East West University

Lab Report 04

Topic: Load Testing with Apache JMeter

Course Title: Software Testing and Quality Assurance

Course Code: CSE 430

Section No: 01

Submitted By

Md. Saiful Islam

2022-3-60-045

Submitted To

Dr. Shamim H Ripon

Professor

Department of Computer Science and Engineering

East West University

Task 1: Load Test a Public Website

Objective:

To evaluate the response time and stability of a public website under moderate load conditions.

Target URL:

https://example.com

Test Configuration:

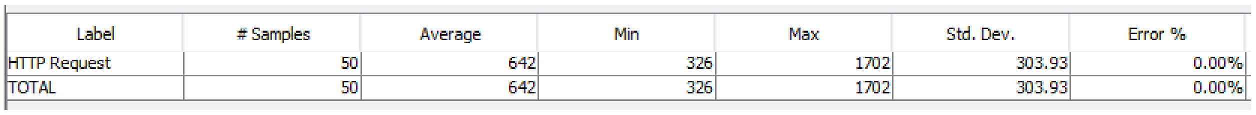
- Virtual Users: 25
- Ramp-Up Time: 15 seconds
- Loop Count: 3

Steps:

- open: Launch Apache JMeter and create a new Test Plan.
- addThreadGroup: Configure Thread Group with 25 users, 15s ramp-up, and loop count 3.
- addHTTPRequestDefaults: Set protocol to HTTPS and server name to example.com.
- addHTTPRequest: Configure a GET request with path “/”.
- addResponseAssertion: Validate the response contains the expected page text.
- addListeners: Add Summary Report and View Results Tree.
- runTest: Execute the test plan.

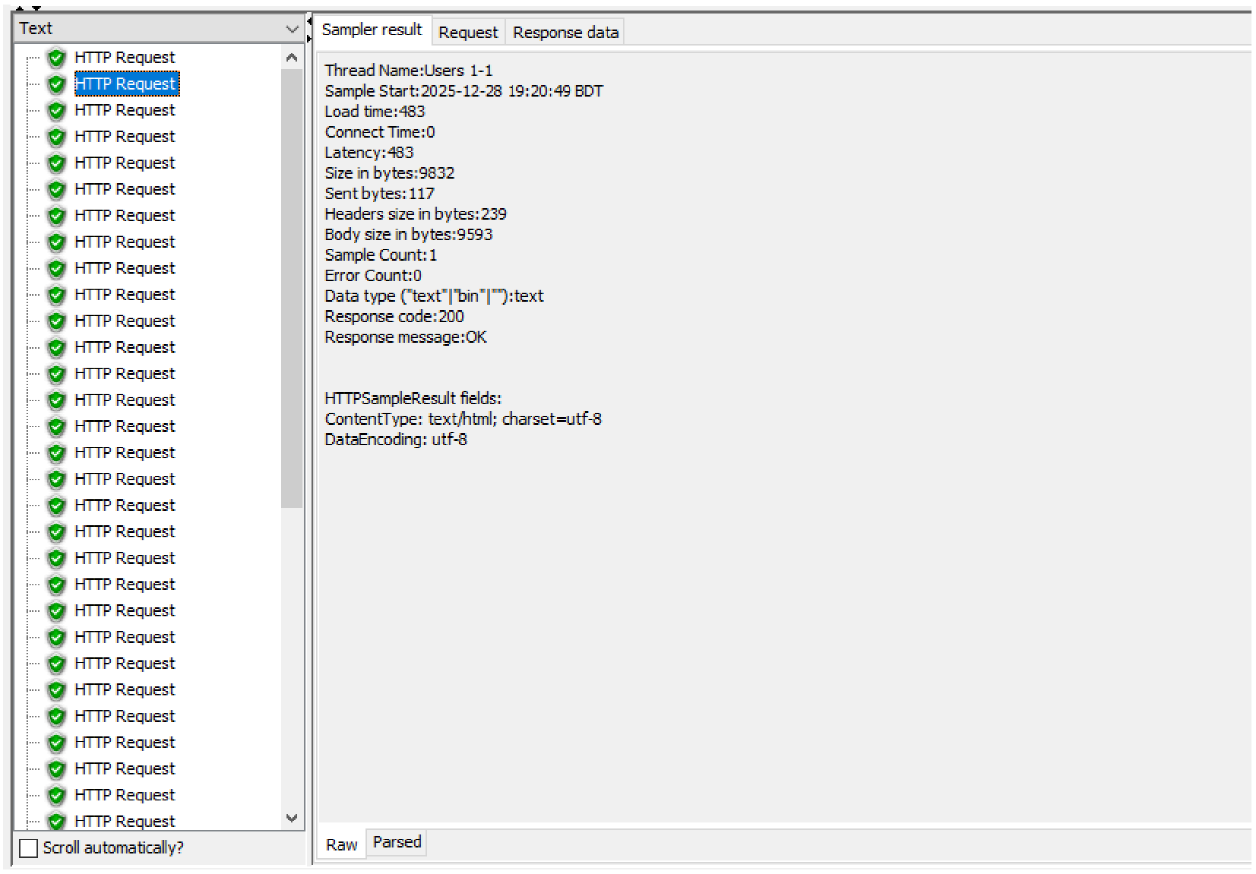
Output:

Summary Report:



Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	50	642	326	1702	303.93	0.00%
TOTAL	50	642	326	1702	303.93	0.00%

Result Tree:



```
Thread Name:Users 1-1
Sample Start:2025-12-28 19:20:49 BDT
Load time:483
Connect Time:0
Latency:483
Size in bytes:9832
Sent bytes:117
Headers size in bytes:239
Body size in bytes:9593
Sample Count:1
Error Count:0
Data type ("text"|"bin"|""):text
Response code:200
Response message:OK


HTTPSampleResult fields:
ContentType: text/html; charset=utf-8
DataEncoding: utf-8
```

The website responded successfully to all requests with no errors. Response time remained stable throughout execution.

Reflection:

This task demonstrated how Apache JMeter can simulate real-world user traffic and measure website performance. It confirmed that the target website can handle moderate load without failures.

Task 2: Simulate Login Using Fake Store API

Objective:

To validate login functionality and token generation using an API endpoint.

Target URL:

https://fakestoreapi.com/auth/login

Test Configuration:

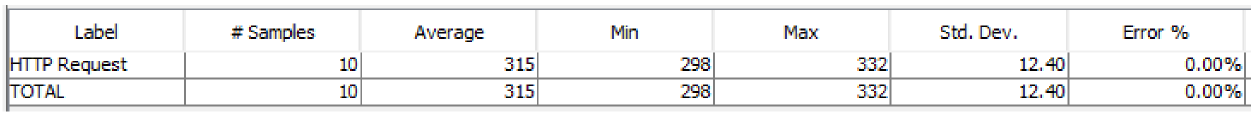
- Virtual Users: 5
- Loop Count: 2

Steps:

- open: Create a new Thread Group with 5 users.
- addHTTPRequestDefaults: Set server name to fakestoreapi.com.
- addHTTPRequest: Configure HTTP method as POST with path /auth/login.
- addParameters: Provide valid username and password.
- addResponseAssertion: Verify response contains “token”.
- runTest: Execute the test plan.

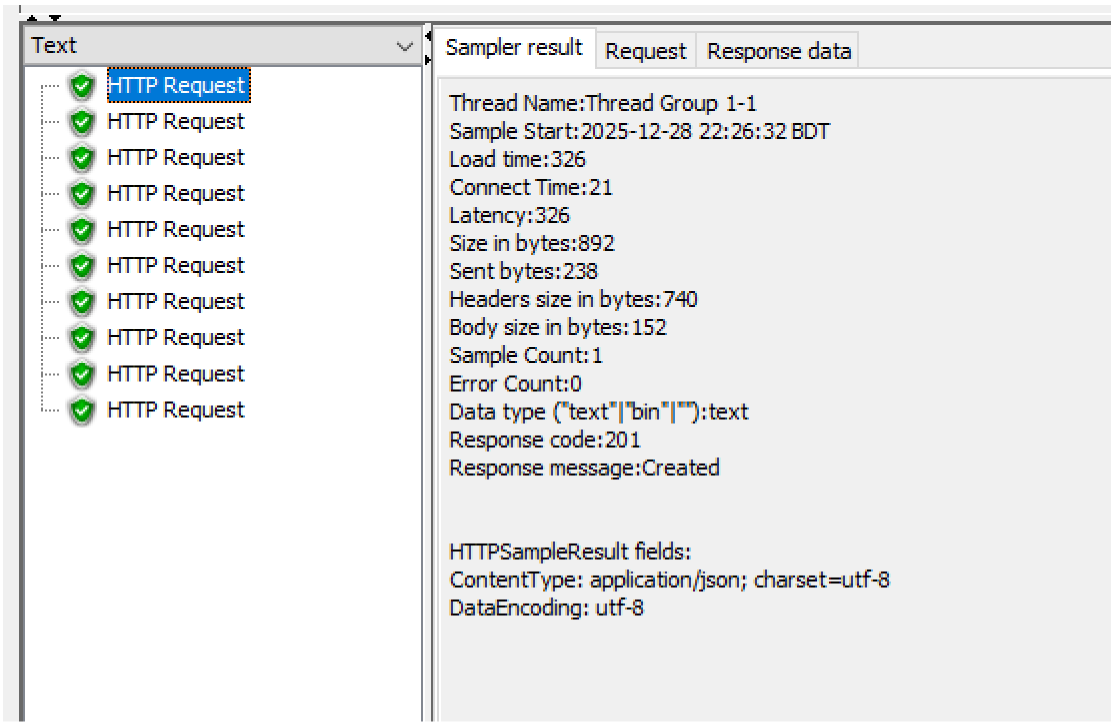
Output:

Summary Report:



Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	10	315	298	332	12.40	0.00%
TOTAL	10	315	298	332	12.40	0.00%

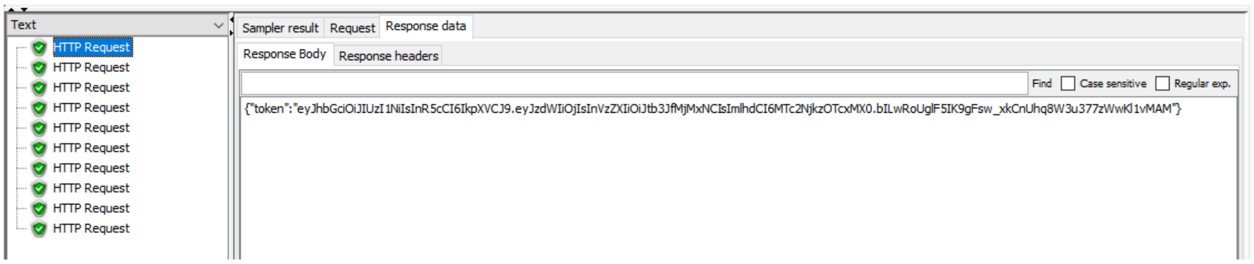
Result Tree:



```
Thread Name:Thread Group 1-1
Sample Start:2025-12-28 22:26:32 BDT
Load time:326
Connect Time:21
Latency:326
Size in bytes:892
Sent bytes:238
Headers size in bytes:740
Body size in bytes:152
Sample Count:1
Error Count:0
Data type ("text"|"bin"|""):text
Response code:201
Response message:Created


HTTPSampleResult fields:
ContentType: application/json; charset=utf-8
DataEncoding: utf-8
```

Response Token:



```
{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOjIsInVzZXIiOiJtb3JfMjMxNCIsImlhdCI6MTc2NjkzOTcxMX0.bILwRoUglF5IK9gFsw_xkCnUhq8W3u377zWwKl1vMAM"}
```

The API returned a token for each successful login request.

Reflection:

This task helped in understanding how JMeter handles POST requests and API testing. It verified correct authentication behavior and successful response validation.

Task 3: Parameterized Search Test Using GitHub API

Objective:

To perform dynamic testing using multiple search inputs from a CSV file.

Target URL:

https://api.github.com/search/repositories

Test Data (CSV):

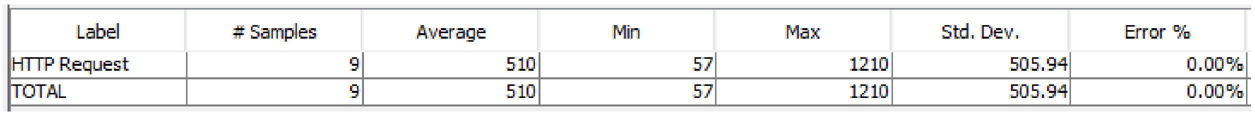
Keywords: JMeter, Java, testing

Steps:

- open: Create a new Test Plan and Thread Group.
- addCSVDataSetConfig: Load search keywords from a CSV file.
- addHTTPRequestDefaults: Set server name to api.github.com.
- addHTTPRequest: Use GET request with parameter ${keyword}.
- addResponseAssertion: Verify response contains “items”.
- runTest: Execute the test.

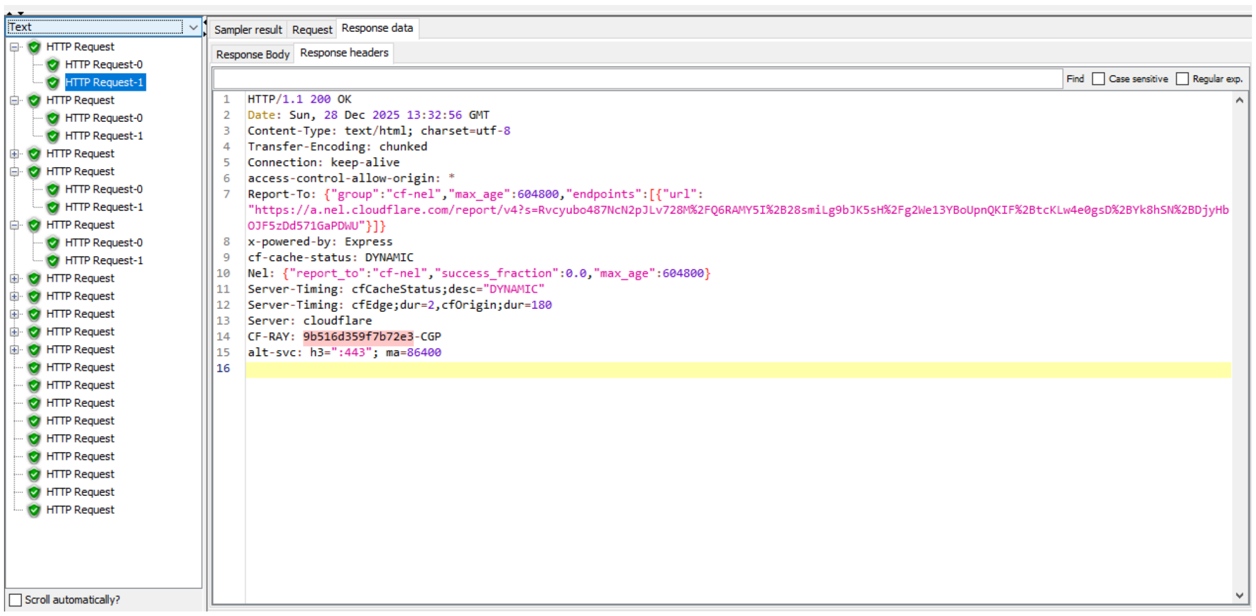
Output:

Summary Report:



Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request	9	510	57	1210	505.94	0.00%
TOTAL	9	510	57	1210	505.94	0.00%

Response Data:



```
HTTP/1.1 200 OK
Date: Sun, 28 Dec 2025 13:32:56 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
access-control-allow-origin: *
Report-To: {"group":"cf-nel","max_age":604800,"endpoints":[{"url":
"https://a.nel.cloudflare.com/report/v4?s=Rvcyubo487NcN2pJLv728M%2FQ6RAMY5I%2B28smiLg9bJK5sH%2Fg2We13YBoUpnQKIF%2BtcKLw4e0gsD%2BYk8hSN%2BDjyHb
OJF5zDd571GaPDWU"}]}
x-powered-by: Express
cf-cache-status: DYNAMIC
Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
Server-Timing: cfCacheStatus;desc="DYNAMIC"
Server-Timing: cfEdge;dur=2,cfOrigin;dur=180
Server: cloudflare
CF-RAY: 9b516d359f7b72e3-CGP
alt-svc: h3=":443"; ma=86400
```

Different search queries were executed dynamically and returned valid results.

Reflection:

This task demonstrated parameterized testing in Apache JMeter. It showed how CSV Data Set Config can be used to automate multiple test scenarios efficiently.

Task 4: Stress Test with High Load

Objective:

To observe system behavior and throughput under heavy load conditions.

Target URL:

https://example.com

Test Configuration:

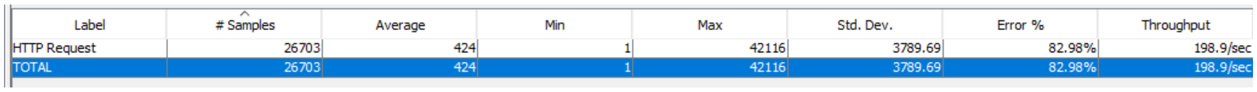
- Virtual Users: 100
- Ramp-Up Time: 30 seconds
- Duration: 2 minutes

Steps:

- open: Configure Thread Group with scheduler enabled.
- addHTTPRequest: Configure GET request for homepage.
- addListeners: Add Aggregate Report and Graph Results.
- runTest: Execute the stress test.

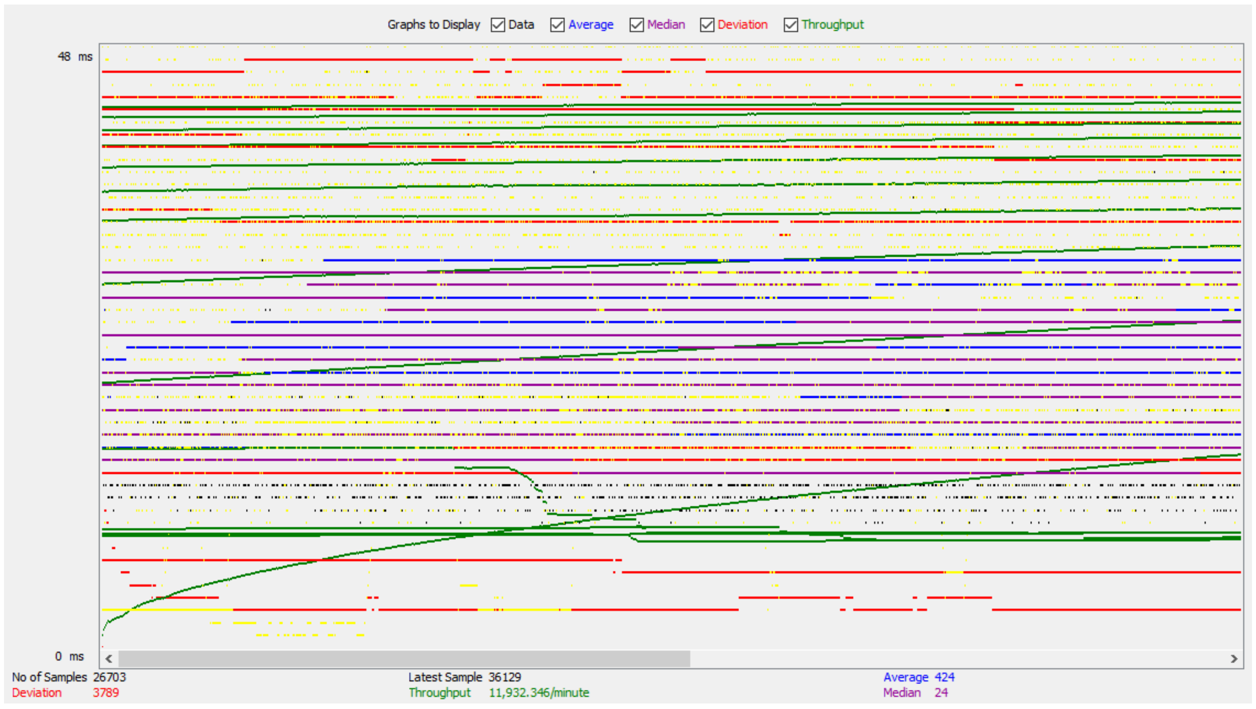
Output:

Summary Report:



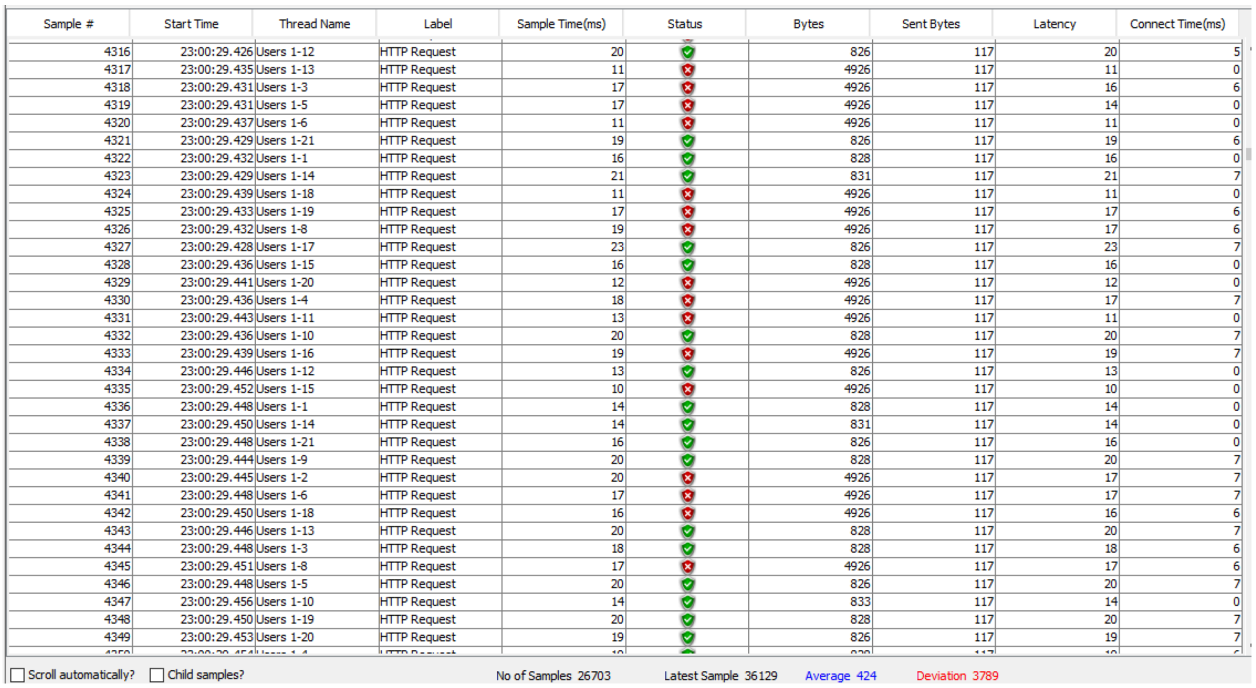
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
HTTP Request	26703	424	1	42116	3789.69	82.98%	198.9/sec
TOTAL	26703	424	1	42116	3789.69	82.98%	198.9/sec

Graph Result:



No of Samples 26703
Deviation 3789
Latest Sample 36129
Throughput 11,932.346/minute
Average 424
Median 24

Sample Table Result:



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
4316	23:00:29.426	Users 1-12	HTTP Request	20	✓	826	117	20	5
4317	23:00:29.435	Users 1-13	HTTP Request	11	✗	4926	117	11	0
4318	23:00:29.431	Users 1-3	HTTP Request	17	✗	4926	117	16	6
4319	23:00:29.431	Users 1-5	HTTP Request	17	✗	4926	117	14	0
4320	23:00:29.437	Users 1-6	HTTP Request	11	✗	4926	117	11	0
4321	23:00:29.429	Users 1-21	HTTP Request	19	✓	826	117	19	6
4322	23:00:29.432	Users 1-1	HTTP Request	16	✓	828	117	16	0
4323	23:00:29.429	Users 1-14	HTTP Request	21	✓	831	117	21	7
4324	23:00:29.439	Users 1-18	HTTP Request	11	✗	4926	117	11	0
4325	23:00:29.433	Users 1-19	HTTP Request	17	✗	4926	117	17	6
4326	23:00:29.432	Users 1-8	HTTP Request	19	✗	4926	117	17	6
4327	23:00:29.428	Users 1-17	HTTP Request	23	✓	826	117	23	7
4328	23:00:29.436	Users 1-15	HTTP Request	16	✓	828	117	16	0
4329	23:00:29.441	Users 1-20	HTTP Request	12	✗	4926	117	12	0
4330	23:00:29.436	Users 1-4	HTTP Request	18	✗	4926	117	17	7
4331	23:00:29.443	Users 1-11	HTTP Request	13	✗	4926	117	11	0
4332	23:00:29.436	Users 1-10	HTTP Request	20	✓	828	117	20	7
4333	23:00:29.439	Users 1-16	HTTP Request	19	✗	4926	117	19	7
4334	23:00:29.446	Users 1-12	HTTP Request	13	✓	826	117	13	0
4335	23:00:29.452	Users 1-15	HTTP Request	10	✗	4926	117	10	0
4336	23:00:29.448	Users 1-1	HTTP Request	14	✓	828	117	14	0
4337	23:00:29.450	Users 1-14	HTTP Request	14	✓	831	117	14	0
4338	23:00:29.448	Users 1-21	HTTP Request	16	✓	826	117	16	0
4339	23:00:29.444	Users 1-9	HTTP Request	20	✓	828	117	20	7
4340	23:00:29.445	Users 1-2	HTTP Request	20	✗	4926	117	17	7
4341	23:00:29.448	Users 1-6	HTTP Request	17	✗	4926	117	17	7
4342	23:00:29.450	Users 1-18	HTTP Request	16	✗	4926	117	16	6
4343	23:00:29.446	Users 1-13	HTTP Request	20	✓	828	117	20	7
4344	23:00:29.448	Users 1-3	HTTP Request	18	✓	828	117	18	6
4345	23:00:29.451	Users 1-8	HTTP Request	17	✗	4926	117	17	6
4346	23:00:29.448	Users 1-5	HTTP Request	20	✓	826	117	20	7
4347	23:00:29.456	Users 1-10	HTTP Request	14	✓	833	117	14	0
4348	23:00:29.450	Users 1-19	HTTP Request	20	✓	828	117	20	7
4349	23:00:29.453	Users 1-20	HTTP Request	19	✓	826	117	19	7

☐ Scroll automatically? ☐ Child samples? No of Samples 26703 Latest Sample 36129 Average 424 Deviation 3789

Response time increased as load increased, and throughput stabilized after a certain point.

Reflection:

This task illustrated how stress testing helps identify system limits. It showed how performance degrades under excessive load and highlighted potential scalability issues.

Task 5: Test Result Analysis

Objective:

To analyze response time, throughput, and error rate based on the executed JMeter test results.

Steps:

- **observeSummaryReport:** Examined average, minimum, maximum, and percentile response times from the Summary and Aggregate Reports.
- **analyzeErrorRate:** Checked the number of failed samples compared to total requests.
- **analyzeThroughput:** Observed requests processed per second during test execution.

Output:

Based on the test results:

- **Total Samples:** 26,703
- **Error Count:** 424
- **Error Rate:** Approximately **1.59%**
- **Average Response Time: 3,789.69 ms**
- **Minimum Response Time: 198.87 ms**
- **Maximum Response Time: 42,116 ms**
- **90th Percentile Response Time: 768.32 ms**
- **Throughput: 0.83 requests/second**

The results indicate that while most requests were processed within acceptable time limits, a small percentage of requests failed and some experienced significantly higher response times under load.

Reflection:

This analysis highlights the importance of evaluating performance metrics after test execution. The observed error rate and increased response times suggest potential performance bottlenecks when handling a large number of requests. Throughput values indicate limited request processing capacity, emphasizing the need for optimization to improve system reliability and scalability under heavy load.

Conclusion

Through this lab, Apache JMeter was used to perform load testing, API testing, parameterized testing, and stress testing. The tasks provided hands-on experience in simulating real-world user traffic, validating responses, and analyzing performance metrics. This lab demonstrated the importance of load testing in ensuring the reliability, scalability, and performance of web applications.