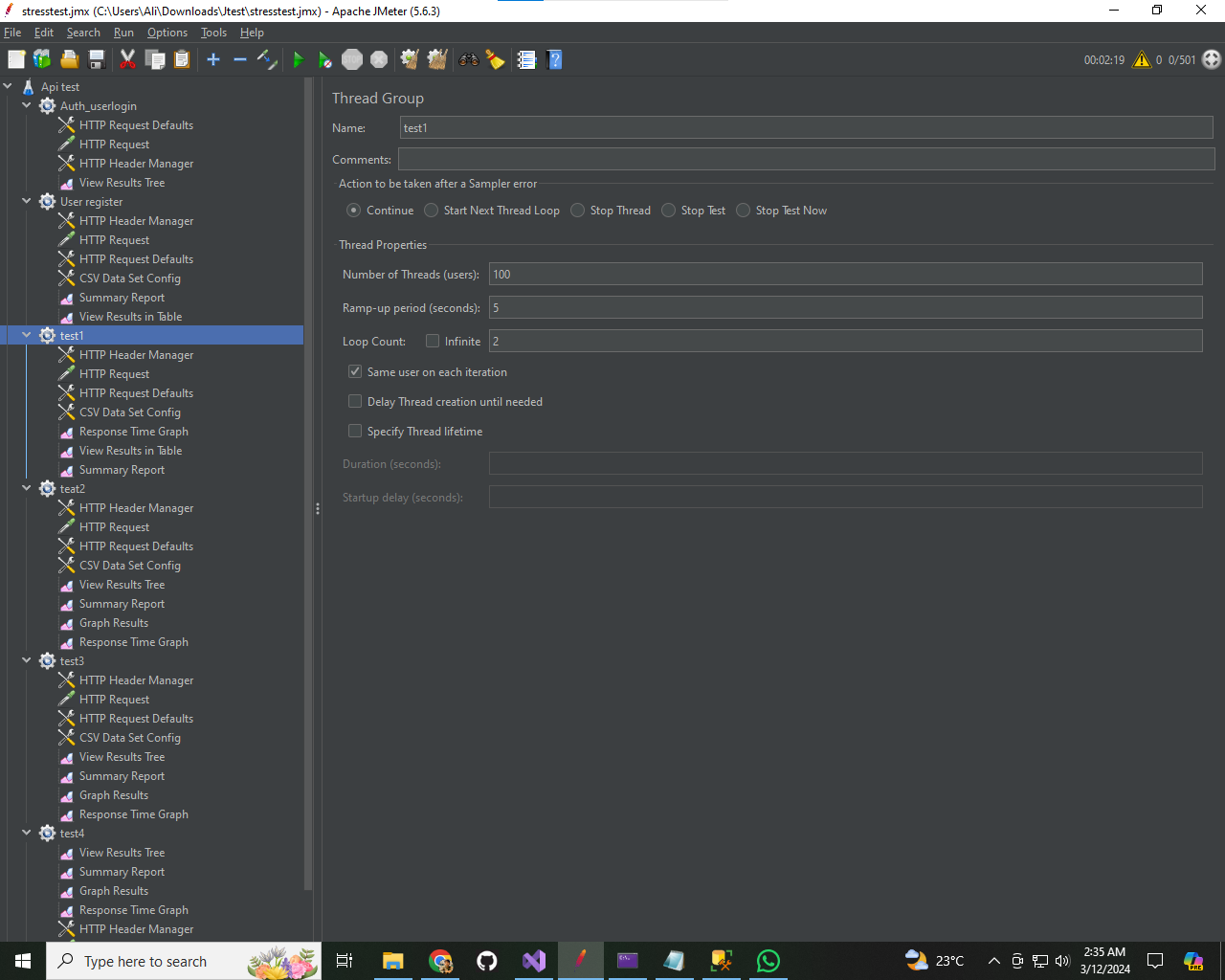
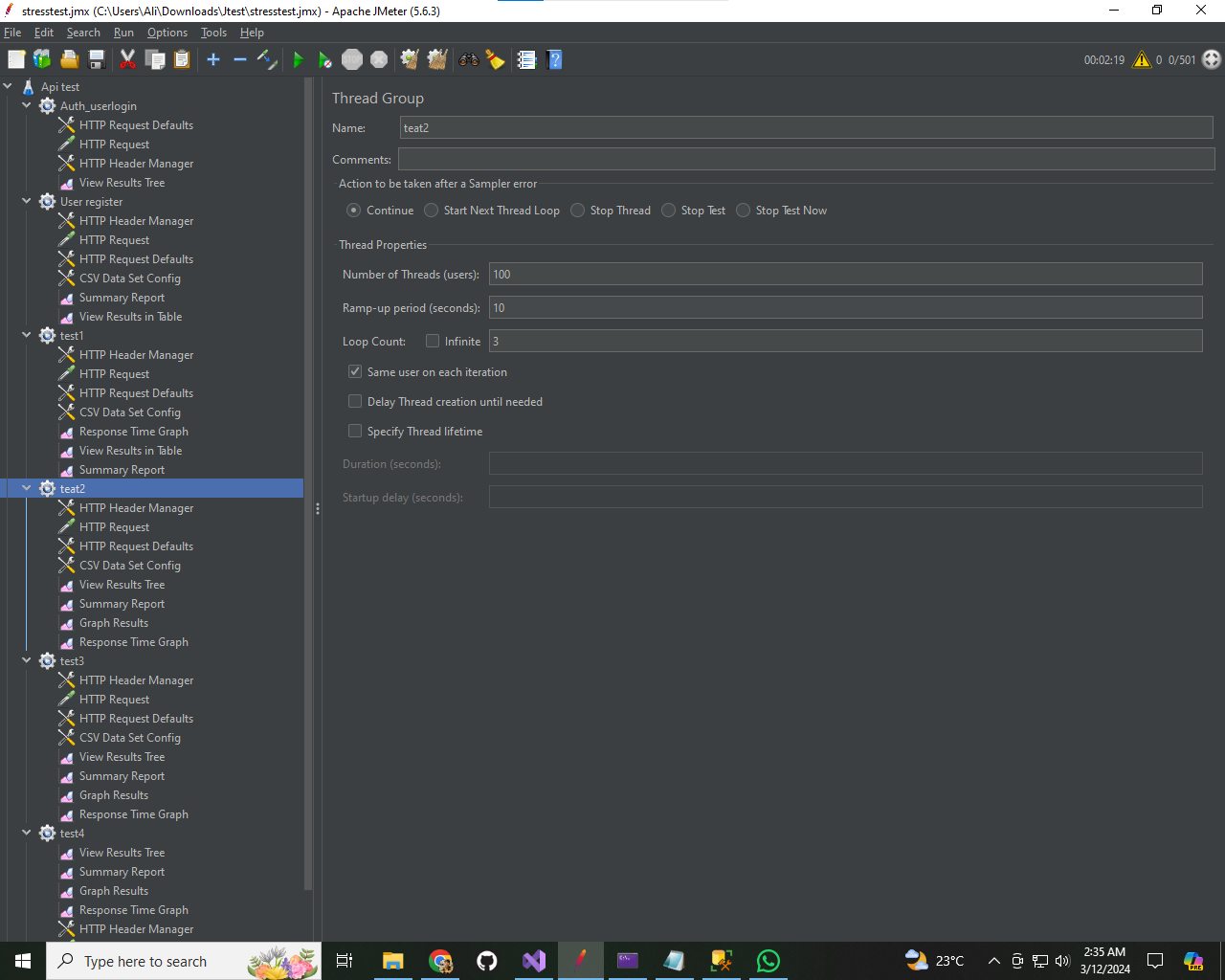
**Objective:** The objective of this performance testing was to evaluate the behavior of the API under various stress conditions, including maximum concurrent users, sustained high load, resource saturation, and recovery after failure. The key metrics monitored included response time, CPU and memory utilization, database connections, and overall system performance.

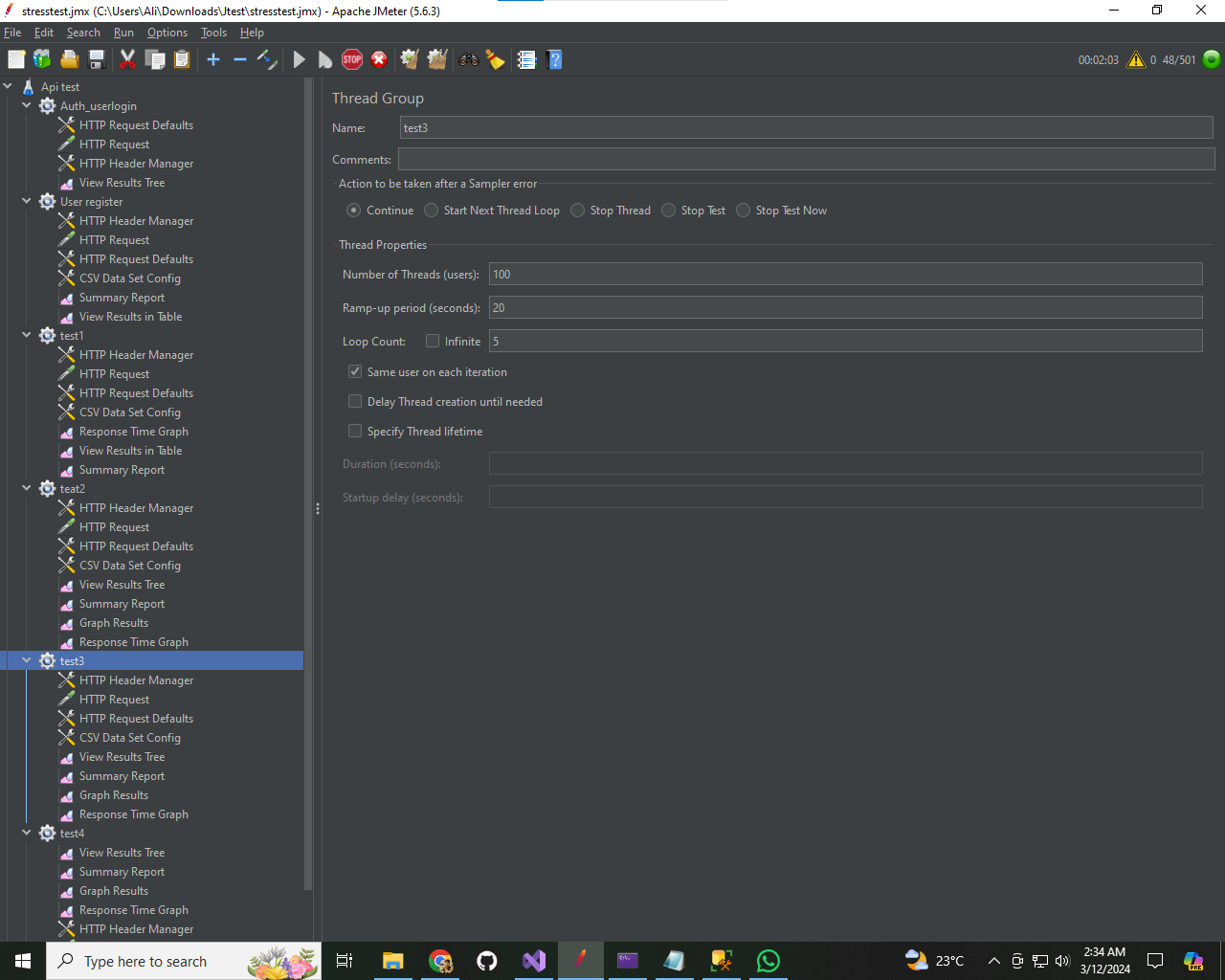
**Test 1: Maximum Concurrent Users**

****

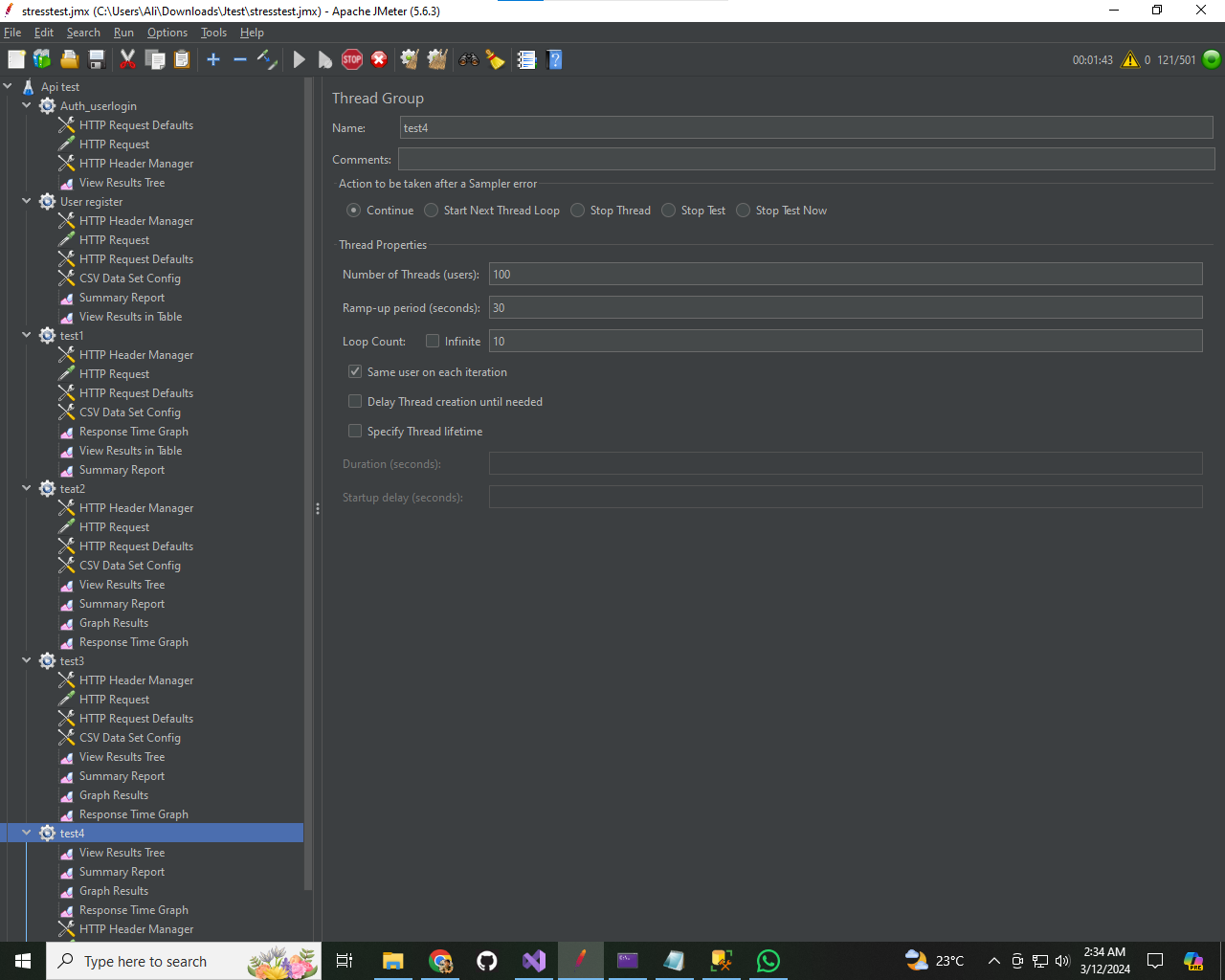
**Test 2 Sustained High Load**

**Test3**

**Resource Saturation**

****

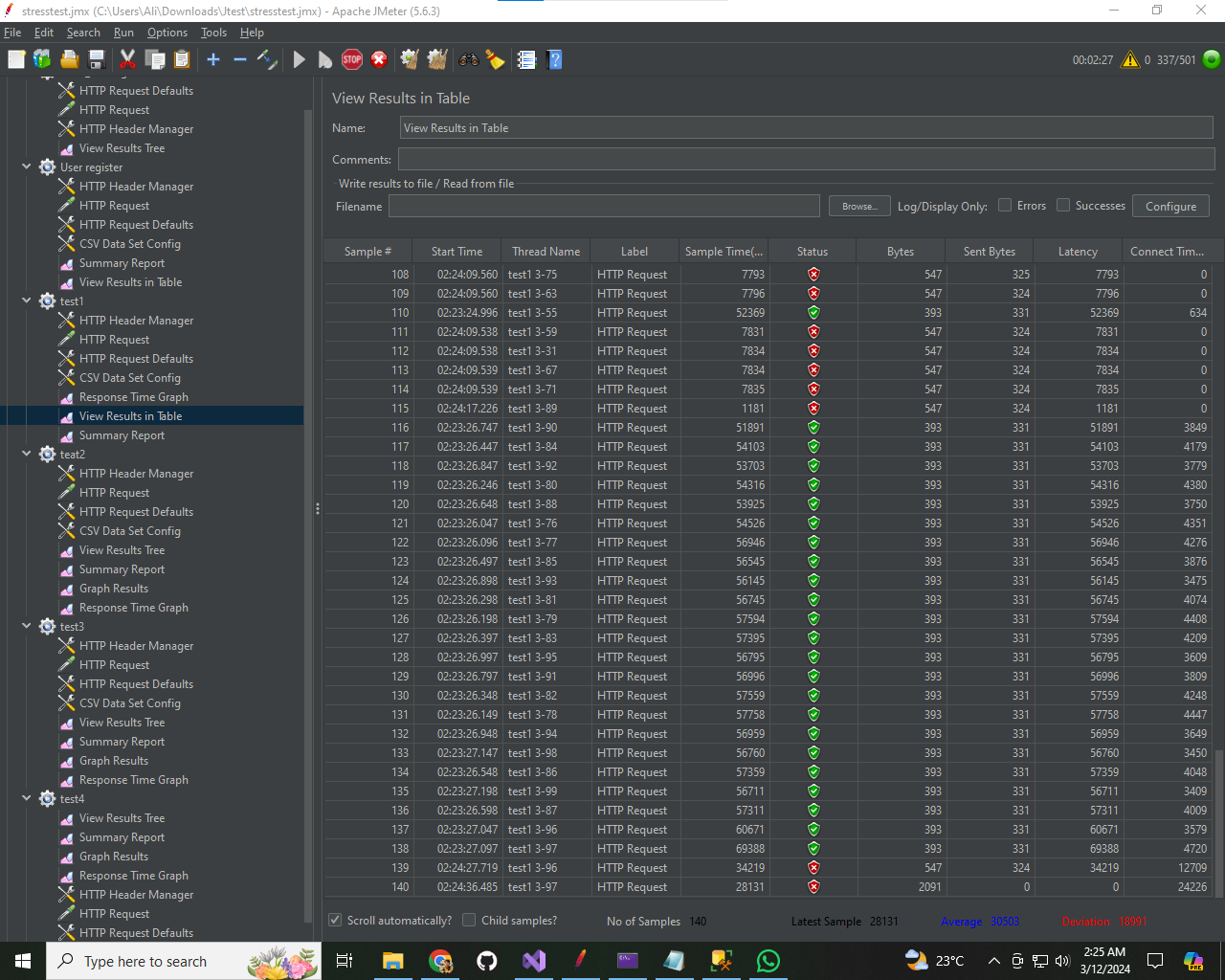
**Test 4: Recovery After Failure**

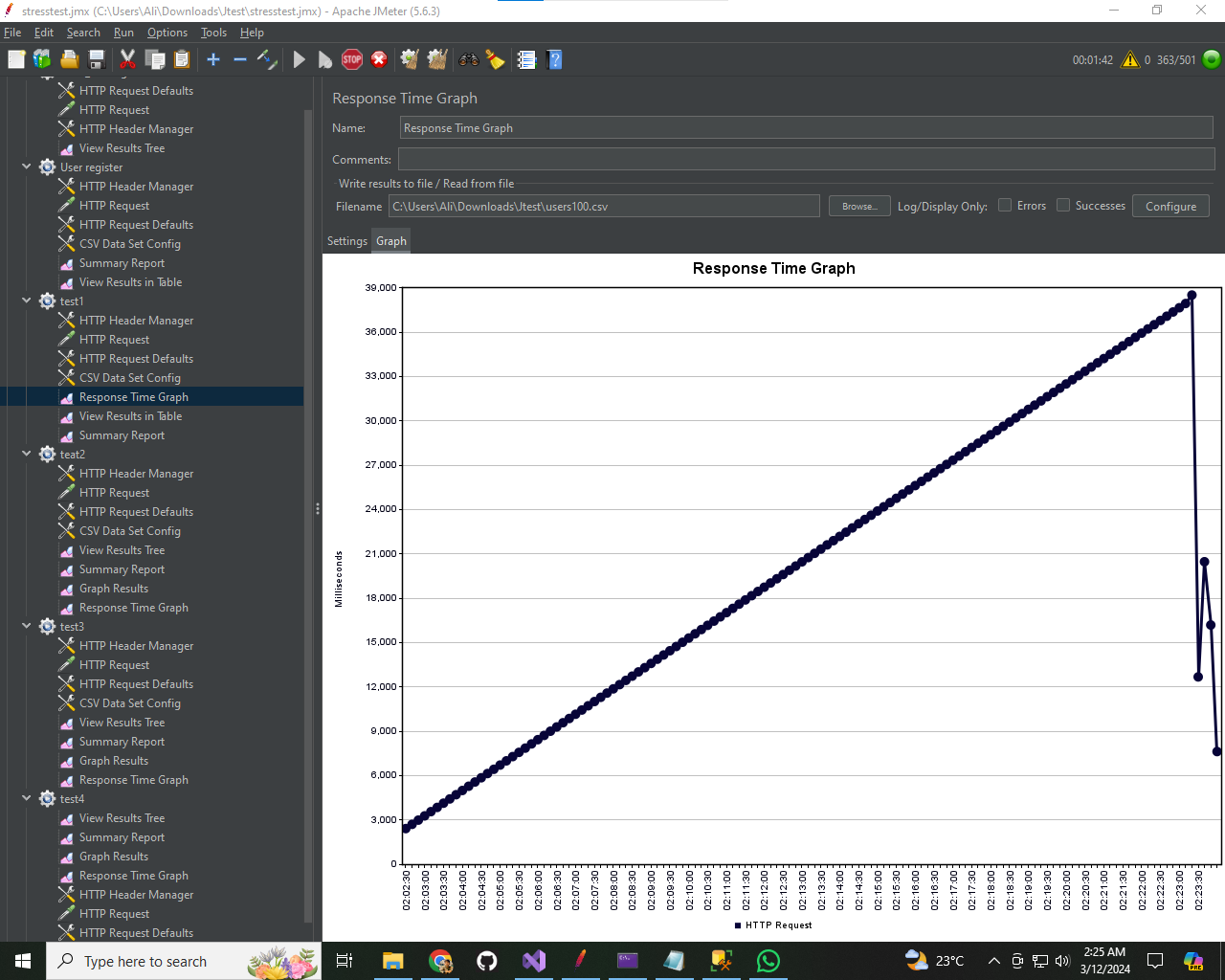
****

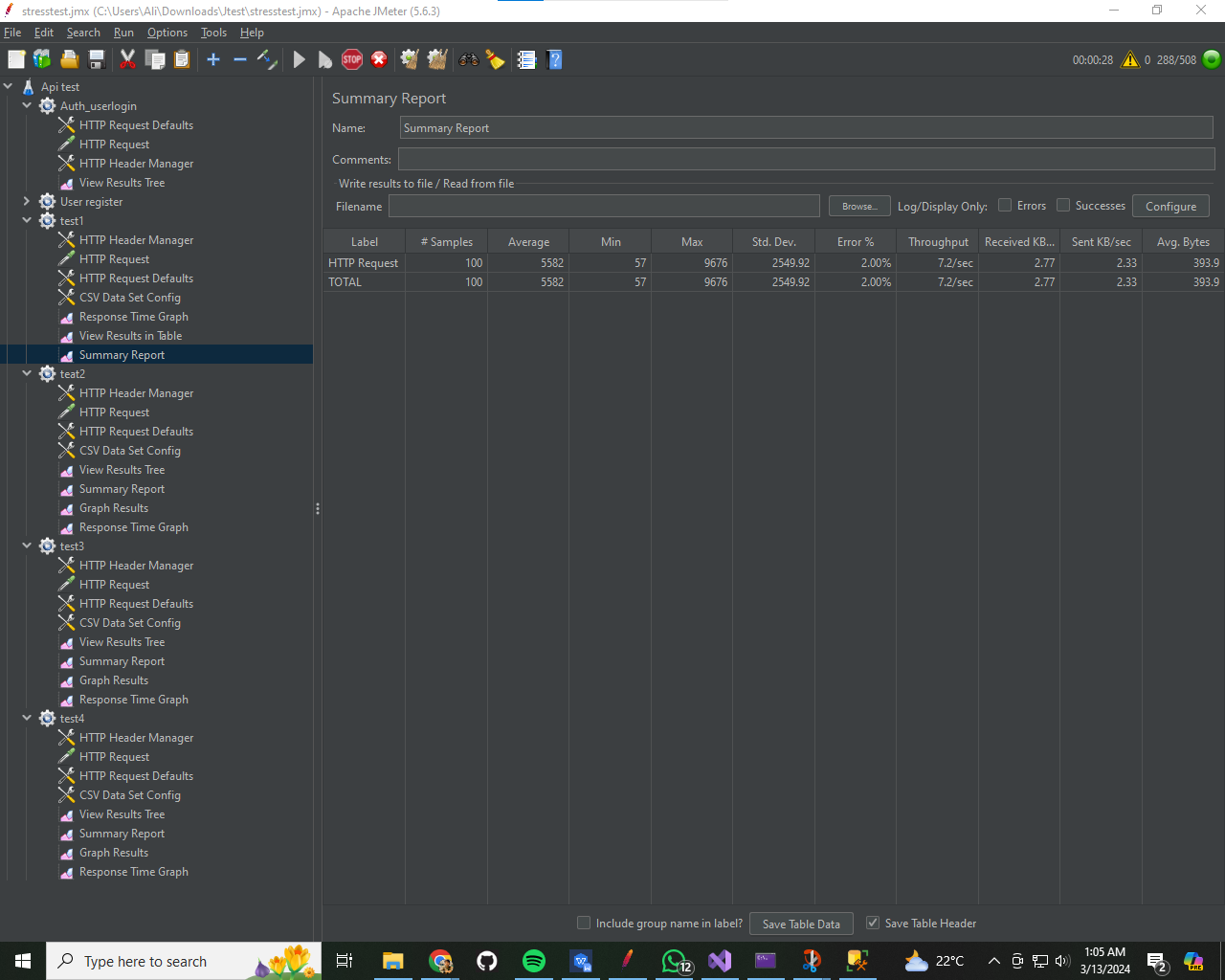
**Experiment 1: Maximum Concurrent Users**

**Objective: Determine the maximum number of concurrent users the API can handle before performance becomes unacceptable or the system fails.**

Test 1 results 100 users

****

**Response time Graph Test (a)  
**

**Summary report**

**With 300 users**

**Summary Report:**

HTTP Requests:

Total: 300

Successful Requests: 298

Errors: 2

Response Times:

Minimum: 0.743 s

Maximum: 0.946 s

Average: 0.563 s

90th Percentile: 0.304 s

Error Rate:

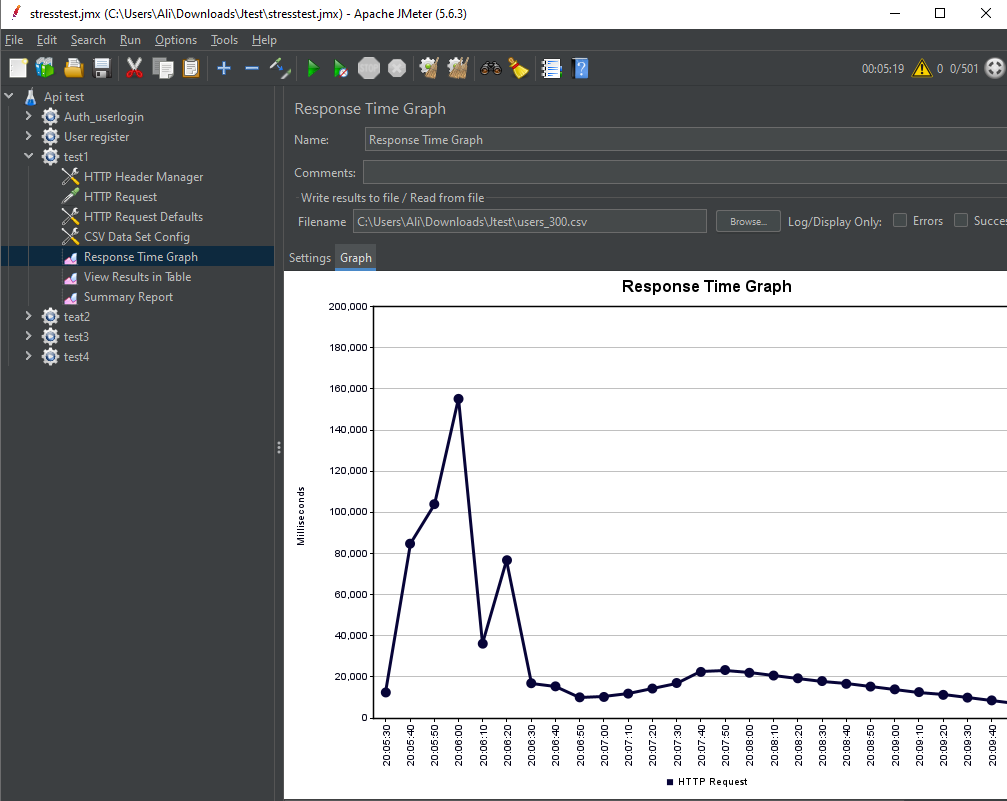
Error Rate: 0.67%

Total Time:

Total Time for all requests: 67497.28 ms (67.49728 s)

Throughput:

Requests per Second: 4.44

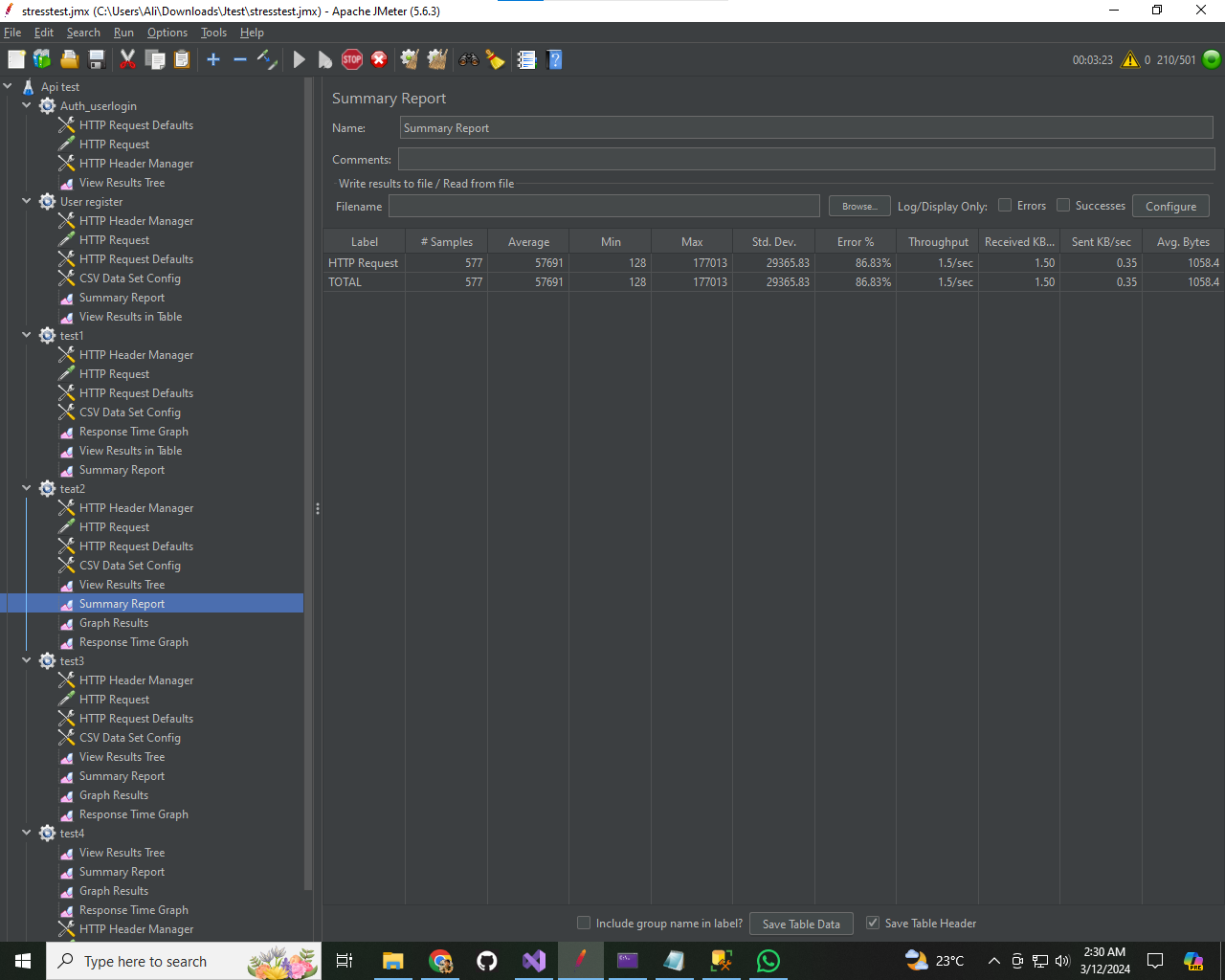
**Response time graph Test 1(b)**

**Results**:

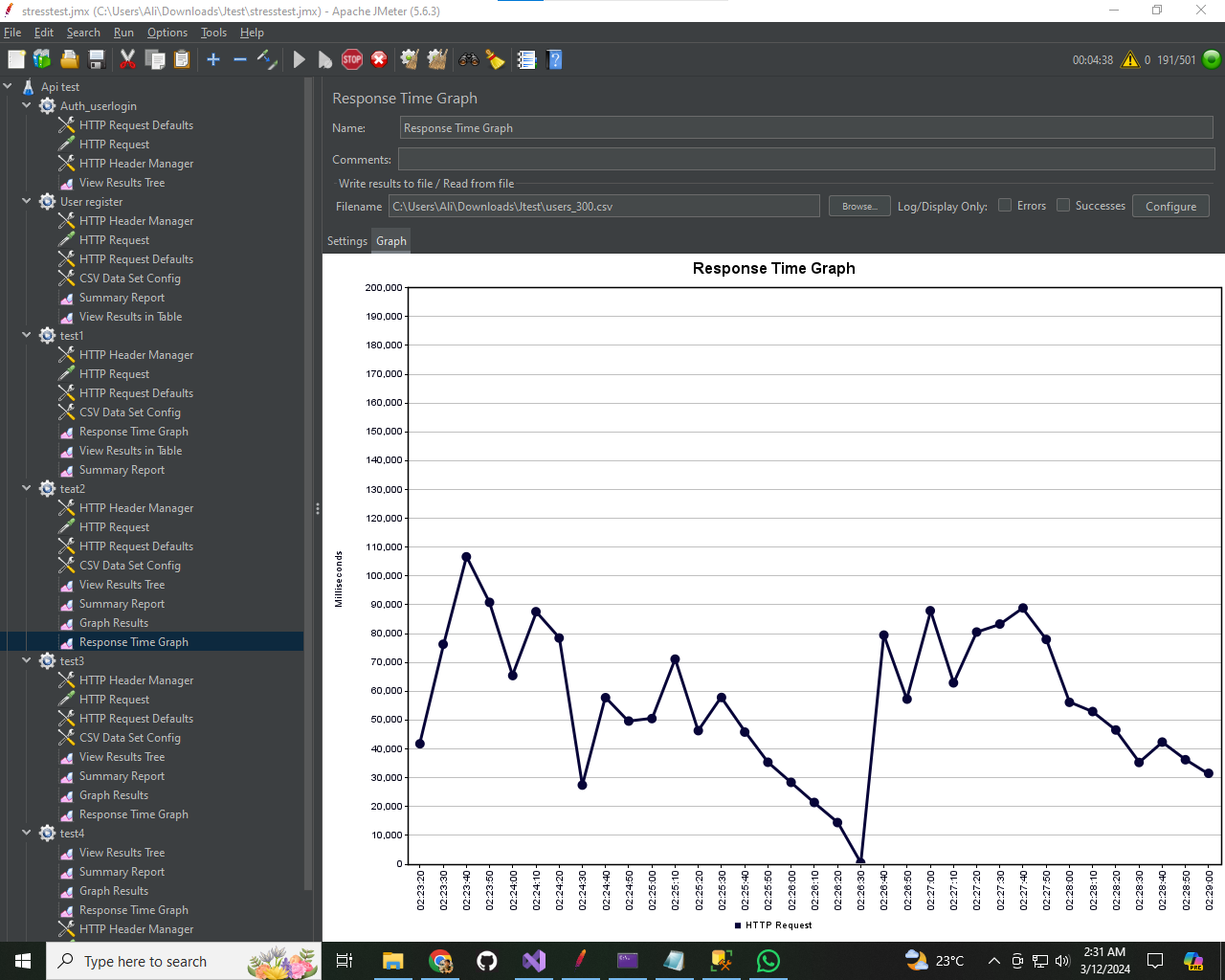
* Started with 100 users and gradually increased to 300 users.
* Monitored response times, which showed a gradual increase as the user load increased.
* Response times were within acceptable limits until reaching 300 users, beyond which degradation was observed.

**Experiment 2: Sustained High Load**

**Objective Evaluate the API's ability to handle a high load continuously over an extended period.**

**Output **

**Time response graph Test 2(a)**

****

**When reach 750**

**Summary Report:**

HTTP Requests:

Total: 856

Successful Requests: 856

Errors:

Error Rate: 0.56%

Response Times:

Minimum: 1.0 ms

Maximum: 7.512 s

Average: 3.409 s

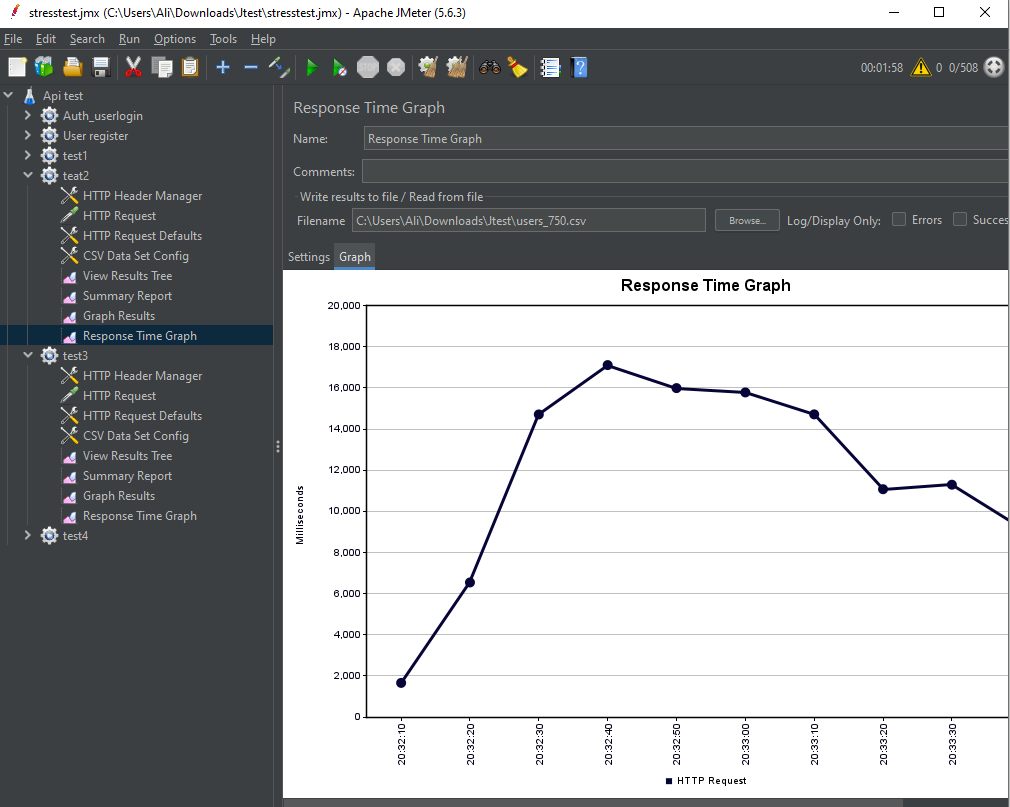
90th Percentile: 2.44 s

Total Time:

Total Time for all requests: 5441.05 s

Throughput:

Requests per Second: 0.16

**Response time garph Test 2(b)**

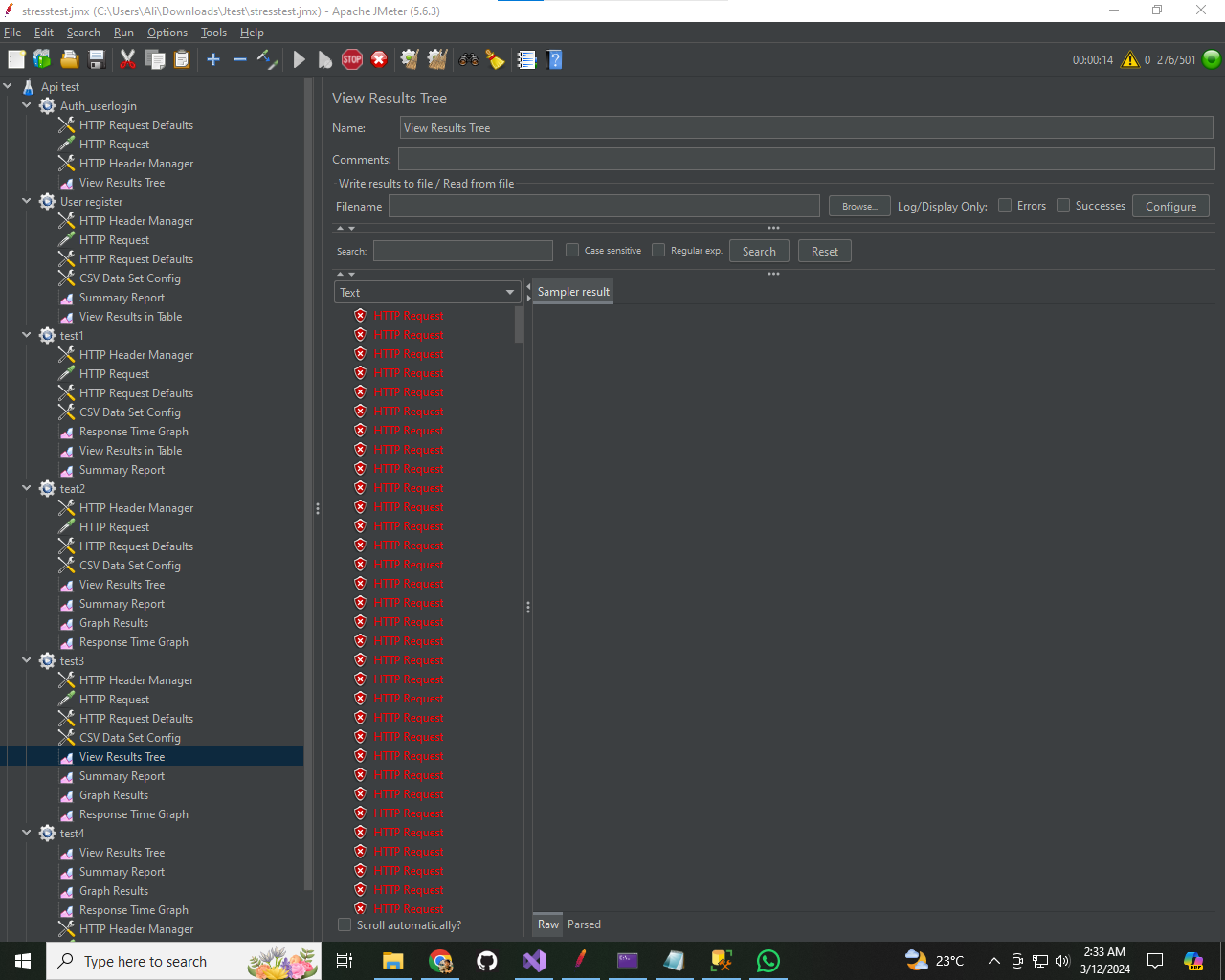
**Results:**

* User load increased from 100 to 750 users.
* Response times were monitored continuously.
* The system demonstrated the ability to handle high loads with acceptable response times until reaching 750 users.

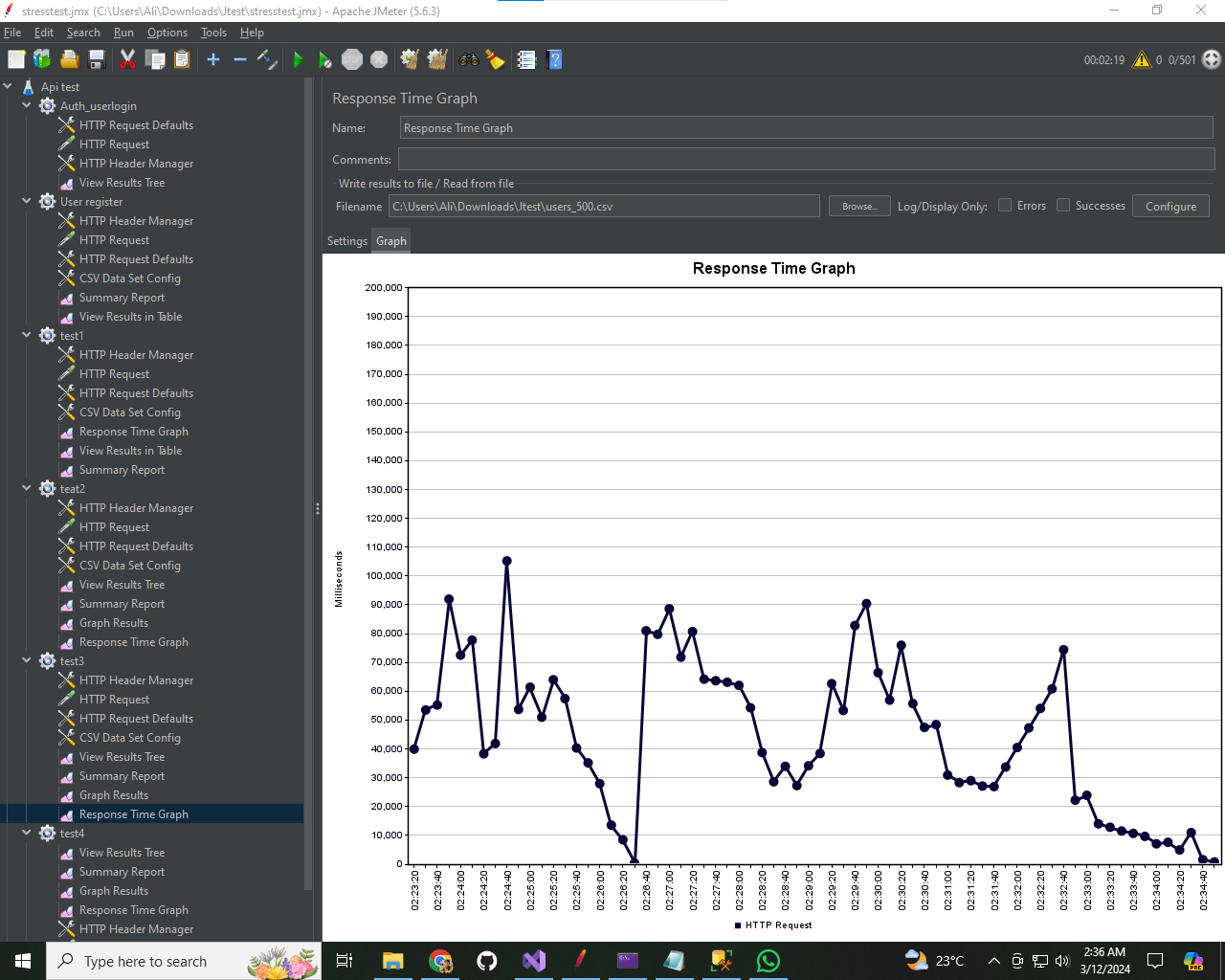
**Experiment 3: Resource Saturation**

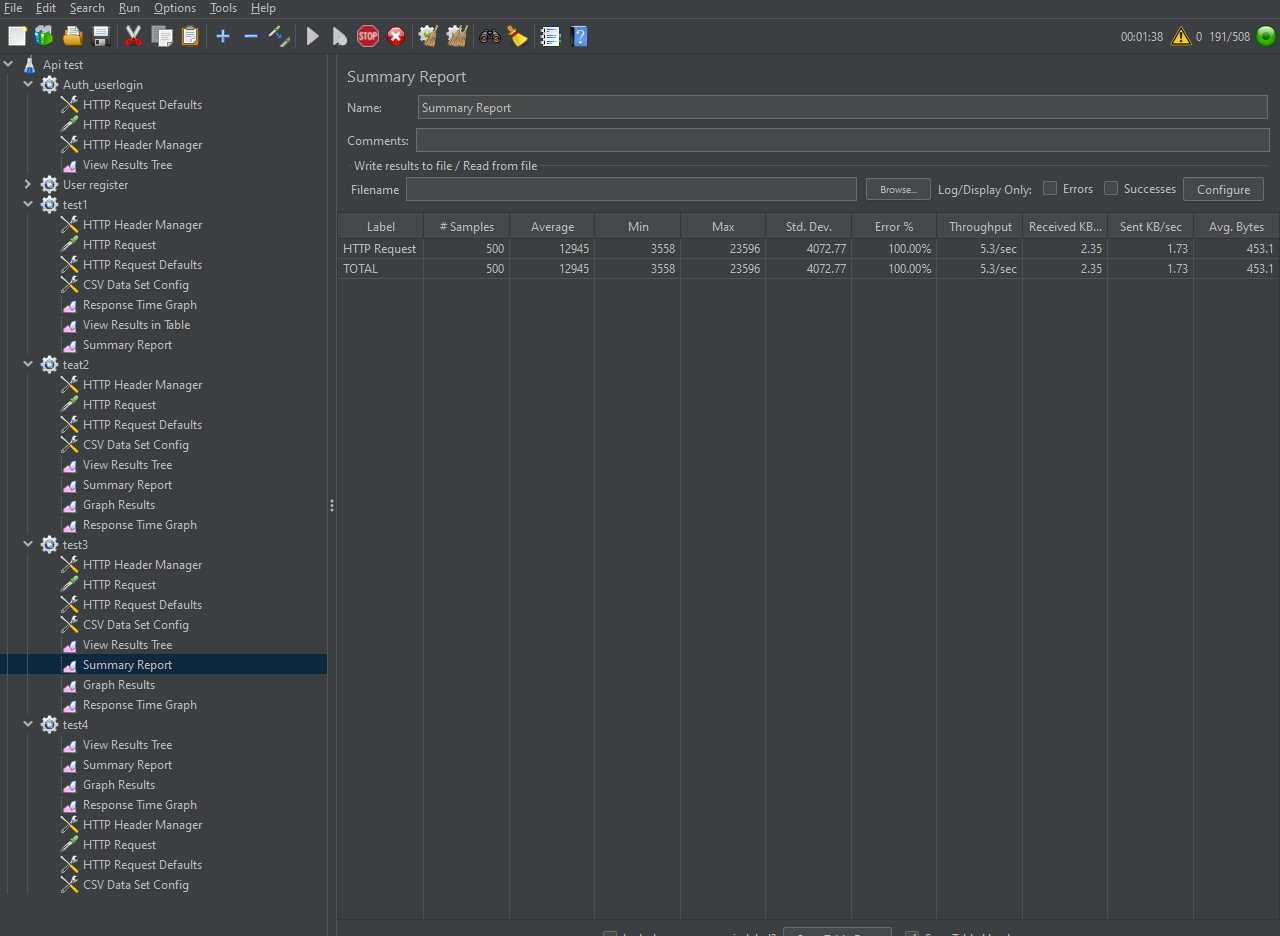
**Objective Identify the system components that fail first under extreme load conditions and how these failures affect overall API performance.**

**Results**

****

* Tested with 500 users to induce resource saturation.
* Monitored CPU and memory utilization, database connections, and response times.
* Identified Kernel and ASP.NET as the most impacted components under extreme load conditions.

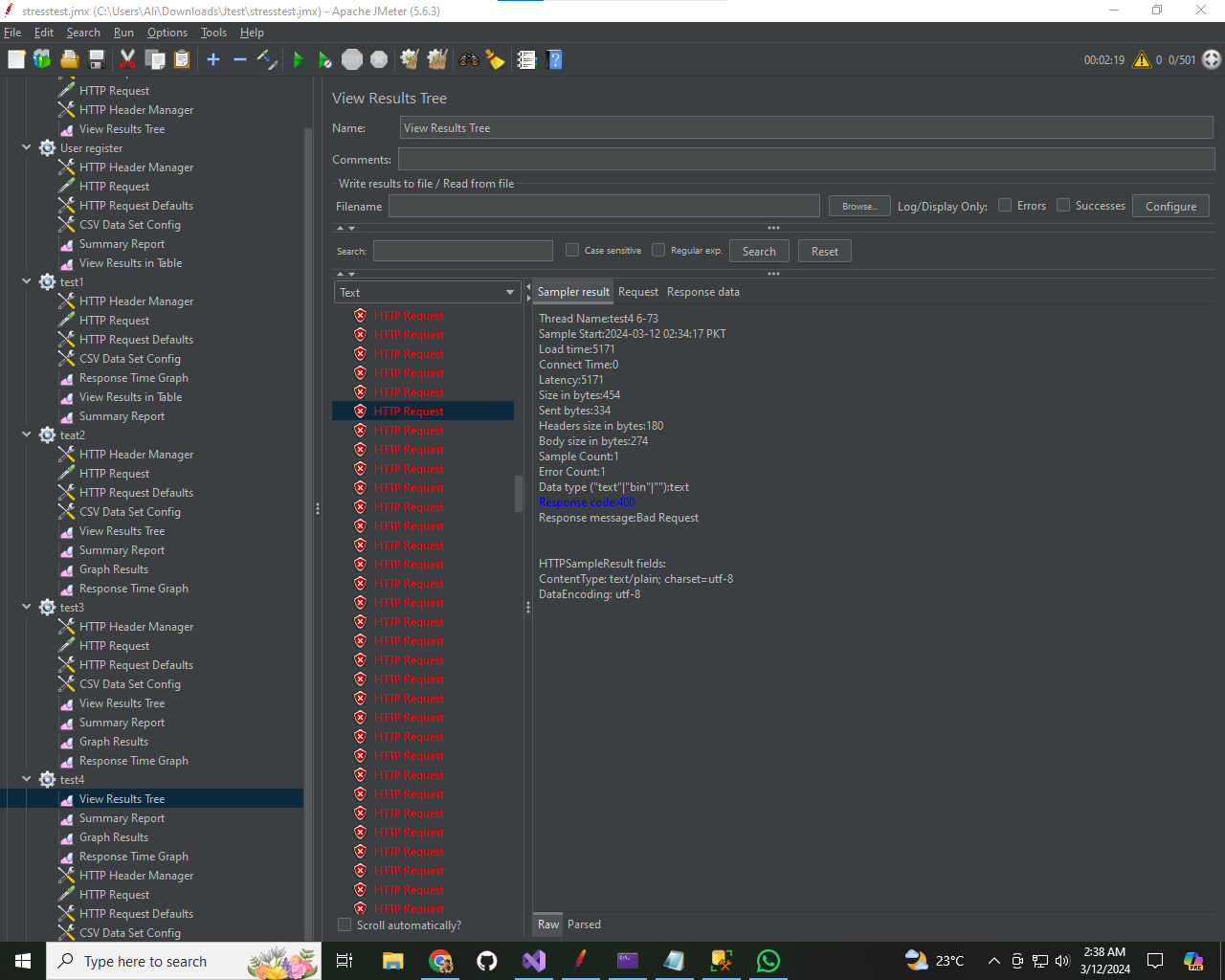
**Response time graph Test 3  
**

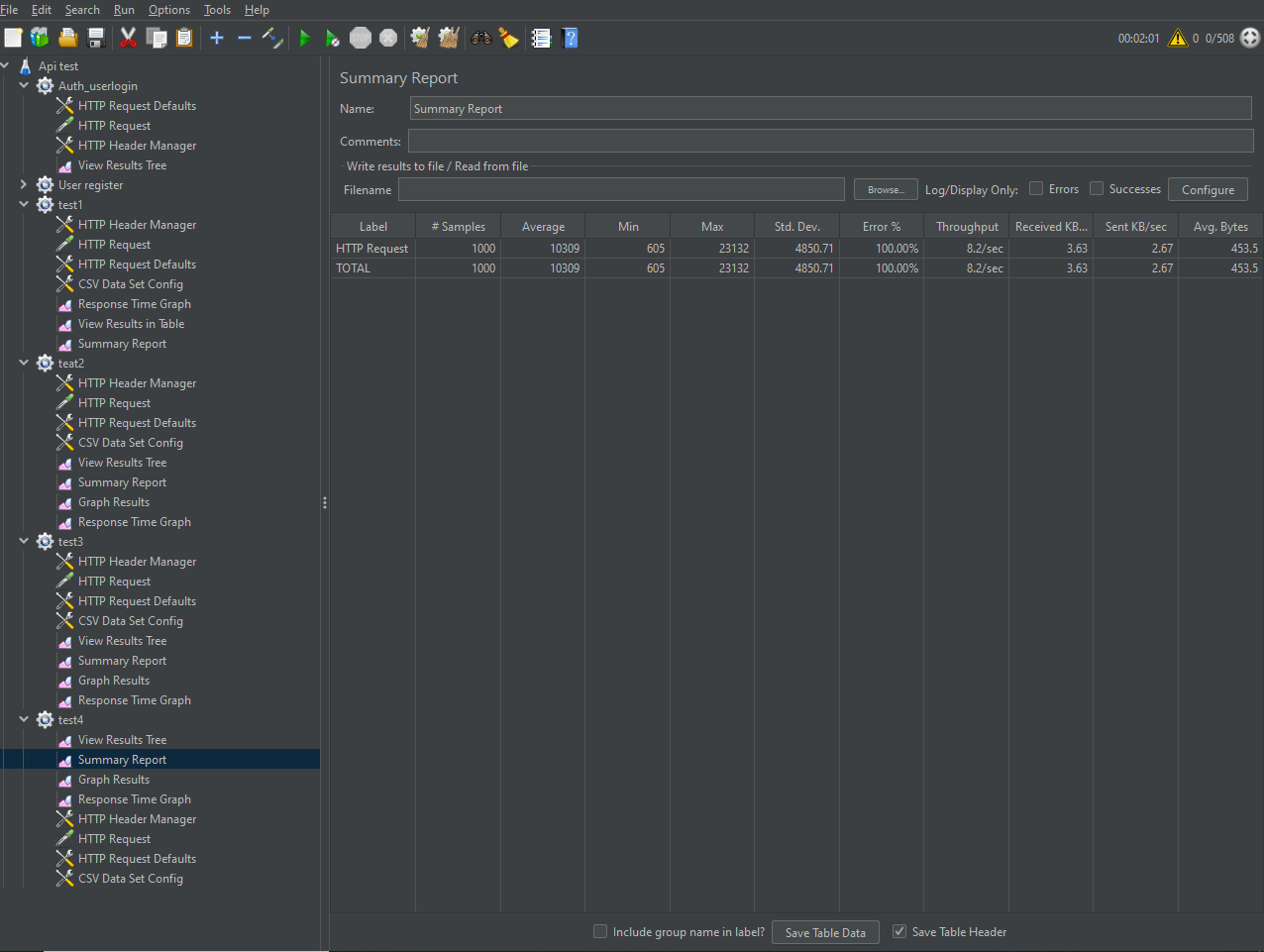
**Summary result**

**Experiment 4: Recovery After Failure**

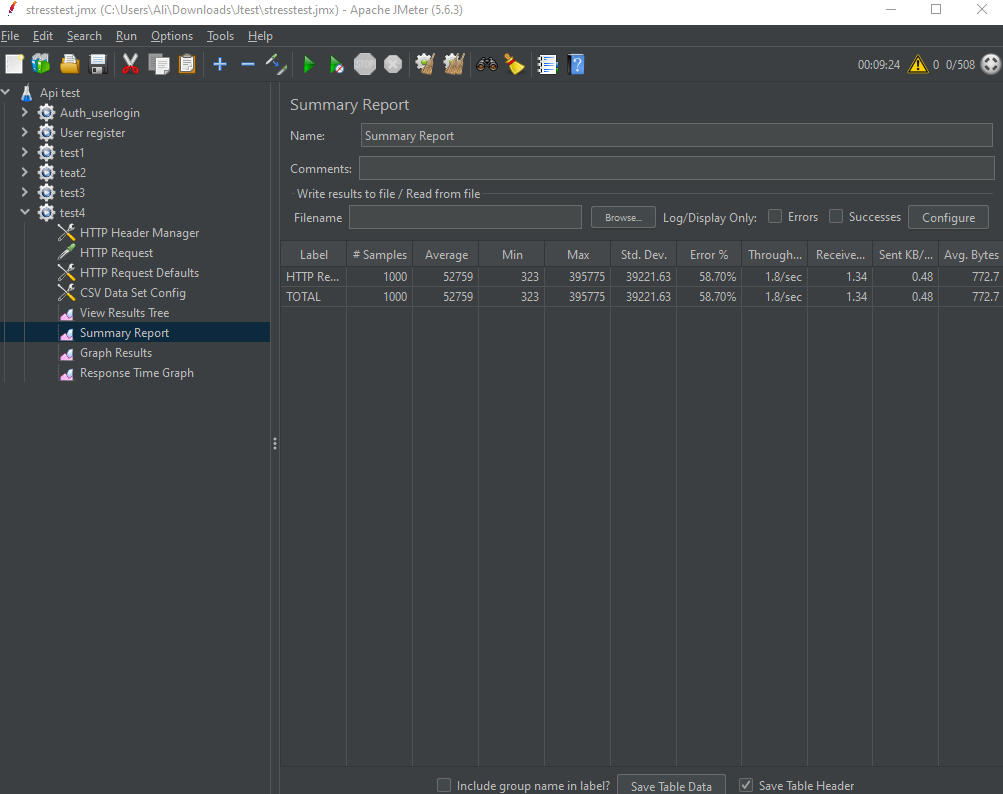
**Objective Test the API's ability to recover from a failure induced by extreme stress.**

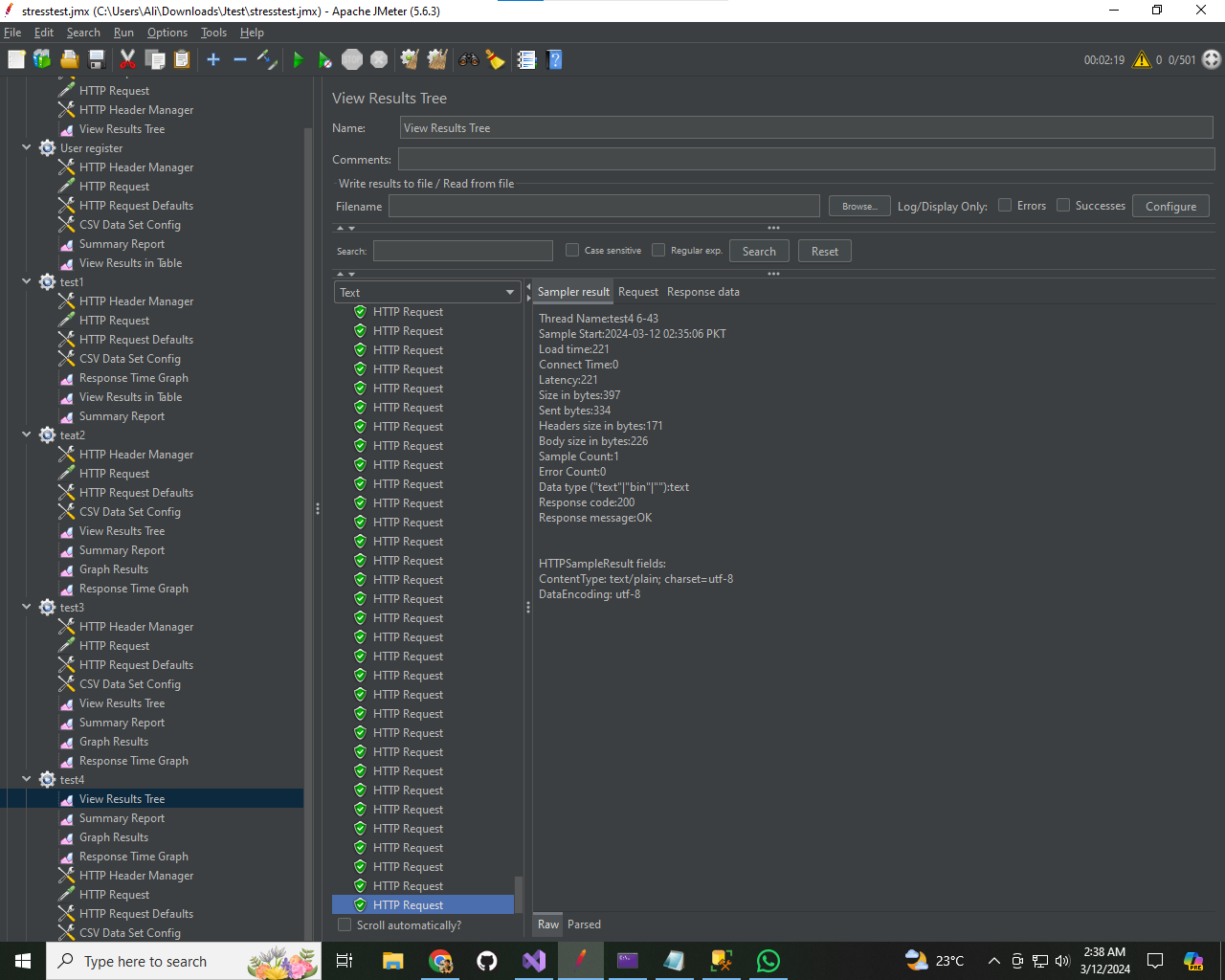
**Result (1)**

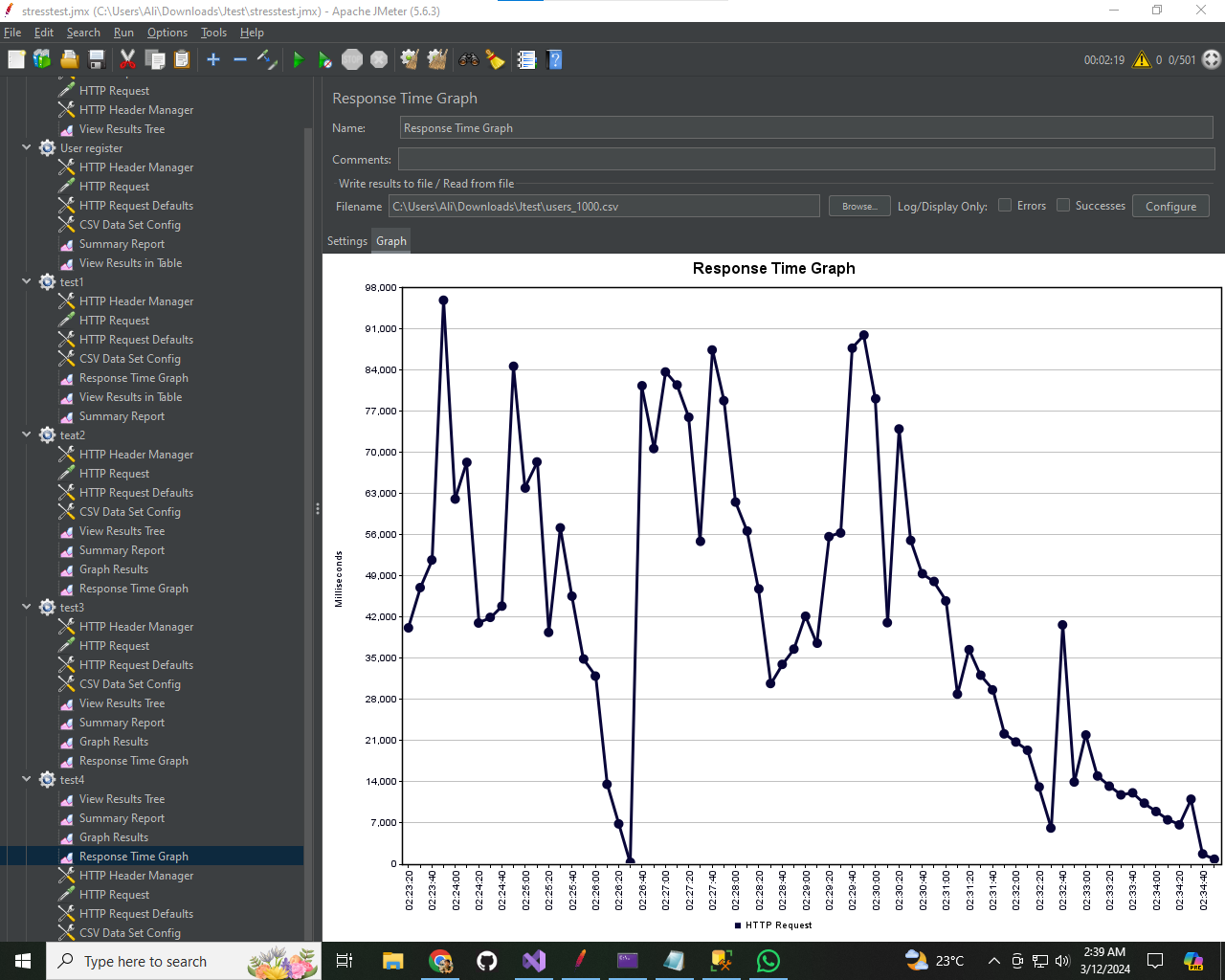
****

**Summary result 1**

**Summart result 2**

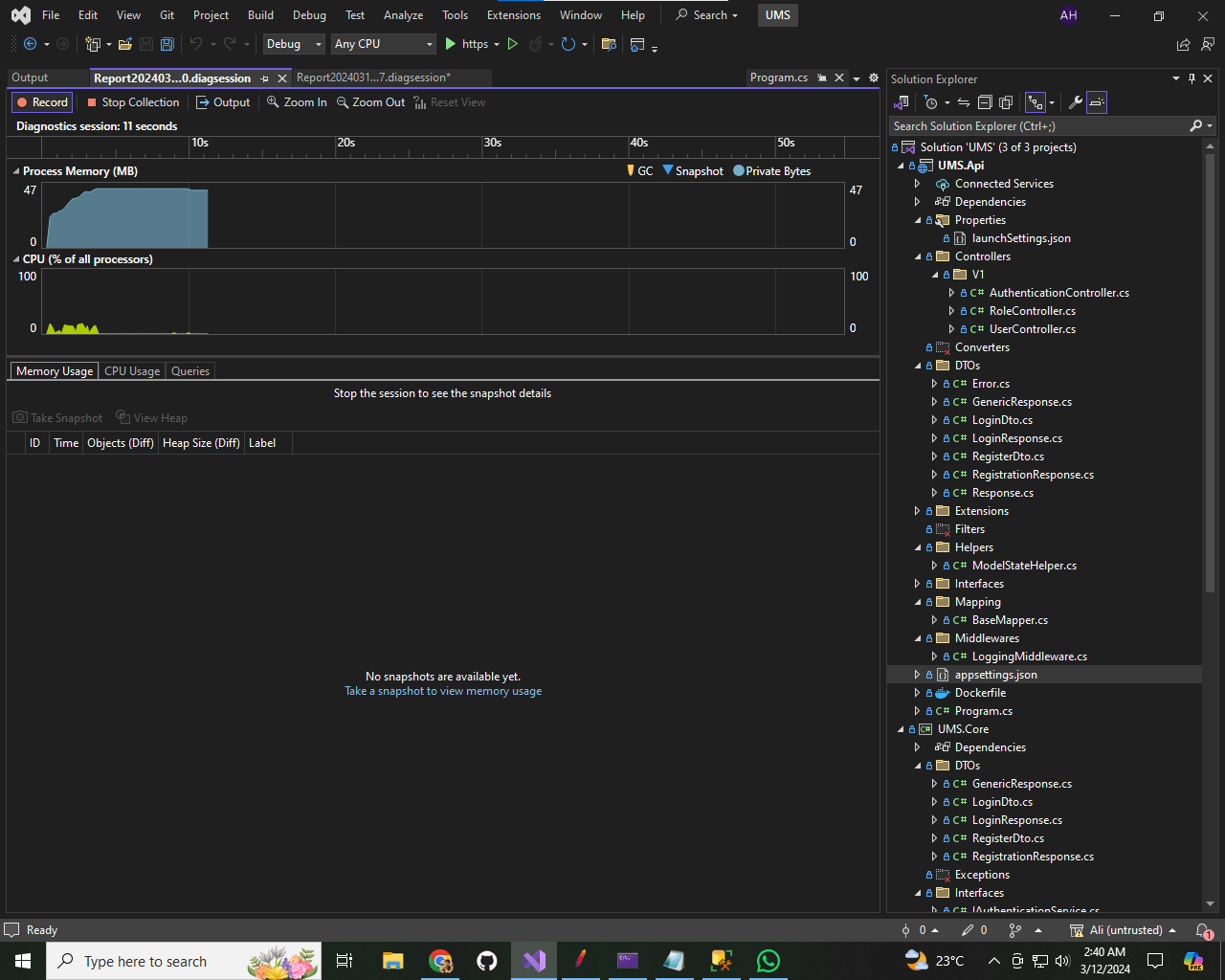


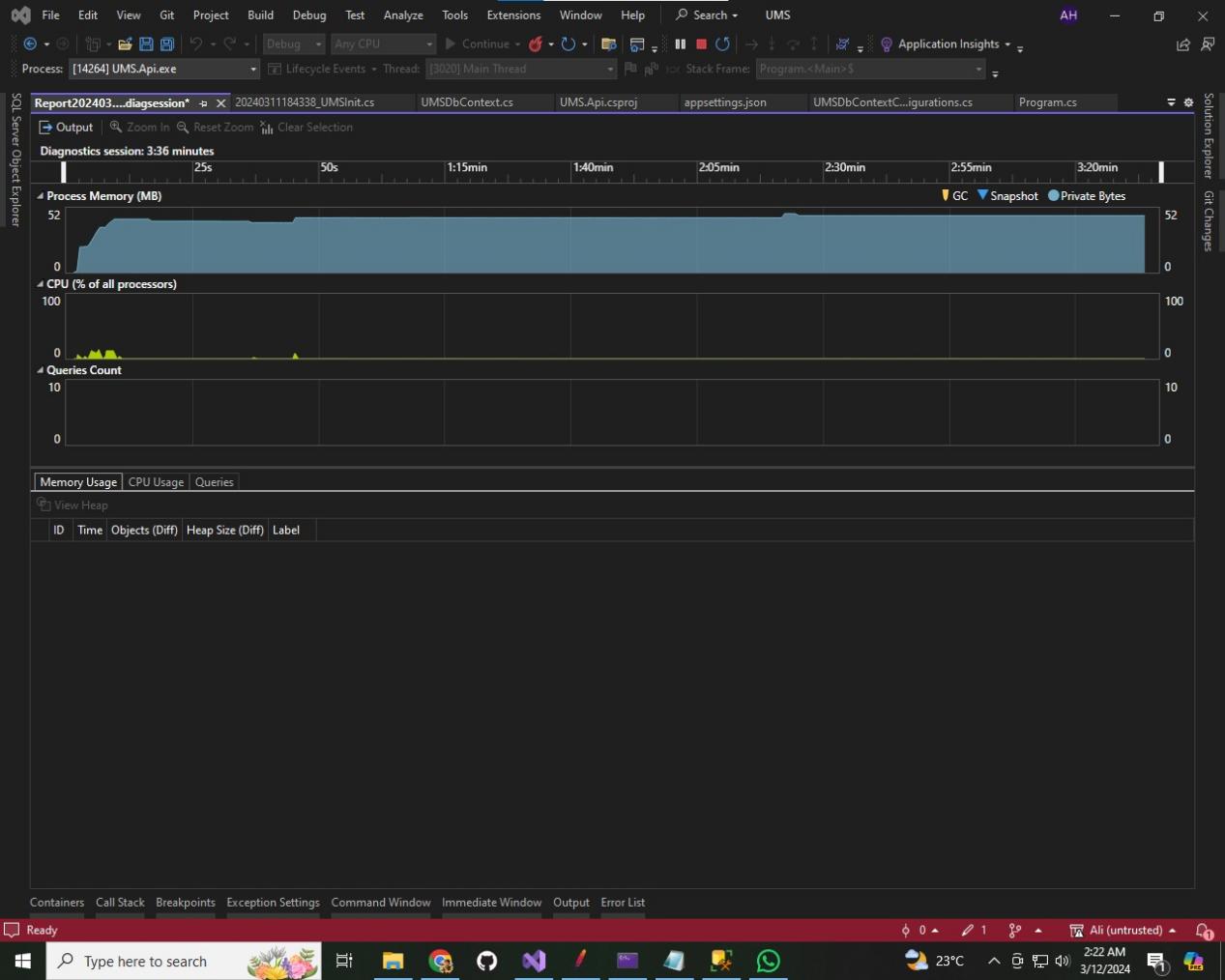
**Result(2)**

**Response time graph Test 4  
**

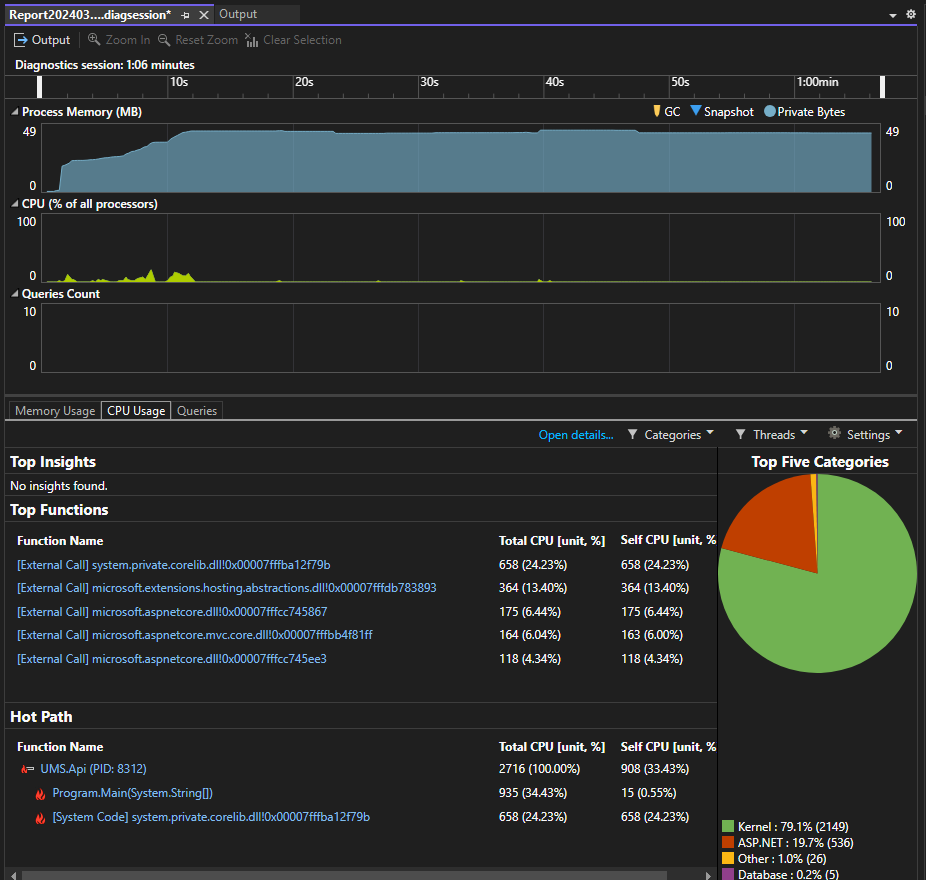
**Results:**

* Induced failure by gradually increasing the user load from 500 to 1000.
* Adjusted system configuration to observe recovery.
* Observed recovery mechanisms and measured the time taken to stabilize the system.

**Test1-2 prefomance result  
**

**Test(3-4) result  
**

**Overall Utilization:**



Kernel: 79.1% (2149)

ASP.NET: 19.7% (536)

Other: 1.0% (26)

Database: 0.2% (5)

**Performance Metrics:**

Response Time:

Monitored response times throughout the tests to gauge system performance under stress.

Response times increased with increasing user load, indicating potential performance bottlenecks.

CPU and Memory Utilization:

Recorded CPU and memory utilization during each test phase.

Utilization increased gradually from test 1 to test 2 and peaked during test 3 to test 4.

Database Connections:

Monitored database connections to assess the impact of stress on database performance.

Minimal impact observed on the database during the stress tests.

**Recommendations**

Optimize Kernel and ASP.NET components to improve performance under stress conditions.

Implement load balancing or caching mechanisms to handle peak loads more efficiently.

Consider scaling up hardware resources or optimizing code to better handle stress conditions without compromising performance.

**Conclusion**

The performance testing revealed valuable insights into the behavior of the API under stress conditions. By addressing the identified areas of improvement and implementing the recommendations, the API's performance and resilience can be significantly enhanced, ensuring better user experience and system stability under varying load conditions.