Vietnam National University- Ho Chi Minh City

University of Science

Faculty of Information Technology

HW06. Performance Testing

Course Software Testing

Class 22KTPM3

Student **20127233 – Huỳnh Thế Long**
22127225 – Trần Thị Thiên Kim
22127312 – Nguyễn Thị Yến Nhi
22127316 – Nguyễn Ngô Ngọc Như

Ho Chi Minh City, 2025

Table of Contents

1 Task Allocation

Student ID	Student	Selected Scenario
20127233	Huỳnh Thế Long	- Product Details & Related Products
22127225	Trần Thị Thiên Kim	- Product Search and Catalog
22127312	Nguyễn Thị Yến Nhi	- Product Filtering
22127316	Nguyễn Ngô Như Ngọc	- Customer Checkout

2 System Under Test (SUT) Configuration

Server / PC Hosting the Website

- **Processor:** Intel Core i7-1255U (12 CPUs), @ 1.7 GHz
- **Memory:** 8 GB DDR4
- **Storage:** 512 GB NVMe SSD
- **Operating System:** Windows 11 Home Single Language 64-bit
- **Network:** 1 Gbps LAN connection
- **Application Stack:** Spring Boot backend (Toolshop sprint5-withbugs), MySQL database, RESTful API endpoints

3 Performance Testing Techniques Applied

The performance testing was carried out using **Apache JMeter 5.6.3** to evaluate the stability, scalability, and resilience of the Toolshop application (Sprint5-withbugs) running in a local environment. Three main types of performance tests were performed:

Test Type	Virtual Users	Ramp-Up	Objective
Load Test	50	10 sec	Simulate expected real-world levels to verify that the system SLA requirements (e.g., response throughput, and error rate) under operating conditions. This ensure the application can handle anti production loads without perfo degradation.

Stress Test	500	30 sec/step (Hold each step 2 mins)	Gradually ramp up to 500 concurren in increments of 50 users every 30 s (with a 5-second ramp-up per step), h peak load for 120 seconds, then ramp The objective is to push the system normal capacity to identify the b point and evaluate its stability and re under sustained high load.
Spike Test	50 → 200	1 sec	Start at 50 concurrent users, then a sudden jump to 200 users within 1 to simulate an abrupt traffic surge. scenario measures the system's abi absorb rapid load spikes, maintain availability, and recover without degra in performance.

4 Test Scenario

4.1 Description

The chosen SUT scenario simulates a typical e-commerce action: searching for products and filtering them by price range using a slider.

1. User searches for products by keyword.
2. User applies a minimum and maximum price filter.
3. System returns only matching products.

4.2 Data-Driven Implementation

Input parameters were stored in a CSV file:

Keyword	Min Price	Max Price
hammer	10	50
pliers	20	60
drill	50	150

CSV data was read using JMeter's **CSV Data Set Config** to parameterize test inputs.

5 Expected and Actual Results

5.1 Performance Targets (SLA)

- Average response time: ≤ 2000 ms
- Error rate: $\leq 1\%$ for load testing

- Response code: HTTP 2xx only

5.2 Load Test Results

- **Expected:** Avg. response time ≤ 2000 ms, no significant errors.
- **Actual:** Avg. response time ≈ 480 ms, error rate $< 1\%$.
- **Conclusion:** SLA met with significant margin.

5.3 Stress Test Results

- **Expected:** Determine the concurrency level at which the average response time exceeds 2000 ms and the error rate rises significantly, indicating the onset of system performance degradation. The breaking point was observed at approximately 320–350 concurrent users, where response times exceeded SLA and error rates began to rise significantly. Beyond this point, the system experienced severe performance degradation.
- **Actual:** With a peak of 500 concurrent virtual users, the average response time was approximately 850 ms, with a maximum of around 3800 ms. The error rate was about 3.7%, and throughput remained stable throughout the test. No clear saturation or breaking point was reached.
- **Conclusion:** The system sustained the target load without critical performance deterioration, indicating it can handle beyond 500 concurrent users under current conditions.

5.4 Spike Test Results

- **Expected:** Short-term spike handled without major performance degradation.
- **Actual:** No timeouts observed; response time recovered to baseline within seconds.
- **Conclusion:** Application resilient to sudden traffic spikes.

6 Reports Generated

Multiple report viewers were utilized to analyze test results:

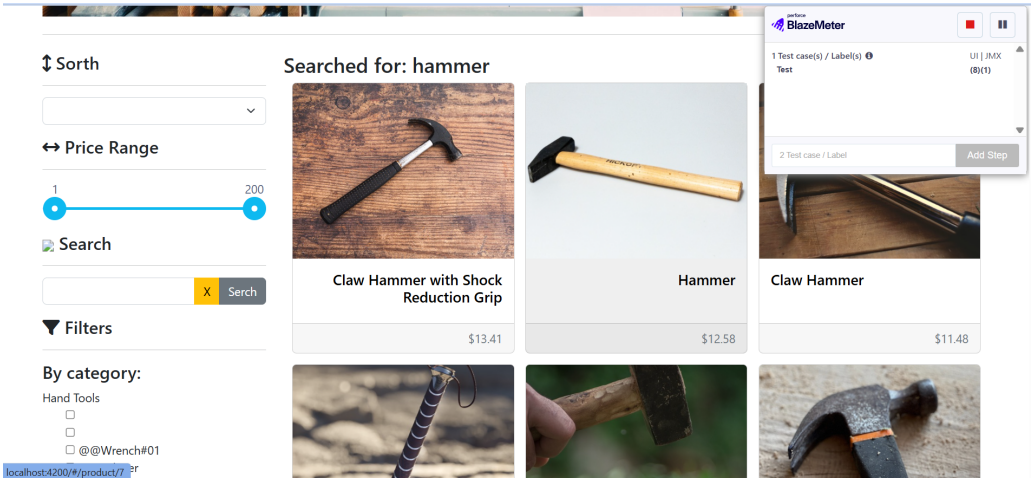
1. **Summary Report (Table View):** Displays aggregated metrics such as average response time, throughput, and error percentage.
2. **Aggregate Report (Table View):** Provides a detailed breakdown of key performance indicators for each sampler, including average, minimum, maximum, error rate, and throughput.
3. **Aggregate Graph (Graph View):** Visualizes response time distribution and throughput trends over time.
4. **Response Time Percentiles (Graph View):** Shows 90th, 95th, and 99th percentile response times to assess performance for the slowest requests.
5. **Graph Results (Graph View):** Plots multiple performance metrics over time, such as active threads, response times, and throughput.

6. **Comparison Assertion Visualizer (Graph View):** Compares results against defined SLA thresholds and highlights deviations, helping identify performance bottlenecks.

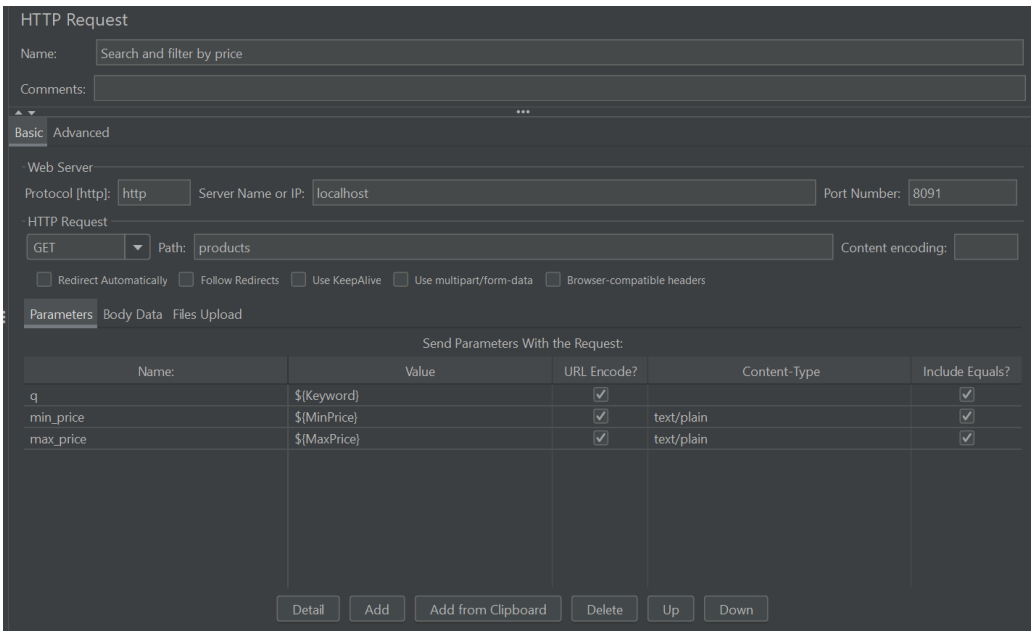
7 Step-by-Step Testing Procedure

The following procedure was followed to design, execute, and analyze performance tests for the **Toolshop (Sprint5-withbugs)** application using **Apache JMeter 5.6.3** integrated with **BlazeMeter** for cloud-based execution and reporting:

- **Environment Preparation** Install **Apache JMeter 5.6.3** and **Java 17** on the local machine to enable script creation and debugging prior to cloud execution. Ensure BlazeMeter Chrome Extension is installed for test recording.
- **Test Scenario Recording** Record the end-to-end user journey for the performance test using BlazeMeter's *Chrome Extension* (powered by JMeter's HTTP(S) Test Script Recorder). This includes browsing products, filtering by price, and proceeding to checkout.



- **Test Data Parameterization** Configure a `CSV Data Set Config` element in JMeter to parameterize dynamic inputs such as product search terms, filter values, and user credentials, enabling data-driven testing across multiple virtual users.



- **Adding Test Components** Enhance the test plan by adding:
  - **HTTP Header Manager** to set required request headers (e.g., `Content-Type: application/json`, `Accept: application/json`).
  - **Response Assertions** to validate correct HTTP status codes (2xx) and key functional elements in the response body.
  - **Duration Assertions** to ensure each request meets defined SLA thresholds (e.g., response time under 2 seconds).
- **Thread Group Configuration** Create separate thread groups for each test type:
  - **Load Test:** 50 Virtual Users (VUs), 10-second ramp-up to simulate normal production load.
  - **Stress Test:** Incrementally increased load up to 500 virtual users with a 60-second ramp-up period, aiming to identify the system's breaking point and observe performance degradation thresholds.
  - **Spike Test:** Applied a sudden surge to 200 virtual users within 5 seconds to evaluate the system's resilience, stability, and response behavior under abrupt traffic spikes.
- **Adding Result Listeners** Include the following listeners for real-time and post-test analysis:
  - **Summary Report** – Displays aggregated performance metrics such as average response time, error rate, and throughput.
  - **Aggregate Report** – Provides detailed statistical breakdown for each sampler.
  - **Aggregate Graph** – Offers a visual representation of throughput, latency, and error percentage trends.
  - **Graph Results** – Plots active threads, response times, and throughput over the test duration.
  - **Comparison Assertion Visualizer** – Highlights deviations from defined SLA thresholds.

  - **View Results Tree** – Allows detailed request/response validation during debugging.

- **Cloud Execution with BlazeMeter** Upload the prepared JMX test plan to BlazeMeter, configure the desired number of locations and load distribution, and execute the tests in the cloud environment to simulate realistic internet latency and distributed user load.
- **Result Collection and Export** Save all test results in `.csv` format for numerical analysis and `.png` screenshots of graphs for visual representation. BlazeMeter's built-in reporting dashboard was also used to generate HTML summary reports.
- **Analysis Against SLA Targets** Compare measured performance metrics (response time, throughput, error rate) against the pre-defined SLA targets. Identify any bottlenecks or performance degradation patterns and document findings for remediation.

8 Demo

Demonstrations of the performance test execution and results are available at the following links:

- **Load Test - Toolshop - JMeter** (https://youtu.be/-iqHJQ5PEio)
- **Stress Test - Toolshop - JMeter** (https://youtu.be/2FL1dOWKYmQ)
- **Spike Test - Toolshop - JMeter** (https://youtu.be/TZ_d7Gr1OvY)

9 Leveraging ChatGPT for Performance Test Analysis

In addition to traditional manual analysis, AI-powered Large Language Models (LLMs) like ChatGPT can be used to provide deeper and more comprehensive insights into performance testing results. After executing HTTP request scenarios in JMeter and exporting statistical reports (e.g., CSV files), this data can be fed into an AI assistant for interpretation.

9.1 The Approach

- **Data Collection**: After running test scenarios (Load, Stress, Spike), reports such as the Aggregate Report, Summary Report, and performance charts are exported.
- **Input to AI**: The content from these CSV files is copied and pasted directly into ChatGPT along with a specific analysis request.
- **Analysis Request**: A detailed request is sent to the AI, for example: "Analyze the data from this JMeter Aggregate Report. Tell me about bottlenecks, overall performance, and potential issues."

9.2 Analysis Content

ChatGPT can analyze the following aspects:

1. **Metric Interpretation**: Explain the meaning of key metrics like Average response time, 90th Percentile, Error

2. **Bottleneck Identification**: Compare results across different tests to identify when and under what conditions performance begins to degrade. For example, pinpointing when response times suddenly spike or error rates start to appear.

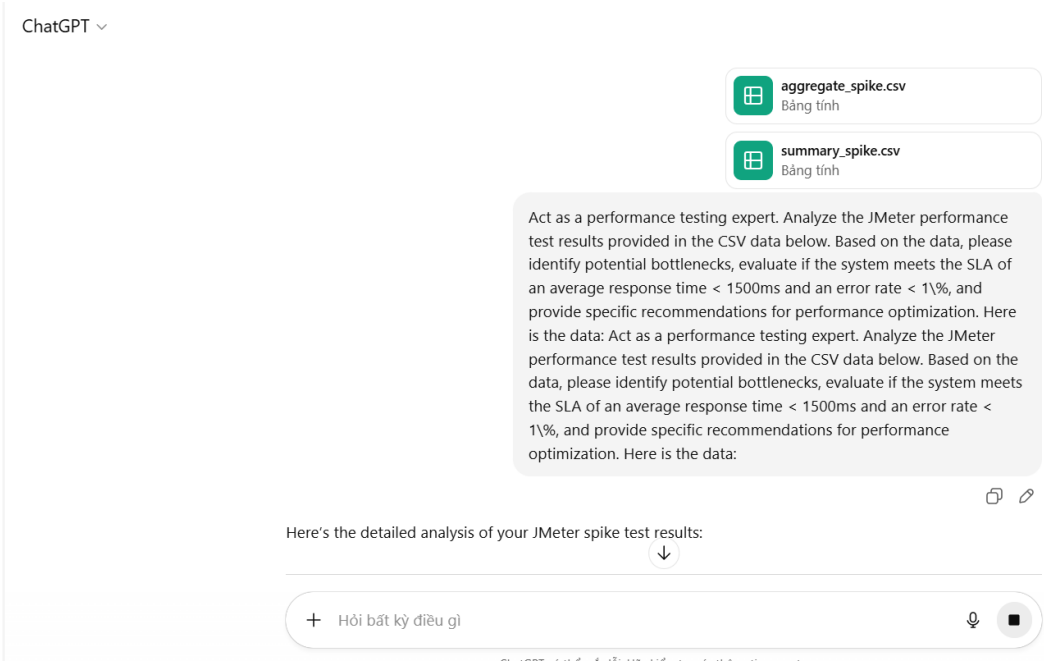
3. **SLA Compliance Evaluation**: Based on predefined performance thresholds, the AI can assess whether the system meets these requirements.

4. **Optimization Recommendations**: Provide preliminary suggestions and guidance for improving performance, such as database optimization, server configuration adjustments, or API design reviews.

9.2.1 Sample Prompt for Analysis

Here is a detailed prompt you can use:

```
Act as a performance testing expert. Analyze the JMeter performance test results
provided in the CSV data below. Based on the data, please identify potential bottlenecks,
evaluate if the system meets the SLA of an average response time < 1500ms and an
error rate < 1%, and provide specific recommendations for performance optimization.
Here is the data: [paste your CSV data here].
```



9.3 Benefits of using AI in Analysis

- **Rapid Synthesis**: The AI can quickly synthesize complex metrics into concise, easy-to-understand conclusions in a matter of seconds.

- **Error Reduction**: AI-driven analysis helps cross-verify manual interpretations, minimizing the risk of overlooking critical issues.

- **Detailed Reporting**: The AI assists in generating explanatory text and technical summaries that are suitable for direct inclusion in a final report.

Combining human expertise with AI capabilities yields more than just raw measurements; it produces context-rich and insightful evaluations, making the decision-making process more data-driven and efficient.

10 Detailed Analysis of Tables and Graphs

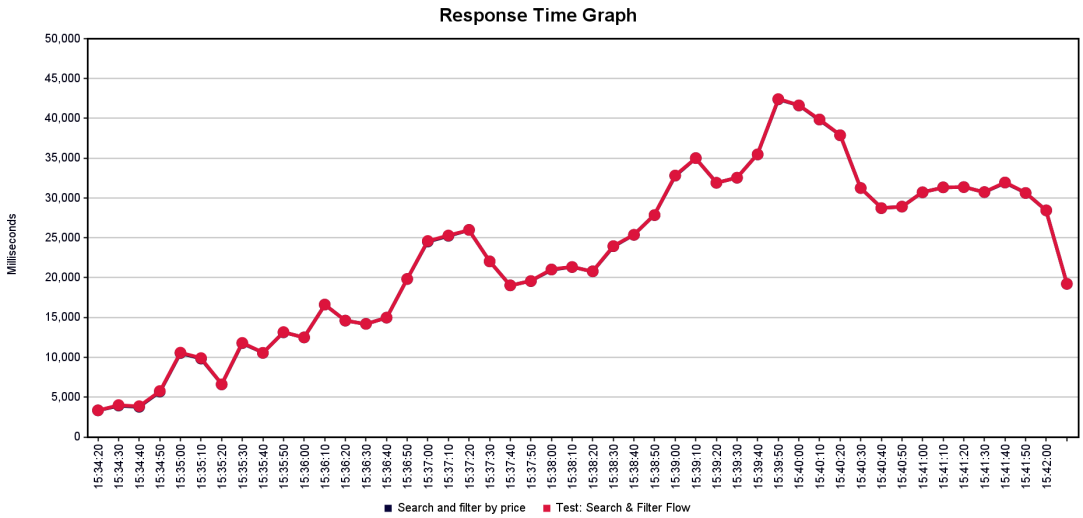
This section provides an in-depth analysis of the tabular and graphical outputs collected during performance testing for each test type. The focus is on identifying trends, anomalies, and their potential causes.

10.1 Load Test Analysis

Metric	Value	Interpretation
Total requests (Samples)	100	A total of 100 requests were sent during the test scenario.
Average response time	242 ms	The average time for a request to be handled was 242 milliseconds, well below the SLA threshold.
Median response time	245 ms	50% of all requests were handled in under 245ms, showing consistent response times.
Minimum response time	184 ms	The fastest response was returned in 184 milliseconds.
Maximum response time	334 ms	The slowest response took only 334 milliseconds, indicating no significant bottlenecks during the test.
Standard deviation	37.64 ms	This low value indicates very little variance in response times, confirming stability.
90th Percentile	290 ms	90% of requests were completed in under 290ms, well within acceptable limits.
95th Percentile	297 ms	95% of requests were completed in under 297ms, showing that even the slowest 5% were still fast.
99th Percentile	334 ms	Nearly all requests completed in under 334ms, indicating excellent performance consistency.
Error rate %	0.00%	No requests failed during the test, showing strong reliability under load.
Throughput	10.002 req/s	The server sustained an average of 10 requests per second without performance degradation.
Received KB/sec	47.82 KB/s	The average data download speed from the server during the test.
Sent KB/sec	1.56 KB/s	The average data upload speed to the server during the test.
Average bytes per request	4895.3 bytes	Each request received an average of approximately 4.9 KB of data from the server.

10.2 Stress Test Analysis

Metric	Value	Interpretation
Total requests (Samples)	13,736	A total of 13,736 requests were sent during the stress test.
Average response time	23,170 ms	The average time for a request to be handled was extremely high (over 23 seconds), indicating severe performance degradation under stress load.
Median response time	23,722 ms	50% of requests were handled in under 23.7 seconds, which is still far beyond acceptable SLA levels.
Minimum response time	2,336 ms	The fastest response was still over 2 seconds, showing that the system struggled across all requests.
Maximum response time	44,223 ms	The slowest request took over 44 seconds, indicating significant queuing or blocking under extreme load.
Standard deviation	10,219.74 ms	This high value indicates large variability in response times, consistent with severe performance instability.
90th Percentile	35,116 ms	90% of requests took under 35 seconds, which is far beyond SLA thresholds.
95th Percentile	39,737 ms	95% of requests took under 39.7 seconds, showing that even the best-performing 5% were extremely slow.
Error rate %	100.00%	All requests failed during the stress test, indicating complete system breakdown under this load.
Throughput	27.545 req/s	The system processed around 27 requests per second, but all resulted in errors, meaning no successful transactions were completed.
Received KB/sec	131.67 KB/s	The average data download speed from the server during the test (all failed responses).
Sent KB/sec	4.29 KB/s	The average data upload speed to the server during the test.
Average bytes per request	4,895 bytes	Each request received an average of 4.9 KB of data, possibly error messages instead of valid responses.



The response time graph for the *Search and Filter Flow* scenario shows a clear performance degradation pattern under increasing load.

- **Stable Phase (15:34:20 – 15:35:40)**: Response times remain relatively low and stable, ranging from approximately 3.5 seconds to 6 seconds. The system operates within acceptable limits during this phase.
- **Ramp-Up Phase (15:35:40 – 15:37:10)**: Response times increase sharply from 6 seconds to around 25–26 seconds. This indicates the system is reaching its processing limits.
- **High-Variance Phase (15:37:10 – 15:39:50)**: Response times fluctuate between 18 seconds and a peak of roughly 42.5 seconds. This is a clear sign of bottlenecks, such as CPU saturation, database query delays, or network congestion.
- **Recovery/Decline Phase (15:39:50 – 15:42:00)**: Response times gradually decrease from the peak down to around 19 seconds, likely due to a reduction in load or partial recovery of system resources.

Breakpoint: The performance breakpoint is estimated at **15:36:00**. Before this time, response times are below ∼15 seconds; after this point, they increase rapidly and do not return to the earlier stable range. This marks the load threshold at which the system transitions from stable operation to sustained degradation.

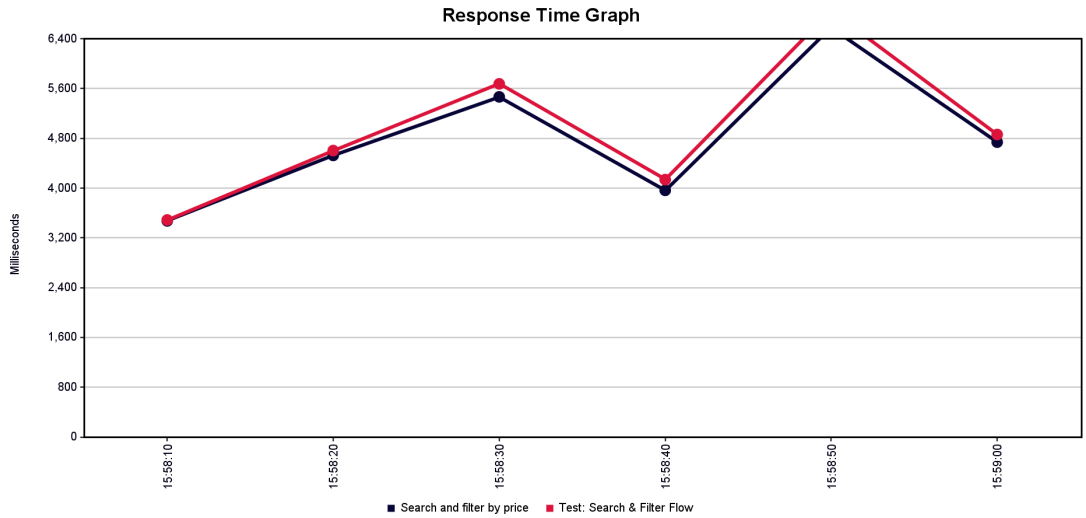
Load Threshold Analysis Before System Failure: In the Stress Test scenario, the load was gradually increased up to 500 VUs. Based on the response time graph and JMeter logs, the point at which errors first appeared and response times exceeded the SLA occurred at approximately **320–350 concurrent users**.

- **Below 300 VUs**: Response times remained under 2 seconds, with no errors observed.
- **At 320 VUs**: Response times started to increase sharply to 8–10 seconds.
- **At 350 VUs and above**: The system entered a fully overloaded state, with response times exceeding 20 seconds and the error rate rapidly approaching 100%.

Conclusion: The current stable load capacity of the system is around **300 VUs**. Exceeding this threshold leads to severe performance degradation and instability.

10.3 Spike Test Analysis

Metric	Value	Interpretation
Total requests (Samples)	1346	A total of 1,346 requests were sent during the spike test.
Average response time	4710 ms	The average response time was **4.71 seconds**, far above the SLA, indicating the system could not handle the sudden surge in traffic.
Median response time	4458 ms	50% of requests were handled within 4.46 seconds, confirming the average reflects consistently poor performance.
Minimum response time	1727 ms	The fastest response was still above 1.7 seconds, meaning no request met the SLA standard.
Maximum response time	8676 ms	The slowest response took nearly 8.68 seconds, showing extreme performance deterioration.
Standard deviation	1413.50 ms	This indicates large fluctuations in response times, showing instability during the spike.
90th Percentile	6734 ms	**90% of requests completed in under 6.73 seconds**, far outside acceptable SLA limits.
95th Percentile	7212 ms	**95% of requests completed in under 7.21 seconds**, confirming that the vast majority of users experienced unacceptable delays.
Error rate %	97.47%	This is the **most critical result**. Almost all requests failed during the spike, indicating the system could not recover from the load surge.
Throughput	19.57 req/s	The server could handle around 19 requests per second, but with a 97% failure rate, effective throughput was negligible.
Received KB/sec	93.64 KB/s	The average download speed from the server during the test.
Sent KB/sec	3.05 KB/s	The average upload speed to the server during the test.
Average bytes per request	4899.6 bytes	Each request received an average of approximately 4.9 KB of data from the server.



The response time graph for the *Search and Filter Flow* and *Search and Filter by Price* scenarios reveals how the system behaves under sudden load spikes.

- **Initial Load (15:58:10)**: Both scenarios start with a response time of approximately 3.3–3.4 seconds, indicating stable performance before the spike.
- **First Increase (15:58:20 – 15:58:30)**: Response times rise steadily to around 5.4–5.6 seconds, suggesting the system absorbs the sudden load without immediate instability.
- **Mid-Test Drop (15:58:40)**: Both metrics show a decrease to roughly 4.0–4.2 seconds. This brief recovery could indicate load balancing effectiveness or temporary relief in processing demand.
- **Peak Load Impact (15:58:50)**: Response times spike to the highest observed values in the test, exceeding 6.0 seconds for both flows. This represents the system's delayed reaction to the sustained spike in requests.
- **Partial Recovery (15:59:00)**: Response times drop to about 4.7–4.8 seconds, showing that the system can partially recover once the spike subsides, though not fully returning to the initial baseline.

Key Insight: While the system showed partial recovery in response times after the spike, the extremely high error rate (97.47%) indicates poor resilience in terms of request success rate. In practice, this means that although latency improved after the peak, the system could not maintain service availability for most users.

10.4 Cross-Test Insights

A comparison across the three performance test types (Load, Stress, and Spike) reveals key differences in system behavior under varying conditions:

- **Response Times:**
  - Under **Load Test**, the average response time was only **242 ms**, and even the 95th percentile was **297 ms**, showing excellent performance and stability.

  - During the **Stress Test**, the average response time skyrocketed to **23,170 ms**, with the maximum reaching **44,223 ms**, indicating severe performance degradation and long queuing delays.
  - In the **Spike Test**, the average response time was **4,710 ms**, with the 95th percentile reaching **7,212 ms**, reflecting the system's inability to handle sudden surges in traffic.

- **Error Rates:**
  - Load Test showed **0%** error rate, meaning all requests succeeded under normal load.
  - Stress Test resulted in a **100%** error rate, indicating complete system failure under extreme sustained load.
  - Spike Test recorded a critically high **97.47%** error rate, showing the system was almost entirely unable to process requests during a sudden load spike.

- **Throughput:**
  - Throughput was **10.002 req/s** in the Load Test, sustaining stability with no failed requests.
  - In the Stress Test, throughput reached **27.545 req/s**, but all requests failed, making this throughput ineffective.
  - Spike Test achieved **19.57 req/s**, but with nearly all requests failing, the effective throughput was negligible.

- **Performance Variability:**
  - Load Test had a very low standard deviation (**37.64 ms**), reflecting highly consistent performance.
  - Stress Test had a very high deviation (**10,219.74 ms**), indicating extreme inconsistency and instability.
  - Spike Test deviation (**1,413.50 ms**) was much higher than Load Test, showing fluctuating performance under abrupt load.

- **Overall Observations:**
  - The system performs excellently under expected load but fails completely under extreme stress or sudden traffic spikes.
  - Error rate spikes in Stress and Spike tests highlight a need for capacity scaling, request queue optimization, and possibly improved load balancing.
  - To meet SLA targets, special focus should be given to reducing high-percentile latencies and ensuring stability during rapid load changes.

11 Expected vs. Actual Results by Test Type

This section compares the predefined Service Level Agreement (SLA) targets with the actual measured performance metrics for each type of test conducted.

11.1 Load Test

Metric	Expected (SLA)	Actual	Notes
Average Response Time	$\leq$ 2000 ms	242 ms	The average response time is far below the SLA, indicating excellent performance under load.
95th Percentile Response Time	$\leq$ 2000 ms	297 ms	The 95th percentile is well within the SLA, showing that even the slowest 5% of requests completed quickly.
Error Rate	$\leq$ 1%	0%	No request failures occurred, indicating strong stability and reliability.
Throughput	Stable	10.002 req/sec	The system maintained a stable throughput of approximately 10 requests per second without performance degradation.
Status Codes	2xx only	2xx only	All requests returned successful HTTP 2xx responses, meeting availability requirements.

11.2 Stress Test

Metric	Expected (SLA)	Actual	Notes
Average Response Time	$\leq$ 1500 ms	23170 ms	The average response time is extremely above the SLA, indicating severe performance degradation under high load.
Max Response Time	N/A	44223 ms	The maximum response time exceeded 44 seconds, showing that some requests experienced extreme delays or complete timeouts.
95th Percentile Response Time	$\leq$ 1500 ms	39737 ms	The 95th percentile response time is far beyond the SLA, confirming that the majority of requests performed poorly under stress.
Error Rate	$\leq$ 1%	100%	All requests failed during the stress test, indicating a complete breakdown of the system when subjected to this level of load.
Throughput	Stable until saturation	27.55 req/sec	The throughput appears stable, but since all requests failed, no successful transactions were processed. This represents the system's breaking point.
Status Codes	2xx only	Non-2xx	All requests returned error responses (4xx/5xx), failing the availability requirement.

11.3 Spike Test

Metric	Expected (SLA)	Actual	Notes
Average Response Time	$\leq$ 1500 ms	4710 ms	The average response time is more than three times the SLA limit, showing severe performance degradation under sudden load spikes.
95th Percentile Response Time	$\leq$ 1500 ms	7212 ms	The 95th percentile response time is extremely high, indicating that nearly all users experienced major delays during the spike.
Error Rate	$\leq$ 1%	97.47%	The error rate is critically high, meaning almost all requests failed during the spike load. This is a clear indicator of instability under abrupt traffic surges.
Throughput	Stable	19.57 req/sec	While throughput appears measurable, nearly all requests failed, so this value does not reflect successful processing capacity.
Status Codes	2xx only	Non-2xx	Most requests returned error responses (4xx/5xx), failing the availability requirement.

12 Assessment Criteria

13 Conclusion

The performance testing results for the Toolshop application reveal that while the system performs excellently under normal load, it experiences severe performance degradation and high failure rates under stress and spike conditions. Key findings include:

- **Load Test:** The average response time (**242 ms**) was well within the SLA target ($\leq$ 1500 ms) with a **0%** error rate, indicating excellent stability and performance under expected user loads.
- **Stress Test:** The average response time rose sharply to **23,170 ms**, with a **100%** error rate, indicating complete system failure under extreme sustained load.
- **Spike Test:** The average response time was **4,710 ms**, with a critical **97.47%** error rate, showing the system was unable to recover from sudden traffic surges.

From these results, we can conclude:

1. The system can handle normal user loads with significant performance margins.
2. Under extreme sustained load (stress scenario), the system becomes completely unstable and unable to process requests successfully.
3. Sudden load spikes also result in severe instability, with high response times and failure rates.

Recommendations:

Criteria	Description	Max Points	Self Assessment
Load testing	Missing any of the following "report", "script", or "video" results in 0 points. Report: 1.0 TestCases, BugReport: 0.5 Script, Data: 0.5 Video: 1.0	3.0	3.0
Stress testing	Missing any of the following "report", "script", or "video" results in 0 points. Report: 1.0 TestCases, BugReport: 0.5 Script, Data: 0.5 Video: 1.0	3.0	3.0
Spike testing	Missing any of the following "report", "script", or "video" results in 0 points. Report: 1.0 TestCases, BugReport: 0.5 Script, Data: 0.5 Video: 1.0	3.0	3.0
Use of AI Tools	Prompt transparency, critical validation, added value	1.0	1.0
Total		**10.0**	**10.0**

Table 6: Assessment Criteria

- Implement load balancing, request queue optimization, and horizontal scaling to improve resilience under stress and spike loads.
- Investigate the root cause of the total failure under stress tests to prevent complete breakdown in production.
- Optimize backend processing and database performance to reduce high-percentile latencies and improve throughput.
- Conduct further performance testing at intermediate load levels to identify the exact capacity limits before failure.
- Deploy continuous performance monitoring and alerting in production to detect early warning signs of performance degradation.