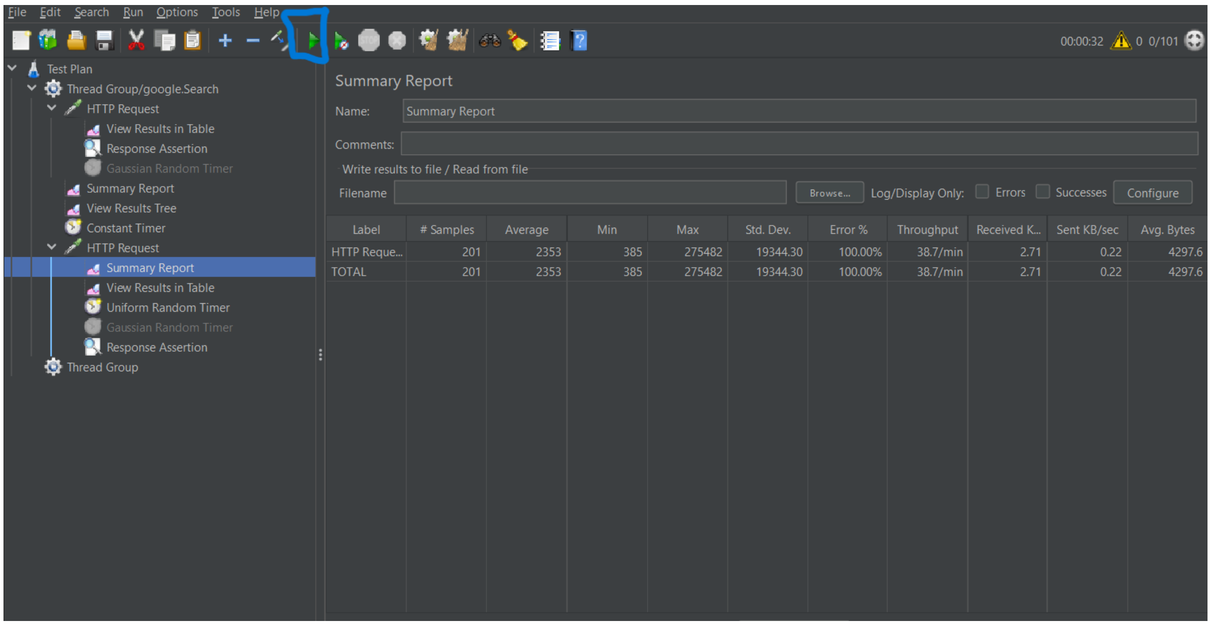
Patterns to Test

Custom failure message

Figure 7

4.6 Save test plan Press Ctrl+S to save the test plan. Now it is ready to run. Every JMeter test plan file is saved in the file system with .jmx extension

4.7 Run test plan Now click on the Play button to run the test plan. Or press Ctrl+R.



Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: ☐ Errors ☐ Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Reque...	201	2353	385	275482	19344.30	100.00%	38.7/min	2.71	0.22	4297.6
TOTAL	201	2353	385	275482	19344.30	100.00%	38.7/min	2.71	0.22	4297.6

Figure 8