PERFORMANCE TEST REPORT

Tested by	Jefri Fahrian
Date	04-06-2025 10:41
Site	https:// thinking-tester-contact-list.herokuapp.com
Endpoints target	1. Sign Up [POST] /users 2. Sign In [POST] /users/login 3. Add Contact [POST] /contacts 4. Get Contact [GET] /contacts/id 5. Update Contact [PUT] /contacts/id 6. Update Partial Contact [PATCH] /contacts/id 7. Delete Contact [DELETE] /contacts/id
Tools	Apache JMeter 5.6.3
Device	Processor 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz RAM 16.0 GB
OS	Windows 11 Home

Objective

1. To detect issues like memory leaks, resource exhaustion, and performance degradation over time.
2. Ensure application can sustain load for many period of time without failure (long-term stability)

Key Metric Captured

1. **Response time:** refers to the total time it takes to receive a complete response from the server after a request is sent.

Metric	Meaning
Average Response Time	Mean response time across all requests
Min Response Time	Fastest single request
Max Response Time	Slowest single request
Percentiles (P90/P95)	% of requests that completed within a certain time
Response Time Over Time	How response time changed during the test

2. **Throughput:** measures the number of requests (or transactions) the system processes per unit of time, typically: **Requests per second** (req/sec) and **Transactions per second** (TPS)

Test Scenario

An **endurance test** is applied to checks how the system behaves under constant load over an extended period of time.

Preconditions: User has logged in

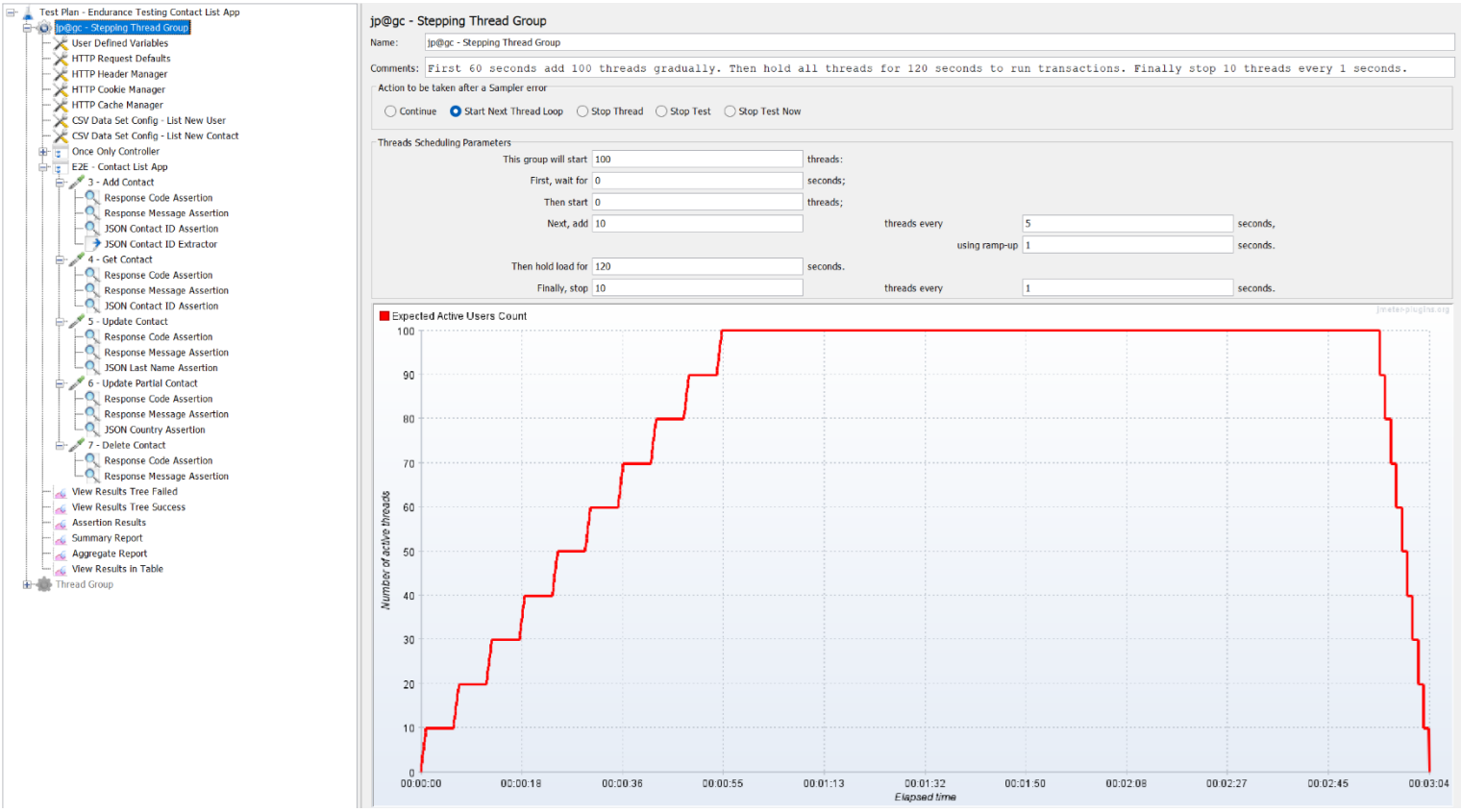
Steps:

1. First 60 seconds add 100 threads (virtual users) gradually (hit endpoint Sign up and Sign in).
2. Then hold all threads for 120 seconds to run transactions (hit endpoint Add, Get, Update, and Delete Contact).
3. Finally stop 10 threads every 1 seconds.

Assertion:

1. Check response code
2. Check response message
3. Check JSON path and value

Jmeter configuration:



Test Report Summary

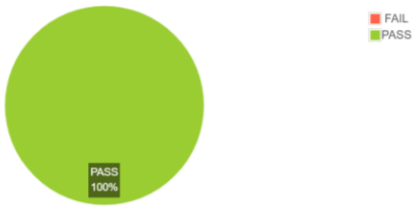
Test and Report information

Source file	"endurance-test-summary-report.jtl"
Start Time	"6/4/25, 10:41 AM"
End Time	"6/4/25, 10:42 AM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.417	**500 ms**	**1 sec 500 ms**	**Total**
0.201	500 ms	1 sec 500 ms	1 - Sign Up
0.641	500 ms	1 sec 500 ms	2 - Sign In
0.217	500 ms	1 sec 500 ms	3 - Add Contact
0.623	500 ms	1 sec 500 ms	4 - Get Contact
0.594	500 ms	1 sec 500 ms	5 - Update Contact
0.478	500 ms	1 sec 500 ms	6 - Update Partial Contact
0.591	500 ms	1 sec 500 ms	7 - Delete Contact
0.000	500 ms	1 sec 500 ms	E2E - Contact List App

Requests Summary

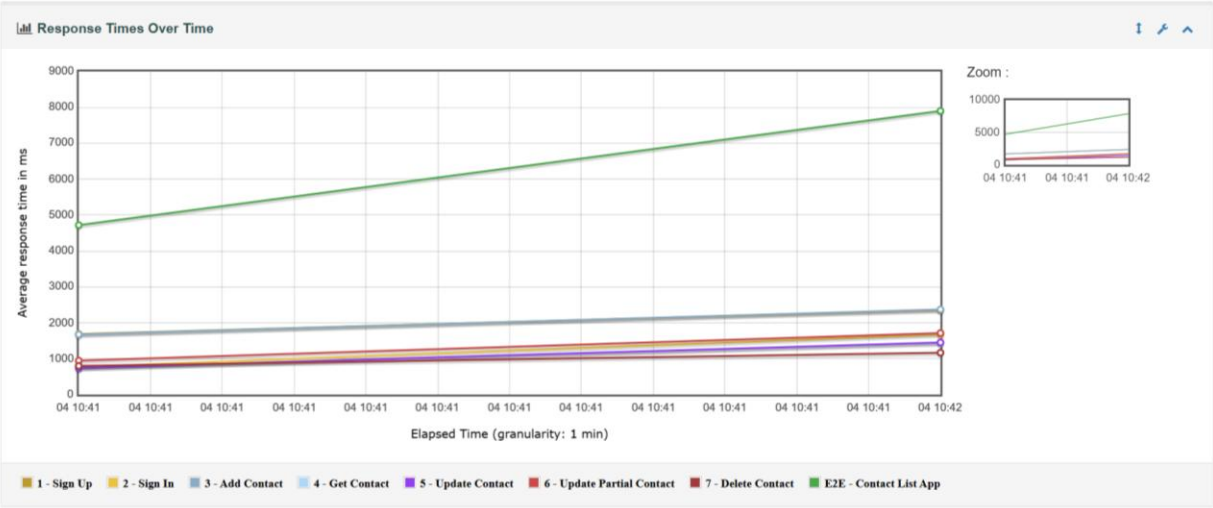


Statistics

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	**2684**	**0**	**0.00%**	**1117.70**	**299**	**3969**	**1003.50**	**2009.00**	**2251.75**	**3000.45**	**44.35**	**44.74**	**25.17**
1 - Sign Up	92	0	0.00%	1775.50	1159	3045	1631.00	2516.50	2686.85	3045.00	1.63	2.03	0.58
2 - Sign In	92	0	0.00%	888.43	322	2060	976.00	1974.90	1996.75	2060.00	1.63	2.03	0.77
3 - Add Contact	500	0	0.00%	1764.81	314	3073	1765.00	2934.00	2985.85	3041.99	8.78	9.34	6.37
4 - Get Contact	500	0	0.00%	855.29	299	2052	955.00	1775.30	1984.80	2027.99	8.69	9.19	3.50
5 - Update Contact	500	0	0.00%	875.11	299	2960	927.50	1869.40	1958.80	2042.98	8.69	9.11	6.87
6 - Update Partial Contact	500	0	0.00%	1123.78	311	3969	999.00	1989.00	2070.60	3022.99	8.69	9.11	4.35
7 - Delete Contact	500	0	0.00%	890.67	302	2076	972.00	1742.80	1993.95	2047.98	8.70	6.43	4.13
E2E - Contact List App	500	0	0.00%	5509.66	1582	11041	5728.50	8848.40	9037.40	10866.73	8.50	42.14	24.60

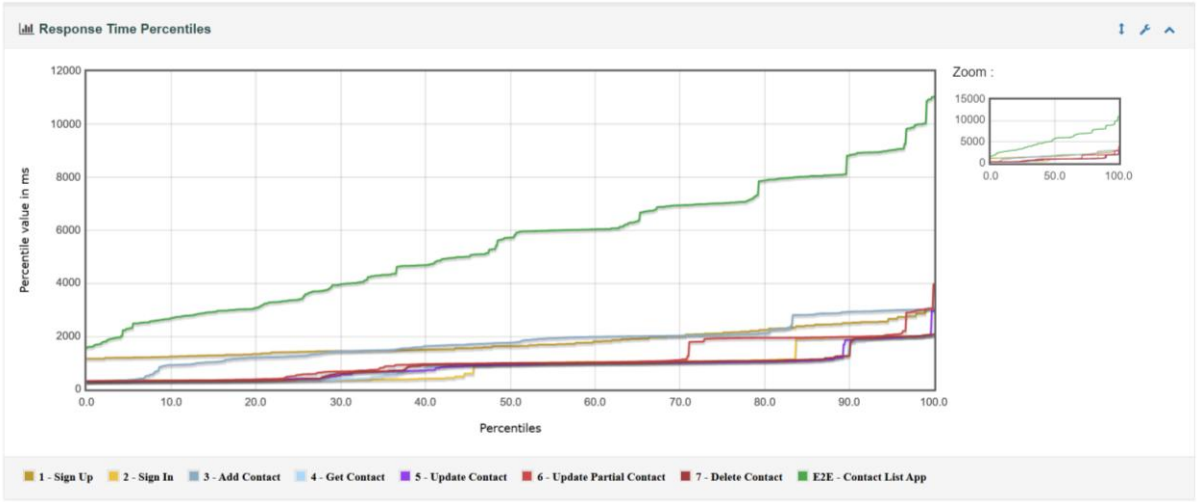
1. Response Time Over Time

Response Time Over Time chart visualize **how the response time of the requests changed throughout the test**.



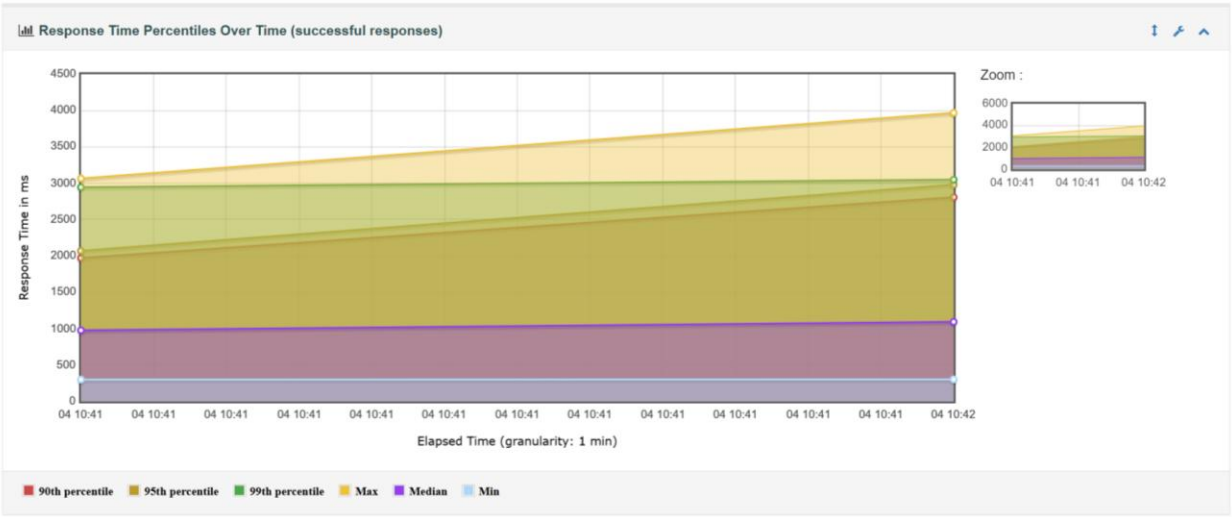
2. Response Time Percentiles

Response Time Percentiles chart gives a detailed look at how the application's **response times are distributed**, focusing on **what percentage of requests fell below specific thresholds**.



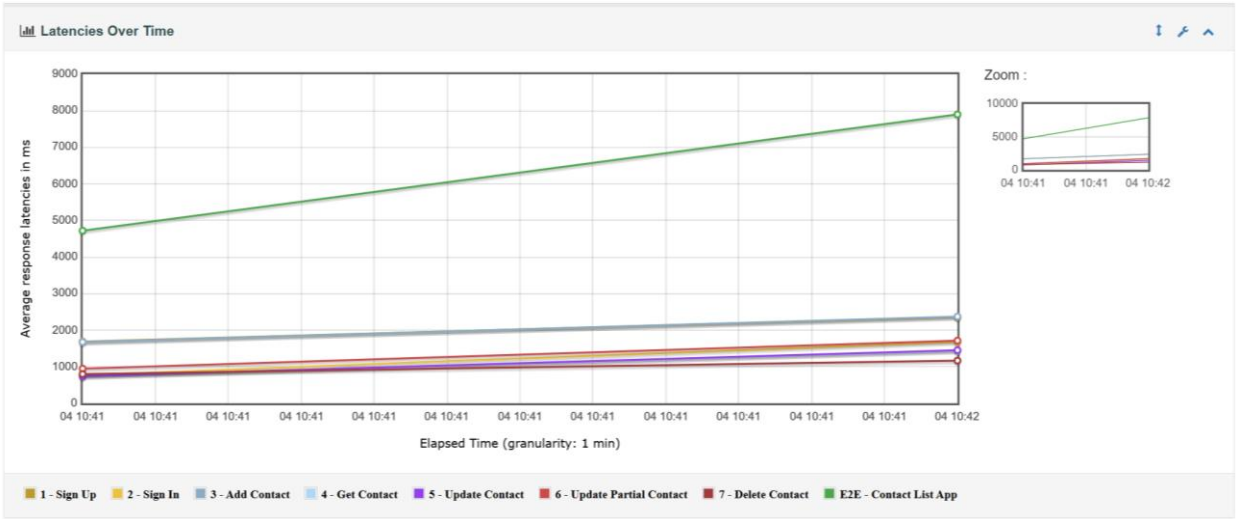
3. Response Time Percentiles Over Time

Response Time Percentiles Over Time chart tracks how the **response time percentiles** evolve **throughout the test run** — helping identify **performance degradation, instability, or spikes**.



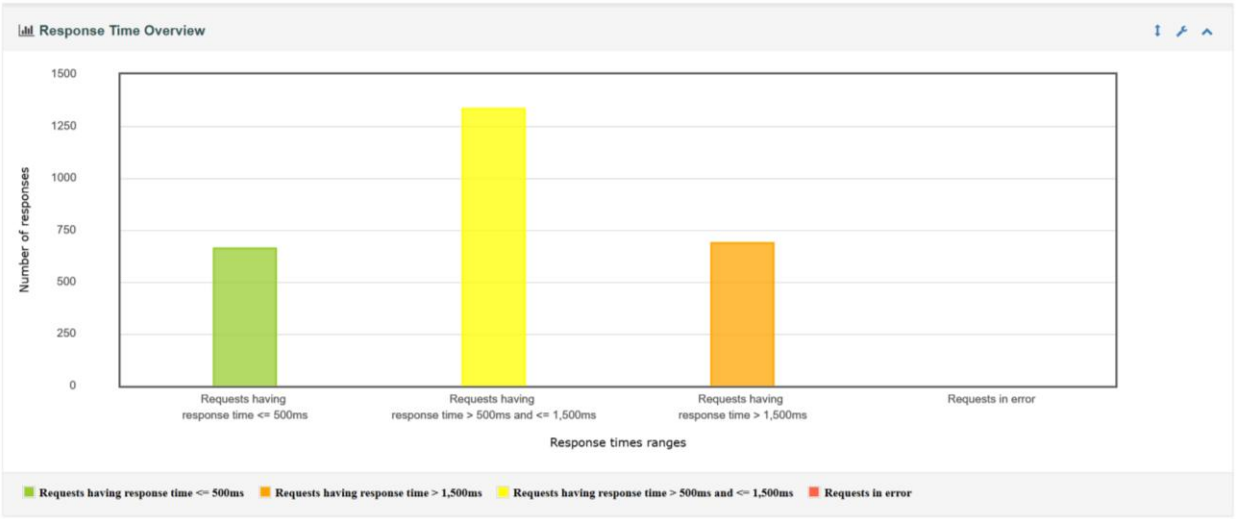
4. Latency Over Time

Time taken to **start receiving** the response, excluding download time. Useful for identifying **network lag** or initial connection delays.



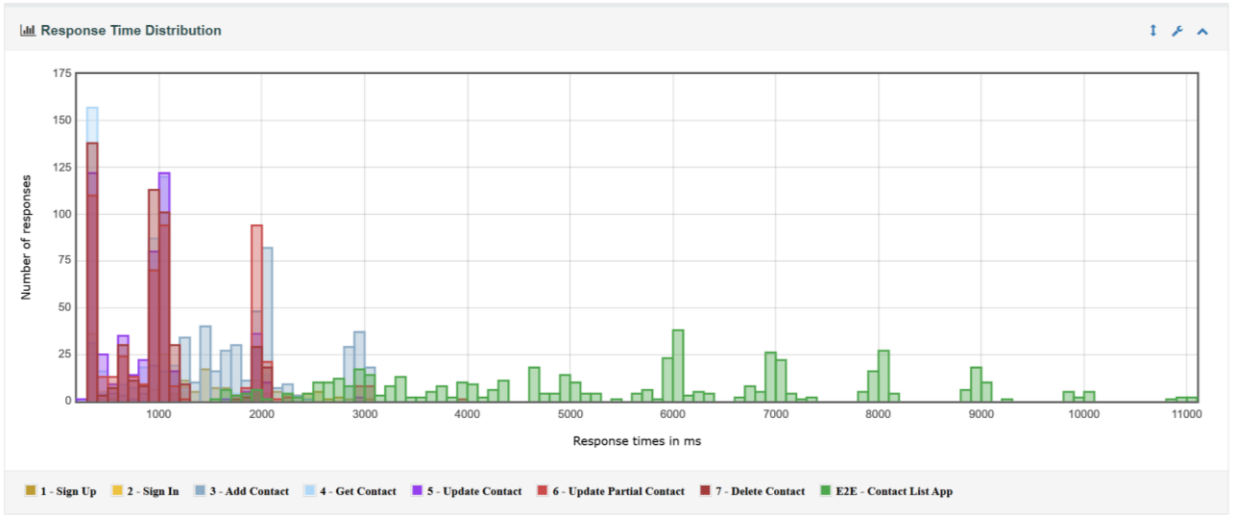
5. Response Time Overview

Response Time Overview chart provides a **high-level summary** of how long it took for the server to **respond to requests**, usually shown **per sampler** (i.e., per endpoint or request type).



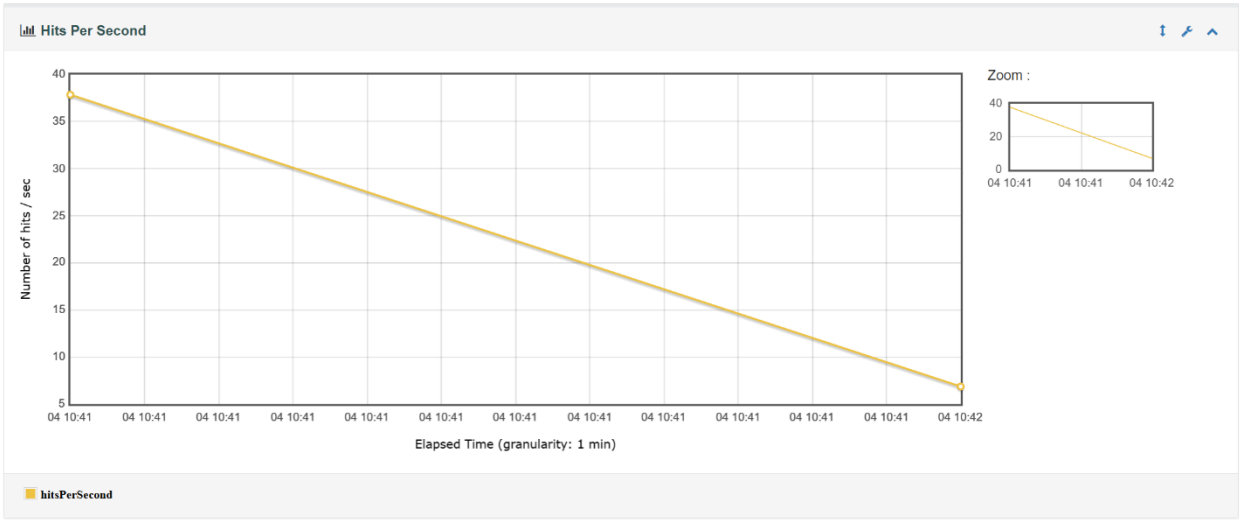
6. Response Time Distribution

Response Time Distribution chart breaks down **how long responses took** — grouped into buckets — so we can see the **distribution of performance across all requests**.



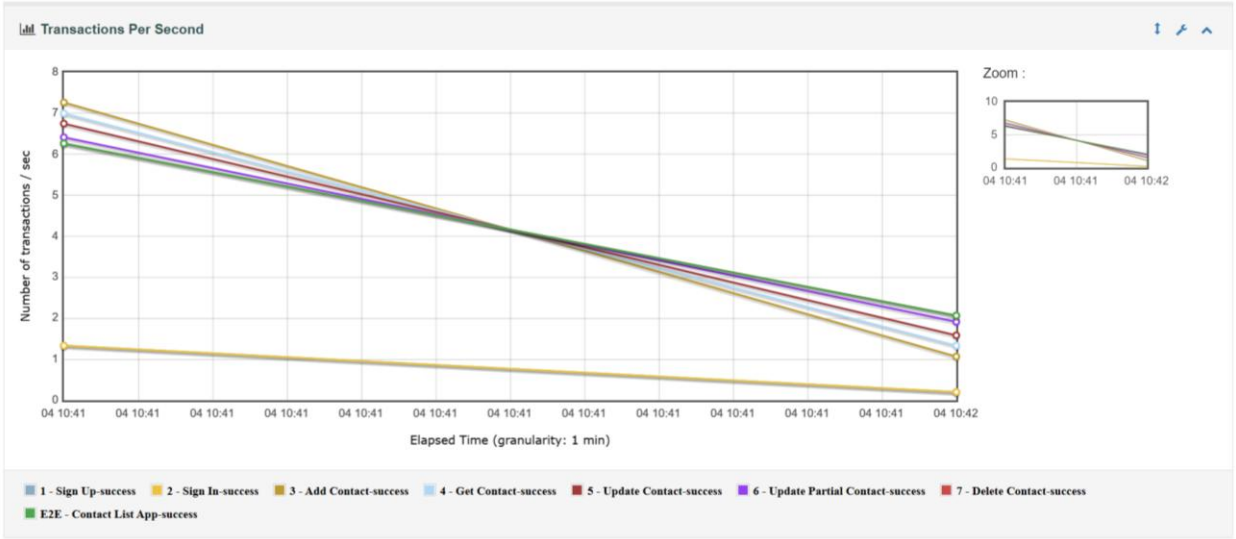
7. Throughput - Hits Per Second

Hits Per Second chart shows how many HTTP requests (or **samplers**) JMeter sent to the server **each second** during the test.



8. Throughput - Transactions Per Second

Transactions Per Second chart shows how many **complete transactions** (requests or samplers) were **successfully executed per second** during the test.



Observation

Metrics	Interpretation
Response Time Over Time	Line chart going up indicating response time is increasing (possible server stress or memory leaks)
Response Time Percentiles	Most users experienced a response time under **2009.00 ms.** Some users waited up to **3000.45** ms, which could indicate occasional performance issues.
Latency Over Time	A growing trend indicates network bottlenecks
Response Time Overview	Most requests having response time **500ms - 1,500 ms**. A small number had slower experiences, possibly due to backend delays or concurrency issues.
Hits Per Second	Drop in hits/sec while threads remain constant indicating server may be overloaded
Transactions Per Second	Line chart drop indicating server slowdown

Recommendation

To increase response time and throughput the site need to optimize application architecture, server configuration, and code:

1. Optimize Backend Response Time

Optimize Database Access

- Use indexed queries.
- Cache frequent queries (e.g., with Redis or Memcached).

Use Caching

- Page caching: cache entire HTML responses for anonymous users.
- Object caching: store frequently accessed data in memory (e.g., Redis).
- HTTP caching: enable browser or CDN caching for static content.

Reduce Response Payload Size

- Compress responses (e.g., Gzip or Brotli).
- Minimize JSON or XML responses — remove unnecessary fields.
- Use pagination for large data responses.

2. Increase Backend Throughput

Increase Concurrency Handling

- Use non-blocking I/O (e.g., Node.js, NIO in Java, asyncio in Python).
- Tune thread pools (e.g., Tomcat or Spring Boot max threads).

Scale Backend Resources

- Vertical scaling: More CPU, memory, or faster disk.
- Horizontal scaling: Add more application servers.
- Use a load balancer (e.g., NGINX, HAProxy) to distribute traffic.

Conclusion

Server performance shows an increase in response time and latency throughout the ± 2684 transactions simultaneously during the testing. So the number of Transactions Per Second and Requests Per Second is decreasing. To achieve a stable performance (under 500ms response time), server optimization is required.