University of Science – Vietnam National University
Faculty of Information Technology

CS423 – Software Testing

HW07 Report
Performance Testing

Student's Name	Student ID	Class
Tran Nhat Thanh	22125093	22TT

Ho Chi Minh City, December 12th, 2025

Contents

1. Overview

A. Objective

The primary objective of this report is to evaluate the performance, stability, and endurance of the *Product Search, Sort, and Filter* flow. This scenario is critical as it represents the main way users interact with the product page. The test aims to assess system behavior under normal load, identify breaking points under stress, and evaluate recovery during sudden traffic spikes.

B. Scope of Testing

The testing process involved the execution of three distinct workload models using *Apache JMeter*:

- **Load Testing:** Conducted with **200** concurrent users to simulate expected daily traffic and establish a performance baseline.
- **Stress Testing:** Scaled up to **1500** concurrent users to determine the system's endurance threshold and identify hardware/software bottlenecks.
- **Spike Testing:** Simulating a sudden surge of **1000** users in a short interval to evaluate the system's recovery time and stability under shock.

C. Methodology & Technical Implementation

To ensure realistic and robust testing, the following technical approaches were applied:

- **Data-Driven Testing:** To prevent server-side caching and simulate real-world variety, a *CSV Data Set Config* was implemented. The dataset includes 30 unique test cases covering various combinations of:
  - **Search Queries** (q:"hammer", "pliers", etc.)
  - **Sorting Parameters** (sort_field, sort_dir)
  - **Filters** (min_price, category_id, brand_ids, eco) This ensures that every virtual user requests different data, stressing the database rather than just the cache.
- **Reporting & Analysis:** Test results were captured and analyzed using a suite of JMeter Listeners to satisfy the requirement:
  1. **Aggregate Report:** For tabular statistical analysis (Average response time, Throughput, etc.).
  2. **Response Time Graph:** For visualizing latency trends over time.
  3. **Active Threads Over Time:** To correlate user load with system performance.
  4. **HTML Dashboard:** Generated via the *Simple Data Writer* (JTL file) to provide a comprehensive, browser-based view of the test result.

2. Test Environment

A. Hardware

The performance tests were executed on my personal laptop which served as both the Load Generator (running JMeter) and the Host for the Software Under Test (SUT).

- **Device Model:** Dell Vostro 5620
- **CPU:** 12th Gen Intel(R) Core(TM) i5-1240P (1.70 GHz)
- **RAM:** 16.0 GB
- **Operating System:** Windows 11 Home Single Language (Version 25H2)

B. Software & SUT Architecture

The environment is defined via *Docker Compose* with the following specific container configurations:

- **Orchestration:** Docker Desktop 4.47.0 (Engine v28.4.0).
- **Application Containers:**
  - **Frontend:** angular-ui running in Development Mode, mapped to port 4200.
  - **Backend:** laravel-api (PHP) serving the business logic.
  - **Web Server:** web exposes HTTP endpoints to the host (8091 and 8000) and provides access to the backend application through Docker's internal network.
- **Database:** mariadb (Alpine Linux version 10.6.11) running on port 3306.
- **Testing Tool:** Apache JMeter 5.6.3 running on Java 24.

C. Environmental Constraints

Because the SUT is deployed locally via Docker containers on the same machine running the JMeter scripts, both the application and the test tool share the same system resources (CPU/RAM). The "Endurance Threshold" determined in this report reflects this specific localized environment and may differ from a production environment where the server and load generator are on separate machines.

3. Test Configurations & Workload Models

A. Scenario 1: Load Test

- **Goal:** Verify system stability under a steady, expected volume of traffic.
- **Thread Group Type:** Standard Thread Group
- **Configuration:**
  - **Target Concurrency:** 200 Users (Threads).
  - **Ramp-Up Period:** 40 seconds (Gradual entry of 5 users/sec).
  - **Duration:** 600 seconds (10 minutes) steady state.
  - **Loop Count:** Infinite (controlled by duration).

B. Scenario 2: Stress Test

- **Goal:** Identify the breaking point and maximum capacity using a "Stepped" load pattern.
- **Thread Group Type:** Stepping Thread Group
- **Configuration:**
  - **Max Threads:** 1500.

- **Initial Load:** Starts immediately with 300 threads.
- **Step Logic:** Adds 300 threads every 10 seconds (with a 3-sec ramp-up per step).
- **Steady State:** Holds the max load (1500 users) for 120 seconds.
- **Ramp-Down:** Rapid decrease of 500 threads every 2 seconds.

- **Why this approach?** The stepping approach allows us to see exactly *which step* causes performance degradation, offering clearer analysis than a linear ramp.

C. Scenario 3: Spike Test

- **Goal:** Evaluate system recovery after a massive, instantaneous surge in traffic.
- **Thread Group Type:** Standard Thread Group
- **Configuration:**
  - **Target Concurrency:** 1000 Users.
  - **Ramp-Up Period:** 1 second (Simulating an instantaneous "click-storm").
  - **Duration:** 30 seconds.
  - **Analysis Focus:** We monitor how quickly response times return to normal after the initial 1-second shock.

4. Test Case Configuration & Execution Steps

A. Data-Driven Test Case Design

To ensure realistic performance analysis, static data was avoided. A *CSV Data Set Config* was implemented to feed dynamic inputs into the test.

- **Source File:** product_search.csv
- **Dataset Volume:** 30 unique test cases covering different search permutations.
- **Parameter Mapping:** The CSV columns were mapped to the HTTP Request parameters as follows:
  - **Search Query (q):** hammer, pliers, drill, etc.
  - **Sorting (sort):** Mapped to ${sort_field},${sort_dir} (price,asc).
  - **Filtering:**
    - Price Range: between=price,${min_price}
    - Categories: by_category=${category_ids}
    - Brands: by_brand=${brand_ids}
    - Eco-Friendly: eco_friendly=${eco}

B. Execution Steps

The following procedure was repeated for each of the three scenarios (Load, Stress, Spike):

1. **Environment Preparation:**
   - Ensure all Docker containers are running and healthy.
   - Verify the SUT is accessible by hitting the API endpoint: http://localhost:8091/products.
2. **JMeter Configuration:**

- Launch Apache JMeter.
- Open the specific JMX file corresponding to the test scenario (Load_Test.jmx, Stress_Test.jmx, or Spike_Test.jmx).
- Verify the Simple Data Writer is configured with a unique filename to prevent overwriting previous test data.

3. **Test Execution:**
   - Click the *Start* button (Green Play Icon) in the toolbar to begin execution.
   - Allow the test to run for the full configured duration.
4. **Result Analysis:**
   - **Step A:** *Internal Listener Review:* Immediately after the test stops, review the three key internal listeners:
     1. *Active Threads Over Time:* Confirm the user load ramped up and held steady as designed.
     2. *Response Time Graph:* Check for visual spikes or upward trends in latency.
     3. *Aggregate Report*: Check the "Error %" and "90th Percentile" columns for immediate pass/fail indicators.
   - **Step B:** *HTML Report Generation:* Open a terminal in the results folder and run the generation command: jmeter -g [filename].jtl -o [output_folder_name]
   - **Step C:** Open index.html in a browser to view the comprehensive dashboard.

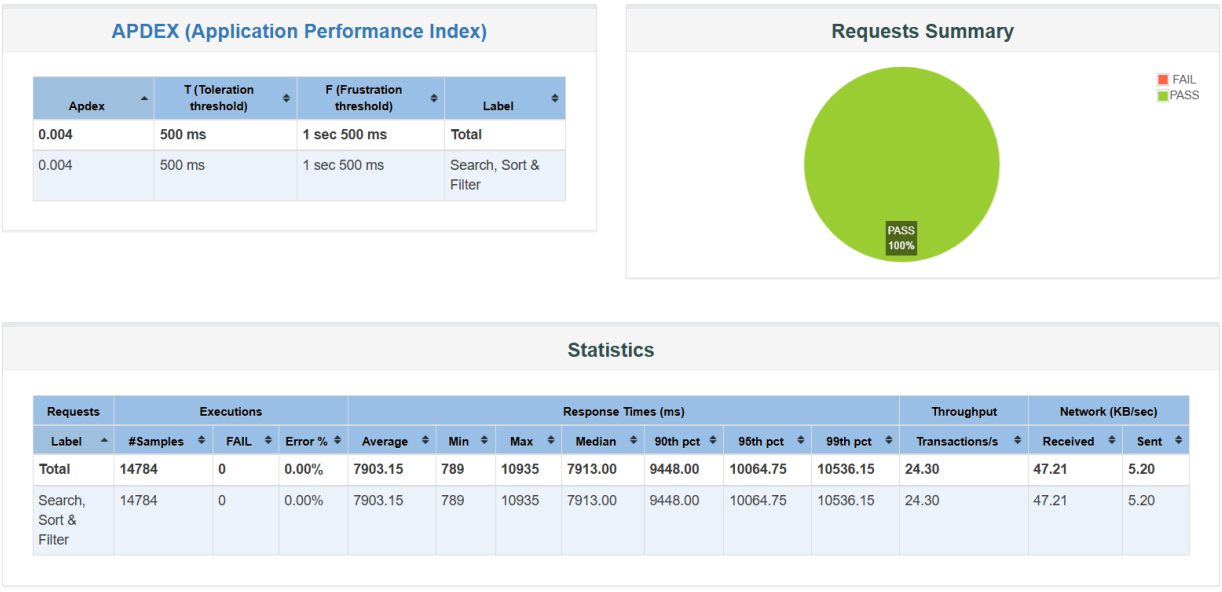
5. Detailed Result Analysis

A. Scenario 1: Load Test

I. Objective

To verify the system's stability and establish a performance baseline under a steady, expected traffic volume of 200 concurrent users.

II. Key Performance Metrics



APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.004	500 ms	1 sec 500 ms	Total
0.004	500 ms	1 sec 500 ms	Search, Sort & Filter

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	14784	0	0.00%	7903.15	789	10935	7913.00	9448.00	10064.75	10536.15	24.30	47.21	5.20
Search, Sort & Filter	14784	0	0.00%	7903.15	789	10935	7913.00	9448.00	10064.75	10536.15	24.30	47.21	5.20

Figure 1: Load Test Overall Statistics.

We can see that:

- **Error Rate:** The system achieved a 0.00% Error Rate (Status: PASS), confirming it can handle expected traffic without crashing.
- **Throughput:** The server consistently processed 24.30 requests per second.
- **Response Time:** The average response time was 7,903 ms, which is quite high but stable.

III. Visual & Graph Analysis

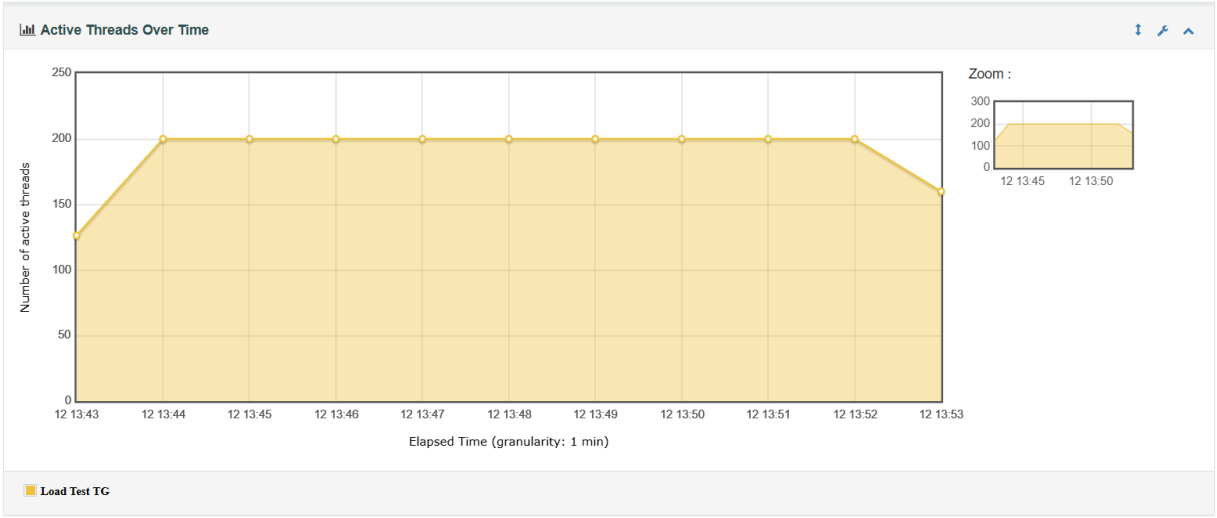


Figure 2: Active Threads showing a steady load of 200 users for 10 minutes.

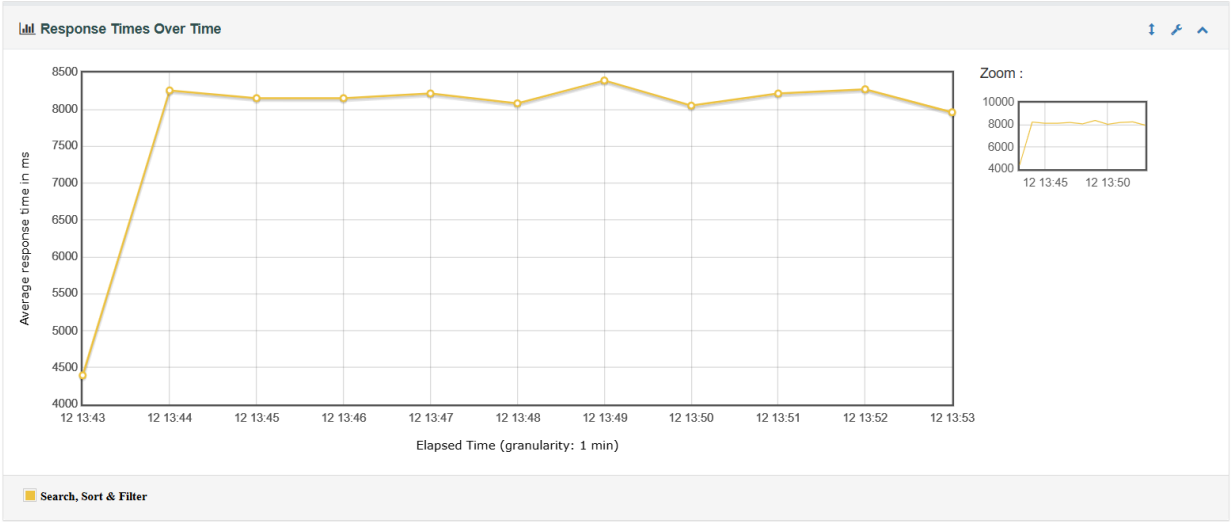


Figure 3: Response Time stabilizing at approximately 8 seconds.

We can observe from both graphs that**:**

- **Stability:** The flat yellow line confirms the load generator successfully maintained 200 concurrent connections for the full duration.
- **Latency Trend:** After the initial ramp-up spike, the response time "flatlined" at ~8000ms. Crucially, it did not continue to rise, indicating there are no memory leaks.

IV. Expected and Actual Result

Metric	Expected Criteria	Actual Result	Status
Error Rate	0.00%	0.00%	PASS
Throughput	> 20 req/sec	24.30 req/sec	PASS
Avg Response Time	< 2,000 ms	7,903 ms	FAIL (High)
90th Percentile	< 4,000 ms	9,448 ms	FAIL (High)

V. Conclusion

The system is *Stable but Slow*. It successfully handled 100% of the traffic, but the baseline latency indicates high processing overhead per request.

B. Scenario 2: Stress Test

I. Objective

To identify the maximum capacity (Endurance Threshold) of the system using a "Stepping" load pattern (increasing users every 10 seconds).

II. Key Performance Metrics

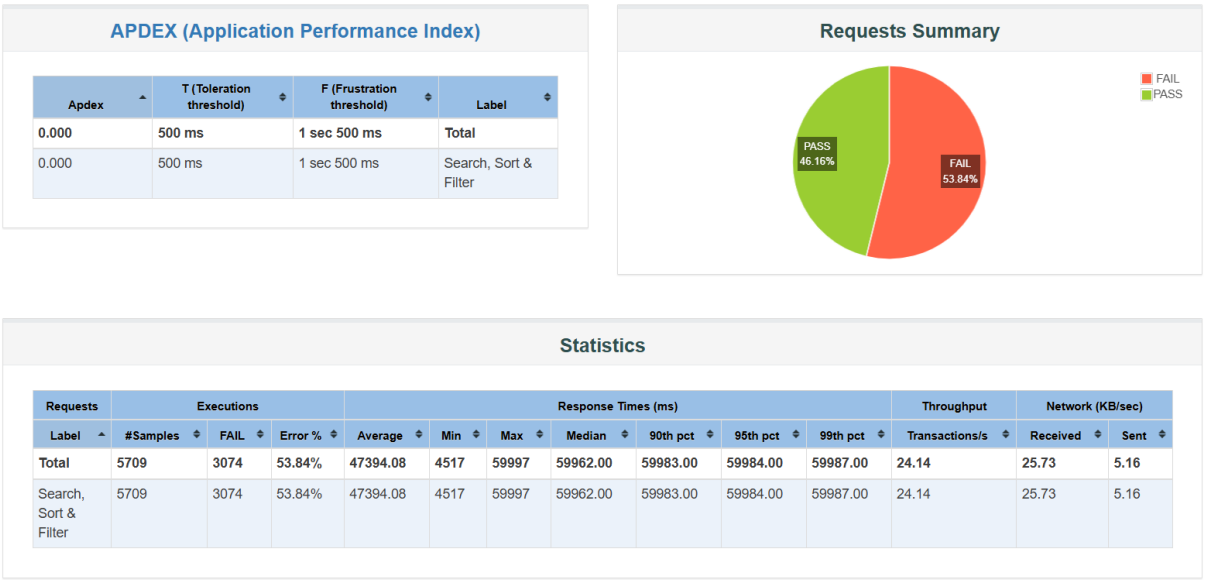


Figure 4: Stress Test Overall Statistics revealing a Critical Failure.

We can see that:

- **Error Rate (53.84%):** The system did not fail immediately. It processed a significant number of requests successfully (46%) before the load became unsustainable.
- **Max Response Time (59,997 ms):** This specific number is the "Smoking Gun." It confirms that the failures were Timeouts. The server accepted the connections, but as the queue grew longer than the server could process, requests older than 60 seconds were dropped, causing the error rate to spike towards the end of the test.
- **Throughput:** consistently capped at ~24 req/sec.

III. Visual & Graph Analysis

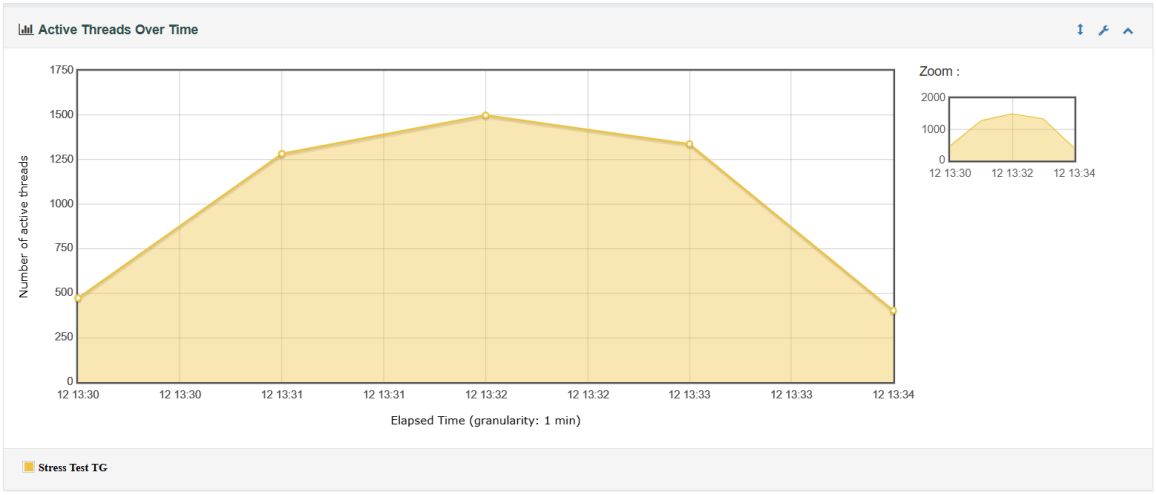


Figure 5: Active Threads successfully ramping up to 1500 and holding the load.

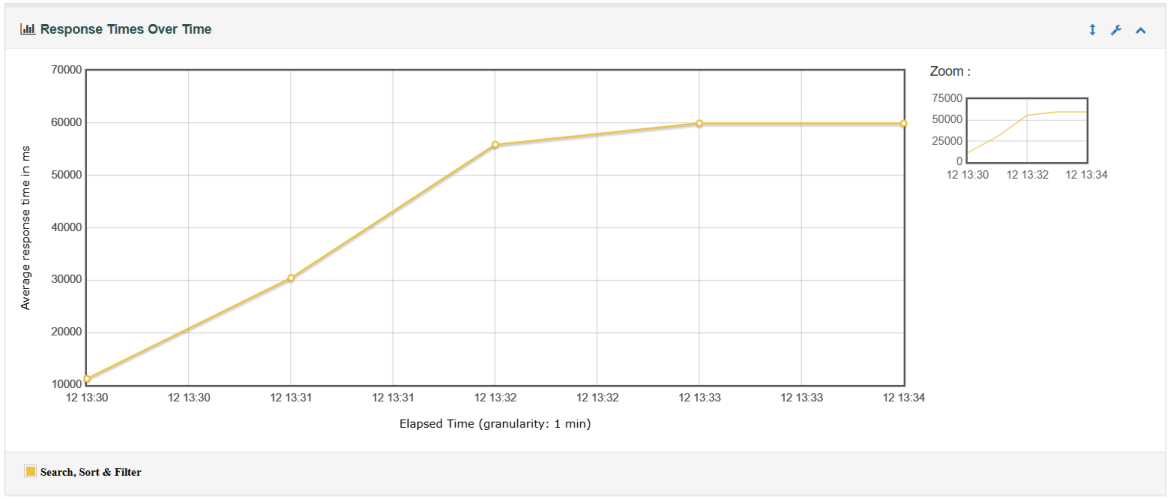


Figure 6: Response Time rising linearly as the backlog of requests accumulated.

We can observe from both graphs that**:**

- **Holding the Load:** As seen in the graph, the system successfully ramped up to 1500 concurrent threads and maintained this connection count for the configured "Hold" duration. The server did not crash or reset connections immediately.
- **The Failure Point:** Although connections were maintained, the system could not process requests fast enough, causing a backlog that led to timeouts during the hold phase.

IV. Expected and Actual Result

Metric	Expected Criteria	Actual Result	Status
Concurrency	Handle 1500 Users	1500 Users (Connected)	PASS
Error Rate	< 1.00%	53.84%	FAIL
Max Response Time	< 10,000 ms	59,997 ms (Timeout)	FAIL
Stability	Sustain load for duration	Failed during "Hold" phase	FAIL

V. Conclusion

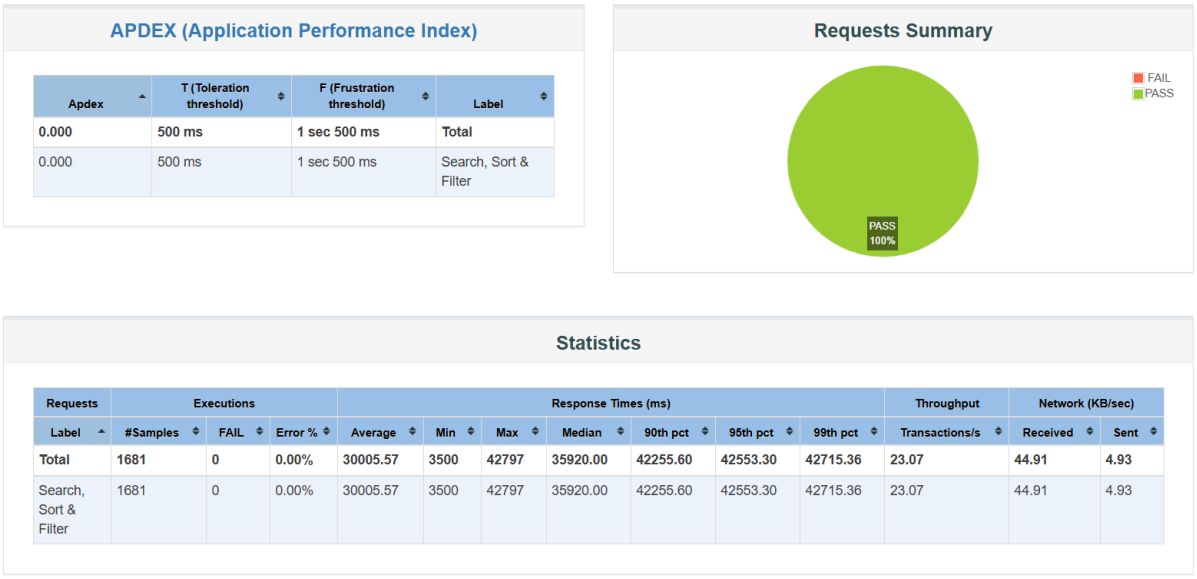
The system sustained 1500 concurrent connections but failed to handle the resulting throughput, with errors increasing once request queues exceeded the 60-second timeout limit.

C. Scenario 3: Spike Test

I. Objective

To evaluate the system's resilience when subjected to a sudden, instantaneous surge of 1000 users in 1 second.

II. Key Performance Metrics



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	Search, Sort & Filter

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	1681	0	0.00%	30005.57	3500	42797	35920.00	42255.60	42553.30	42715.36	23.07	44.91	4.93
Search, Sort & Filter	1681	0	0.00%	30005.57	3500	42797	35920.00	42255.60	42553.30	42715.36	23.07	44.91	4.93

Figure 7: Spike Test Overall Statistics showing no errors but severe latency.

We can see that:

- **Resilience:** Surprisingly, the system achieved a 0.00% Error Rate (Status: PASS). It did not crash immediately.
- **Lag:** The Average Response Time jumped to 30,005 ms, making the site unusable for real users during the spike.

III. Visual & Graph Analysis

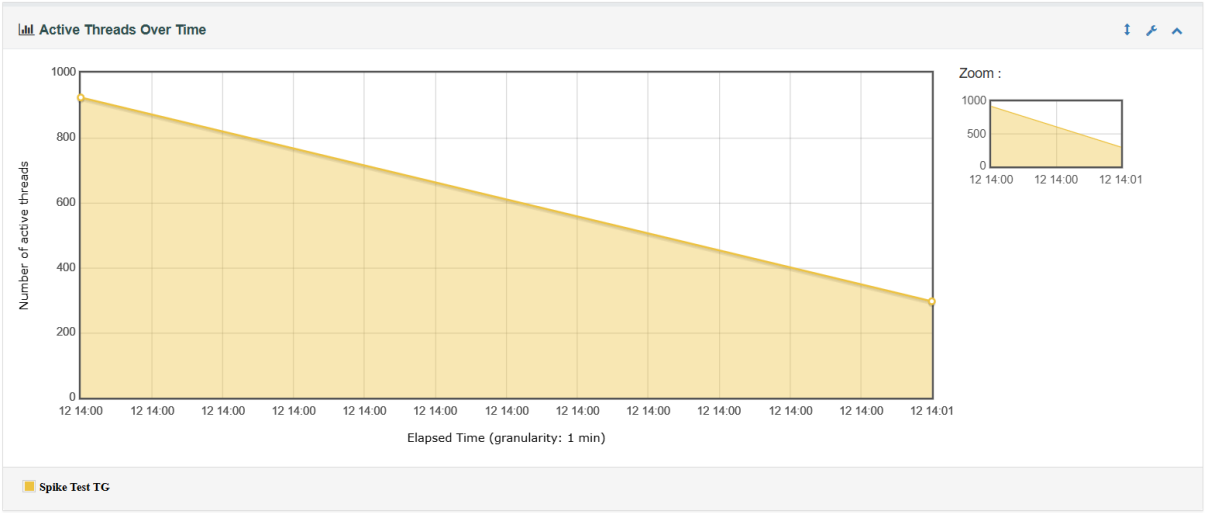


Figure 8: Active Threads decreasing as the test duration cut off unfinished requests.

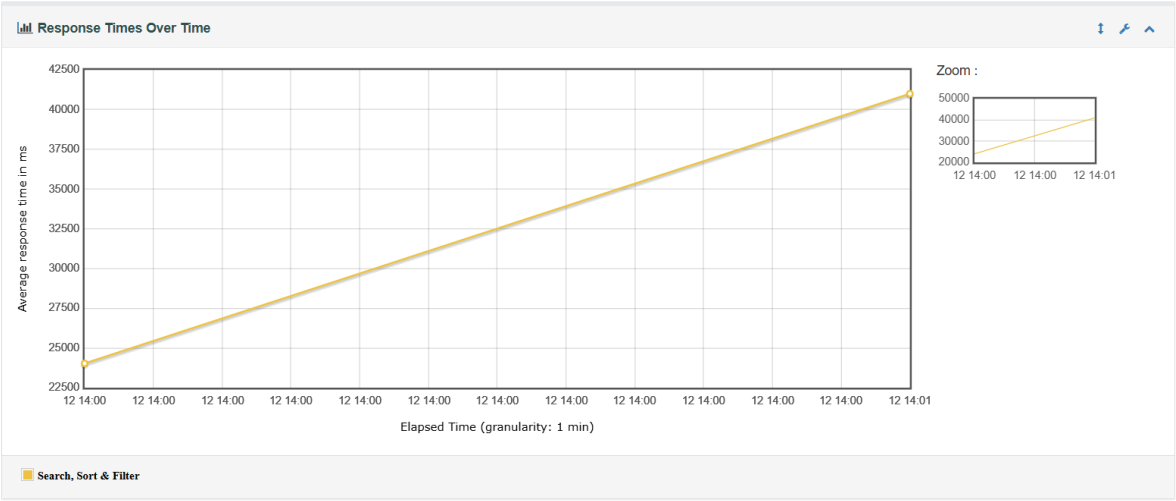


Figure 9: Response Time increasing linearly over the 30-second test.

We can observe from both graphs that**:**

- **The "Snowball" Effect:** The perfect diagonal line in the Response Time graph is critical evidence. It shows that the server never recovered from the initial shock. It spent the entire 30 seconds processing the backlog, with latency getting worse every second.
- **Recovery Failure:** If the system had recovered, the line in Figure 9 would have spiked and then dropped back down. The fact that it stayed high proves the server was overwhelmed.

IV. Expected and Actual Result

Metric	Expected Criteria	Actual Result	Status
System Crash	No Crash (0% Errors)	0.00% Errors	PASS
Recovery Time	Return to baseline in < 10s	Did not recover (>30s)	FAIL
Avg Response Time	< 5,000 ms during spike	30,005 ms	FAIL

V. Conclusion

The system demonstrated *High Technical Resilience* (no crash) but *Poor Recovery*. It successfully queued the spike but lacked the CPU power to process it in time.

D. Determination of Endurance Threshold

Based on the combined results of the Load, Stress, and Spike tests, the system's endurance threshold on my personal machine is determined to be *approximately 200 concurrent users*.

- **Evidence of Stability:** At this level, as demonstrated in the Load Test, the system maintained a 0.00% error rate, stable throughput (~24 requests per second), and consistent response times without progressive degradation. This indicates that the system can reliably sustain this load over time.
- **Evidence of Failure:** Beyond 200 concurrent users, the Stress Test revealed severe performance degradation. Although the system was able to accept up to 1500 concurrent connections, response times increased sharply, request queues accumulated, and timeout-related errors occurred once the system's processing capacity was exceeded.
- **Confirmation:** Similarly, the Spike Test showed that a sudden surge of 1000 users did not cause an immediate crash but resulted in extreme latency and lack of recovery, confirming that such loads are not sustainable.

Conclusion: Therefore, *200 concurrent users* represents the maximum sustainable workload for this system under the constraints of my personal hardware configuration. Higher levels of concurrency exceed the system's endurance capacity and lead to unacceptable latency and reliability issues.

6. Bug Report

Bug 1: Stress Test Failure (Sustained Load)

- **Description:** The system successfully establishes 1500 concurrent connections but cannot process the request volume over time. A backlog accumulates during the "Hold" phase, eventually causing requests to hit the 60-second server timeout.
- **Evidence:**
  - Active Threads graph shows stable 1500 users (System didn't crash).
  - Error rate spiked to 53.84% only towards the end of the test.
  - Max Response Time hit exactly 59,997 ms (Timeout).

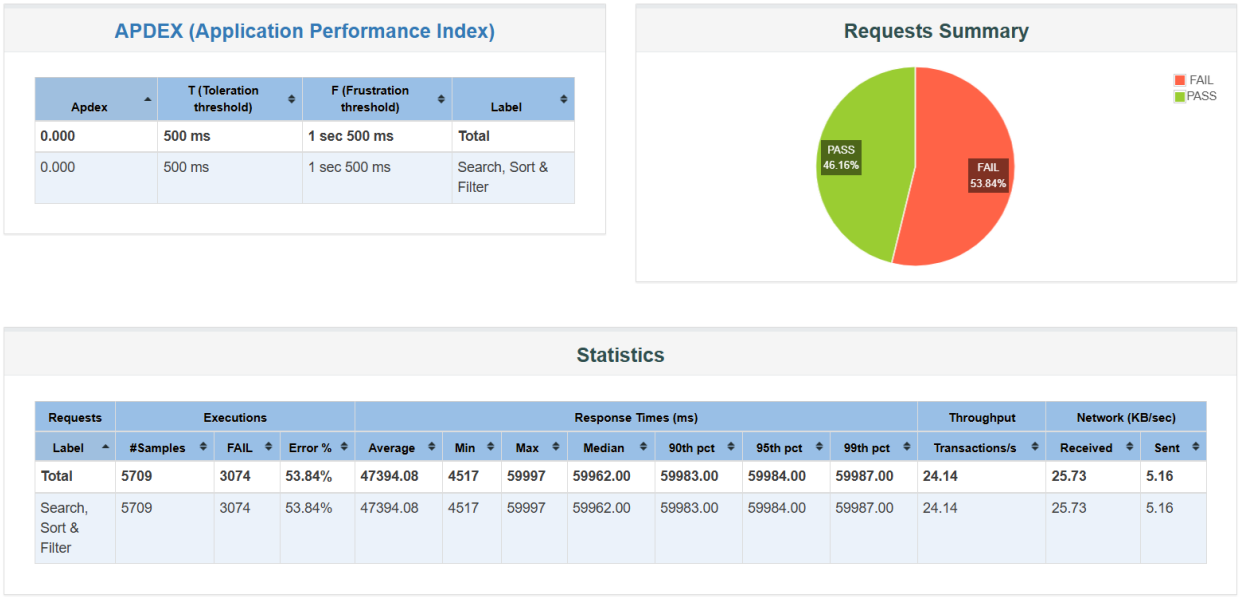


Figure 10: Overall results show high error rates and long response times under load.

Bug 2: Spike Test Latency (No Recovery)

- **Description:** When subjected to a sudden burst of 1000 users, response times degrade linearly and do not return to baseline. The server spends the entire test duration clearing the initial queue.
- **Evidence:** Response Time Graph shows a perfect diagonal rise from 24s to 41s without leveling off.

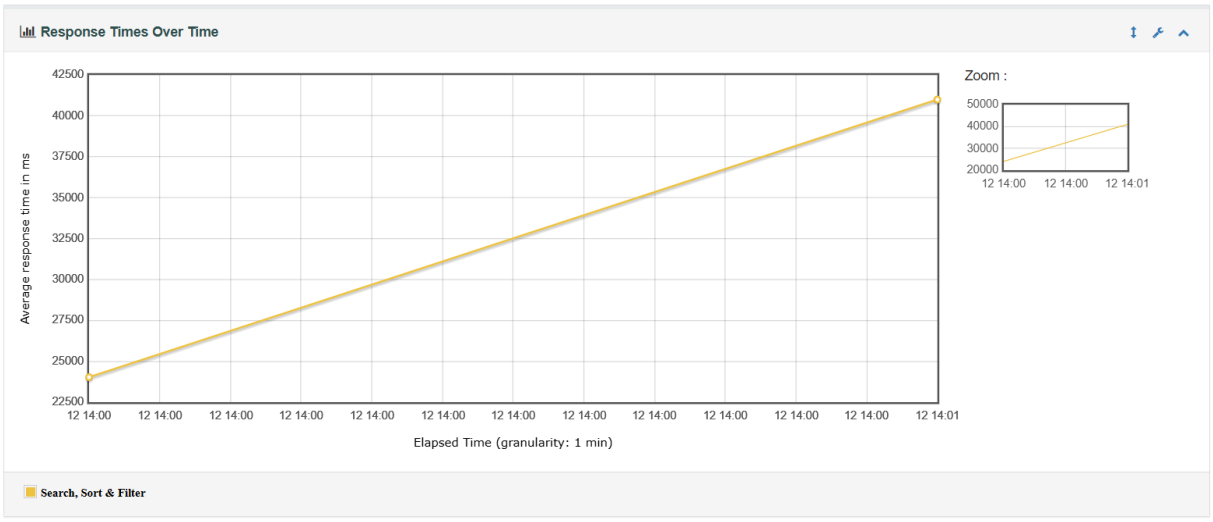
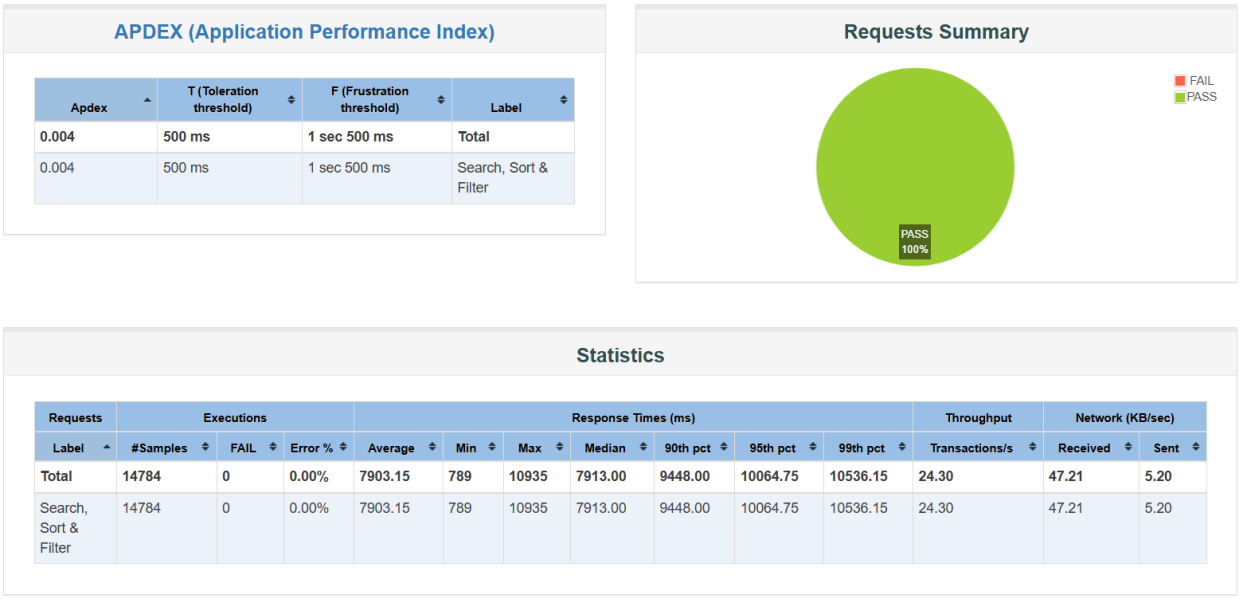


Figure 11: Response times increase steadily from 24s to 41s with no sign of recovery.

Bug 3: High Baseline Latency

- **Description:** Even under low, stable load (200 users), the application averages ~8 seconds per request, which is poor for user experience.
- **Evidence:** Load Test Analysis shows consistent 7,903 ms average response time with 0% errors.



APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.004	500 ms	1 sec 500 ms	Total
0.004	500 ms	1 sec 500 ms	Search, Sort & Filter

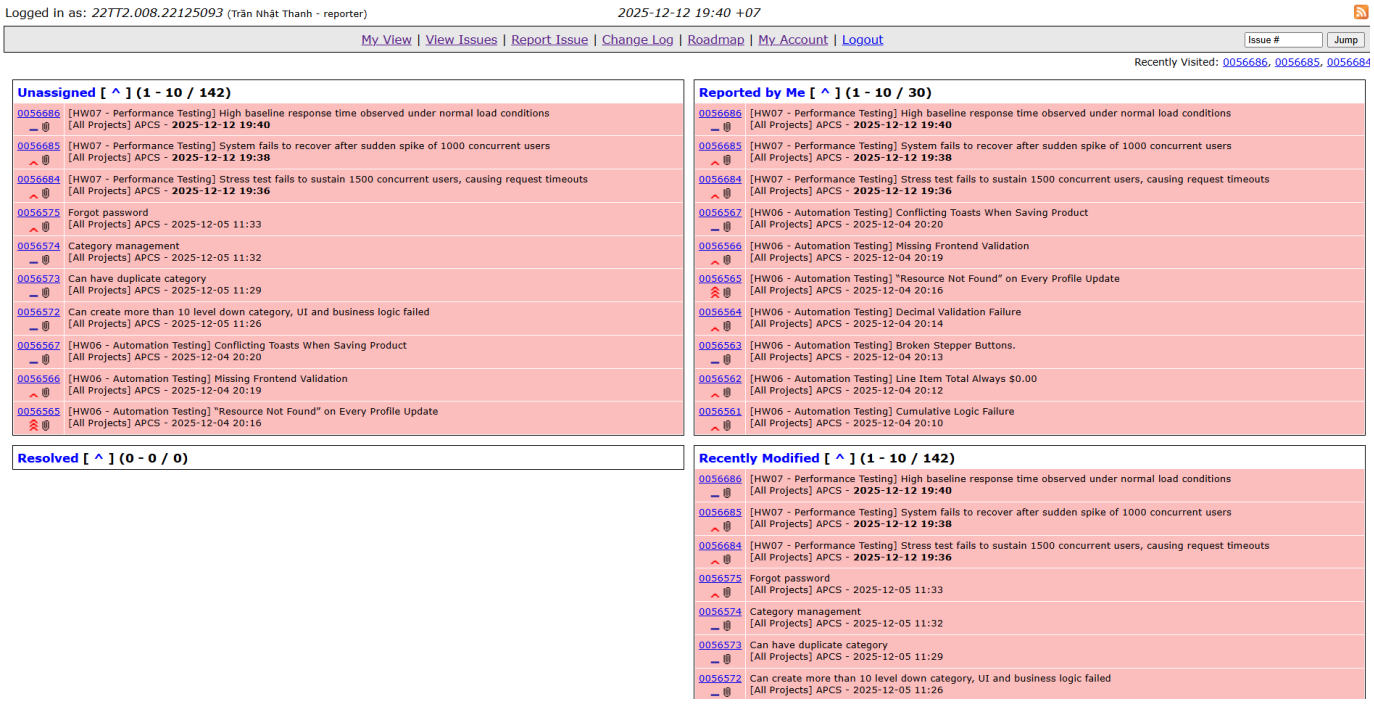
Requests Summary

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	14784	0	0.00%	7903.15	789	10935	7913.00	9448.00	10064.75	10536.15	24.30	47.21	5.20
Search, Sort & Filter	14784	0	0.00%	7903.15	789	10935	7913.00	9448.00	10064.75	10536.15	24.30	47.21	5.20

Figure 12: Average response time remains around 7,903 ms with no errors observed.

The updated bugs on Mantis



Logged in as: *22TT2.008.22125093* (Trần Nhật Thanh - reporter) *2025-12-12 19:40 +07*

My View | View Issues | Report Issue | Change Log | Roadmap | My Account | Logout — Issue # — Jump

Recently Visited: 0056686, 0056685, 0056684

Unassigned [^] (1 - 10 / 142)

0056686	[HW07 - Performance Testing] High baseline response time observed under normal load conditions [All Projects] APCS - **2025-12-12 19:40**
0056685	[HW07 - Performance Testing] System fails to recover after sudden spike of 1000 concurrent users [All Projects] APCS - **2025-12-12 19:38**
0056684	[HW07 - Performance Testing] Stress test fails to sustain 1500 concurrent users, causing request timeouts [All Projects] APCS - **2025-12-12 19:36**
0056575	Forgot password [All Projects] APCS - 2025-12-05 11:33
0056574	Category management [All Projects] APCS - 2025-12-05 11:32
0056573	Can have duplicate category [All Projects] APCS - 2025-12-05 11:29
0056572	Can create more than 10 level down category, UI and business logic failed [All Projects] APCS - 2025-12-05 11:26
0056567	[HW06 - Automation Testing] Conflicting Toasts When Saving Product [All Projects] APCS - 2025-12-04 20:20
0056566	[HW06 - Automation Testing] Missing Frontend Validation [All Projects] APCS - 2025-12-04 20:19
0056565	[HW06 - Automation Testing] "Resource Not Found" on Every Profile Update [All Projects] APCS - 2025-12-04 20:16

Reported by Me [^] (1 - 10 / 30)

0056686	[HW07 - Performance Testing] High baseline response time observed under normal load conditions [All Projects] APCS - **2025-12-12 19:40**
0056685	[HW07 - Performance Testing] System fails to recover after sudden spike of 1000 concurrent users [All Projects] APCS - **2025-12-12 19:38**
0056684	[HW07 - Performance Testing] Stress test fails to sustain 1500 concurrent users, causing request timeouts [All Projects] APCS - **2025-12-12 19:36**
0056567	[HW06 - Automation Testing] Conflicting Toasts When Saving Product [All Projects] APCS - 2025-12-04 20:20
0056566	[HW06 - Automation Testing] Missing Frontend Validation [All Projects] APCS - 2025-12-04 20:19
0056565	[HW06 - Automation Testing] "Resource Not Found" on Every Profile Update [All Projects] APCS - 2025-12-04 20:16
0056564	[HW06 - Automation Testing] Decimal Validation Failure [All Projects] APCS - 2025-12-04 20:14
0056563	[HW06 - Automation Testing] Broken Stepper Buttons. [All Projects] APCS - 2025-12-04 20:13
0056562	[HW06 - Automation Testing] Line Item Total Always $0.00 [All Projects] APCS - 2025-12-04 20:12
0056561	[HW06 - Automation Testing] Cumulative Logic Failure [All Projects] APCS - 2025-12-04 20:10

Resolved [^] (0 - 0 / 0)

Recently Modified [^] (1 - 10 / 142)

0056686	[HW07 - Performance Testing] High baseline response time observed under normal load conditions [All Projects] APCS - **2025-12-12 19:40**
0056685	[HW07 - Performance Testing] System fails to recover after sudden spike of 1000 concurrent users [All Projects] APCS - **2025-12-12 19:38**
0056684	[HW07 - Performance Testing] Stress test fails to sustain 1500 concurrent users, causing request timeouts [All Projects] APCS - **2025-12-12 19:36**
0056575	Forgot password [All Projects] APCS - 2025-12-05 11:33
0056574	Category management [All Projects] APCS - 2025-12-05 11:32
0056573	Can have duplicate category [All Projects] APCS - 2025-12-05 11:29
0056572	Can create more than 10 level down category, UI and business logic failed [All Projects] APCS - 2025-12-05 11:26

7. Appendix

- **Demonstration Video:** Please check this playlist for the demo videos: HW07 – Performance Testing.
- **JMeter Script:** Please check the folder Script for the 3 jmx files.

Self – Assessment

Criteria	Outcomes	Grade	Self-Assessed Grade
1	**Load Testing**	**40**	**40**
	1.1 Report	15	15
	1.2 Script	10	10
	1.3 Data	5	5
	1.4 Video	10	10
2	**Stress Testing**	**30**	**30**
	2.1 Report	10	10
	2.2 Script	5	5
	2.3 Data	5	5
	2.4 Video	10	10
3	**Spike Testing**	**30**	**30**
	3.1 Report	10	10
	3.2 Script	5	5
	3.3 Data	5	5
	3.4 Video	10	10
	Total	**100**	**100**