Introduction

Apache JMeter is a popular open-source tool utilized for assessing the performance, load capacity, and behavior of web applications. The primary objective of this lab report was to develop a wide JMeter Test Plan. This involved configuring Thread Groups, HTTP Samplers, Assertions, Listeners, Loop Controllers, and Timers.

For this lab report, I selected Facebook and LinkedIn as the example web applications. I aimed to simulate numerous users interacting with Facebook and LinkedIn to observe their response under pressure and to gain hands-on experience with a variety of JMeter components functioning together in a structured testing environment.

Test Plan Design

The test plan includes four separate Thread Groups. Each one shows specific JMeter functions. In this lab report, I use some key elements of JMeter and configure some parameters as per the instructions to build this test plan. These components and configures include:

Components :

JMeter Components	Purpose
• Thread Groups	• Indicate total number of virtual users and executed setting
• HTTP Samplers	• HTTPS/HTTP request will be making requests to our server
• Listeners	• Results generated will be retrieved and displayed
• Assertions	• Check all received responses against expected responses.
• Loop Controllers	• Number of times sampler will be sent based on defined repetitions.
• Timer	• Simulated real user actions by including realistic wait times between requests.

Configurations (Based on my ID: 232-35-268):

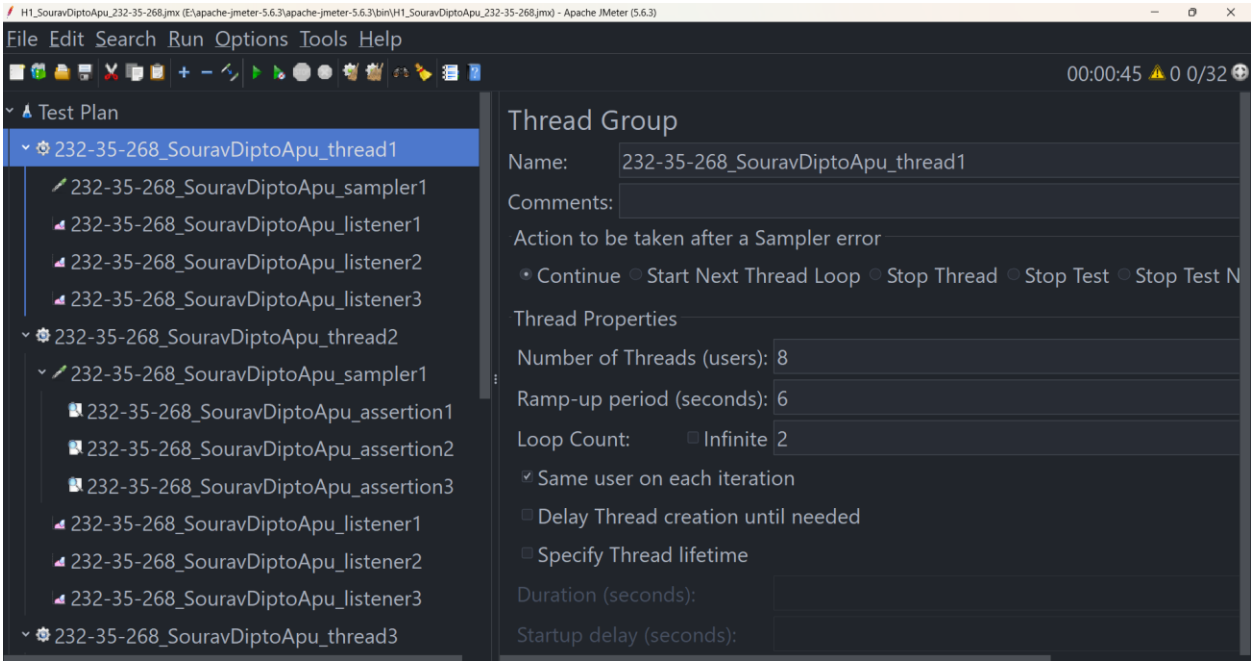
1. Number of Users (Threads): 8 (Last digit = 8)
2. Ramp-up Period: 6 seconds (2nd last digit = 6)
3. Loop Count: 2 (3rd last digit = 2)
4. Timer Delay: 6000 ms (2nd last digit = 6)
5. Loop Controller Count: 8 (Last digit = 8)

Thread Group 1:

Name: 232-35-268_SouravDiptoApu_thread1
Purpose: To execute basic HTTP requests with configured user load and visualize results using multiple listeners.
Configuration:

Number of Users: 8
Ramp-up Period: 6 seconds
Loop Count: 2

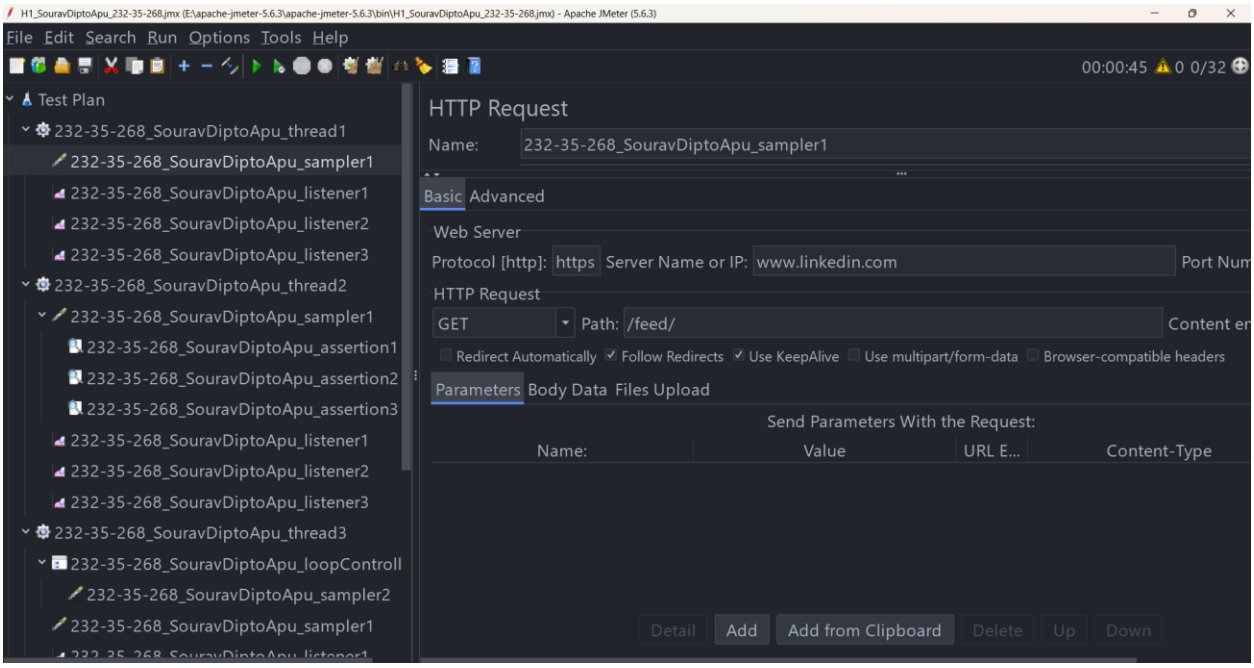
Thread Group 01 Screenshot:



Elements Included:

HTTP Sampler:
Name: 232-35-268_SouravDiptoApu_sampler1
Purpose: Send GET request to linkedin feed.
URL: https://www.linkedin.com
Path: /feed/

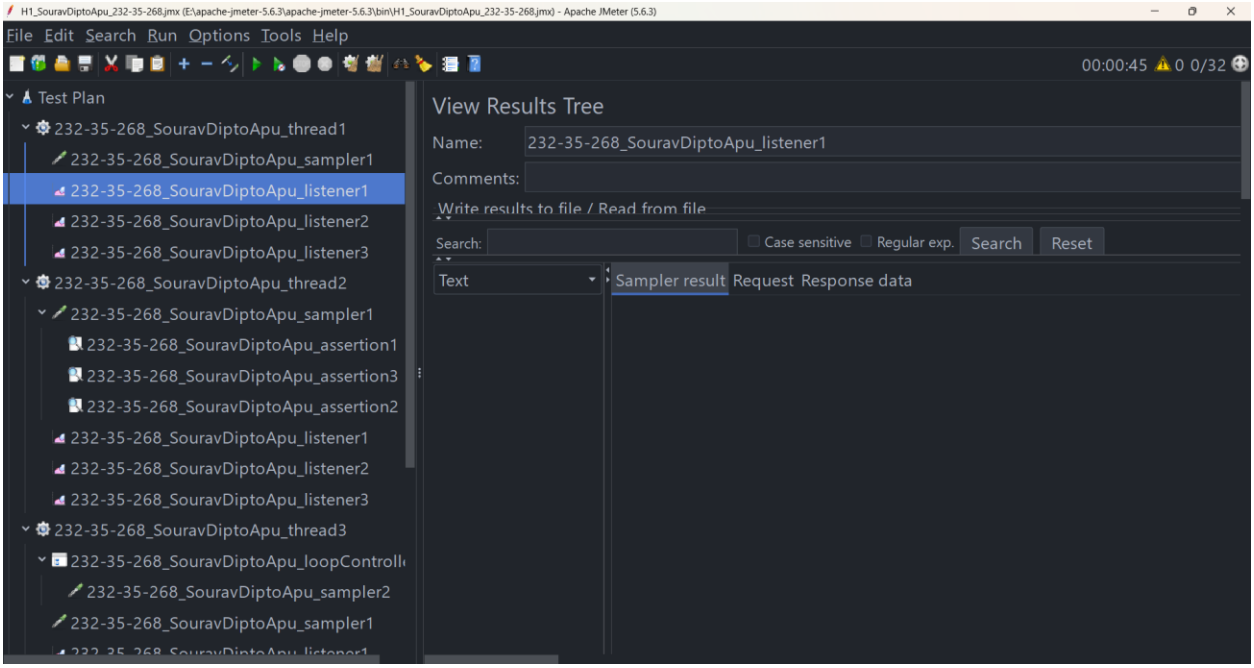
HTTP Sampler Screenshot:

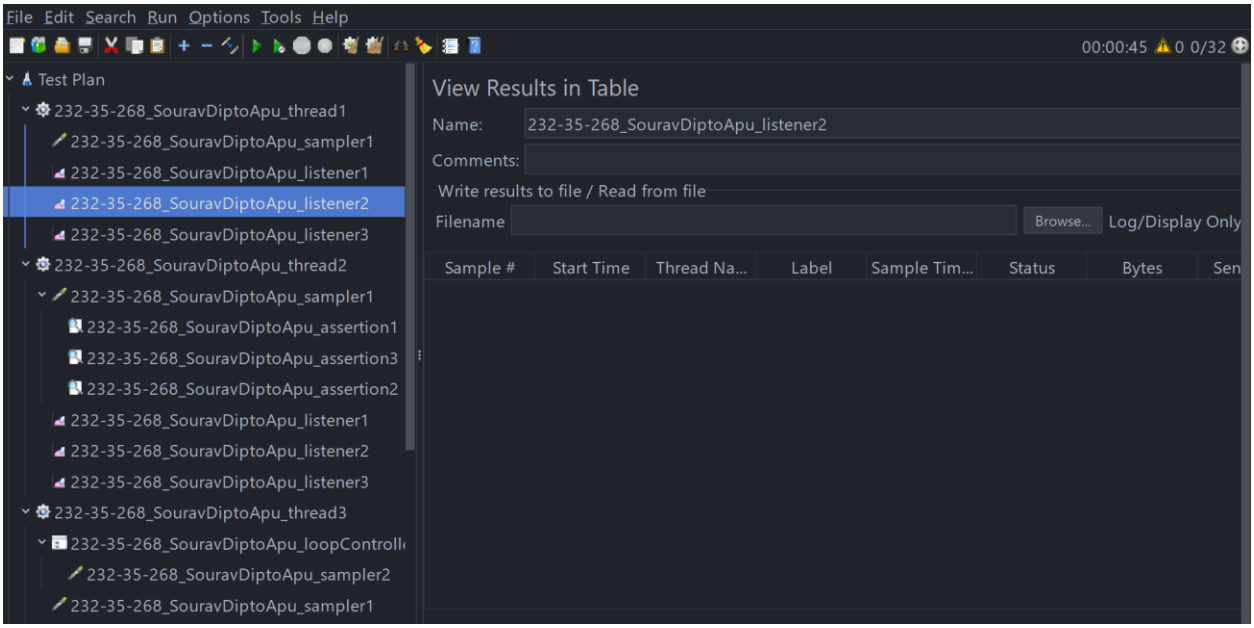


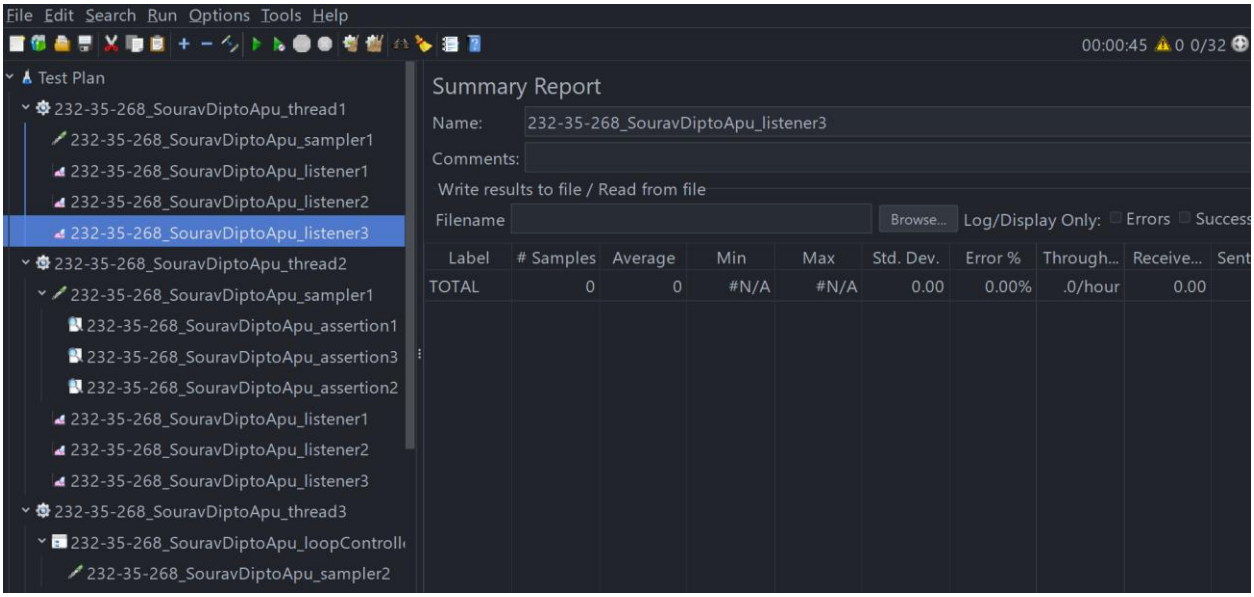
Listeners:

232-35-268_SouravDiptoApu_listener1
232-35-268_SouravDiptoApu_listener2
232-35-268_SouravDiptoApu_listener3

Listeners screenshot (including all Listeners) :


H1_SouravDiptoApu_232-35-268.jmx (E:\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\H1_SouravDiptoApu_232-35-268.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:00:45 0 0/32
Test Plan
232-35-268_SouravDiptoApu_thread1
232-35-268_SouravDiptoApu_sampler1
232-35-268_SouravDiptoApu_listener1
232-35-268_SouravDiptoApu_listener2
232-35-268_SouravDiptoApu_listener3
232-35-268_SouravDiptoApu_thread2
232-35-268_SouravDiptoApu_sampler1
232-35-268_SouravDiptoApu_assertion1
232-35-268_SouravDiptoApu_assertion3
232-35-268_SouravDiptoApu_assertion2
232-35-268_SouravDiptoApu_listener1
232-35-268_SouravDiptoApu_listener2
232-35-268_SouravDiptoApu_listener3
232-35-268_SouravDiptoApu_thread3
232-35-268_SouravDiptoApu_loopControll
232-35-268_SouravDiptoApu_sampler2
232-35-268_SouravDiptoApu_sampler1
View Results Tree
Name:
232-35-268_SouravDiptoApu_listener1
Comments:
Write results to file / Read from file
Search:
Case sensitive
Regular exp.
Search
Reset
Text
Sampler result
Request
Response data


File Edit Search Run Options Tools Help
00:00:45 0 0/32
Test Plan
232-35-268_SouravDiptoApu_thread1
232-35-268_SouravDiptoApu_sampler1
232-35-268_SouravDiptoApu_listener1
232-35-268_SouravDiptoApu_listener2
232-35-268_SouravDiptoApu_listener3
232-35-268_SouravDiptoApu_thread2
232-35-268_SouravDiptoApu_sampler1
232-35-268_SouravDiptoApu_assertion1
232-35-268_SouravDiptoApu_assertion3
232-35-268_SouravDiptoApu_assertion2
232-35-268_SouravDiptoApu_listener1
232-35-268_SouravDiptoApu_listener2
232-35-268_SouravDiptoApu_listener3
232-35-268_SouravDiptoApu_thread3
232-35-268_SouravDiptoApu_loopControll
232-35-268_SouravDiptoApu_sampler2
232-35-268_SouravDiptoApu_sampler1
View Results in Table
Name:
232-35-268_SouravDiptoApu_listener2
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only
Sample #
Start Time
Thread Na...
Label
Sample Tim...
Status
Bytes
Sen

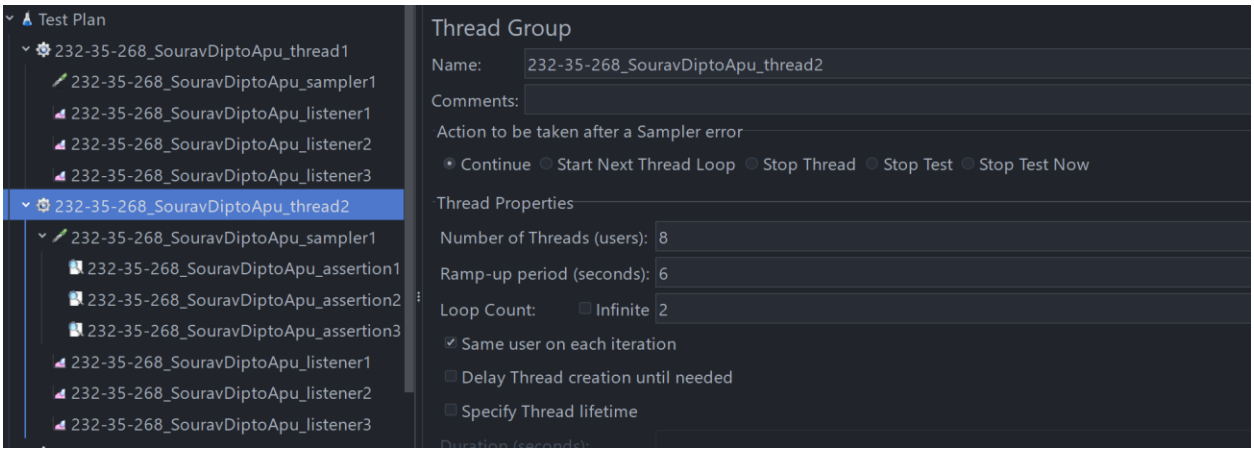


Thread Group 2:

Name: 232-35-268_SouravDiptoApu_thread2
Purpose: To verify the correctness of the server response by using different types of assertions during the performance test.
Configuration Details:

Number of Users (Threads): 8
Ramp-up Period: 6 seconds
Loop Count: 2

Screenshot:



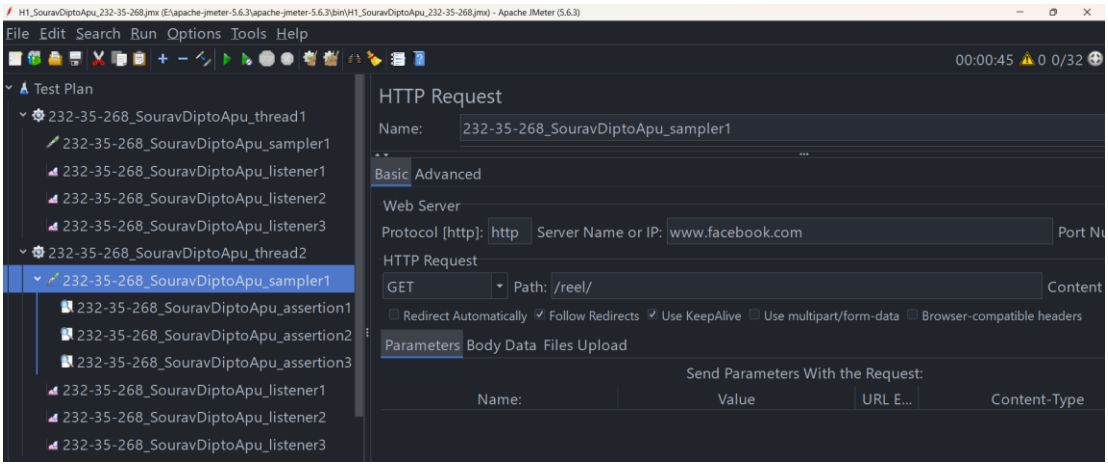
HTTP Sampler:

Name: 232-35-268_SouravDiptoApu_sampler1
Purpose in Test Plan:
To send an HTTP GET request to the Facebook homepage and collect the server response for validation.

Configuration Details:

Protocol: https
Server Name: www.facebook.com
HTTP Method: GET
Path: /reel/

Screenshot:



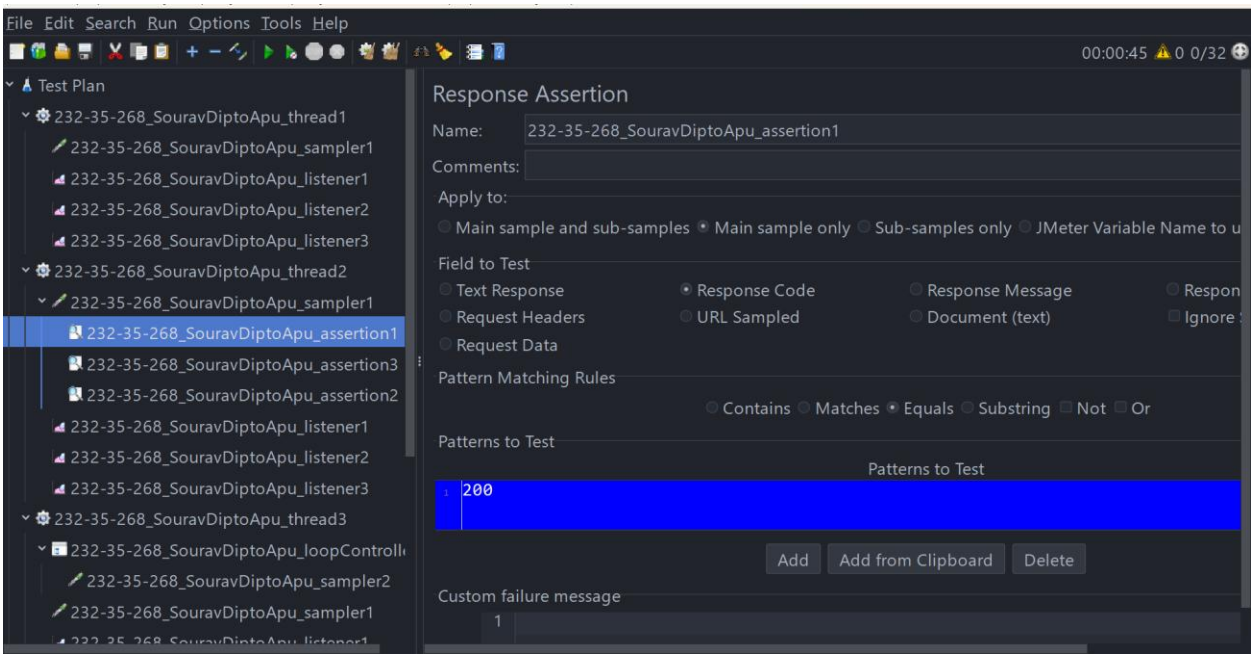
Assertions:

1. Response Assertion
Name: 232-35-268_SouravDiptoApu_assertion1
Purpose in Test Plan: To check whether the server response contains the expected text.

Configuration Details:

Test Field: Response Body
Pattern Matching Rule: Contains
Expected Text: Facebook

Screenshot:



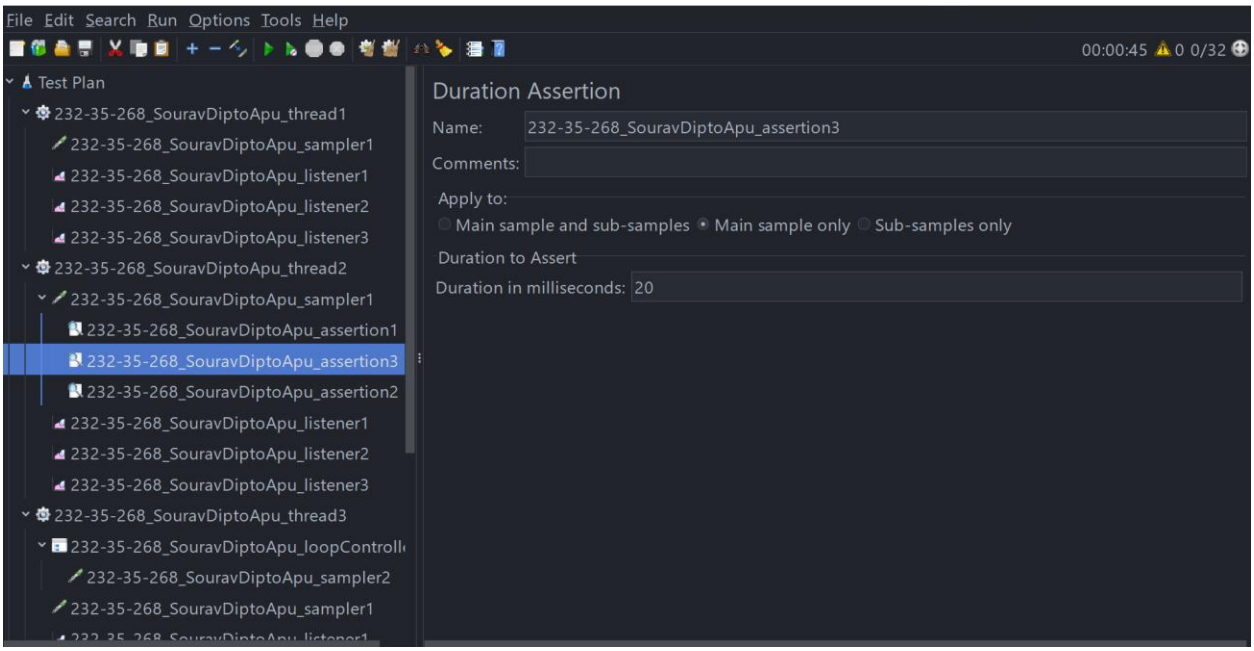
2. Duration Assertion (Intentional Failure)
Name: 232-35-268_SouravDiptoApu_assertion3
Purpose in Test Plan: To ensure that the server response time does not exceed the specified limit.

Configuration Details:

Maximum Response Time: 20 ms

Explanation of Duration Assertion Error:
Because the maximum permitted response time was set at just 20 milliseconds, which is unreasonably low for big web apps like Facebook, linkedin, the Duration Assertion failed. As anticipated, the assertion failed because the actual response time exceeded the configured limit.

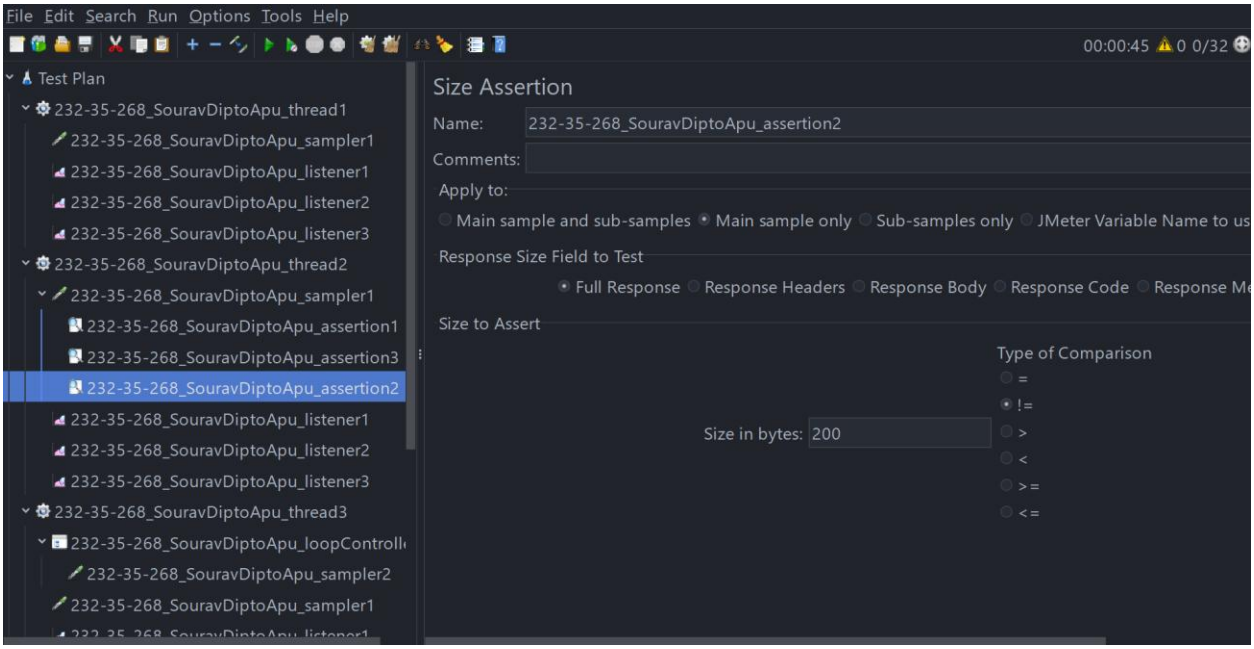
Screenshot:



3. Size Assertion:
Name: 232-35-268_SouravDiptoApu_assertion2
Purpose in Test Plan: To validate the size of the server response.
Configuration Details:

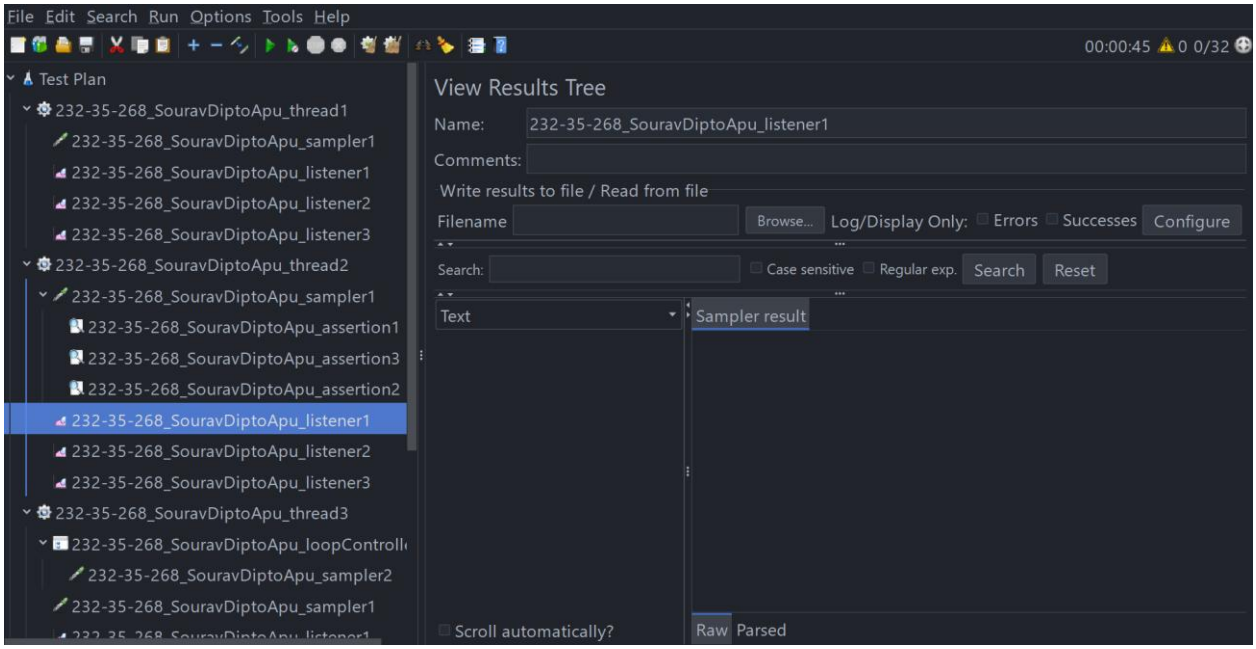
Expected Size (Bytes): 200
Condition: Not Equals

Screenshot:

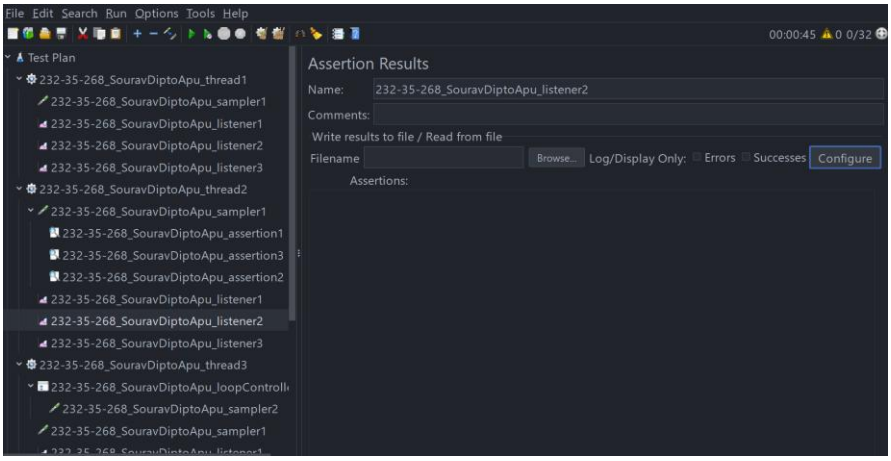


Listeners

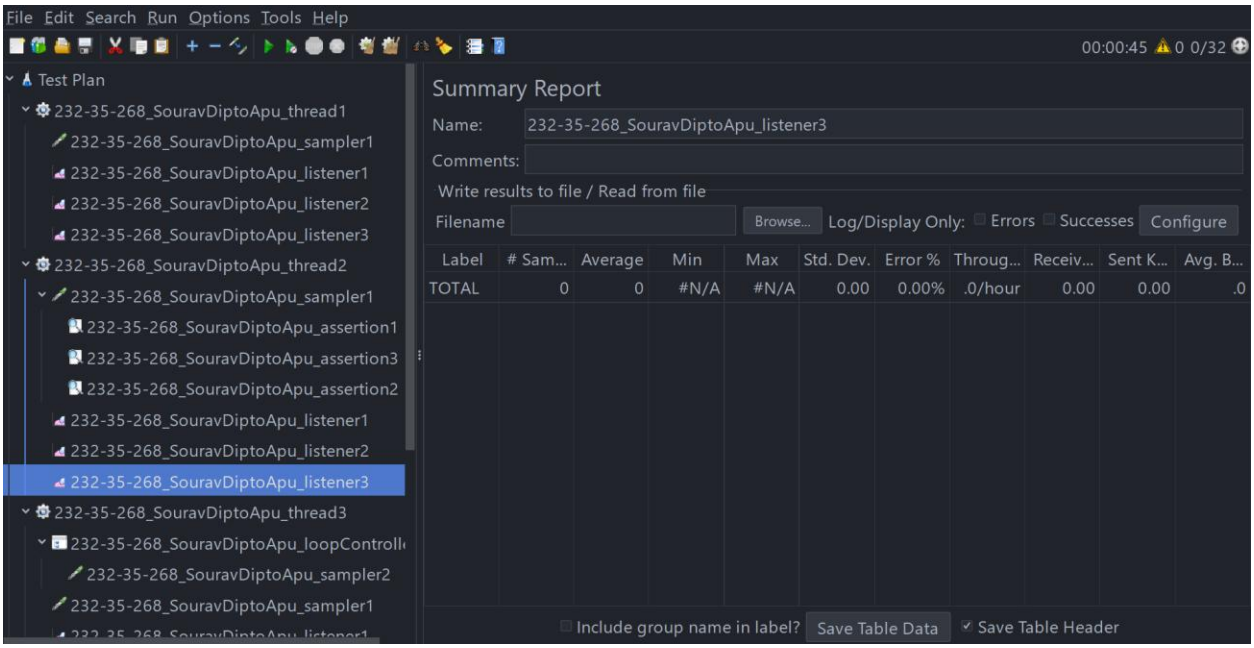
View Results Tree:
Name: 232-35-268_SouravDiptoApu_listener1
Purpose in Test Plan: To display detailed request and response information for each sample.
Configuration Details: Default
Screenshot:



Assertion Results:
Name: 232-35-268_SouravDiptoApu_listener2
Purpose in Test Plan: To show which assertions pass or fail during execution.
Configuration Details: Default
Screenshot:



Summary Report:
Name: 232-35-268_SouravDiptoApu_listener
Purpose in Test Plan: To summarize performance metrics like average time, throughput, and error rate.
Configuration Details: Default
Screenshot:

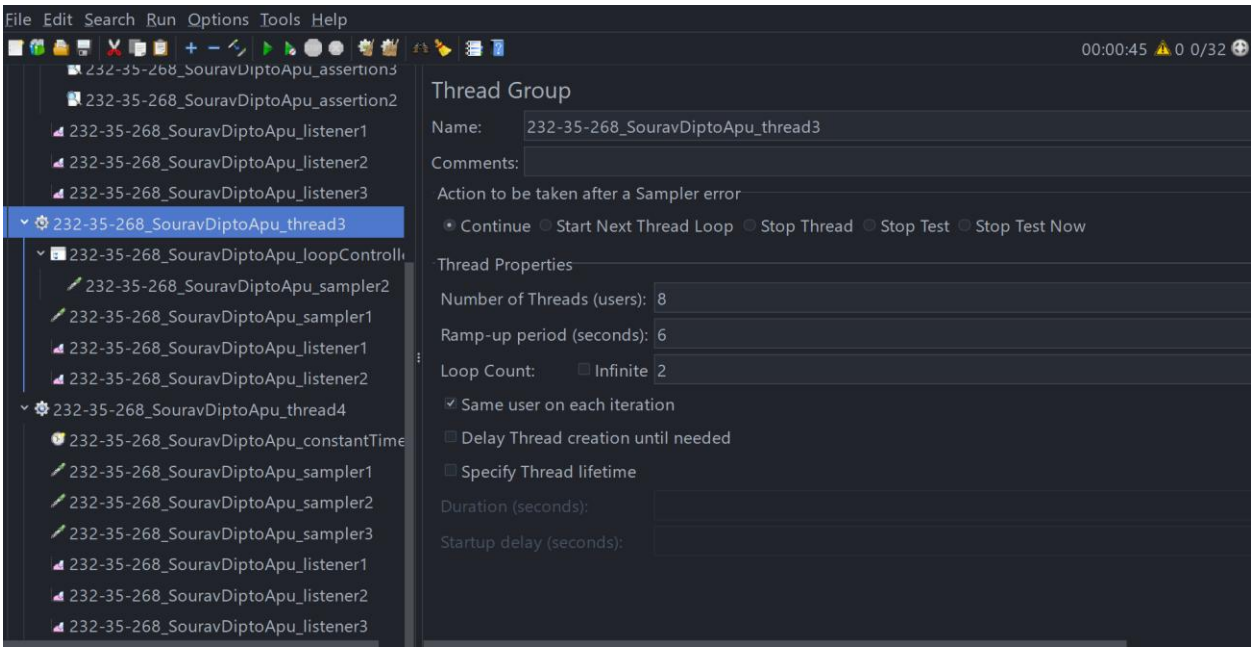


Thread Group 3

Name: 232-35-268_SouravDiptoApu_thread3
Purpose in Test Plan: To repeat a specific HTTP request multiple times inside a Loop Controller.
Configuration Details:

Number of Users (Threads): 8
Ramp-up Period: 6 seconds
Loop Count: 2

Screenshot:

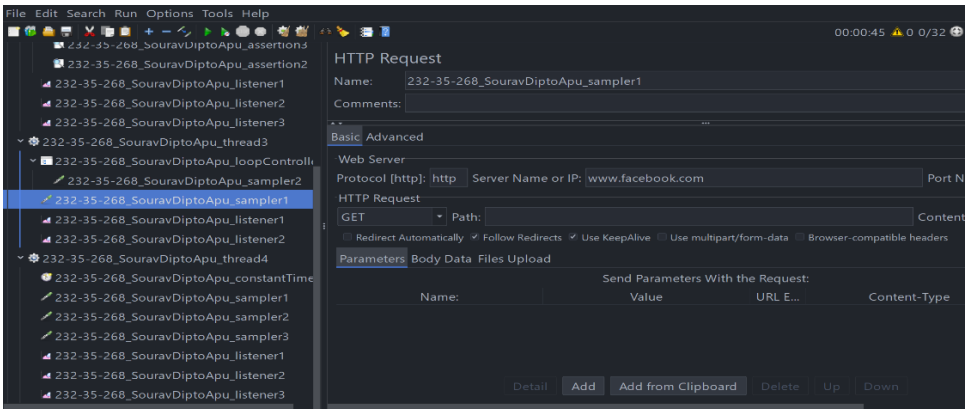


HTTP Sampler

Name: 232-35-268_SouravDiptoApu_sampler1
Purpose in Test Plan: To send a GET request to Facebook homepage before the Loop Controller executes.
Configuration Details:

Protocol: https
Server Name: www.facebook.com
HTTP Method: GET
Path: N/A

Screenshot:

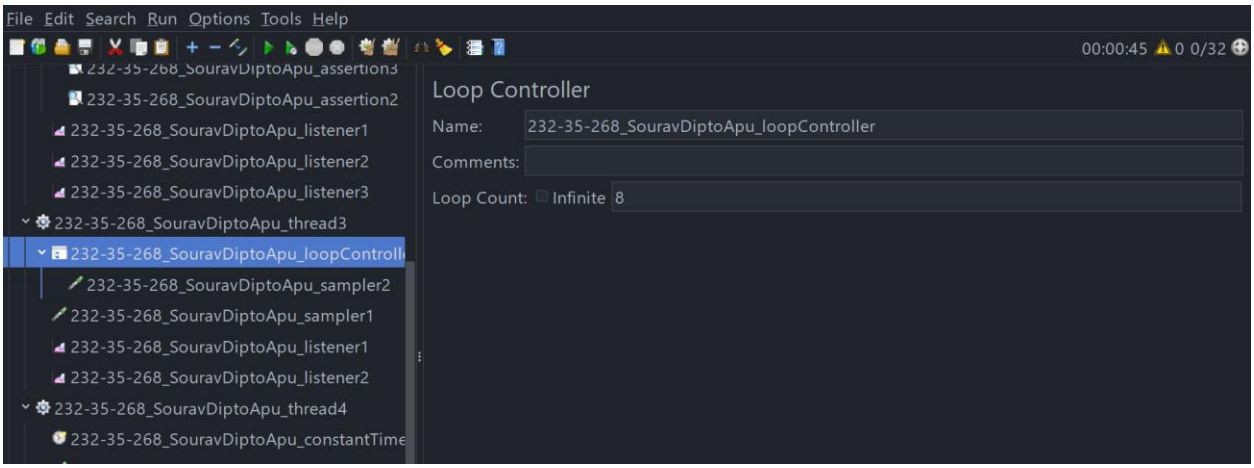


Loop Controller

Name: 232-35-268_SouravDiptoApu_loopController
Purpose in Test Plan: To repeat a sampler multiple times according to Loop Count.
Configuration Details:

Loop Count: 8 (last digit of student ID)

Screenshot:

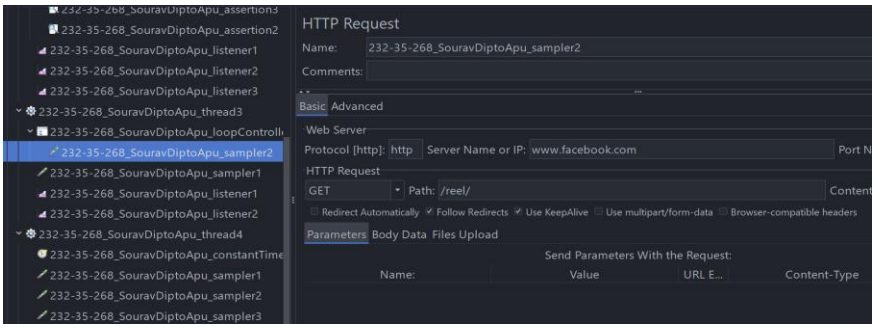


HTTP Sampler inside Loop Controller

Name: 232-35-268_SouravDiptoApu_sampler2
Purpose in Test Plan: To repeatedly send a GET request to Facebook homepage inside the loop.
Configuration Details:

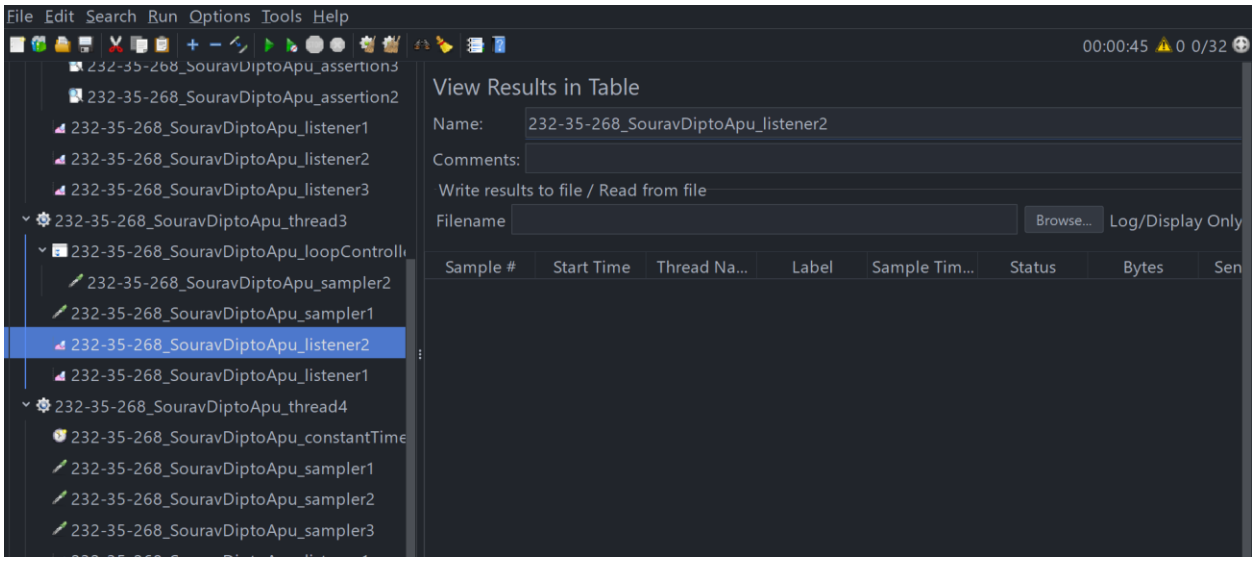
Protocol: https
Server Name: www.facebook.com
HTTP Method: GET
Path: /reel/

Screenshot:

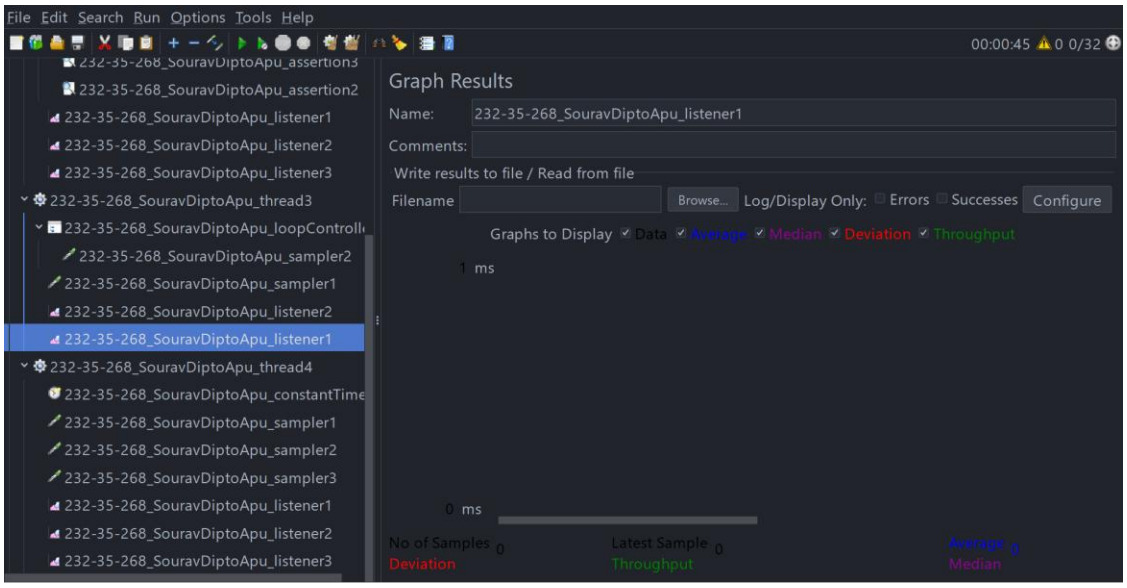


Listeners:

View Results Tree:
Name: 232-35-268_SouravDiptoApu_listener2
Purpose in Test Plan: To display detailed request and response information for each sample.
Configuration Details: Default
Screenshot:



Graph Result:
Name: 232-35-268_SouravDiptoApu_listener1
Purpose in Test Plan: To visualize performance trends like response time and throughput in a graph format.
Configuration Details: Default
Screenshot:

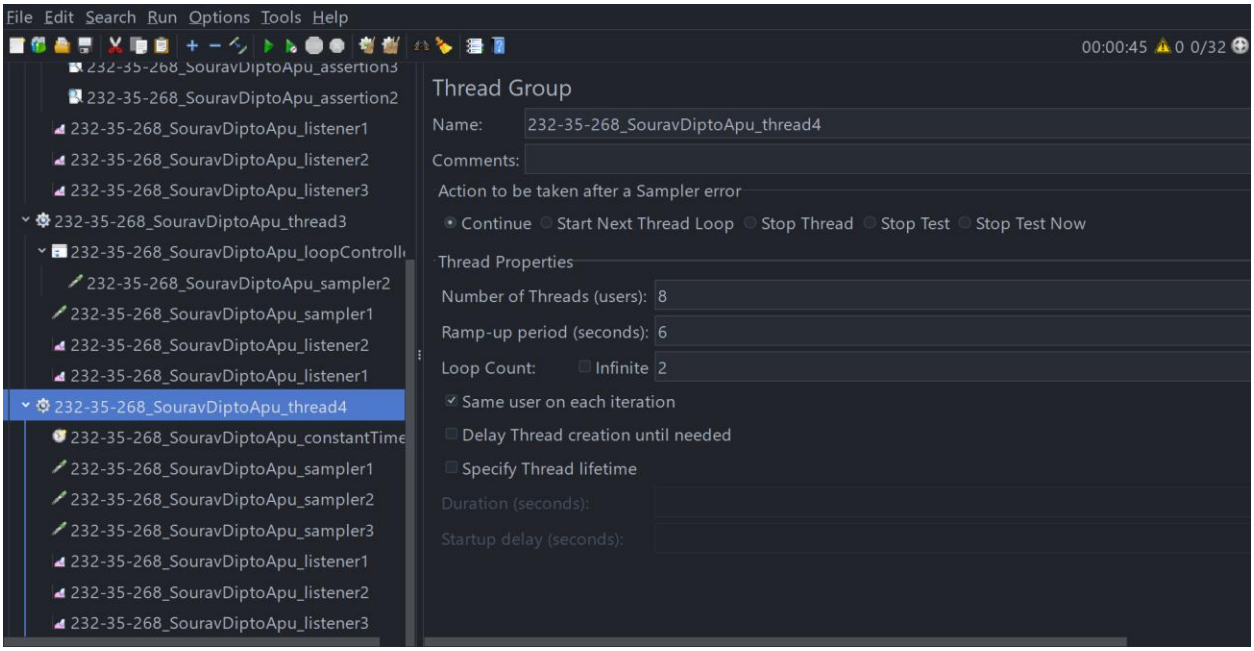


Thread Group 4:

Name: 232-35-268_SouravDiptoApu_thread4
Purpose in Test Plan: To simulate delays between user requests using a Constant Timer.
Configuration Details:

Number of Users (Threads): 8
Ramp-up Period: 6 seconds
Loop Count: 2

Screenshot:

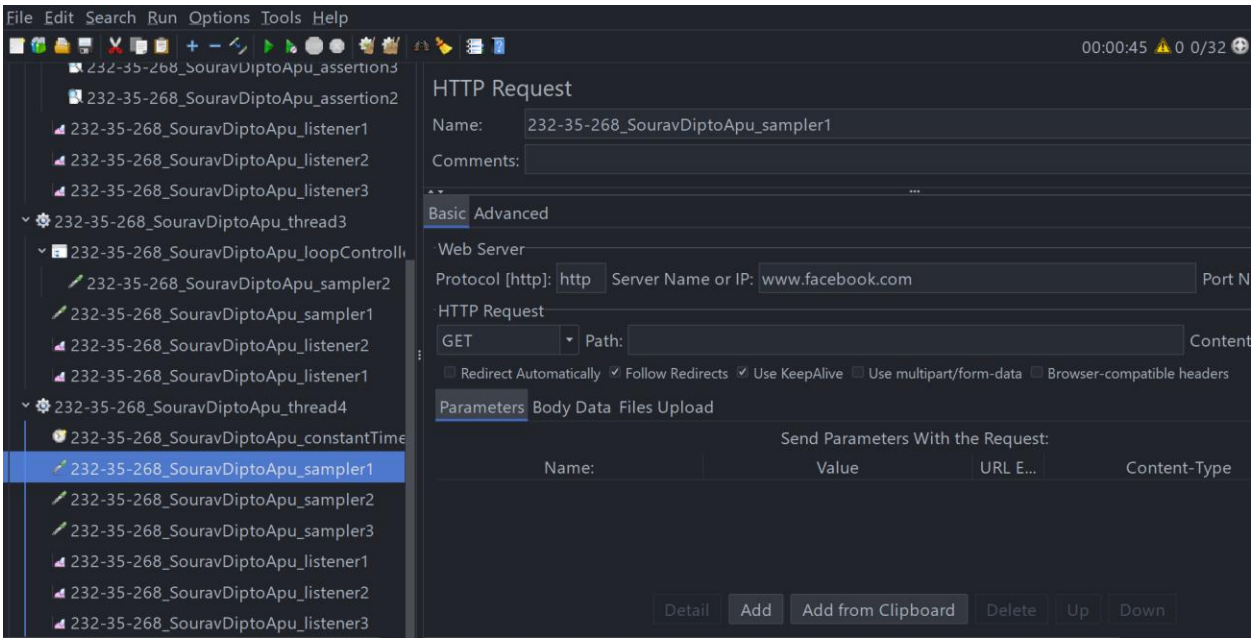


HTTP Samplers

HTTP Sampler 1
Name: 232-35-268_SouravDiptoApu_sampler1
Purpose in Test Plan: To send a GET request to Facebook homepage.
Configuration Details:

Protocol: https
Server Name: www.facebook.com
HTTP Method: GET
Path: N/A

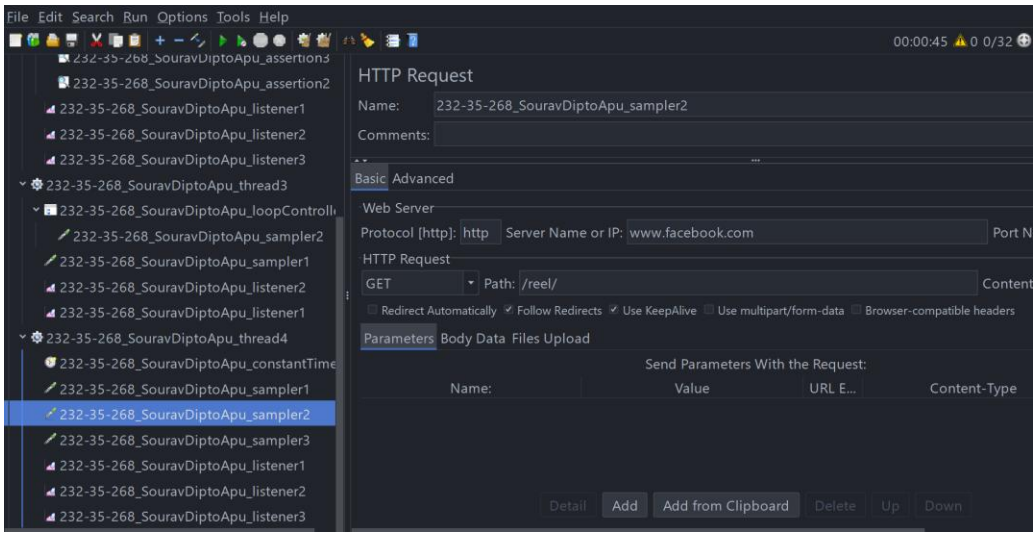
Screenshot:



HTTP Sampler 2
Name: 232-35-268_SouravDiptoApu_sampler2
Purpose in Test Plan: To send another GET request to Facebook homepage.
Configuration Details:

Protocol: https
Server Name: www.facebook.com
HTTP Method: GET
Path: /reel/

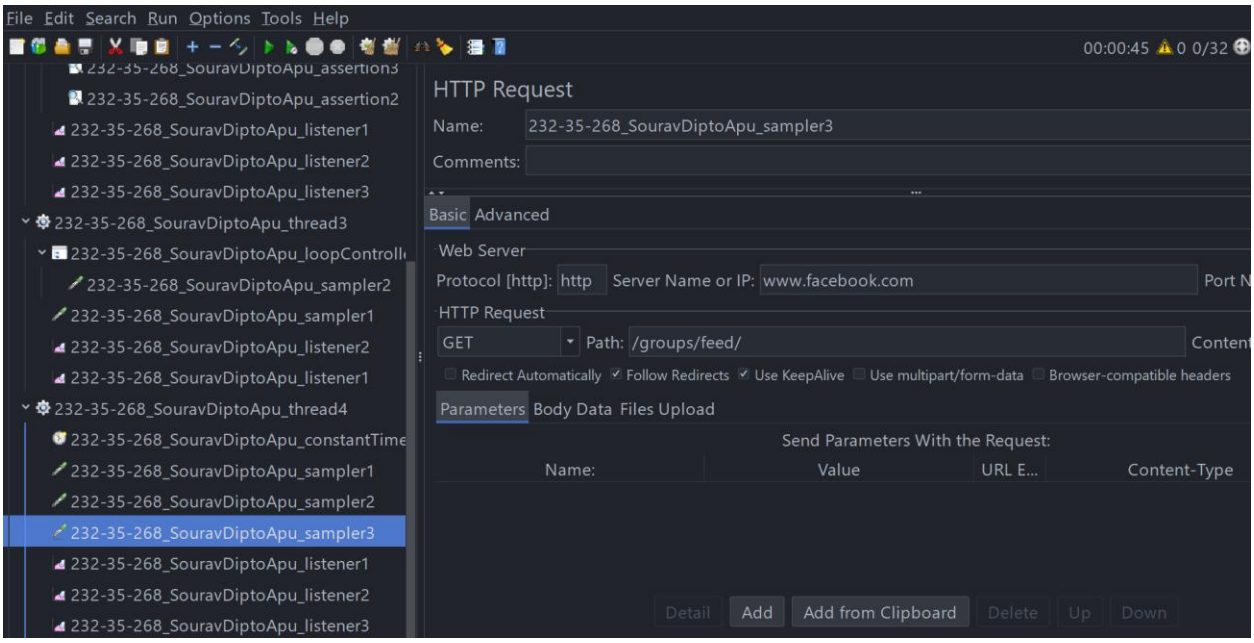
Screenshot:



HTTP Sampler 3
Name: 232-35-268_SouravDiptoApu_sampler3
Purpose in Test Plan: To send a third GET request to Facebook homepage.
Configuration Details:

Protocol: https
Server Name: www.facebook.com
HTTP Method: GET
Path: /groups/feed/

Screenshot:

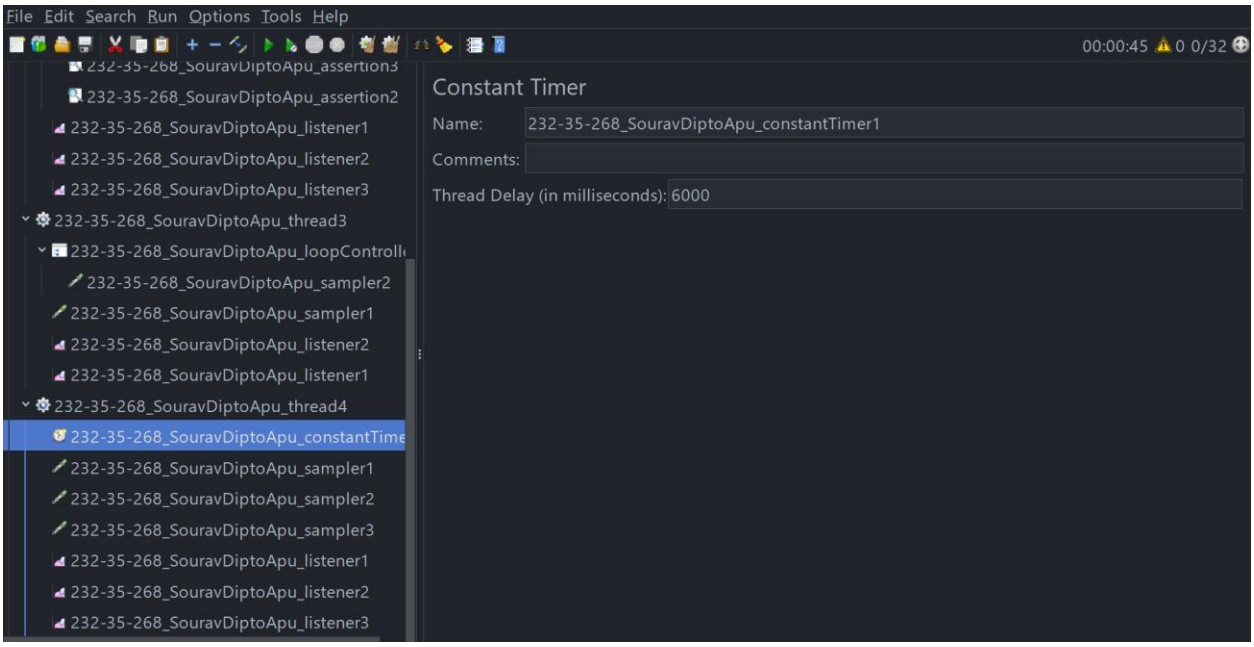


Timer

Constant Timer
Name: 232-35-268_SouravDiptoApu_constantTimer1
Purpose in Test Plan: To introduce a fixed delay between consecutive requests.
Configuration Details:

Thread Delay: 6000 ms (2nd last digit of ID = 6 → 6 seconds)

Screenshot:

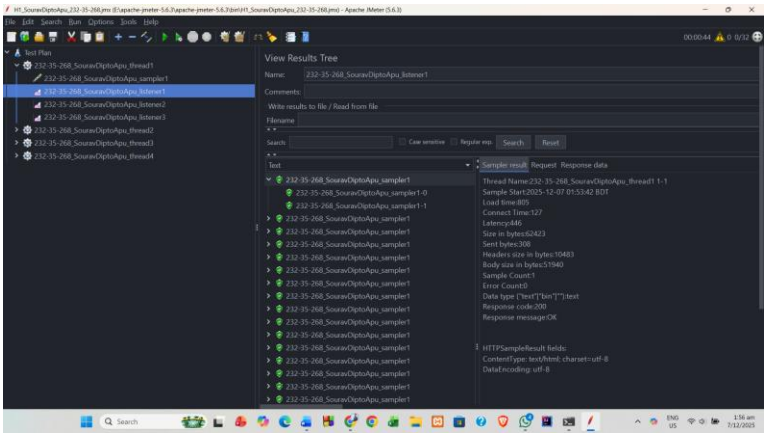


Execution and Results

Thread Group 1:
Listener: View Results Tree
Name: 232-35-268_SouravDiptoApu_listener1

View Results Tree Analysis:

For that, I used this listener, the View Results Tree, to verify whether my base requests were successful or not. I have observed that every sample appears with a green success icon, meaning that all the requests coming from Thread Group 1 were processed by the server of Linkedin without any issues. The response code was 200 and the response message was OK - thus, no connection error or any issue with the server occurred in this basic load test.

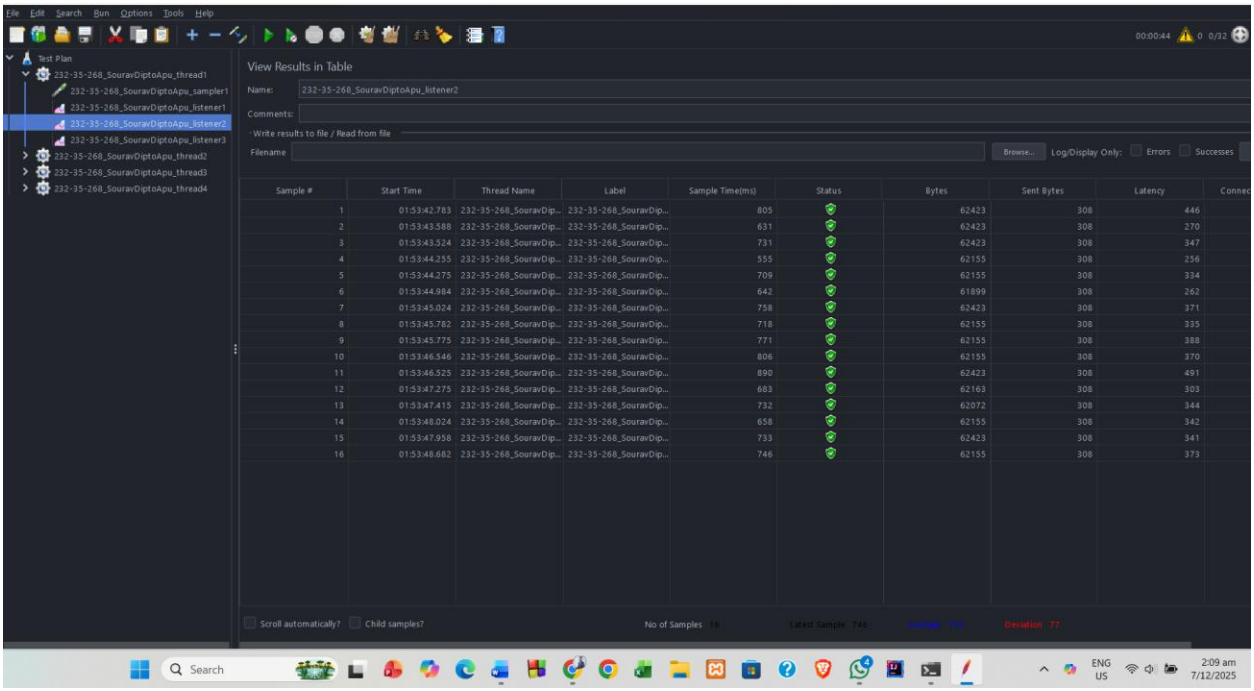


Thread Group 1:
Listener: View Results in Table
Name: 232-35-268_SouravDiptoApu_listener2

View Results in Table Analysis:

I used the View Results in Table listener to get a clear, step-by-step view of how the test ran in Thread Group 1. The table shows that 16 samples were executed, which matches the setup of 8 users running 2 loops each, resulting in 16 total requests.

All the requests showed a green checkmark in the Status column, meaning every request was processed successfully without any errors. The table also provided useful performance data. The average response time was around 723 ms, and the average latency was about 323 ms. These results confirm that the test configuration was correct and the server handled the load smoothly.



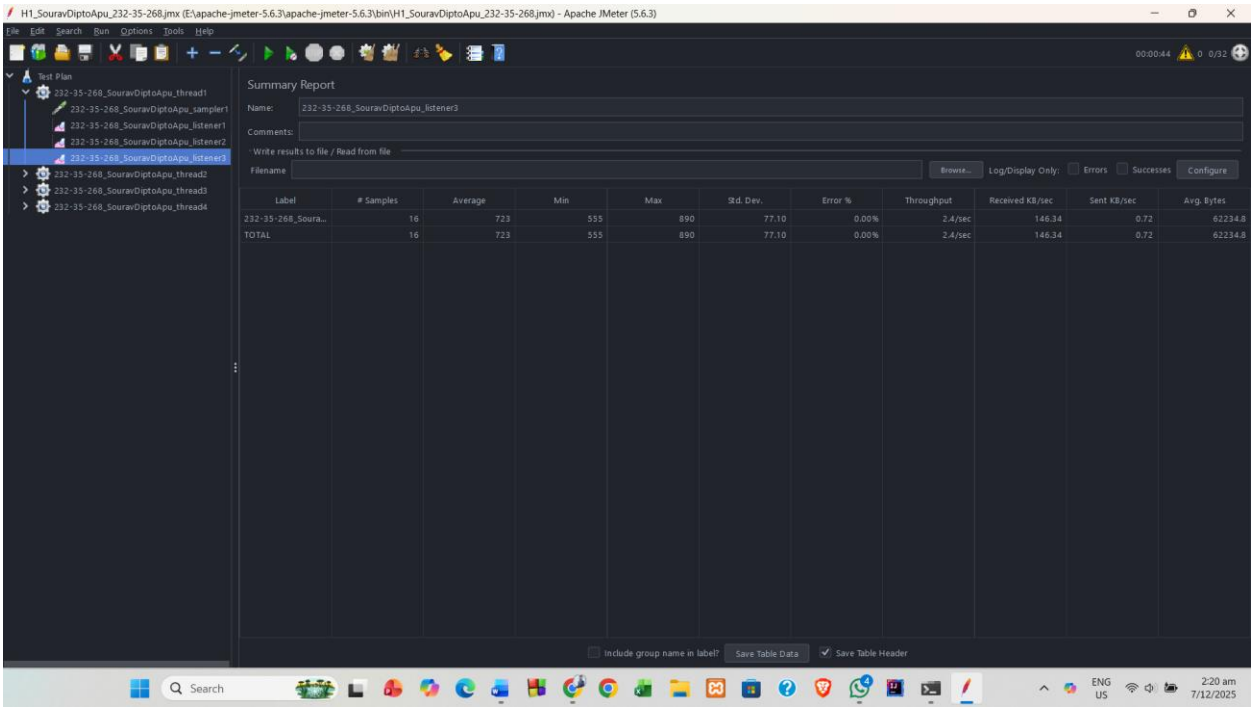
Thread Group 1:
Listener: Summary Report
Name: 232-35-268_SouravDiptoApu_listener3

Summary Report Analysis:

I used the Summary Report to see the overall metrics for my basic requests in Thread Group 1. The report confirmed that 16 samples were executed in total for the label
The key performance indicators I observed were:

Average Time: The average response time was 723 milliseconds.
Error Rate: The test had an Error % of 0.00%. This means the Google server successfully returned a correct response for every single request I sent.
Throughput: The system processed requests at a rate of 2.4 requests per second.
Deviation: The standard deviation was 77.10 milliseconds, which is relatively low compared to the average, showing that the response times were consistent across the 16 requests.

Overall, this report confirms that my basic load test configuration worked correctly and the target application handled the initial 8-user load efficiently.



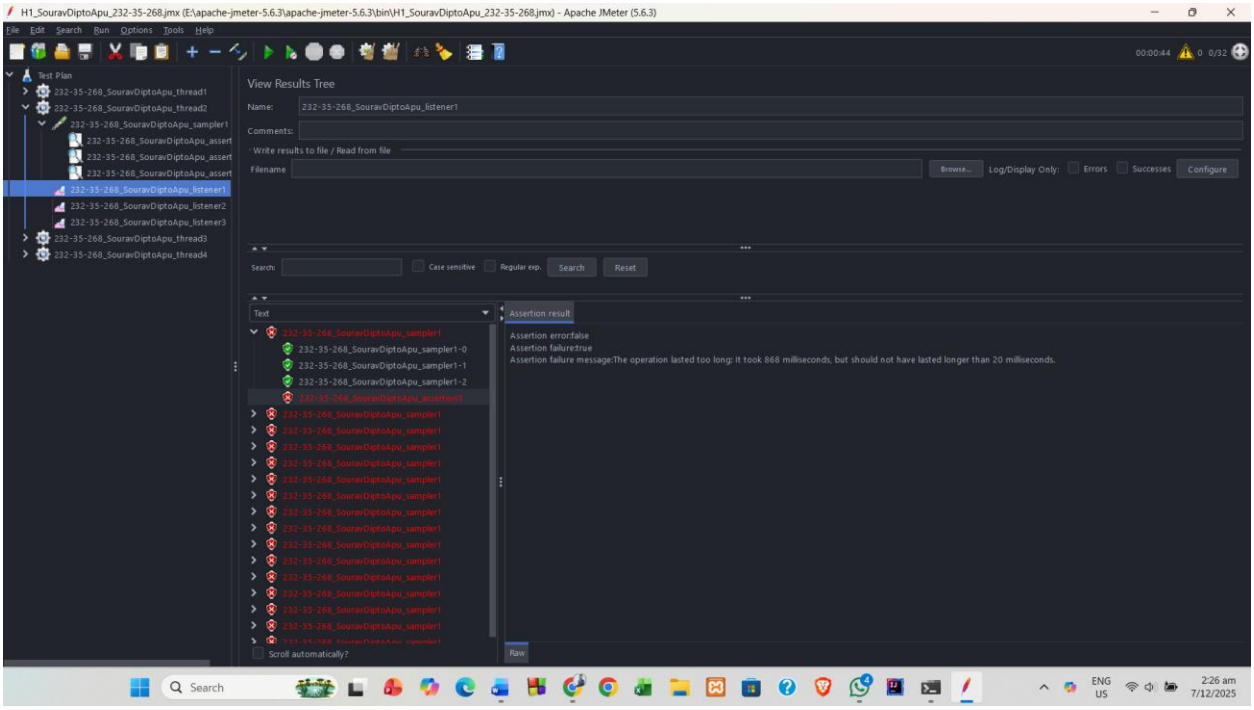
Thread Group 2:

Listener: View Results Tree
Name: 232-35-268_SouravDiptoApu_listener1

View Results Tree Analysis:

I used the View Results Tree listener to check the status of the assertion tests in Thread Group 2. The results showed that the sampler failed, which was indicated by red icons for most executions. After locating the failed sampler, I checked the Assertion Result tab and found the failure message stating that the operation took 805 milliseconds, which was longer than the allowed maximum of 20 milliseconds.

This confirms that the Duration Assertion failed as intended. The real response time was much higher than the configured limit, which successfully created the expected error for the test.

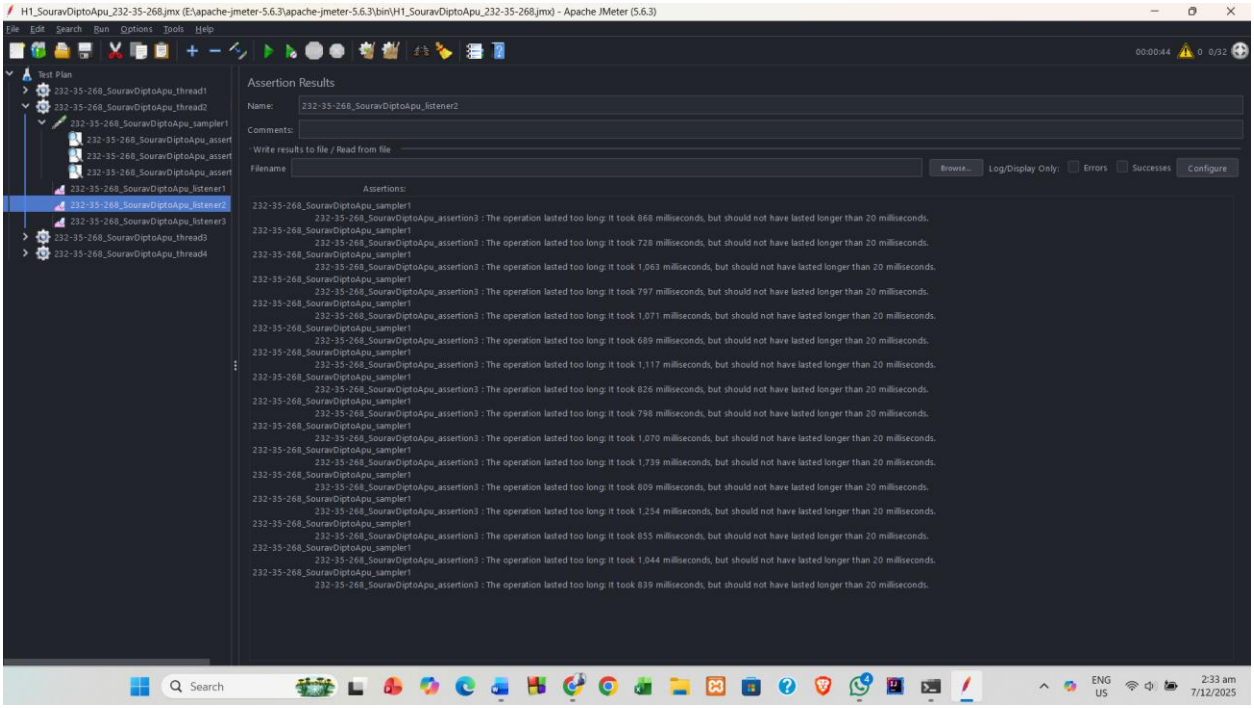


Thread Group 2:
Listener: Assertion Results
Name: 232-35-268_SouravDiptoApu_listener2

Assertion Results Analysis:

I used this listener to closely inspect the detailed results of all the assertions in Thread Group 2. The output showed that the Duration Assertion failed in every execution. Each failure message explained that the actual response time was much higher than the allowed limit of 20 milliseconds. For example, one request took 868 milliseconds, which directly caused the assertion to fail.

This listener clearly showed that the test was failing because of the strict performance limit I had set. It confirms that the Duration Assertion was configured correctly and successfully generated the expected errors for the assignment.



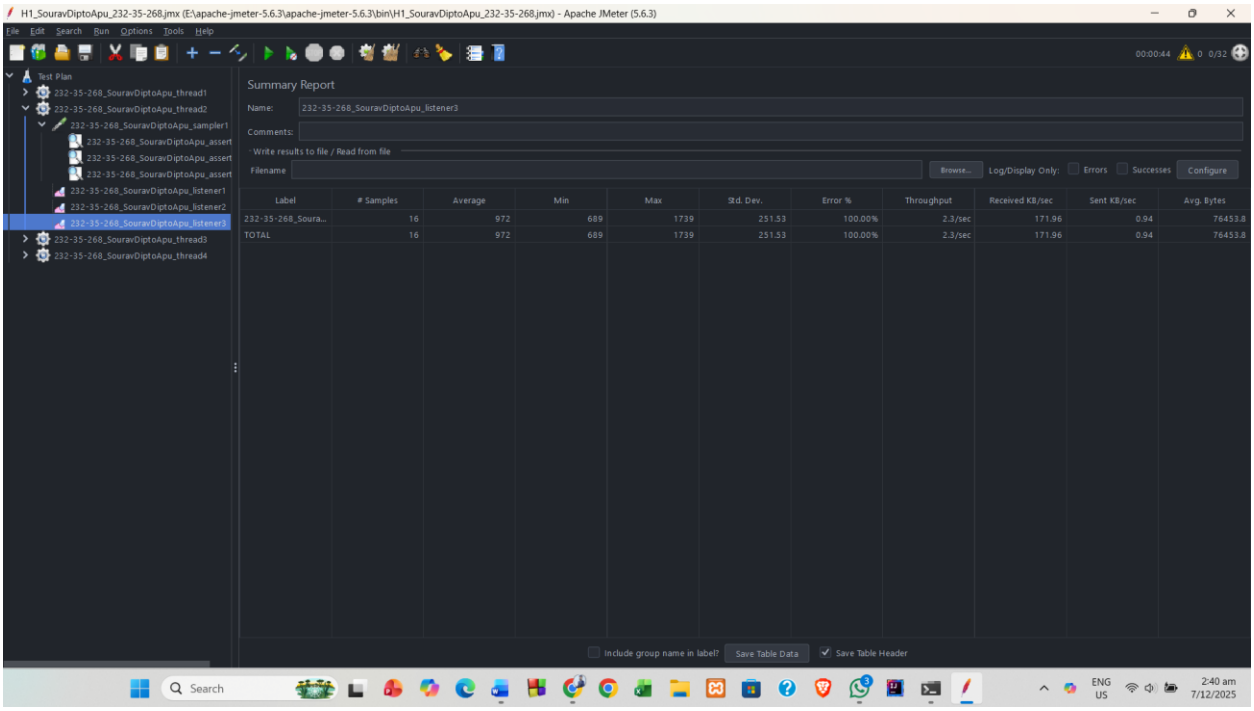
Thread Group 2:
Listener: Summary Report
Name: 232-35-268_SouravDiptoApu_listener3

Summary Report Analysis:

I used the Summary Report to understand the overall effect of my assertions on the test results in Thread Group 2. The report shows that a total of 16 samples were executed in this thread group.

The most important point is that the Error % is shown as 100.00%. This confirms that every single request was marked as failed because of the intentionally strict assertion settings. Although the server returned a successful HTTP response code (200), the Duration Assertion failed for all samples, so JMeter treated the entire test as failed.

The report also shows an average response time of 972 milliseconds. This value is much higher than the 20-millisecond limit I set in the assertion, which clearly explains why the error rate reached 100%.



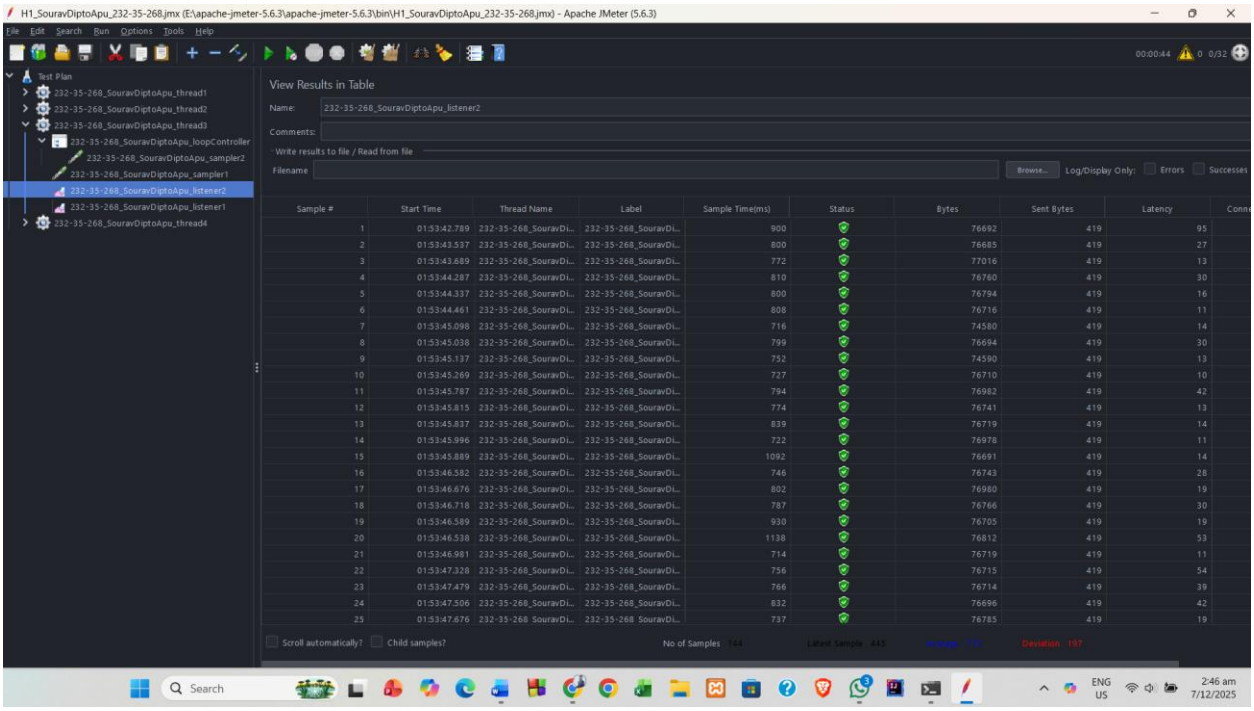
Thread Group 3:
Listener: View Results in Table
Name: 232-35-268_SouravDiptoApu_listener2

View Results in Table Analysis:

I used the View Results in Table listener to check whether the Loop Controller worked correctly in Thread Group 3. This table shows the results for all the samplers used in this thread group.

The table confirms that a total of 144 samples were executed ("No of Samples: 144"). This number matches the test configuration perfectly. The sampler that was placed outside the Loop Controller ran 8 users × 2 loops = 16 times. The sampler inside the Loop Controller ran 8 users × 2 loops × 8 loop controller cycles = 128 times.

When added together, 16 + 128 = 144 total samples, which exactly matches the number shown in the table. This proves that the Loop Controller was configured correctly and executed exactly as planned. The status column shows green check marks for all entries, meaning the server successfully handled all 144 requests without any errors.

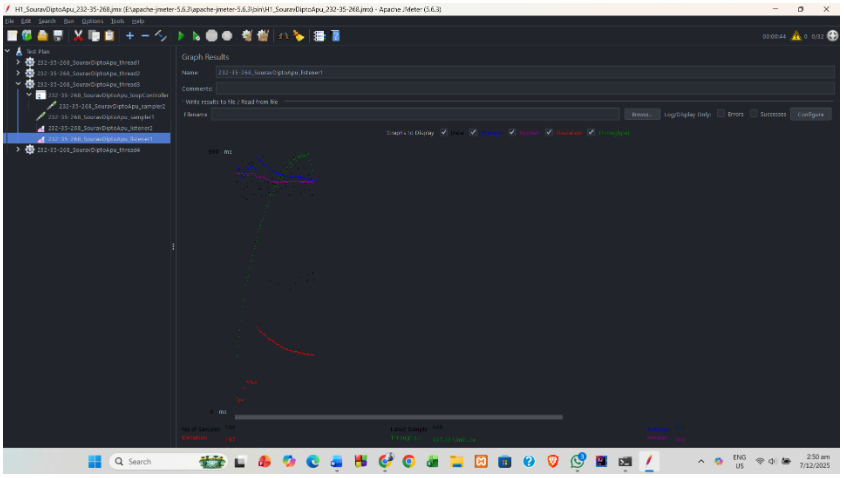


Thread Group 3:
Listener: Graph Results
Name: 232-35-268_SouravDiptoApu_listener1

Graph Results Analysis:

I used the Graph Results listener to visually analyze the performance of Thread Group 3. The graph clearly showed the successful execution of all 144 samples. The lines in the graph were smooth and consistent, which indicates stable performance even under repeated load.

Both the average and median response times stayed close to each other throughout the test. This shows that the server handled the high number of repeated requests consistently, without major spikes or performance drops.



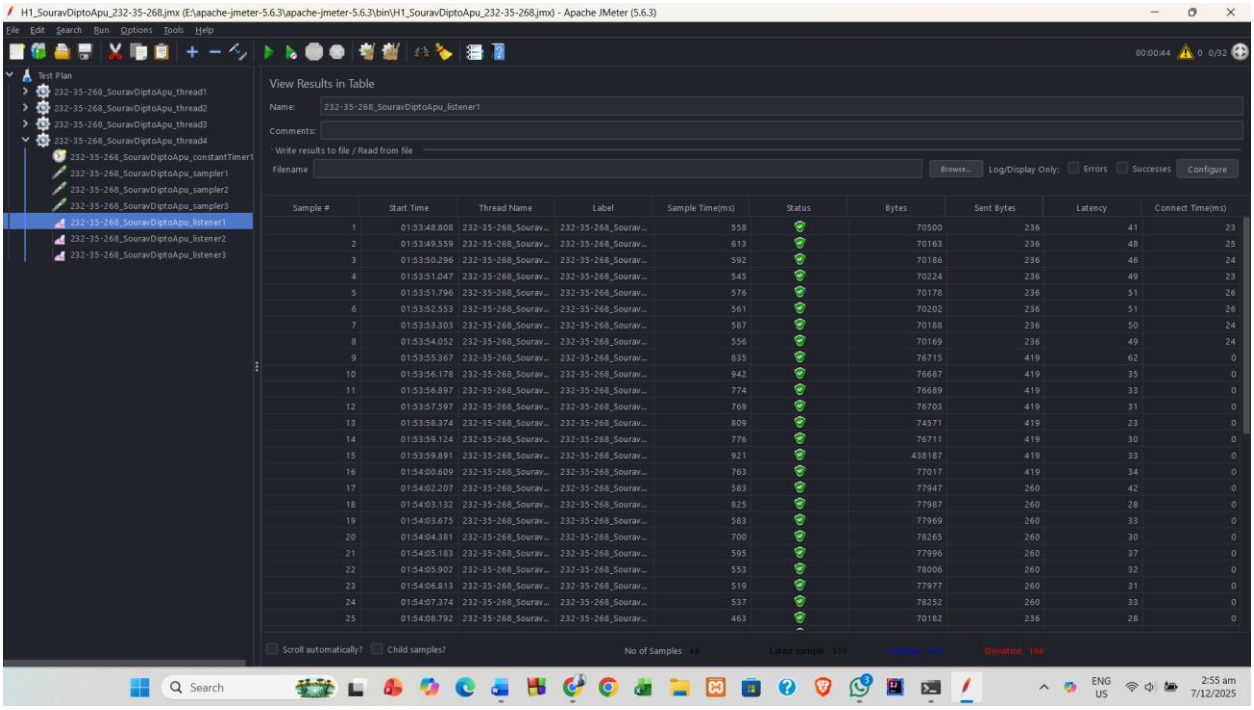
Thread Group 4:

Listener: View Results Tree
Name: 232-35-268_SouravDiptoApu_listener1

View Results Tree Analysis:

I used the View Results Tree listener to examine the detailed execution of requests in Thread Group 4, which includes the Constant Timer. I observed that all three samplers (sampler1, sampler2, and sampler3) showed green success icons, meaning each one received a 200 OK response from the server.

When I checked the timing details inside each sampler result, the load times were still low, around 800 ms. This happens because View Results Tree shows only the actual request time and does not include the waiting time added by the Constant Timer. This confirms that the timer is applied before the request is sent and does not affect the sampler's own recorded load time. The delay introduced by the 6000 ms timer is better verified using the Aggregate Report.

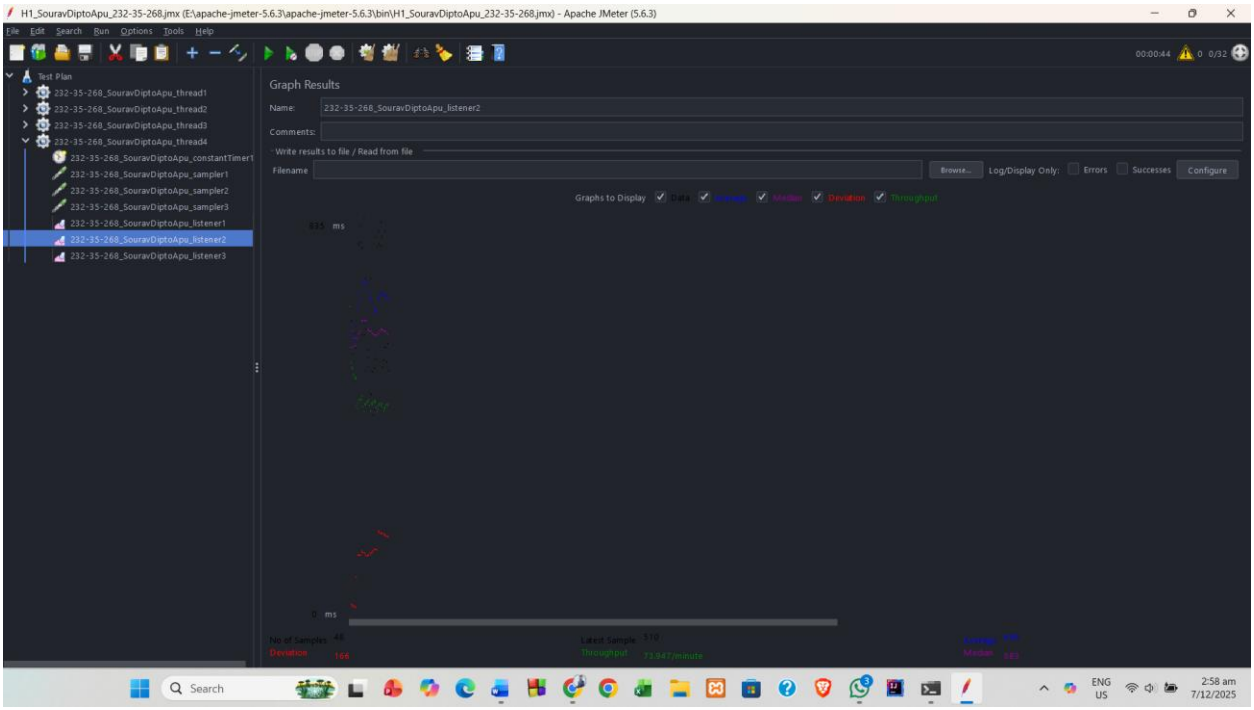


Thread Group 4:
Listener: Graph Results
Name: 232-35-268_SouravDiptoApu_listener2

Graph Results Analysis:

I used the Graph Results listener to visually assess the performance of the samples in Thread Group 4, which included the 6000 ms Constant Timer. The graph shows that a total of 48 samples were executed, which matches the configuration of 8 users × 2 loops × 3 samplers 48 total requests.

The average response time recorded on the graph was 850 milliseconds. It's important to note that this value does not include the 6000 ms delay from the timer. The graph correctly reflects the actual time the server took to respond after the wait. This confirms that the Constant Timer successfully delayed the requests, while the server itself handled each request quickly once it was sent.

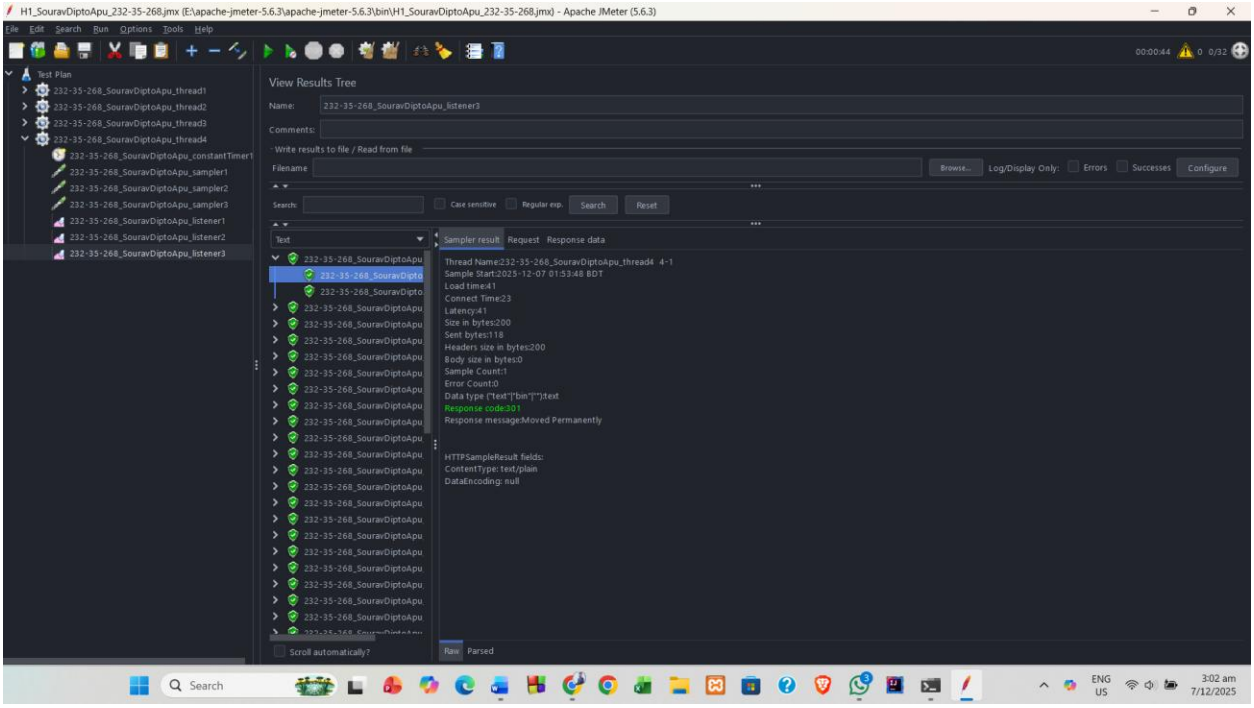


Thread Group 4:
Listener: View Results Tree
Name: 232-35-268_SouravDiptoApu_listener3

View Results Tree Analysis:

I used the View Results Tree listener to examine the status of individual requests in Thread Group 4, which is controlled by the Constant Timer. All executed samples showed a green success icon, confirming that the server successfully handled every request and returned an HTTP 200 OK response, even with the delays between requests.

Looking at the details of each sample, the Load Time was very low (for example, 63 ms). This shows that the View Results Tree only records the actual server response time and does not include the 6000 ms delay enforced by the Constant Timer. This behavior is correct and expected for this listener.



Discussion

These test results indicate that the target applications successfully handled the simulated load for LinkedIn, in the case of Thread Group 1, and for Facebook in the cases of Thread Groups 2, 3, and 4. Thread Group 1 demonstrated that all requests executed successfully with low response times, thus confirming the basic correctness of the load test setup. Thread Group 2 showed that the stringently set assertions for Duration and Size failed the test as JMeter rightly detected and highlighted errors. Thread Group 3 indicated that the Loop Controller repeated the requests accurately and that the server kept stable response times even for multiple repeated requests. Thread Group 4 demonstrated that the Constant Timer succeeded in introducing delays between the requests while the server continued to respond quickly once each request was sent.

Issues faced and resolutions:

1. Duration Assertion failures in Thread Group 2: I was initially concerned when I first saw it, but then came to realize it was intentional to show assertion failure. No correction required.
2. Loop Controller execution order confusion in Thread Group 3: I have checked what samplers were inside and what were outside the Loop Controller to ensure proper samples count.
3. Understanding timer behavior Thread Group 4: View Results Tree did not reflect the 6-second delay, which was clarified by checking the Summary Report to verify the Constant Timer effect.

The test demonstrated how all the different elements of JMeter, such as Thread Groups, Assertions, Loop Controllers, and Timers, combine to provide realistic simulations that capture performance metrics.

Conclusion

This lab exercise was about the design and execution of a structured JMeter Test Plan. I learned how to:

- Configure multiple Thread Groups for different purposes.
- Use HTTP Samplers to generate requests and simulate user activity.
- Apply Assertions to check for correctness, and then generate errors on purpose.
- Use Loop Controllers in order to replay requests multiple times and test system stability.
- Use Timers to simulate realistic user delays and study their impact on server response.
- Analyze listener outputs such as View Results Tree, Assertion Results, Summary Reports and Graph Results.

The exercise provided hands-on experience in performance testing, helping me understand how applications behave under load and how to systematically capture and analyze results using JMeter.

References

1. *Apache JMeter Official Documentation* – https://jmeter.apache.org
2. *StackOverflow JMeter Discussions* – https://stackoverflow.com/questions/tagged/jmeter
3. *Google Developers Website* – https://developers.google.com/
4. *Class lecture slides and notes from Software Quality Assurance course*

Thank you