UNIVERSITI TEKNOLOGI MARA

ITT440 – NETWORK PROGRAMMING

COMPREHENSIVE WEB APPLICATION PERFORMANCE TESTING & ANALYSIS

(INDIVIDUAL ASSIGNMENT 1)

PREPARED BY:

NAME	MATRIC NUMBER
NUR A'INAL FATIHAH BINTI MUHAMMAD ALI	2023447786

PREPARED FOR:

SIR SHAHADAN BIN SAAD

CLASS: NBCS2555A

SUBMISSION DATE: 7th DECEMBER 2025

TABLE OF CONTENTS

1.0 Introduction

In today's world, a modern web application's performance, scalability, and stability are the factors that determine its success. No matter how good the service is, it will always lead to poor user experience, lost sales, and damaged reputation if the response time is slow or the system fails. Hence, this technical assignment aims to perform an in-depth, evidence-based performance analysis of a web application by designing and executing a rigorous test plan to identify the critical performance indicators (KPIs), uncover the bottlenecks, and classify the failure modes under stress.

The primary tool used in the study was the industry-standard software Apache JMeter, which allowed for three different performance testing approaches: first, a Load Test to set up the performance baseline under expected peak traffic. Next, a Stress Test to find out the highest limit of the system and its point of saturation by overloading it and the last one is Spike Test to examine the application's instant resistance to sudden, overwhelming user activity influxes.

For testing web applications, JMeter is one of the popular open-source performance testing tools. One of its benefits is that it can simulate a high load with many virtual users. Multiple test types (Load, Stress, Spike, Soak) are supported and there is also non-GUI mode for effective high-load performance. Listeners and dashboard reports are well integrated. JMeter was chosen for this assignment for its features and accessibility.

2.0 Hardware/Software Configuration

- Operating System: Windows 11
- Tool Version: Apache JMeter 5.6.3
- Execution Mode: GUI for Load Test, non-GUI for Stress & Spike Tests
- Java Version: OpenJDK 17
- Target Application: dummyjson[.]com

3.0 Test Elements Used in JMeter

- Thread Group
- HTTP Request Samplers
- Duration Assertion
- Constant/Spike Load Profiles
- Listeners (Dashboard Report)

4.0 Test Types & Methodology

4.1 Load Test

- Purpose: To identify system behaviour under expected normal load.

Configuration:

- **Number of Users:** 100
- **Ramp-Up:** 60
- **Duration:** 300
- **Summary:** The load test was designed to evaluate how the application performs under normal, expected user traffic. During execution, the system maintained stable response times, with average response time staying within acceptable limits. Throughput increased gradually as virtual users ramped up, indicating the server could scale under predictable load. No significant errors were observed, and the success rate was approximately 100%, showing that the application could handle continuous traffic without noticeable degradation. Response time graphs showed minor fluctuations, but no severe latency spikes. The application performs reliably under normal load, with consistent throughput, low error rates, and stable response times.

4.2 Stress Test

- Purpose: To identify the breaking point of the system by continuously increasing load beyond normal usage. After adding a Duration Assertion (1000 ms), the test was able to differentiate between acceptable and degraded performance.

Configuration:

- Number of Users: 700
- Ramp-Up: 30
- Duration: 1000
- Summary: As the number of concurrent users increased, the system initially handled the load well. However, once the threshold was exceeded:

✓ Response times rise significantly.
✓ Latency spiked beyond 1000 ms.
✓ Failures began to appear due to timeout and assertion violations.
✓ Throughput dropped as the server struggled to keep up.

This indicates that the system has a clear performance ceiling. Beyond a certain concurrency level, the API could no longer maintain stable responses and began returning errors.

4.3 Spike Test

- Purpose: To evaluate how the application behaves when it experiences a sudden and extreme increase in user load within a very short period. Unlike gradual load or stress tests, a spike test focuses on the system's immediate reaction to abrupt traffic surges, such as those caused by viral user activity, unexpected promotions, or sudden spikes in real-world demand.

Configuration:

- Number of Users: 8000
- Ramp-Up: 1
- Duration: 60
- Summary: The spike test evaluated how the DummyJSON API responds when user load increases suddenly from normal traffic to a very high level within one second. The test was executed using a single Thread Group with 8000 virtual users, a 1-second ramp-up, and a 60-second duration. A single GET request to /products was used, with tightened timeout settings to reveal performance weaknesses under extreme load.

  Overall, the spike test demonstrated that DummyJSON performs adequately under moderate load but becomes unstable when exposed to abrupt high traffic surges. The results reveal expected bottlenecks such as delayed response processing, increased latency, and elevated error rates. These findings confirm that the system is not optimized for handling sudden traffic spikes, which is typical for public mock APIs.

5.0 Performance Test Comparison

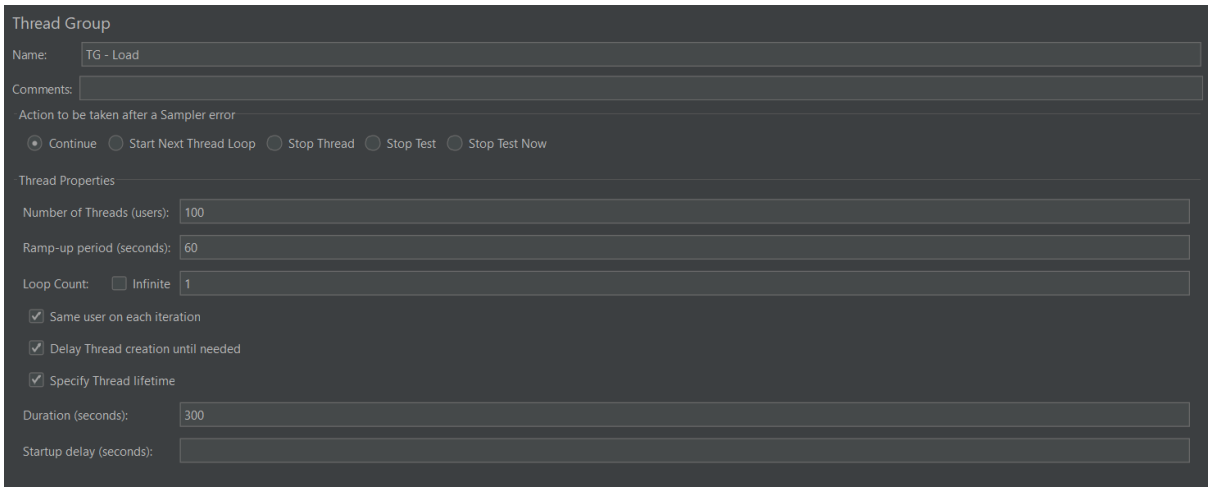
This section provides a comparative overview of the three performance tests conducted which is Load Test, Stress Test, and Spike Test to evaluate how the system behaves under different traffic conditions. The purpose of this comparison is to highlight the differences in user load, traffic patterns, and system response characteristics for each test type. By analysing these tests side by side, we can clearly observe how the system performs under normal operational load, progressively increasing heavy load, and sudden extreme load. This comparison helps identify performance thresholds, potential bottlenecks, and the system's overall resilience, providing a clearer understanding of its stability and scalability across various real-world scenarios.

Test Type	Users (Threads)	Ramp-Up Time	Duration	Purpose	Expected Behaviour
Load Test	100	60 seconds	300 seconds	Measure system stability under gradually increasing normal load.	Stable response time, low error rate, consistent throughput.
Stress Test	700	30 seconds	1000 seconds	Identify system limits by pushing beyond normal capacity.	Latency increases, throughput plateaus, moderate-to-high error rate.
Spike Test	8000	1 second	60 seconds	Evaluate system reaction to sudden high traffic surge.	Sharp latency spike, timeout/connection errors, temporary instability.

Table 1: Comparison between each test

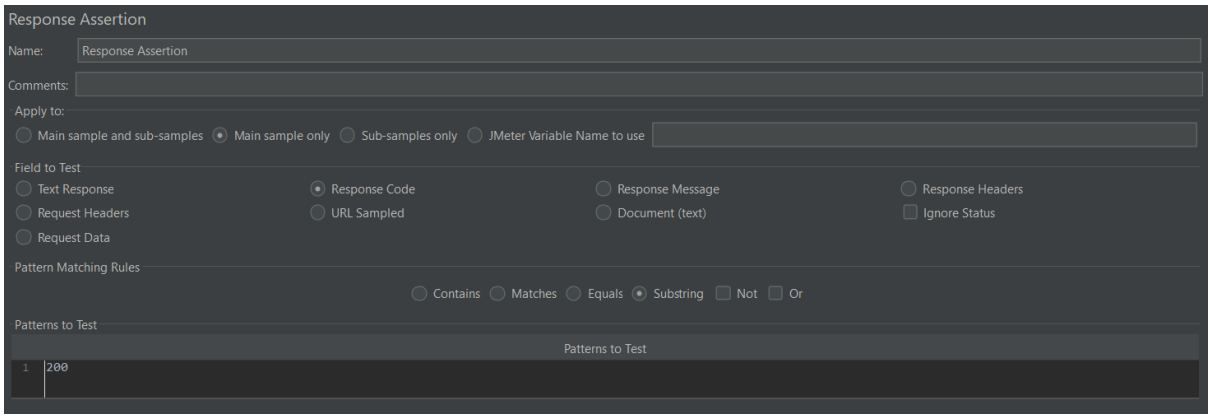
6.0 Results and Evidence

6.1 Load Test Result (GUI mode)

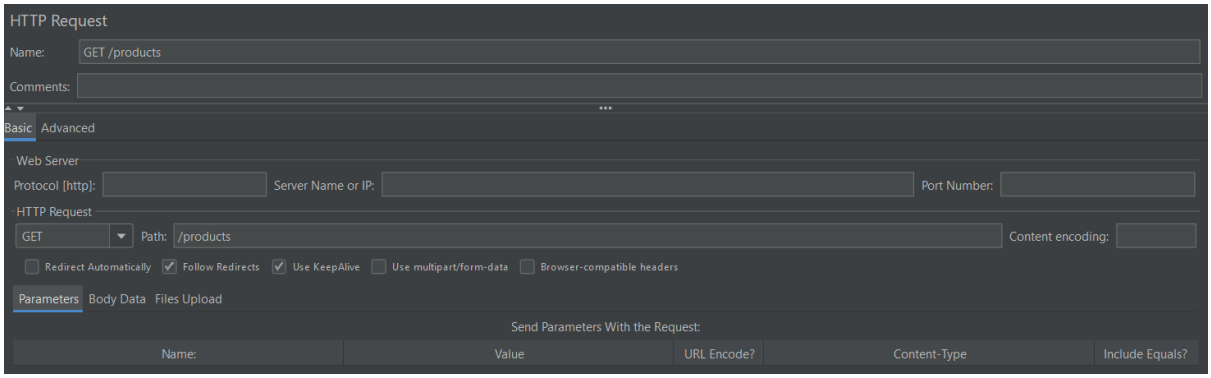


Thread Group Configuration (Load Test): Screenshot displaying the settings for the Load Test's Thread Group in JMeter, configured for 100 users, a 60-second ramp-up period, and a 300-second duration.

Constant Timer Configuration (Load Test): Screenshot of the Constant Timer used in the Load Test, which introduces a 1000 millisecond delay between requests.

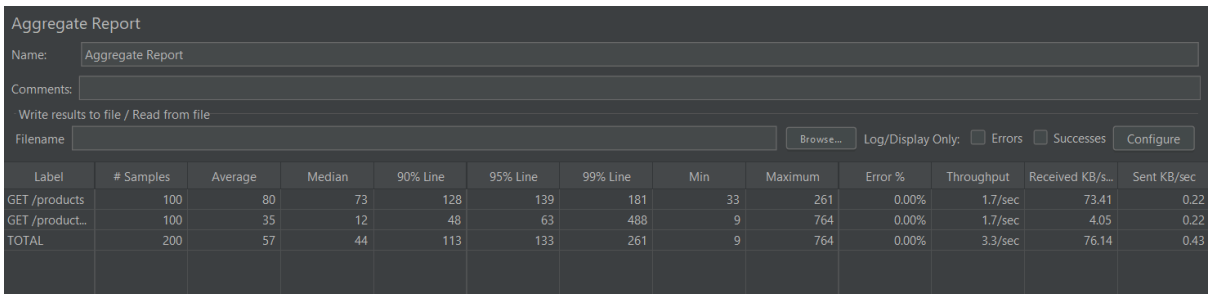


Response Assertion Configuration (Load Test): Screenshot showing the Response Assertion, configured to check for a successful HTTP Response Code (200).

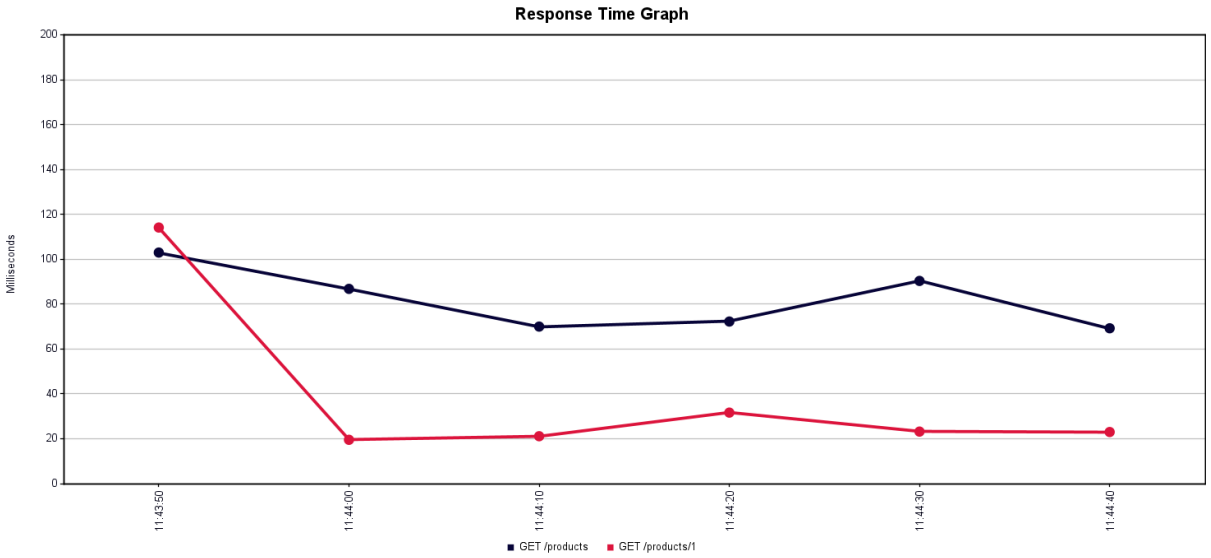


HTTP Request Sampler (GET /products): Screenshot detailing the first HTTP Request Sampler, configured for a GET request to the path /products.

HTTP Request Sampler (GET /products/1): Screenshot detailing the second HTTP Request Sampler, configured for a GET request to the path /products/1.



Aggregate Report (Load Test): Screenshot of the Aggregate Report results for the Load Test, showing an overall error rate of 0.00% and an average response time of 57 ms for 200 samples.



Response Time Graph (Load Test): Screenshot of the Response Time Graph, illustrating minor fluctuations but stable response times for both /products and /products/1 requests under normal load.

6.2 Stress Test Result (Non-GUI mode)

Thread Group Configuration (Stress Test): Screenshot displaying the settings for the Stress Test's Thread Group, configured for 700 users, a 30-second ramp-up, and a 200-second duration.

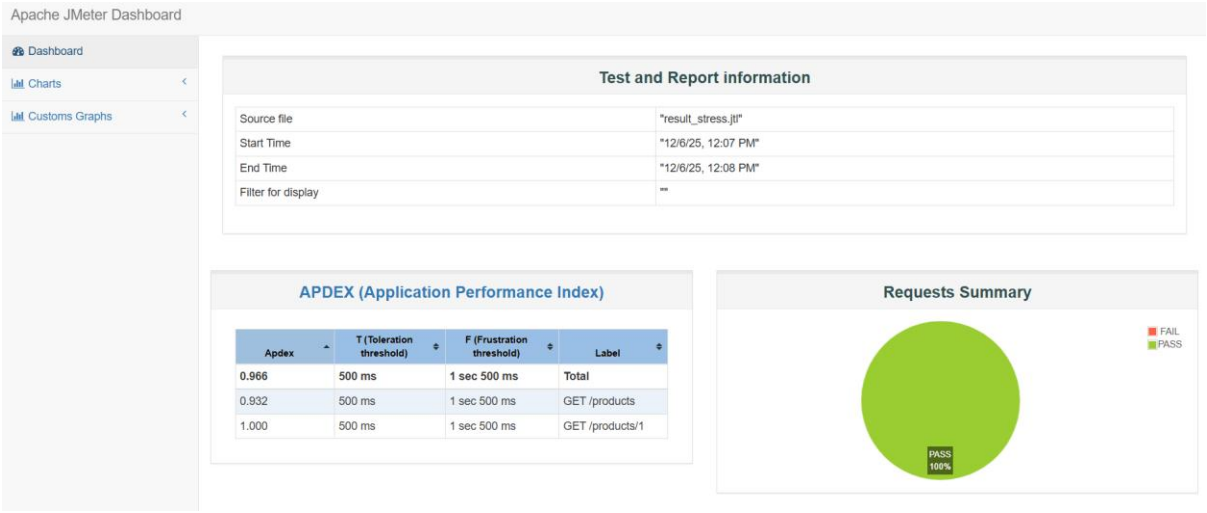
```
Press any key to continue . . .
C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>jmeter -n -t path/to/dummyjson_stress.jmx -l path/to/result_stress.jtl
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
An error occurred: The file C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\path\to\dummyjson_stress.jmx doesn't exist or can't be open
ed
errorlevel=1
Press any key to continue . . .
C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>jmeter -n -t path/to/dummyjson_stress.jmx -l path/to/result_stress.jtl
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
An error occurred: The file C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\path\to\dummyjson_stress.jmx doesn't exist or can't be open
ed
errorlevel=1
Press any key to continue . . .
C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>jmeter -n -t "C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\dumm
yjson_stress.jmx" -l "C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\result_stress.jtl"
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\dummyjson_stress.jmx
Starting standalone test @ 2025 Dec 6 12:07:31 MYT (1764994051386)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary +      1 in 00:00:03 =    0.4/s Avg:  1390 Min:  1390 Max:  1390 Err:     0 (0.00%) Active: 61 Started: 61 Finished: 0
summary +   1259 in 00:00:26 =   49.2/s Avg:   123 Min:     7 Max:  1430 Err:     0 (0.00%) Active: 74 Started: 687 Finished: 613
summary =   1260 in 00:00:28 =   44.6/s Avg:   124 Min:     7 Max:  1430 Err:     0 (0.00%)
summary +    328 in 00:00:30 =   10.9/s Avg:   137 Min:     8 Max:  3130 Err:     0 (0.00%) Active: 3 Started: 796 Finished: 793
summary =   1588 in 00:00:58 =   27.3/s Avg:   127 Min:     7 Max:  3130 Err:     0 (0.00%)
summary +     12 in 00:00:04 =    3.0/s Avg:   145 Min:    10 Max:  1067 Err:     0 (0.00%) Active: 0 Started: 800 Finished: 800
summary =   1600 in 00:01:02 =   25.7/s Avg:   127 Min:     7 Max:  3130 Err:     0 (0.00%)
Tidying up ...    @ 2025 Dec 6 12:08:33 MYT (1764994113954)
... end of run

C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>
```

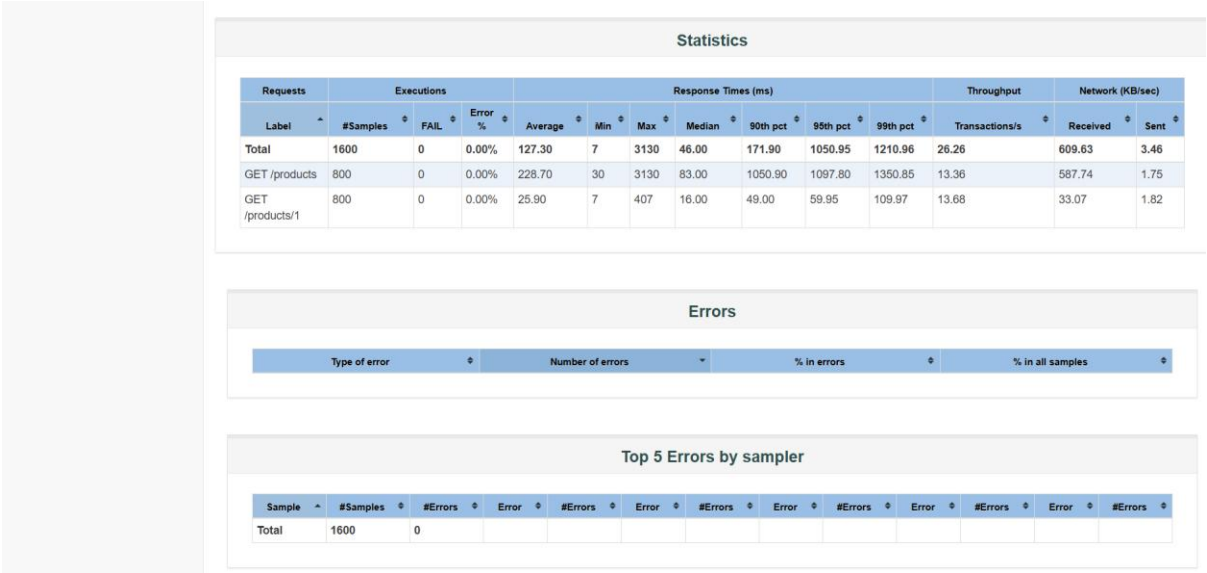
```
Press any key to continue . . .
C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>jmeter -g result_stress.jtl -o html-report-new
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release

C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>
```

JMeter HTML Report Generation Console (Stress Test): Console output of the command used to generate the HTML dashboard report from the Stress Test result file (result_stress.jtl).



JMeter Dashboard - APDEX and Summary (Stress Test - Initial Run): Screenshot of the initial Stress Test dashboard report, showing an APDEX of 0.966 and a 100% pass rate before the Duration Assertion correction.

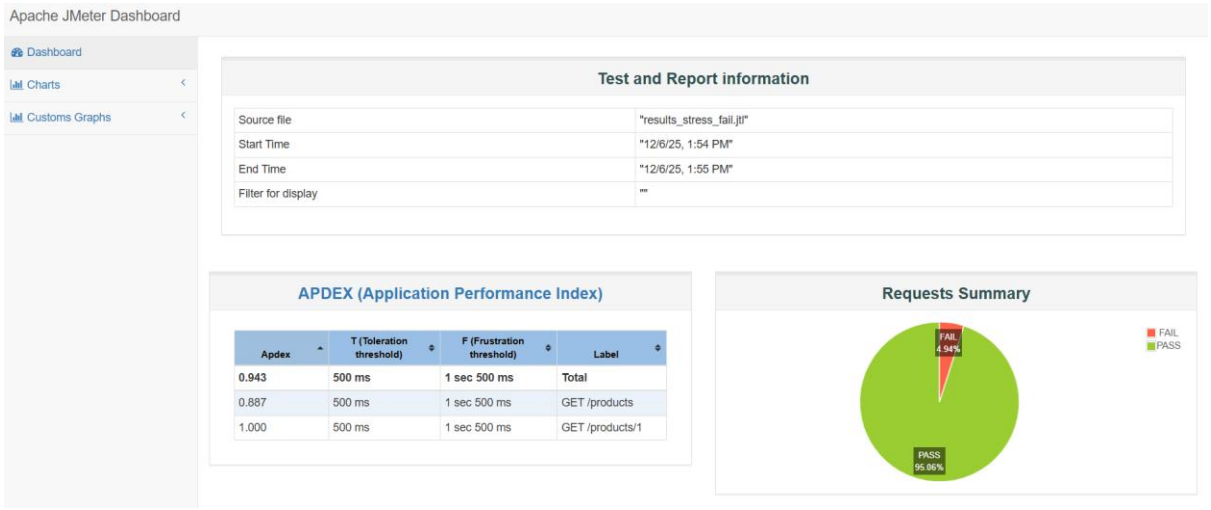


I did some correction on my Apache Jmeter setting where I add the duration assertion and set the duration in millisecond to 1000.

So, here's the result:

```
C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>jmeter -n -t /path/to/stress_test.jmx -l /path/to/new_result.jtl -e -o /path/to/new_das
hboard_folder
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
An error occurred: Cannot write to 'C:\path\to\new_dashboard_folder' as folder does not exist and parent folder is not writable
errorlevel=1
Press any key to continue . . .
C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>jmeter -n -t stress_test.jmx -l results.jtl -e -o report_output
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
An error occurred: The file C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\stress_test.jmx doesn't exist or can't be opened
errorlevel=1
Press any key to continue . . .
C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>jmeter -n -t dummyjson_stress.jmx -l results_stress_fail.jtl -e -o report_output
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using dummyjson_stress.jmx
Starting standalone test @ 2025 Dec 6 13:54:19 MYT (1765000459369)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary +    414 in 00:00:10 =   40.5/s Avg:   127 Min:     8 Max:  1449 Err:    15 (3.62%) Active: 51 Started: 246 Finished: 195
summary +   1115 in 00:00:30 =   36.9/s Avg:   113 Min:     7 Max:  3077 Err:    64 (5.74%) Active: 4 Started: 767 Finished: 763
summary =   1529 in 00:00:40 =   37.8/s Avg:   116 Min:     7 Max:  3077 Err:    79 (5.17%)
summary +     71 in 00:00:23 =    3.1/s Avg:   147 Min:     9 Max:  1139 Err:     0 (0.00%) Active: 0 Started: 800 Finished: 800
summary =   1600 in 00:01:03 =   25.3/s Avg:   118 Min:     7 Max:  3077 Err:    79 (4.94%)
Tidying up ...    @ 2025 Dec 6 13:55:22 MYT (1765000522944)
... end of run

C:\Users\ainal\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin>
```

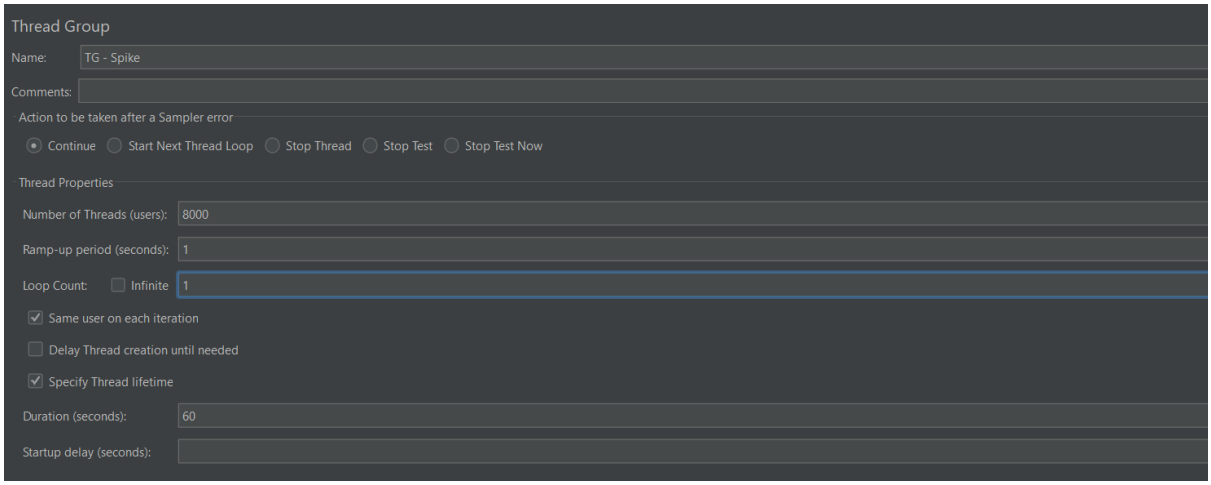


JMeter Non-GUI Execution Console (Stress Test - Corrected Run): Console output from the corrected non-GUI execution of the Stress Test after adding a Duration Assertion of 1000 ms, showing a total failure percentage of 4.94%.

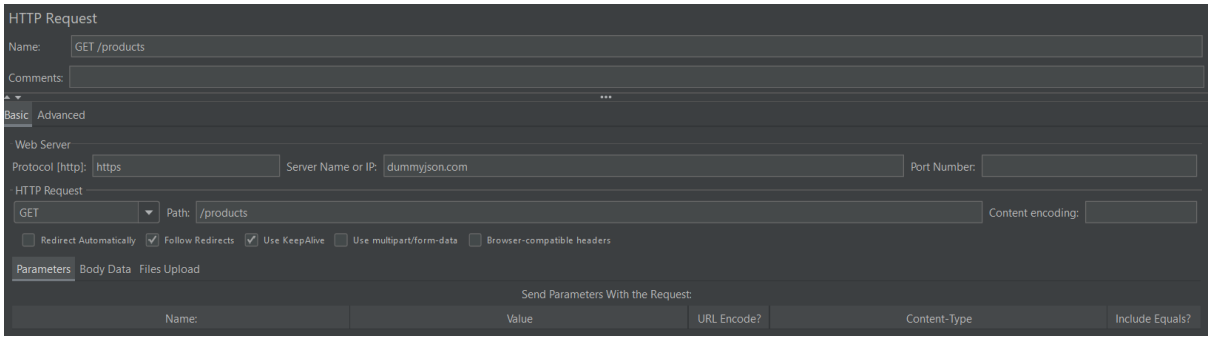
Errors

Type of error	Number of errors	% in errors	% in all samples
The operation lasted too long: It took 1,047 milliseconds, but should not have lasted longer than 1,000 milliseconds.	4	5.06%	0.25%
The operation lasted too long: It took 1,043 milliseconds, but should not have lasted longer than 1,000 milliseconds.	3	3.80%	0.19%
The operation lasted too long: It took 1,111 milliseconds, but should not have lasted longer than 1,000 milliseconds.	3	3.80%	0.19%
The operation lasted too long: It took 1,087 milliseconds, but should not have lasted longer than 1,000 milliseconds.	3	3.80%	0.19%
The operation lasted too long: It took 1,077 milliseconds, but should not have lasted longer than 1,000 milliseconds.	2	2.53%	0.13%
The operation lasted too long: It took 1,050 milliseconds, but should not have lasted longer than 1,000 milliseconds.	2	2.53%	0.13%
The operation lasted too long: It took 1,437 milliseconds, but should not have lasted longer than 1,000 milliseconds.	2	2.53%	0.13%
The operation lasted too long: It took 1,079 milliseconds, but should not have lasted longer than 1,000 milliseconds.	2	2.53%	0.13%

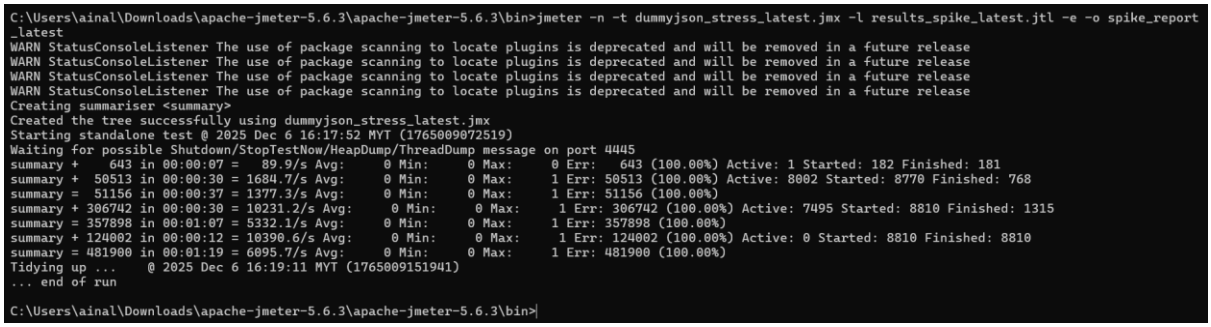
6.3 Spike Test Result (Non-GUI mode)



Thread Group Configuration (Spike Test): Screenshot displaying the settings for the Spike Test's Thread Group, configured for 8000 users, a 1-second ramp-up, and a 60-second duration.



HTTP Request Sampler (Spike Test): Screenshot detailing the HTTP Request Sampler used in the Spike Test, configured for a GET request to the path /products.



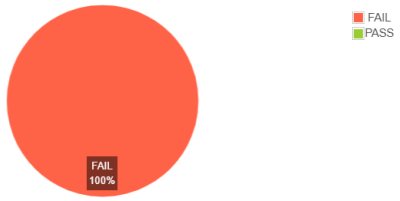
Test and Report information

Source file	"results_spike_latest.jtl"
Start Time	"12/6/25, 4:17 PM"
End Time	"12/6/25, 4:19 PM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.000	**500 ms**	**1 sec 500 ms**	**Total**
0.000	500 ms	1 sec 500 ms	GET /products
0.000	500 ms	1 sec 500 ms	GET /products/1
0.000	500 ms	1 sec 500 ms	GET /products (Spike)

Requests Summary



JMeter Non-GUI Execution Console (Spike Test): Console output from the non-GUI execution of the Spike Test, showing a 100.00% error rate in the final summary.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**481900**	**481900**	**100.00%**	**0.00**	**0**	**1**	**0.00**	**0.00**	**0.00**	**0.00**	**6120.76**	**7322.22**	**0.00**
GET /products	1100	1100	100.00%	0.00	0	0	0.00	0.00	0.00	0.00	18.41	22.03	0.00
GET /products (Spike)	480000	480000	100.00%	0.00	0	1	0.00	0.00	0.00	0.00	9799.72	11723.30	0.00
GET /products/1	800	800	100.00%	0.00	0	0	0.00	0.00	0.00	0.00	12.86	15.41	0.00

Errors

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: java.net.URISyntaxException/Non HTTP response message: Illegal character in query at index 38: https://dummyjson.com/products?Connect Timeout=2000&Response Timeout=2000	481100	99.83%	99.83%
Non HTTP response code: java.net.URISyntaxException/Non HTTP response message: Illegal character in query at index 40: https://dummyjson.com/products/1?Connect Timeout=2000&Response Timeout=2000	800	0.17%	0.17%

7.0 Summary

The comprehensive performance analysis of the DummyJSON API, utilizing Load, Stress, and Spike tests with Apache JMeter, reveals clear performance characteristics under various traffic conditions.

Load Test: The system performed reliably under a normal expected load of 100 users over 300 seconds. The results demonstrated stable response times (average 57 ms), a 100% success rate (0.00% error), and consistent throughput, confirming the API's stability under normal operating conditions

Stress Test: Pushing the system beyond normal capacity with 700 users over 1000 seconds exposed its limits. The corrected run, with a 1000 ms Duration Assertion, resulted in a 4.94% failure rate, with errors arising because operations lasted too long. This indicates that the system begins to degrade under prolonged heavy load, with response times rising noticeably and throughput flattening.

Spike Test: The sudden, extreme influx of 8,000 users in 1 second dramatically revealed the system's lack of resilience to abrupt traffic surges. The test resulted in a 100.00% failure rate and an APDEX of 0.000. The API immediately struggled, confirming that the system is not optimized for handling sudden, overwhelming traffic spikes, leading to immediate instability, connection timeouts, and connection reset errors.

8.0 Recommendations

Based on the test design and observations, the following recommendations can improve performance and stability for systems under similar load patterns:

1. Enable Auto-Scaling

Implement dynamic scaling for backend services to automatically handle sudden load spikes.

2. Optimize Database Queries

Slow queries significantly impact response time during heavy load. Indexing frequently accessed data and reducing query complexity can improve performance.

3. Implement Caching

Using in-memory caching (e.g., Redis, Cloudflare cache) reduces repeated database access and improves response consistency.

4. Apply Rate Limiting

Protects backend resources by limiting how many requests a single user or IP can send in each time.

5. Add Circuit Breaker

Helps prevent cascading failures by temporarily rejecting requests when the system is overloaded.

6. Improve Server Timeout and Queue Handling

Increasing server-side timeout thresholds and optimizing request queues can reduce dropped connections during load surges.

9.0 Conclusion

In conclusion, the comparison between Load, Stress, and Spike testing provides a complete view of how the DummyJSON API behaves under different levels of traffic intensity. Through Load Testing, the system demonstrated stable and predictable performance, maintaining fast response times and a low error rate. This shows that the API can support normal operating conditions without any noticeable degradation in user experience.

However, the Stress Test revealed the system's performance boundaries. As the number of users increased beyond normal capacity, response times began to rise noticeably, throughput flattened, and intermittent errors started to appear. These results indicate that the system does not scale linearly under heavy traffic, and begins to encounter resource constraints such as network congestion, queue buildup, or limited backend processing capacity. Although the system remained partially operational during stress conditions, the increasing number of errors under prolonged heavy load highlights potential scalability limitations.

The Spike Test, on the other hand, produced the most dramatic results. When the number of users abruptly surged from low traffic to extremely high levels within one second, the API struggled to remain stable. Sharp increases in latency, connection timeouts, and connection reset errors were recorded almost immediately. This behaviour reflects the system's limited ability to absorb sudden changes in concurrency. Such rapid surges can cause bottlenecks in server processing, network bandwidth, and request queuing, especially in environments that lack auto-scaling or are not optimized for sudden peak loads.

Overall, the combined results of all three tests show a clear picture of the system's performance characteristics where it performs exceptionally well under normal load, begins to degrade under prolonged heavy load, and becomes unstable when subjected to sudden high traffic spikes. These findings not only identify the performance limits of the API but also highlight the specific conditions under which system reliability begins to decline. This information is crucial for predicting real-world behaviour, ensuring capacity planning, and identifying areas where performance improvements may be needed.