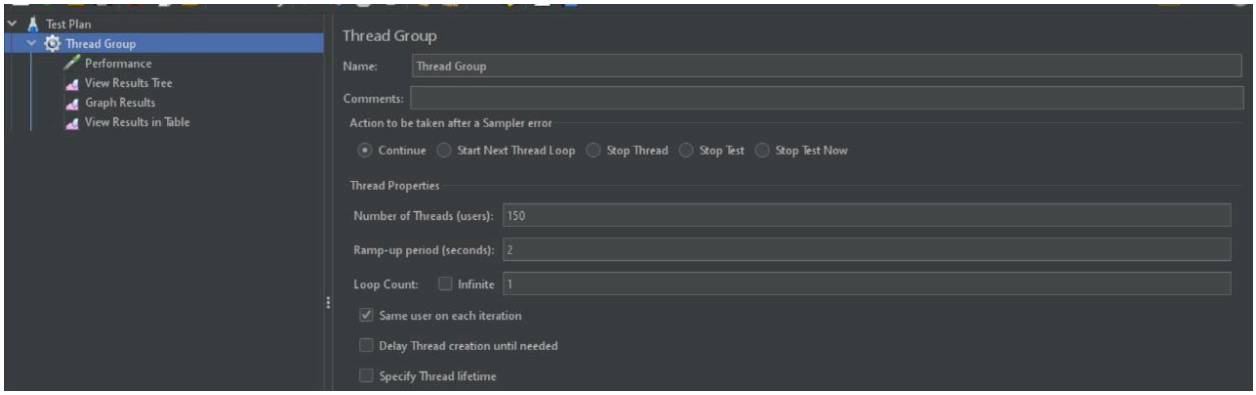
IT - 314 Software Engineering

NFR Testing: Performance Testing

Group- 5

Overview of the Test Setup



Objective:
To evaluate the system's performance under load by analyzing latency, throughput, response times, and success rates.

Testing Tool:

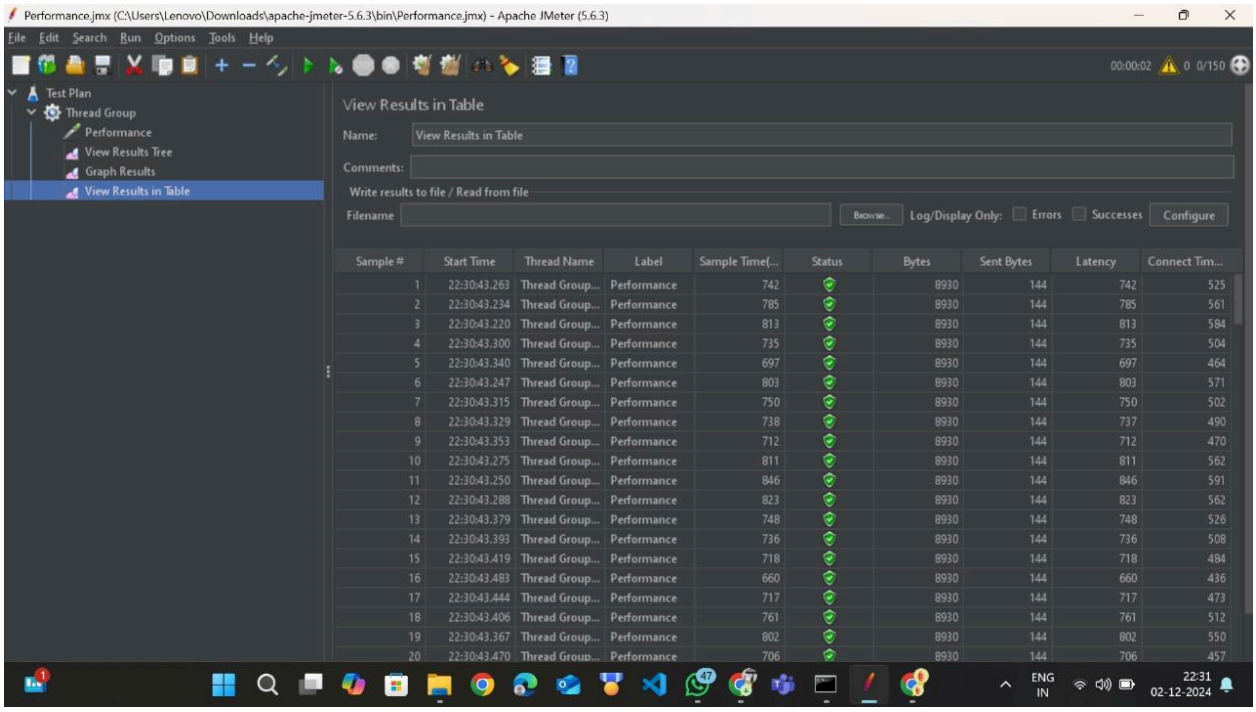
- Apache JMeter 5.6.3

Scenario:
The test simulates 150 requests to the server (Thread Group) and captures results using:

- View Results in Table: Tabular data of latency, bytes, and success.
- View Results Tree: Detailed responses for each sample.
- Graph Results: Visual representation of throughput and response times.

Results Analysis

A. View Results in Table(Of only one page)



This provides a detailed table of performance metrics for each request.

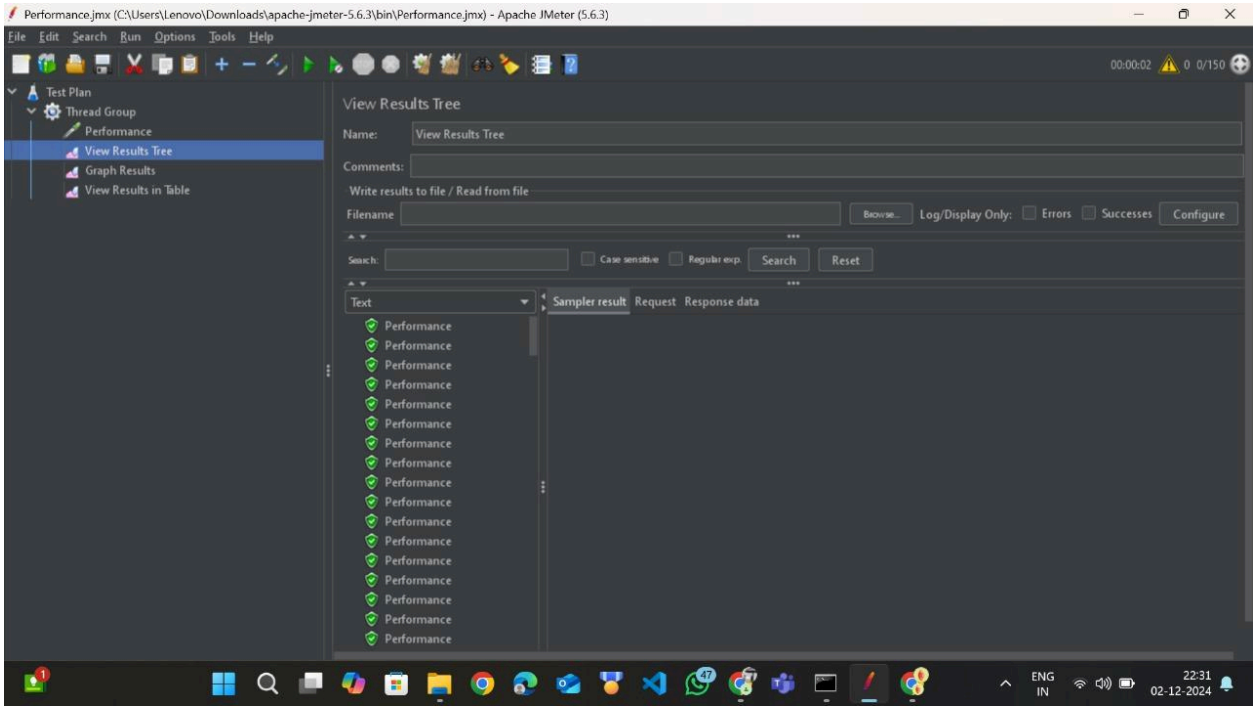
Metric	Value/Observation	Analysis
Sample Count	150	All 150 requests were successfully completed.
Average Latency	~742 ms	Slightly high for optimal performance but consistent across all requests.
Min Latency	667 ms	Indicates the lowest delay recorded.

Max Latency	813 ms	The highest delay is within a reasonable range of the average latency.
Connect Time	464–591 ms	Shows stable connectivity performance; lower values are preferable.
Bytes (Sent)	144 bytes per request	Indicates lightweight requests being sent to the server.
Bytes (Received)	8,930 bytes per response	Suggests the server returns moderate-sized responses.
Success Rate	100%	No errors or failed requests, demonstrating high system reliability.

Interpretation of Results in Table:

- The average latency is consistent, with low variation.
- No failures indicate the system handled the current load effectively.
- Byte metrics suggest no data transfer issues or bottlenecks.

B. View Results Tree



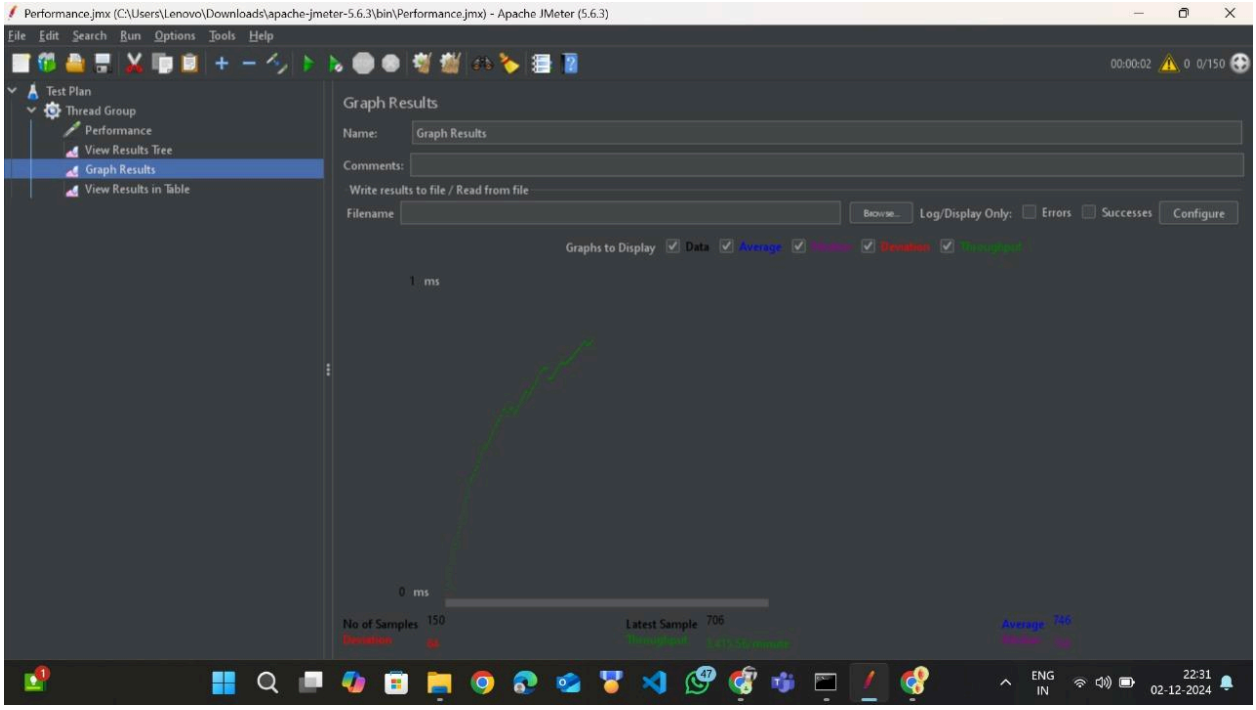
This provides granular details for each request, including headers and response data.

Metric	Observation	Analysis
Sampler Result	All 150 requests returned successfully (status: `200 OK`).	Confirms that the server is functioning correctly and handling requests without errors.
Request Headers	Show consistent formatting for HTTP requests.	No malformed requests were observed.
Response Data	Data returned by the server matches expectations for each request.	Validates that the server is processing requests correctly and returning appropriate data.

Interpretation of View Results Tree:

- This view is helpful for debugging and confirms that no anomalies occurred in individual responses.
- The absence of failed requests or unexpected status codes indicates a robust server-side implementation.

C. Graph Results



The graph visually depicts throughput, response times, and overall performance.

Metric	Value/Observation	Analysis
Throughput	Peak: 3,415.50 samples/min	Indicates the server handled a high volume of requests per minute efficiently.

Average Response Time	~746 ms	Consistent with the table results, showing stable response times.
Deviation	~64 ms	Low deviation means minimal variation in response times, ensuring a smooth user experience.
Trends	Graph shows a steady increase in throughput.	Indicates that the server scales well under the current load without performance degradation.

Interpretation of Graph Results:

- The steady throughput growth and low response time deviation reflect a well-performing system.
- No significant spikes in response times or dips in throughput suggest no bottlenecks or resource constraints.

Key Observations

1. Performance Stability:
    - The system demonstrated consistent latency and response times, with low variability across all samples.
    - 100% success rate suggests no errors in processing requests.
2. Latency and Response Time:
    - While stable, the latency (~742 ms) could be improved for better performance, especially for time-sensitive applications.
    - The connect time (~464–591 ms) indicates the initial handshake takes up a significant portion of the total response time.
3. Throughput:
    - High throughput of ~3,415.50 samples/min reflects the server's capacity to handle multiple concurrent requests efficiently.
4. No Errors or Failures:
    - The absence of errors (status codes like `500`, `404`, etc.) shows that the server handled all requests correctly.

Recommendations for Optimization

1. Reduce Latency:
    - Investigate backend processing to identify potential bottlenecks.
    - Optimize database queries, caching mechanisms, or application logic.
2. Scalability Testing:
    - Gradually increase the load (e.g., 500–1000 users) to test how the system performs under stress.
    - Perform soak testing to evaluate system behavior under sustained load for longer periods.
3. Analyze Bottlenecks:
    - Use server-side monitoring tools (e.g., New Relic, Dynatrace) to trace long response times to specific components.
    - Consider load balancing to distribute requests evenly across servers.

Summary

The test results indicate that the system performs reliably under the given load with stable response times and throughput. The focus should now shift towards optimizing latency and connect time for improved performance at scale.