**TEST SUMMARY REPORT**

**For**

**Simple Gallery System**

| Version | Author | Date | Remarks |
| --- | --- | --- | --- |
| 1.0.0 | Victor kipngetich | 11/10/2024 | Initial |

**Contents**

[.1 Summary 3](#_Toc179901919)

[.2 Purpose 3](#_Toc179901920)

[.3 Test Items 3](#_Toc179901921)

[.4 Testing Environment: 3](#_Toc179901922)

[.5 Load testing performance 4](#_Toc179901923)

[.6 Stress testing performance metrics 8](#_Toc179901924)

[.7 Conclusion 11](#_Toc179901925)

[.8 Test Activity Schedule 11](#_Toc179901926)

## Summary

On 12th October 2024, I conducted performance Test for the login page of the Simple Gallery system to assess its performance under load and stress conditions. The primary objective was to verify whether the login functionality met the specified system requirements and effectively validated user credentials.

During the testing, I simulated a range of user loads to evaluate the system's responsiveness and stability. Load testing aimed to determine how the system performs under expected user traffic, while stress testing sought to identify its limits by applying higher-than-normal loads

## Purpose

* **Ensure Stability**: Determine the application stability and reliability during high user loads to prevent potential failures that could disrupt user experience.
* **Identify Limits**: Establish the system capacity limits by applying stress conditions to uncover any weaknesses that need to be addressed

## Test Items

The following modules/functions were the focus points for the QA/UAT exercise.

* **Response Time**: Measure the time taken for the login page to respond under different user loads to ensure it meets performance benchmarks.
* **Error Rate**: Track the frequency of failed login attempts, assessing the system’s robustness under both normal and peak loads.
* **Latency**: Monitor the time delay between a user action (such as submitting login credentials) and the system's response, especially under heavy load conditions.

|  |  |  |
| --- | --- | --- |
| **Tool** | **IP** | **Function** |
| JMeter | 8080 | Script Generation |
| Chrome | 8080 | Running the program |
| Firebase console | -- | Authentications |
| Xammp server | 8080 | Server environment |

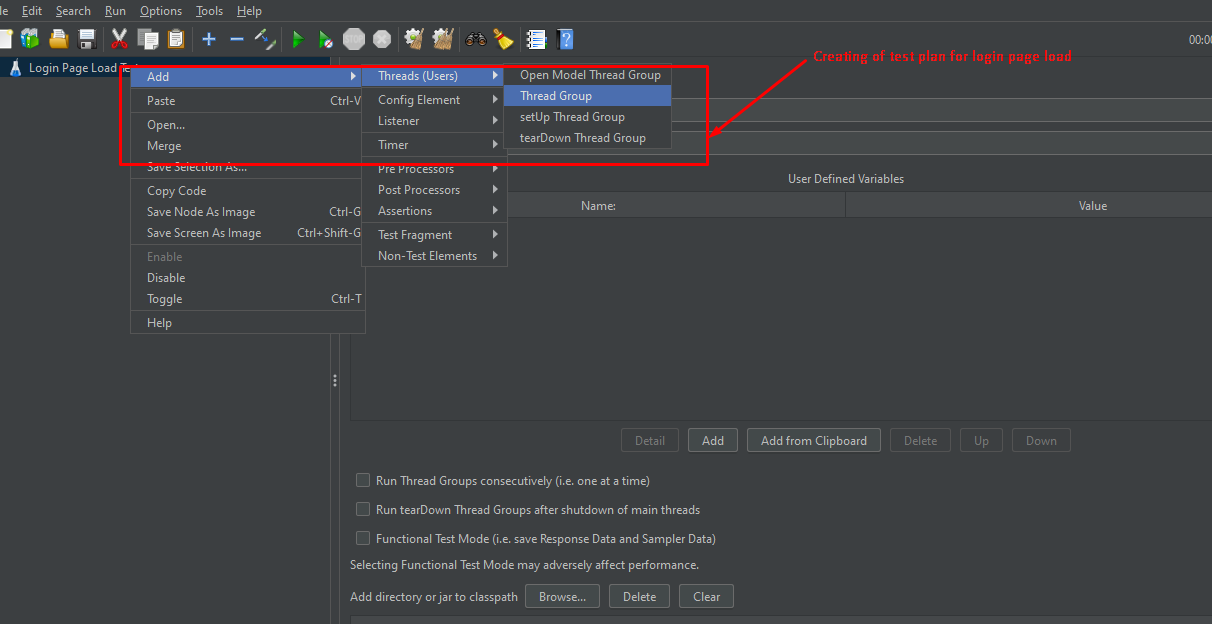
## ****Testing Environment****:

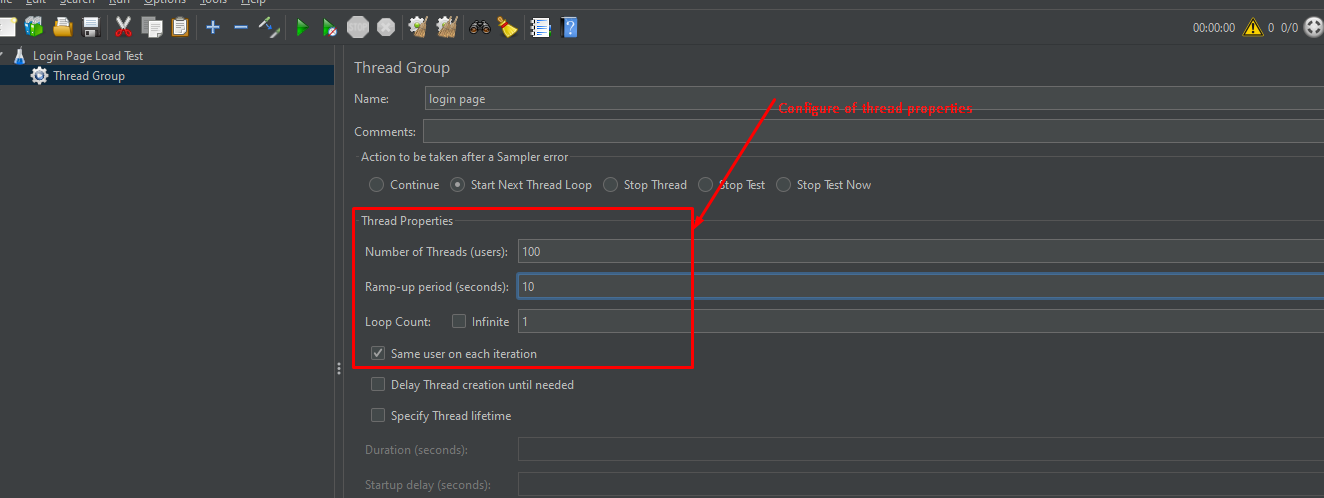
* **URL**: <http://localhost/gallery/login.php>

## Load testing performance

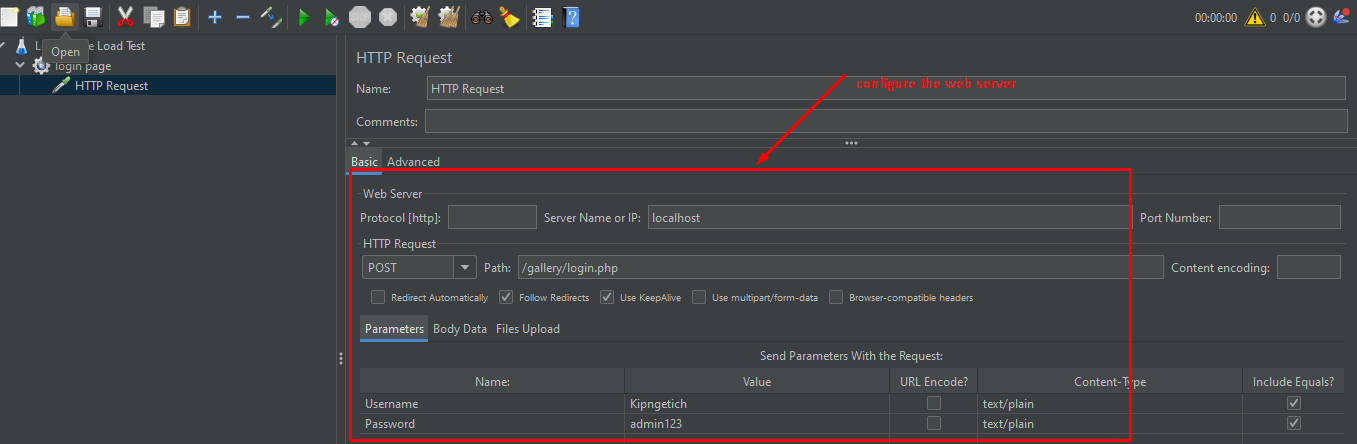
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User** | **Pass** | **Fail** | **App CPU** | **DB CPU** | **App Memory Utilization** | **DB Memory Utilization** |
| 100 every 10 secs | 100 | 0 | 17.8 | 4.6 | 29987 | 1508736 |
| 200 every 10 secs | 200 | 0 | 25.4 | 7.7 | 30445 | 1509186 |
| 500 every 10 secs | 500 | 0 | 27.1 | 6.2 | 30966 | 1509409 |

1. The test plan for load testing involves creating a Thread Group to simulate 100 users with a ramp-up time of 10 seconds and a loop count of 1. This setup will help assess the system's responsiveness and stability under a moderate load by configuring HTTP Request samplers to define the requests and adding listeners to monitor performance metrics, as illustrated in the graph below

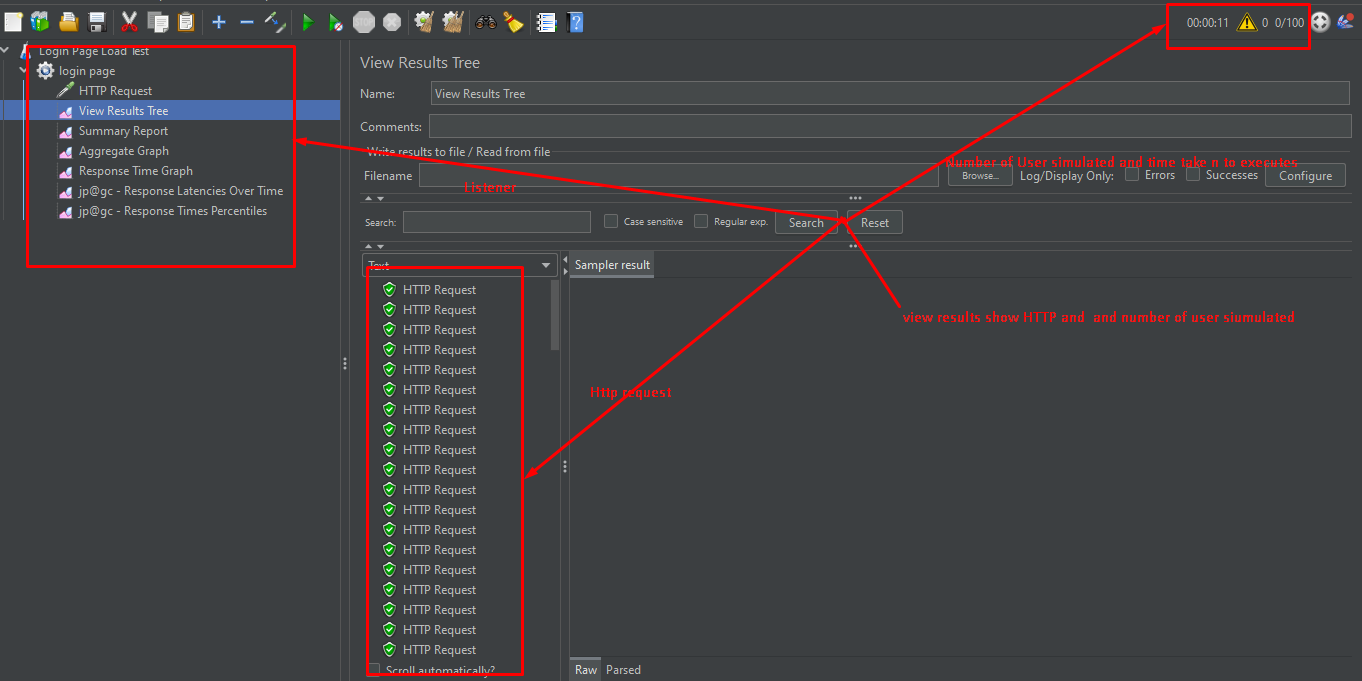




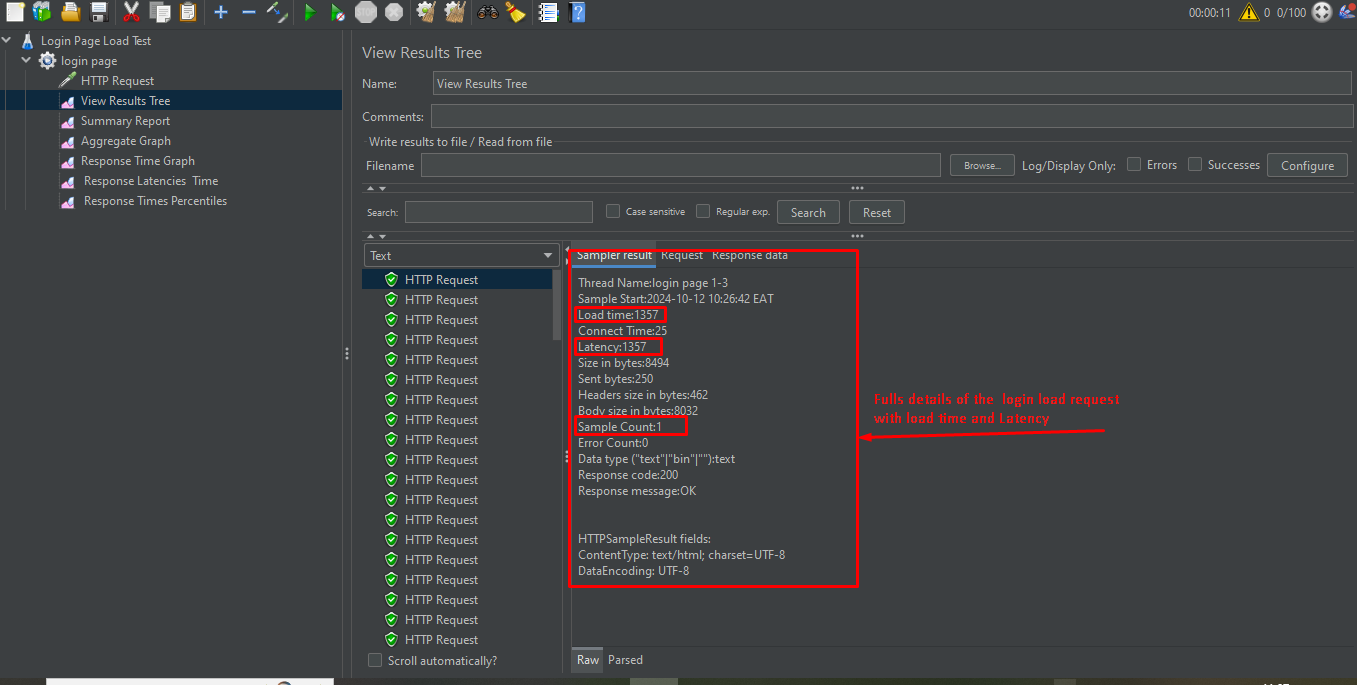
1. Creating and configuring web server of web application



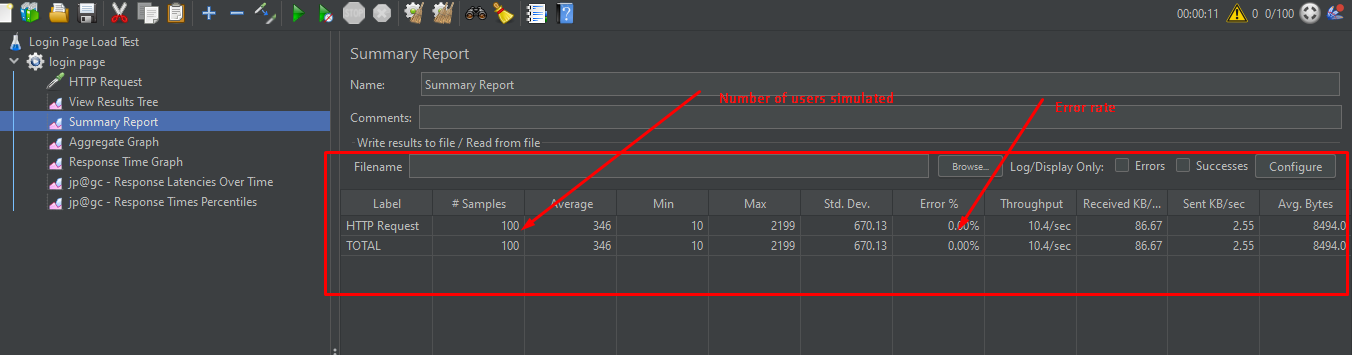
1. Adding listeners in JMeter allows you to monitor and analyze the performance of your application during testing as illustrated in the graph below



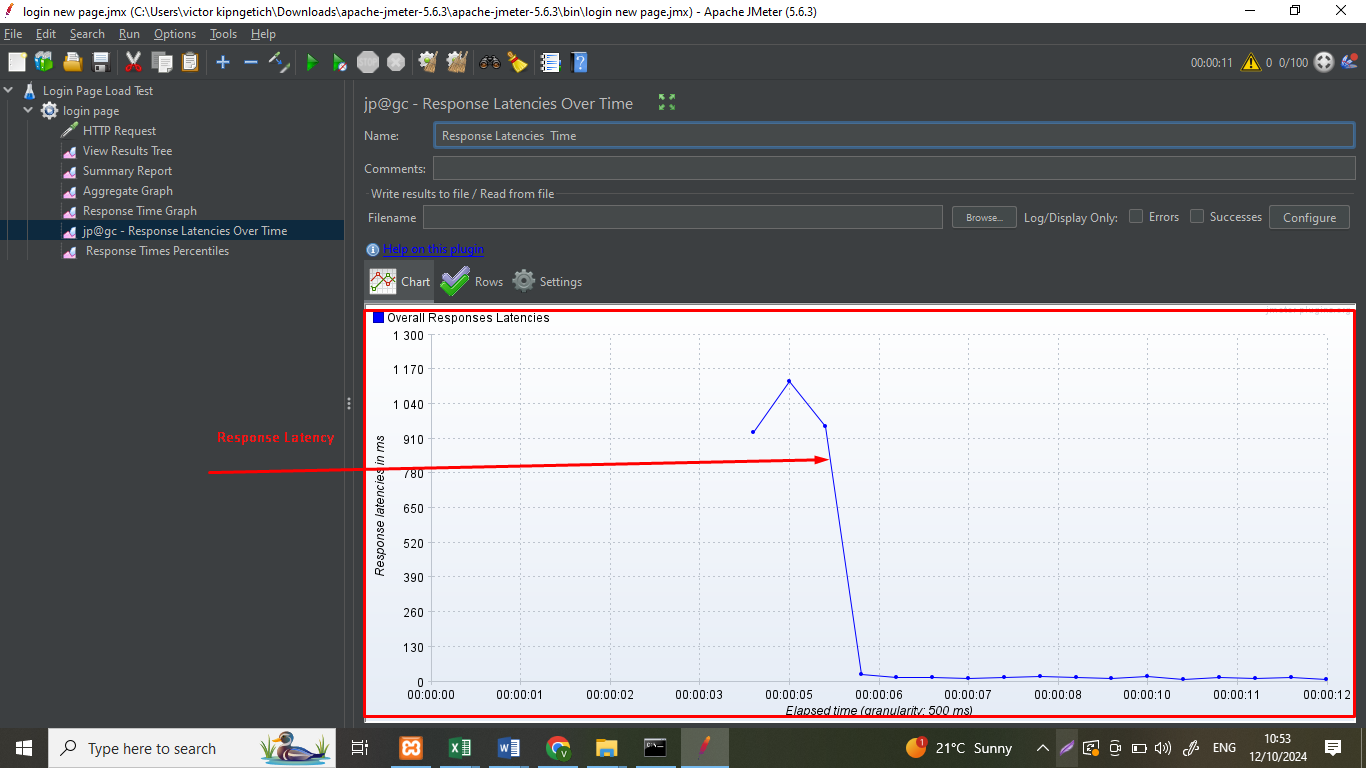
1. Upon clicking the 'View Results' listener in JMeter, you will see more detailed information, including start time, load times, connection time, latency, and status. This data provides insights into the performance of each request, allowing for a comprehensive analysis of the application's responsiveness and potential areas for improvement, as illustrated in the graph below



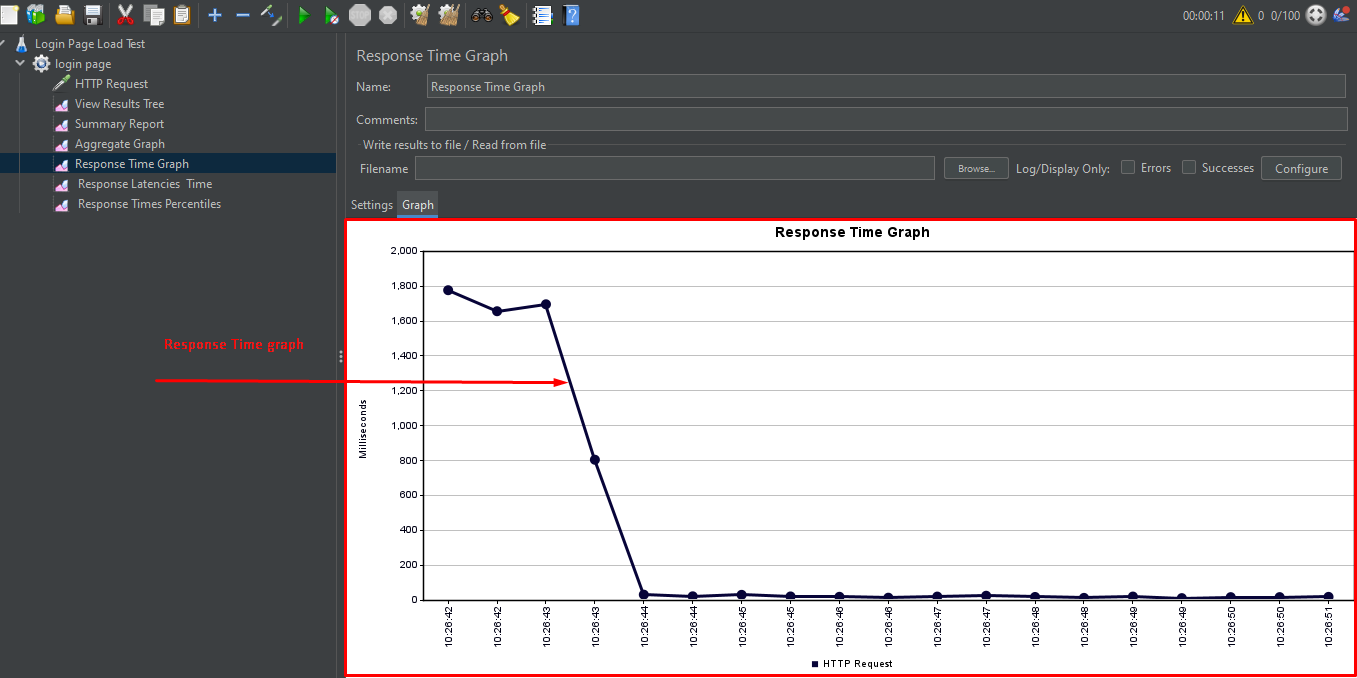
1. Upon clicking the 'Summary Report' listener in JMeter, you will see a comprehensive summary of the requests, including the number of users simulated, error rates, average response times, and additional performance metrics. This report provides a quick overview of the test results, helping to identify trends and assess the overall performance of the application under load, as illustrated in the graph below



1. Response latency graph



1. Response time graph



## Stress testing performance metrics

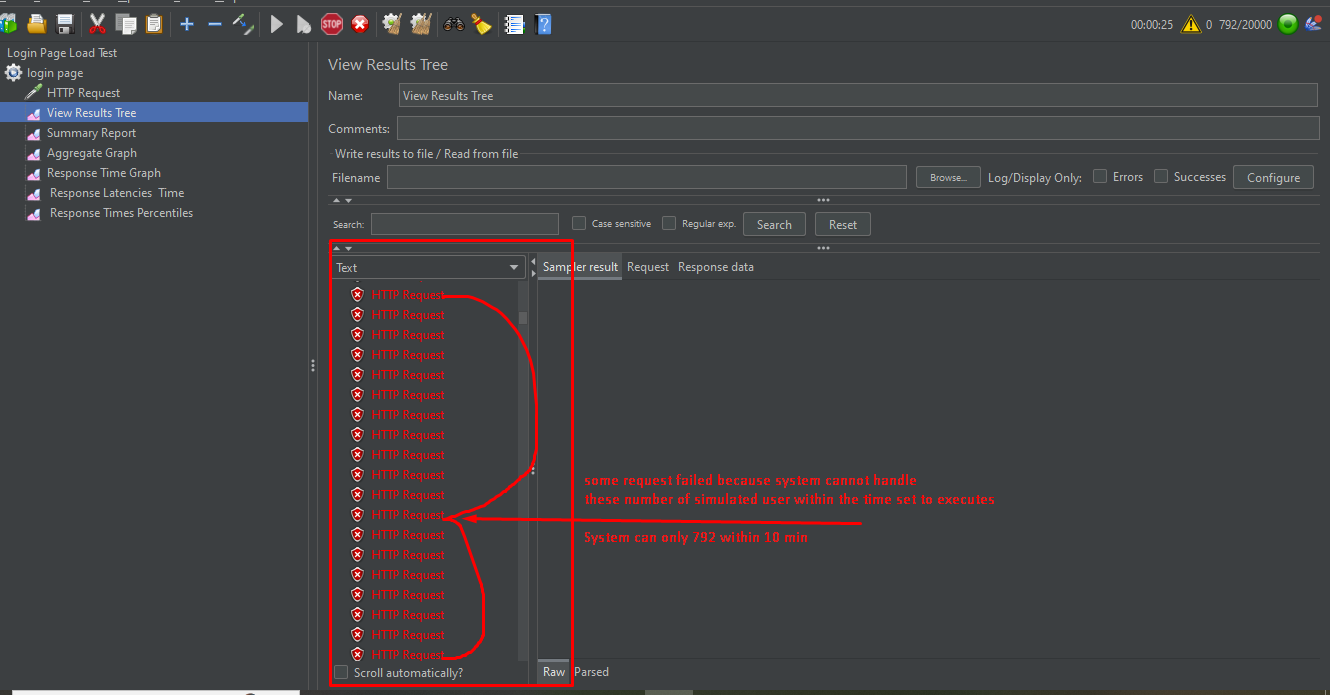
**Number of users simulated =20000 users**

**Ramp up time 10 seconds**

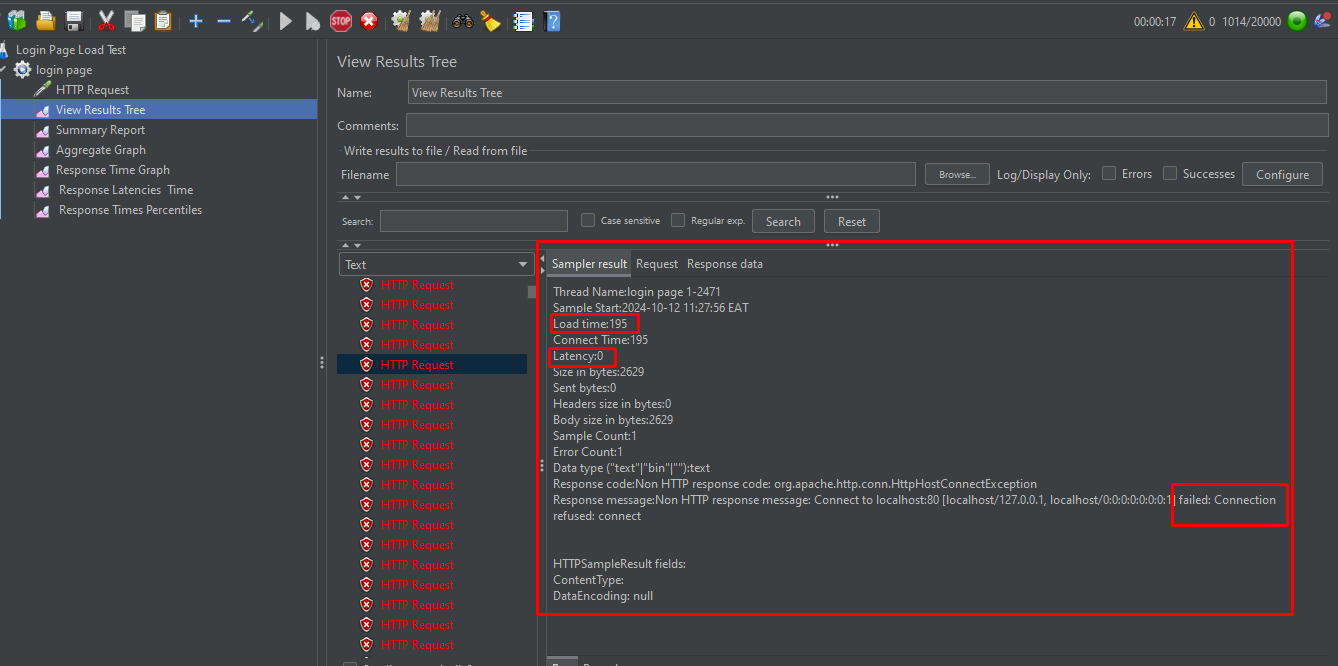
**Loop count 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User** | **Pass** | **Fail** | **App CPU** | **DB CPU** | **App Memory Utilization** | **DB Memory Utilization** |
| 20000 every 10 secs | 1014 | 18986 | 70.8 | 12.6 | 2998789 | 15087362348 |
| 25000 every 10 secs | 811 | 24189 | 86.4 | 30.7 | 3044556 | 15091861347 |
| 30000 every 10 secs | 676 | 29324 | 92.1 | 50.2 | 3096645 | 15094099038 |

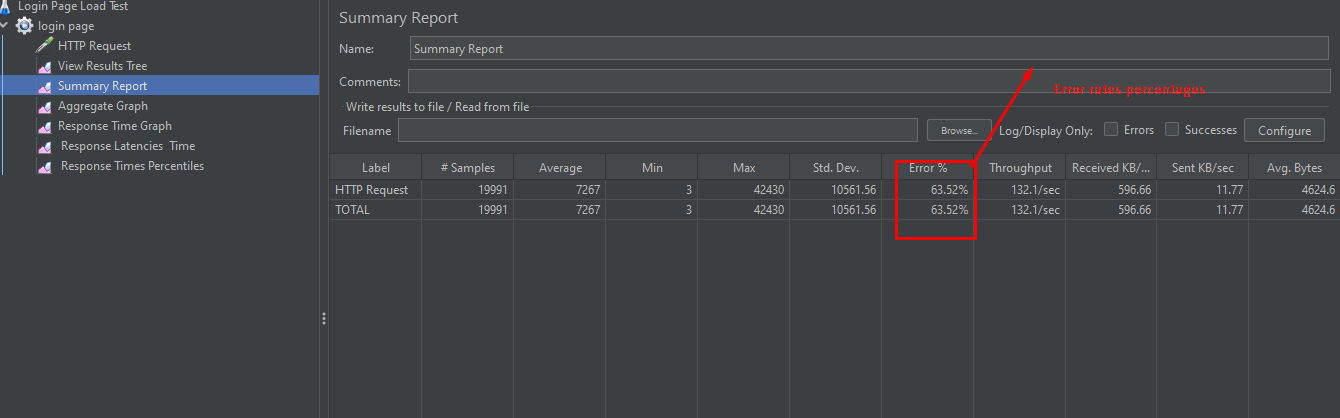
1. During the simulation involving 20,000 threads and various user properties, we observed that some HTTP requests successfully completed, while others failed. The failures were primarily due to the inability of those requests to be validated within the designated 10-second ramp-up time. This ramp-up period is crucial for allowing the system to adjust to the increased load and establish stable connections, as illustrated in the graph below



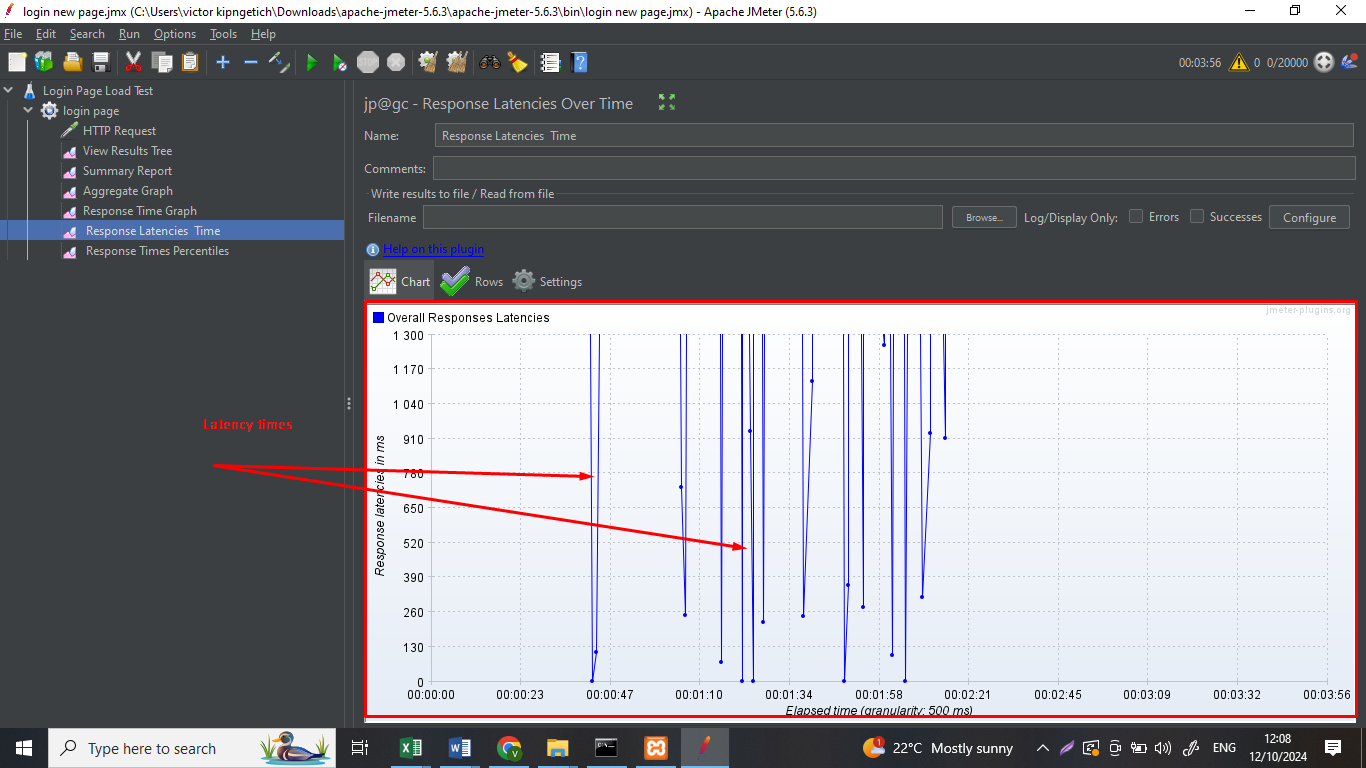
1. Clicking on the HTTP Request reveals that the login failed because the users could not complete execution within the allocated time or ramp-up period, resulting in a latency time of 0, as illustrated in the graph below



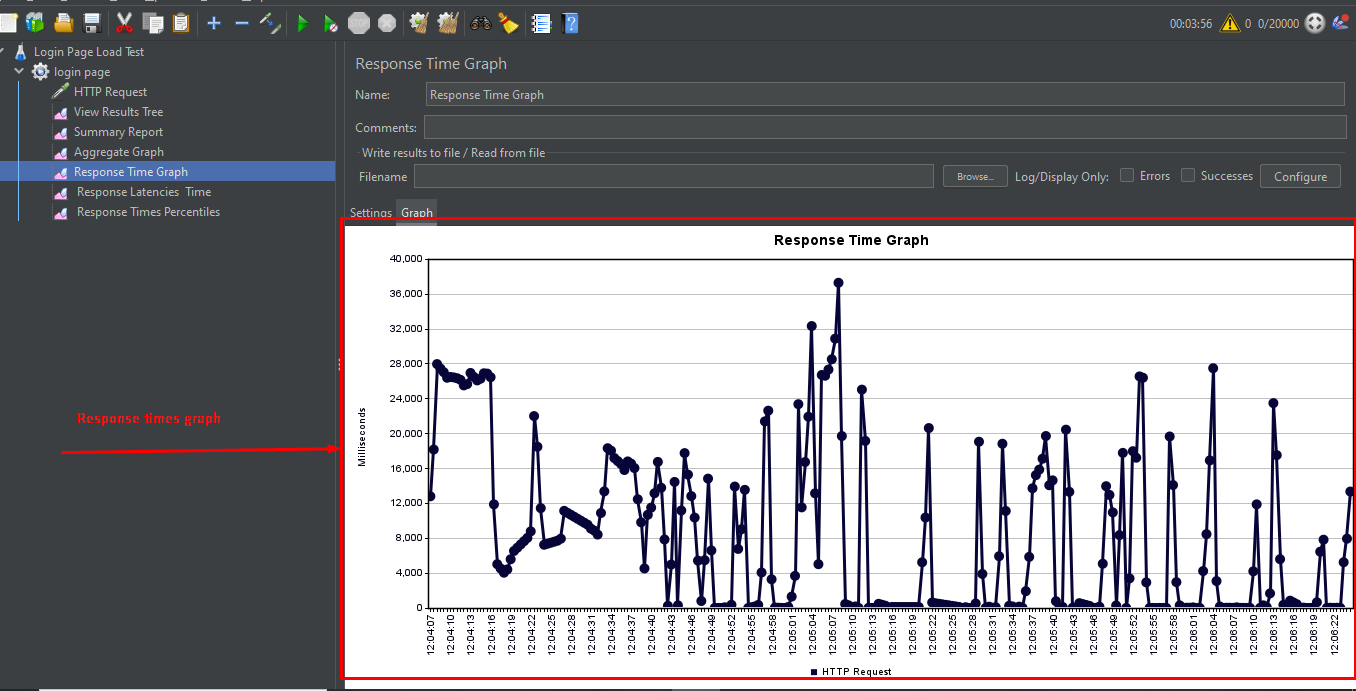
1. The request error rate currently stands at 63.52%, indicating that the majority of requests encountered failures during processing. This high error rate not only reflects significant performance issues but also suggests that the system is struggling to handle the load effectively. As a result, my laptop experienced noticeable slowdowns, affecting overall usability.



1. **Latency Time**: During stress testing, the latency time exhibits frequent fluctuations and appears inconsistent. These variations are attributed to the high number of simulated users, which strain the system's resources. As the number of concurrent user’s increases, the time it takes for the system to respond to requests becomes unstable, leading to unpredictable response times.



1. Response time graph – The response time graph during stress testing exhibits significant fluctuations, indicating inconsistent performance. These variations suggest that the system struggles to maintain stable response times under load, which could impact user experience and application reliability. The erratic nature of the graph highlights potential issues with resource allocation, network latency, or processing capacity, warranting further investigation to identify and address the underlying causes as illustrated in the graph below



## Conclusion

The simulation with 20,000 threads revealed significant performance issues:

Load testing

* **Response Time**: The response time graph displayed consistent performance under load, which can positive affect user experience.
* **Error Rate**: The error rate reached 0.00% showing that the all of HTTP requests passed highlighting the system manage the load effectively.

Stress Testing

* **Response Time**: The response time graph displayed considerable fluctuations, indicating inconsistent performance under stress, which can negatively affect user experience.
* **Error Rate**: The error rate reached 63.52%, showing that the majority of HTTP requests failed, highlighting the system's struggle to manage the stress effectively.
* **Latency Time**: Latency times fluctuated frequently due to the high number of simulated users, leading to unpredictable response times and potential user dissatisfaction.

## Test Activity Schedule

|  |  |  |
| --- | --- | --- |
| **#** | **Activity** | **Number of Days** |
| 1 | Test data creation | 2 (11th October 2024) |
| 2 | Test script creation/debugging | 4 (12th October 2024) |
| 3 | Test Execution and reporting | 2 (12th October 2024) |