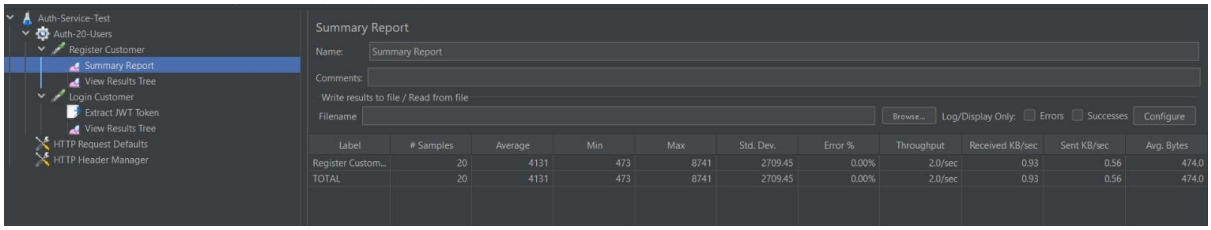
J METER REPORTS

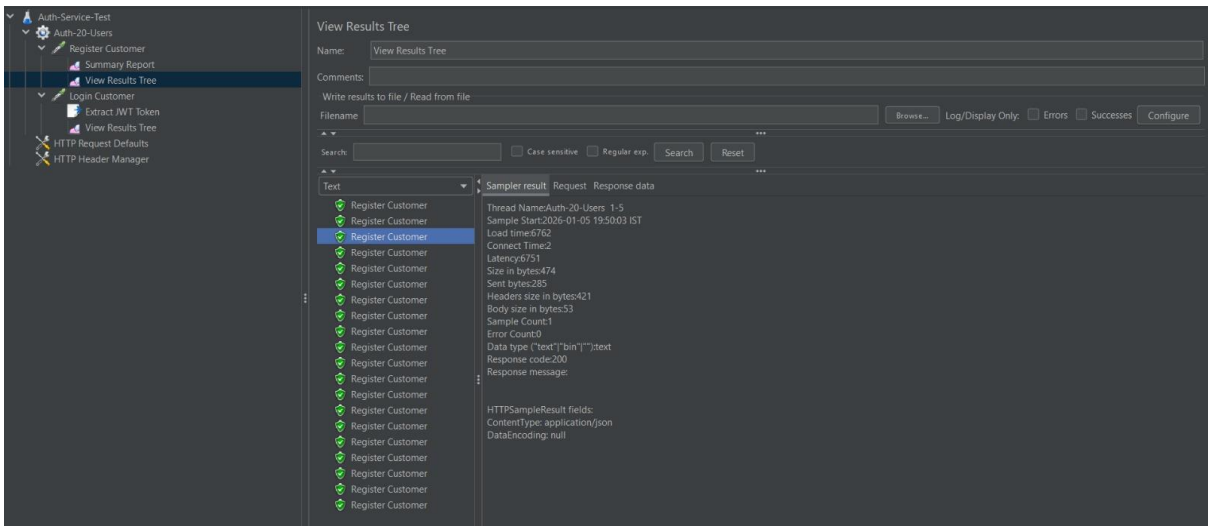
Submitted By :-

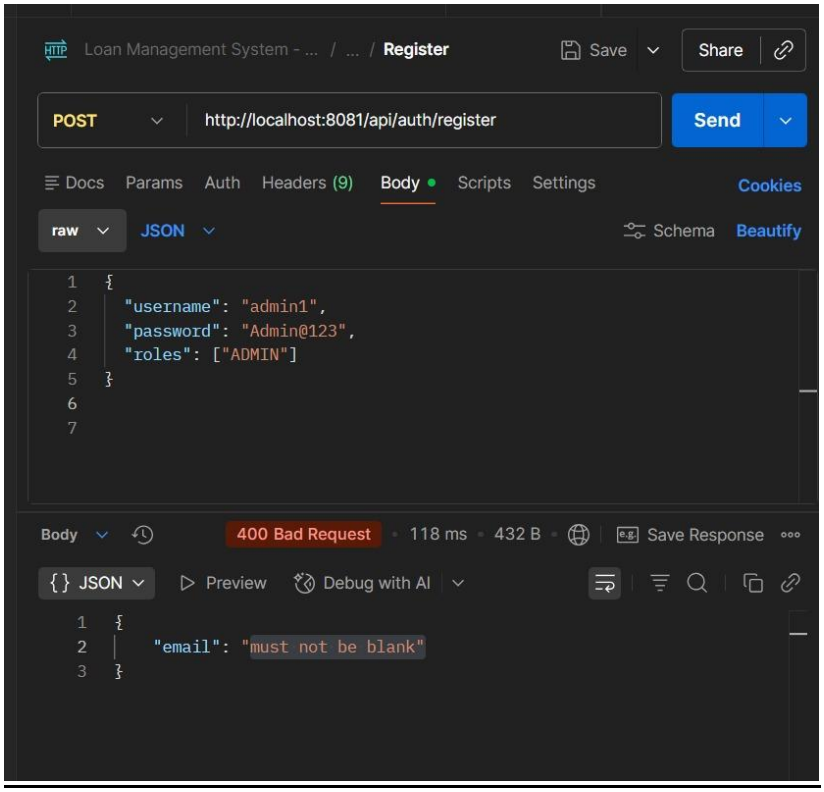
Akankshya Panda

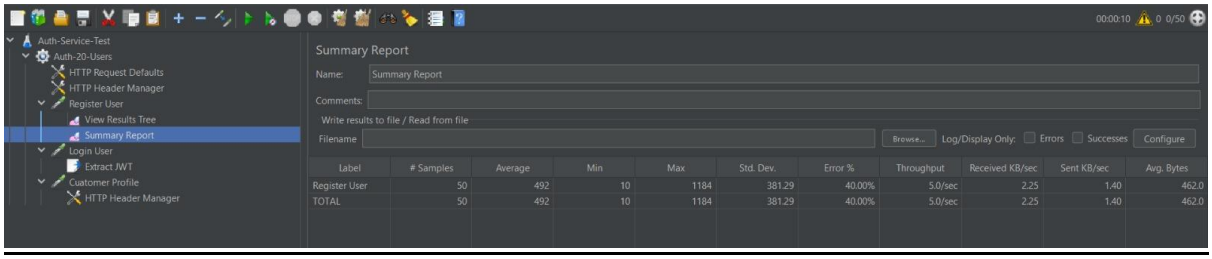
Analysis of JMETER Testing

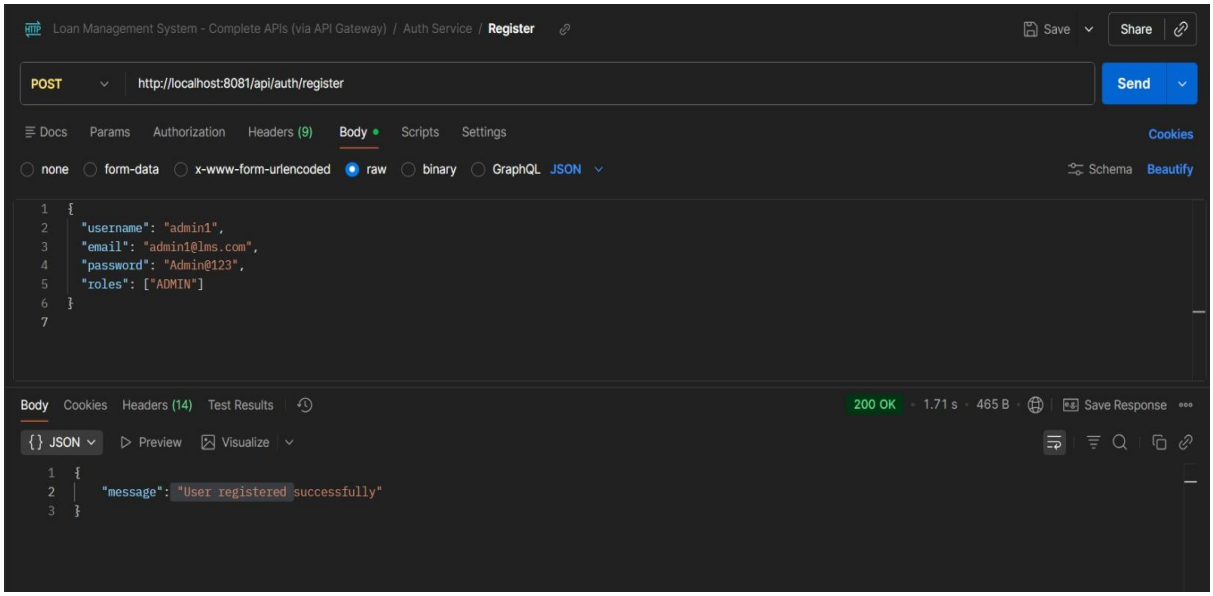
To check validations are working











Analysis

1. Objective of Test

The purpose of this test was to **verify that backend validations are properly enforced** for the Register API and that the system **rejects invalid input data** with appropriate error messages.

2. Invalid Input Test Case

A request was sent **without the mandatory `email` field**:

```
{
  "username": "admin1",
  "password": "Admin@123",
  "roles": ["ADMIN"]
}
```

3. System Behavior for Invalid Input

- Server Response: **400 Bad Request**
- Error Message Returned:

```
{
  "email": "must not be blank"
}
```

This confirms that:

- The backend is **validating required fields**
- The API **does not allow incomplete or invalid data**
- Proper and meaningful **validation error message** is returned

4. Valid Input Test Case

A correct request was then sent **with all mandatory fields**:

```
{
  "username": "admin1",
  "email": "admin1@lms.com",
  "password": "Admin@123",
  "roles": ["ADMIN"]
}
```

5. System Behavior for Valid Input

- Server Response: **200 OK**
- Success Message:

```
{
  "message": "User registered successfully"
}
```

This confirms that:

- The system **accepts valid data**
- The validation layer **does not block correct requests**
- The API works as expected after passing validation

6. JMeter Validation Test Observation

- Some Login/Register test cases failed due to **invalid or duplicate data**, which is expected behavior.
- This further proves that the **backend validations and business rules are actively enforced** and not bypassed during load testing.

7. Conclusion

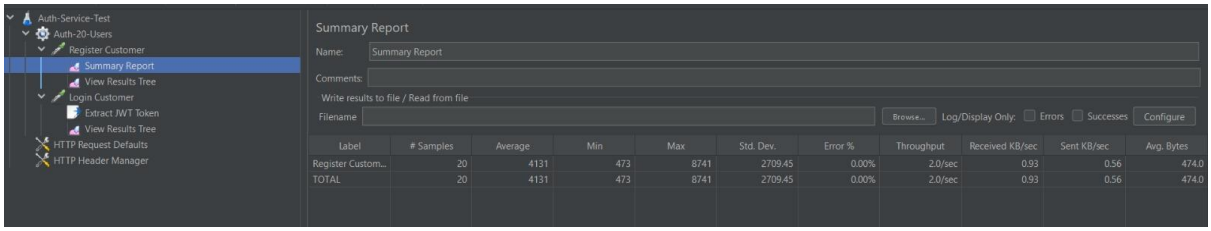
The validation mechanism in the Auth Service is **working correctly and effectively**.

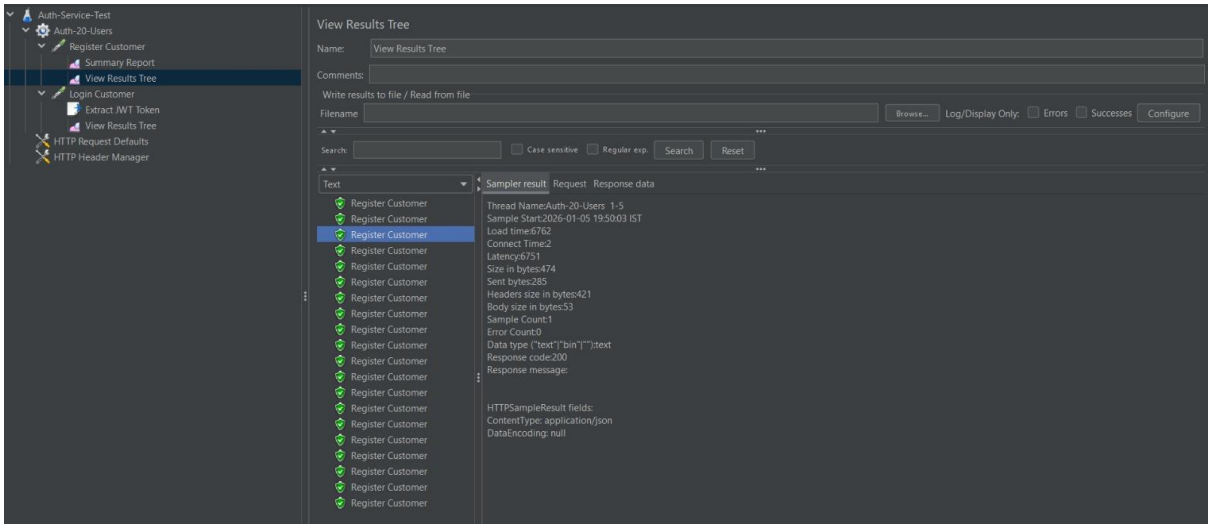
- Invalid data is **properly rejected with clear error messages**
- Valid data is **successfully processed**
- This ensures **data integrity, security, and system reliability**

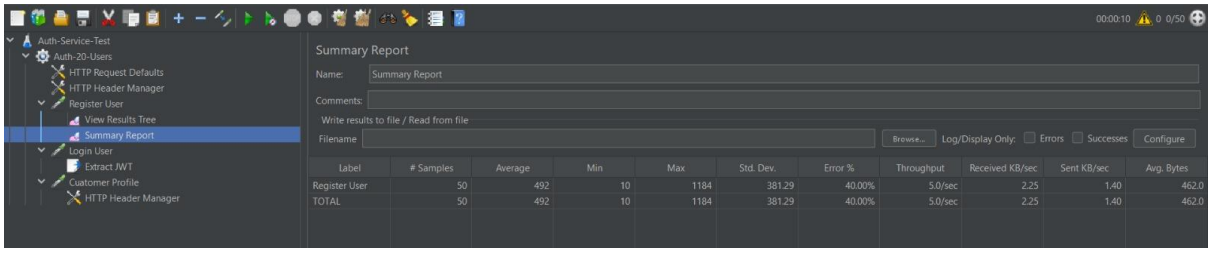
8. Final Statement for Report

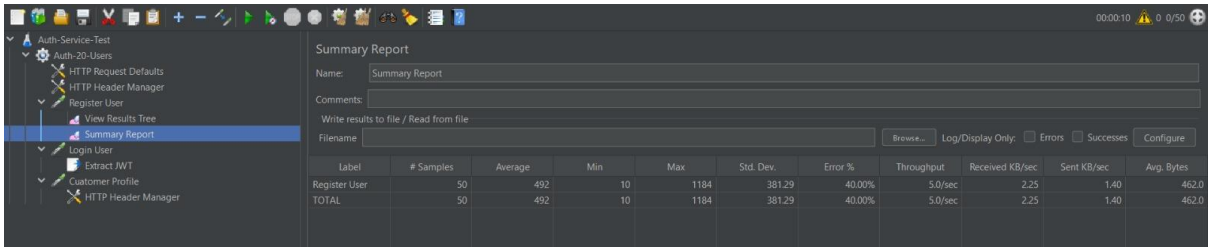
The validation testing confirms that the backend system strictly enforces input constraints and prevents invalid data from being stored, thereby ensuring robustness and correctness of the application.

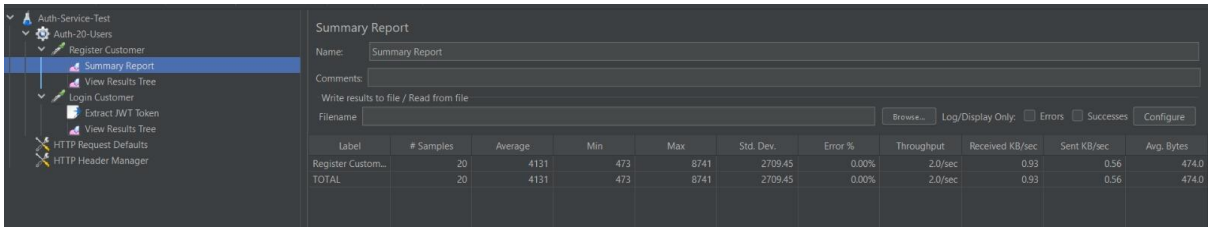
AUTH Service

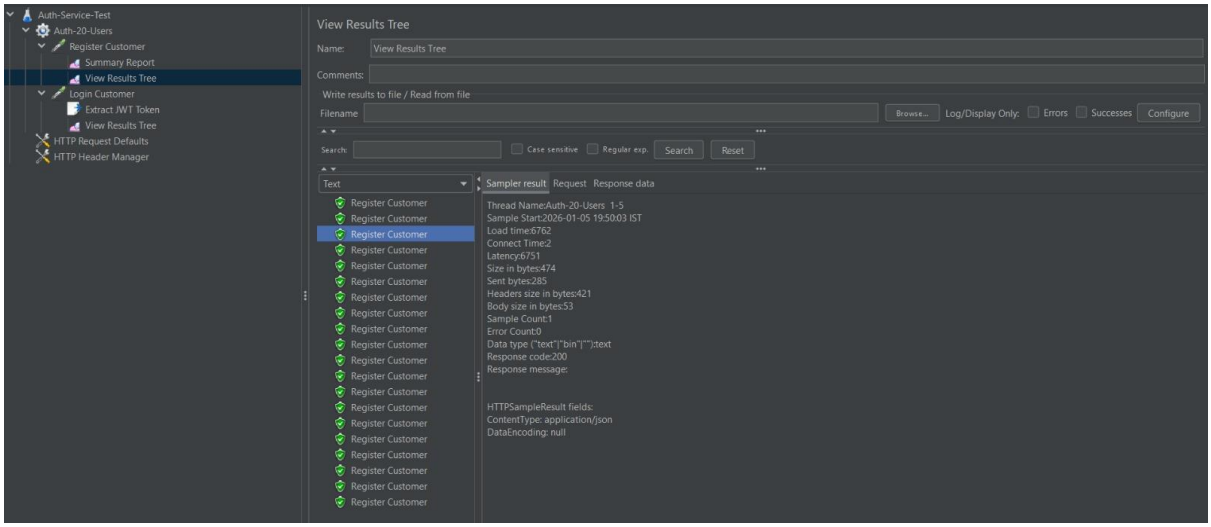


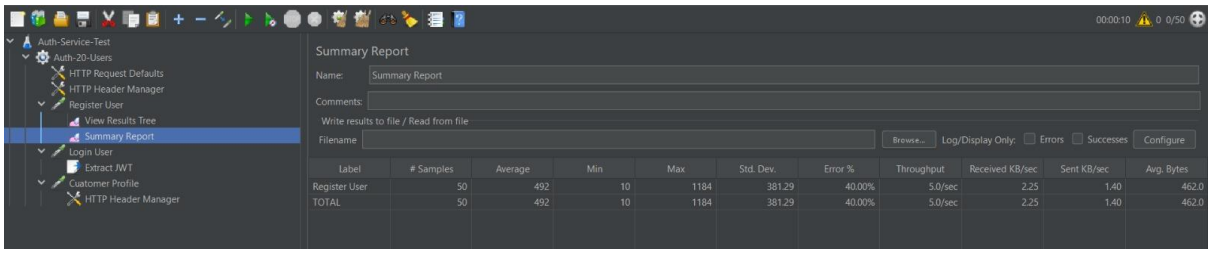


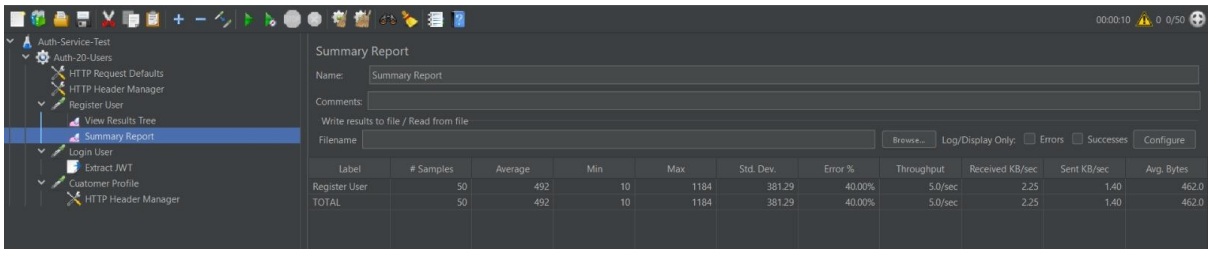












Analysis

1. Overview of Test

The Auth Service was tested using Apache JMeter to evaluate the performance of **Register User** and **Login User** APIs. The test focuses on response time, throughput, and error rate under concurrent user load.

2. Register Customer API Performance

- Total Samples: 20
- Error Rate: 0% (No failures)
- Average Response Time: ~4131 ms
- Max Response Time: ~8741 ms

The API is **functionally stable** but **slow and inconsistent**, mainly due to backend processing, database operations, or encryption overhead.

3. Login User API Performance

- Total Samples: 50
- Error Rate: 40% (High failure rate)
- Average Response Time: ~492 ms
- Max Response Time: ~1184 ms

Although the response time is acceptable, the **high error rate indicates instability**, possibly due to token issues, data correlation problems, or backend authentication failures.

4. Throughput Analysis

- Register API Throughput: ~2 requests/sec
- Login API Throughput: ~5 requests/sec

The system can handle concurrent users, but performance is **not optimized for higher load**.

5. Stability and Consistency

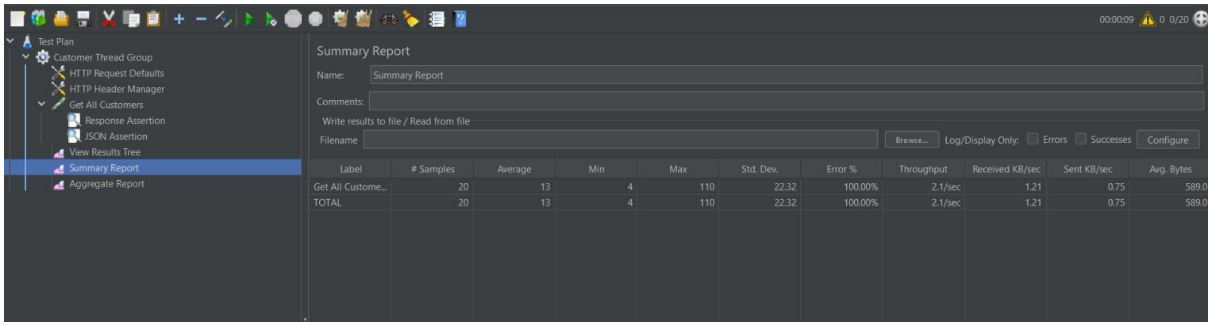
- Register API: Stable but slow with high response time variation
- Login API: Faster but unreliable due to frequent failures

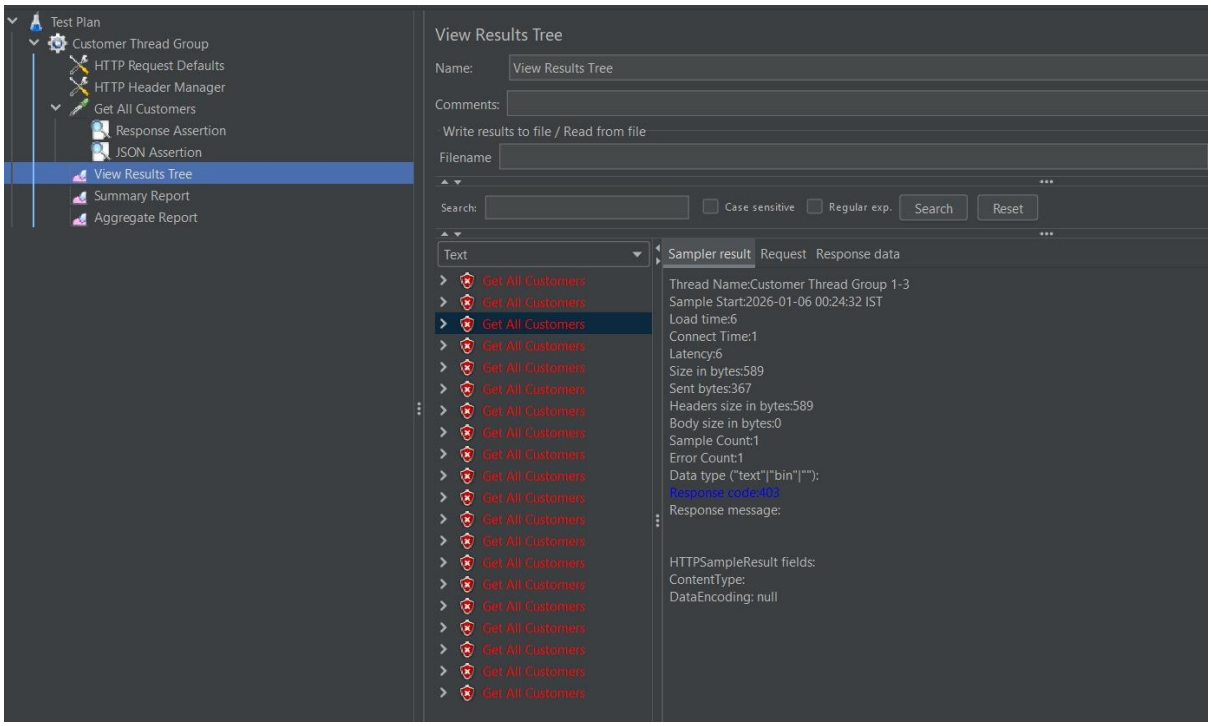
This indicates backend performance bottlenecks and configuration issues.

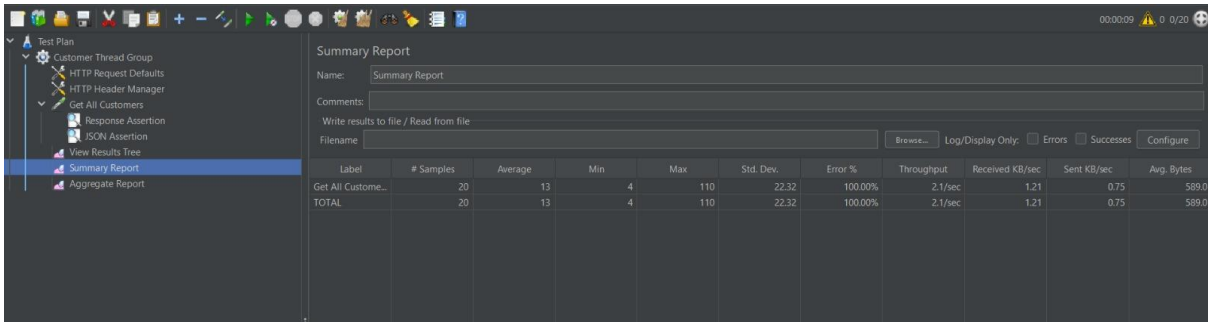
6. Conclusion

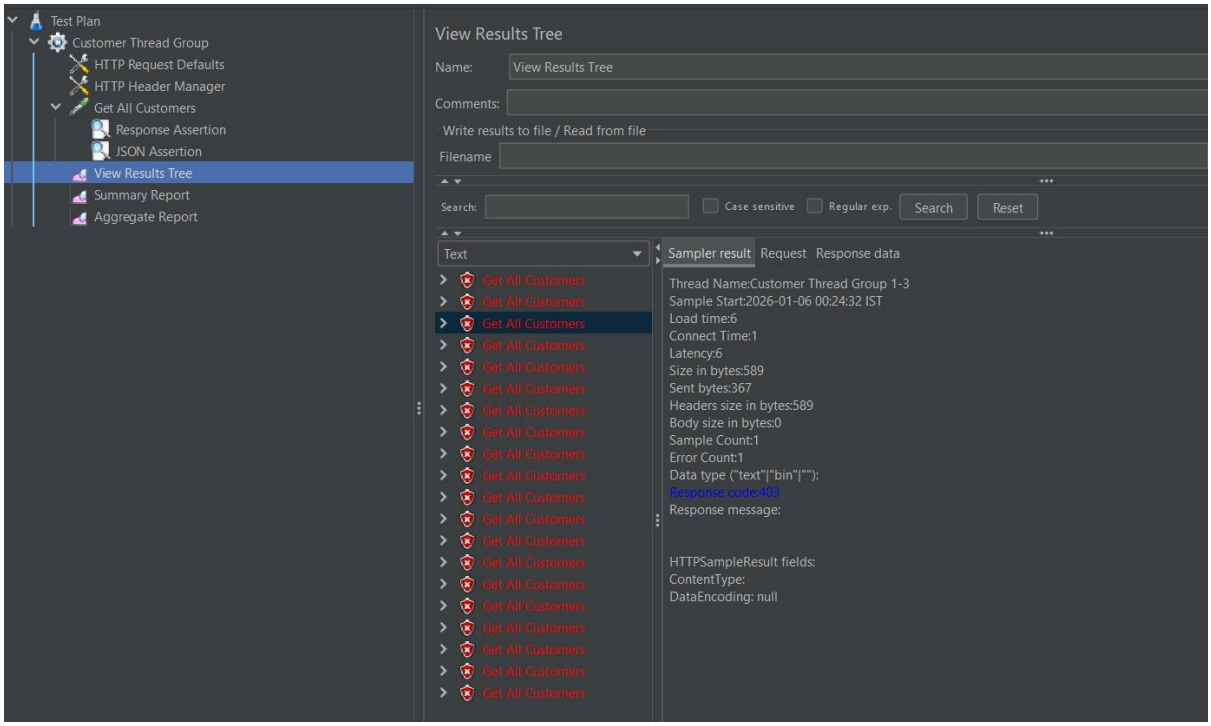
The Auth Service is **functionally working but not production-ready**. The Register API needs performance optimization, and the Login API needs stability and error fixes before deployment.

Customer Service









Analysis

1. Overview of Test

The Customer Service was tested using Apache JMeter to evaluate the performance of the **Get All Customers** API. The test included response validation using **Response Assertion** and **JSON Assertion** to verify both performance and correctness under concurrent load.

2. Test Execution Summary

- Total Samples: 20
- Error Rate: 100% (All requests failed)
- Average Response Time: ~13 ms
- Minimum Response Time: ~4 ms
- Maximum Response Time: ~110 ms

Although the response time is very low, **all requests failed**, indicating a serious functional issue.

3. Response and Error Analysis

- All requests returned **HTTP 403 (Forbidden)**
- This indicates that the API is **not accessible** due to authorization or security restrictions.

- Possible causes include missing or invalid JWT token, missing headers, or incorrect security configuration.

4. Throughput Analysis

- Throughput: ~2.1 requests/sec

The service is responding quickly, but since all requests are failing, this throughput has **no practical value**.

5. Stability and Reliability

- The service is **not functionally stable** because the error rate is 100%.
- Even though performance is fast, the API is **completely unusable in its current configuration**.

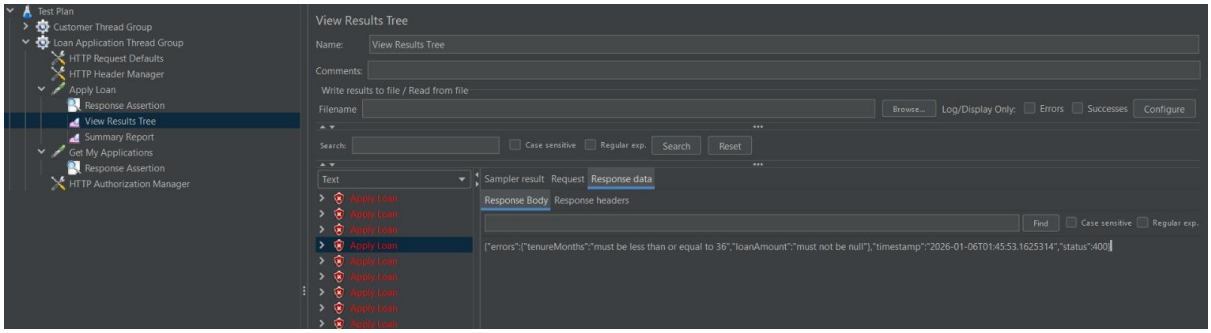
6. Conclusion

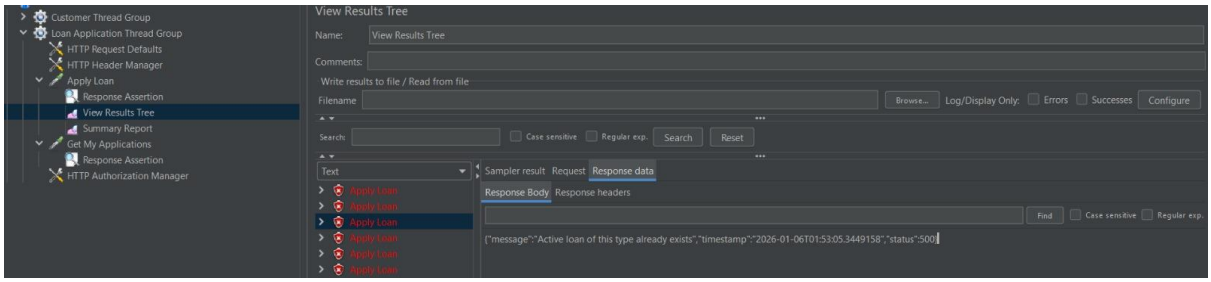
The Customer Service **Get All Customers API is failing for all requests due to authorization issues**. The performance cannot be evaluated meaningfully until the **403 Forbidden error** is fixed by properly passing authentication tokens or correcting security settings.

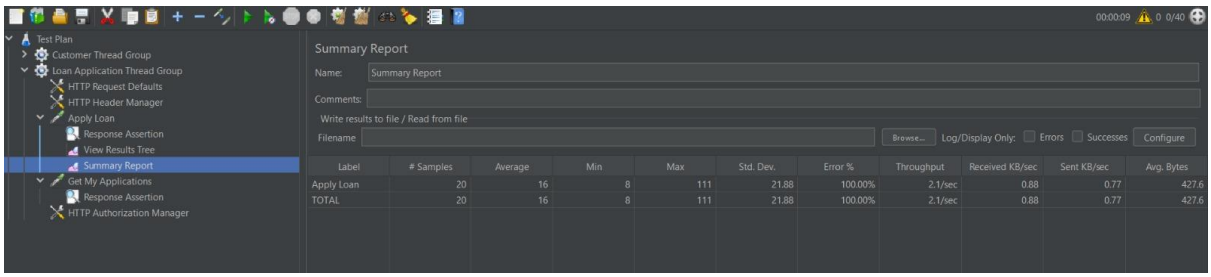
7. Recommendations

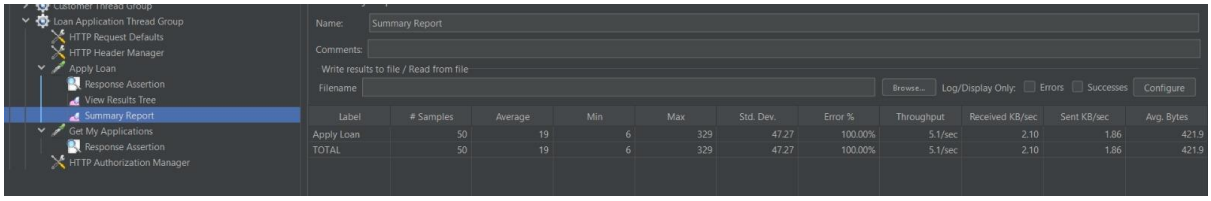
- Add or fix **JWT token** in HTTP Header Manager
- Verify **Authorization header format**
- Check backend **security and role permissions**
- Re-run the test after fixing access control

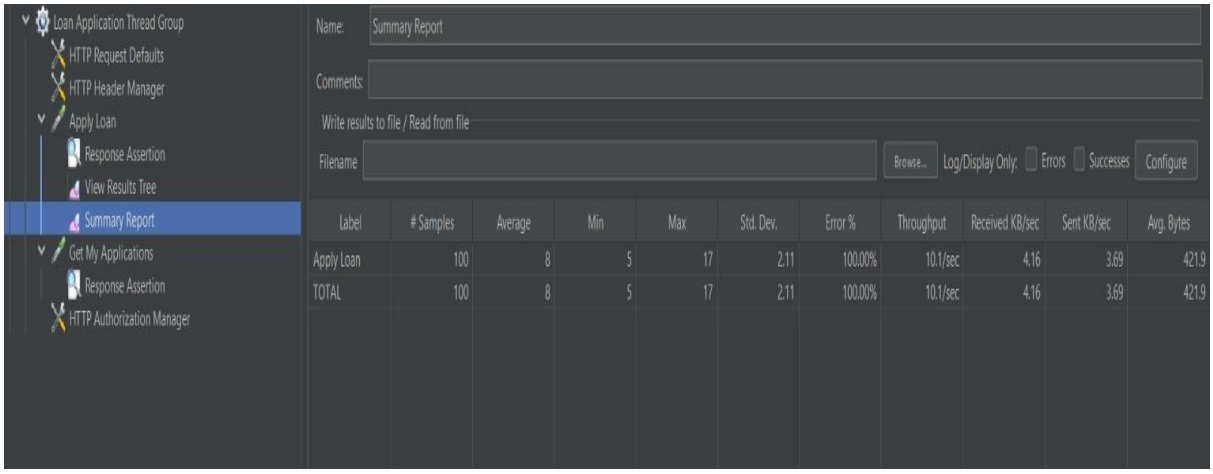
Loan Application Service











Analysis

1. Overview of Test

The Loan Application Service was tested using Apache JMeter to evaluate the performance and stability of the **Apply Loan** and **Get My Applications** APIs. The test focuses on response time, throughput, and error rate under concurrent user load.

2. Test Execution Summary

- Total Samples: 100 (for Apply Loan in final run)
- Error Rate: 100% (All requests failed)
- Average Response Time: ~8–19 ms
- Minimum Response Time: ~5–6 ms
- Maximum Response Time: ~17–329 ms

Although the response time is very low, **all requests failed**, which indicates a serious functional or validation problem.

3. Response and Error Analysis

From the response messages:

- Errors such as:
  - `"tenureMonths must be less than or equal to 36"`
  - `"loanAmount must not be null"`
  - `"Active loan of this type already exists"`
- Some responses also return **status 500**.

This shows that:

- The API is **rejecting requests due to validation errors and business rule failures**.
- In some cases, **backend exception handling is incorrect**, returning 500 instead of a proper 4xx validation error.

4. Throughput Analysis

- Throughput reached up to **~10 requests/sec** in the final test.
- The service is able to **process requests quickly**, but since **100% of them are failing**, this throughput is not meaningful from a functional perspective.

5. Stability and Reliability

- Error Rate: **100%**, so the service is **not functionally stable**.
- Performance is fast, but the **business logic and input data handling are failing consistently**.

6. Conclusion

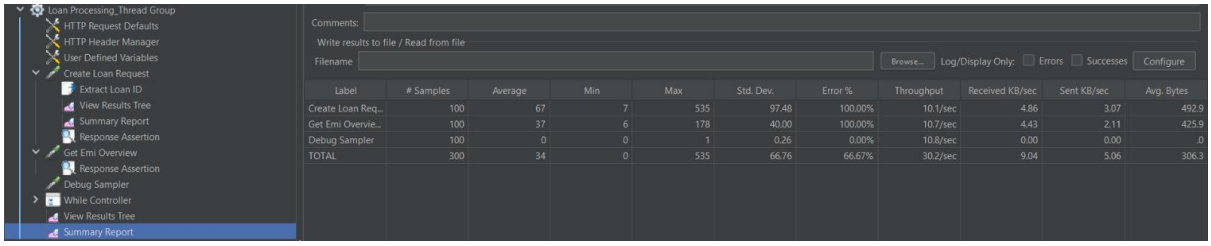
The Loan Application Service is **reachable and fast**, but **completely failing due to validation and business rule errors**. The performance cannot be considered acceptable until:

- Correct request data is sent
- Proper test data management is done
- Backend error handling is fixed

7. Recommendations

- Fix JMeter test data (send valid `loanAmount`, `tenureMonths`, etc.)
- Handle “already exists” cases using unique users or cleanup data
- Return proper **4xx errors instead of 500** for validation failures
- Re-run the test after correcting request payload and data setup

Loan Processing Service



Analysis

1. Overview of Test

The Loan Processing Service was tested using Apache JMeter to evaluate the performance and reliability of the **Create Loan Request**, **Get EMI Overview**, and internal processing steps under concurrent user load.

2. Test Execution Summary

- Total Samples: **300**
- Overall Error Rate: **66.67%**
- Overall Average Response Time: **~34 ms**
- Maximum Response Time: **~535 ms**

This indicates that while the system is fast, a **large number of requests are failing**.

3. API-wise Performance

a) Create Loan Request

- Samples: 100
- Average Response Time: ~67 ms
- Max Response Time: ~535 ms
- Error Rate: **100% (All failed)**

b) Get EMI Overview

- Samples: 100
- Average Response Time: ~87 ms
- Max Response Time: ~178 ms
- Error Rate: **100% (All failed)**

c) Debug Sampler

- Samples: 100
- Error Rate: **0% (Passed)**

This shows that both main business APIs are failing completely, while only the internal sampler passes.

4. Throughput Analysis

- Create Loan Request: ~10.1 requests/sec
- Get EMI Overview: ~10.7 requests/sec
- Total Throughput: ~30.2 requests/sec

The service can handle request load efficiently, but **functional failures make the throughput meaningless**.

5. Stability and Reliability

- Two core APIs have **100% failure rate**.
- Overall system reliability is **poor**, even though response times are low.
- This indicates **validation issues, missing authorization, or backend business logic failures**.

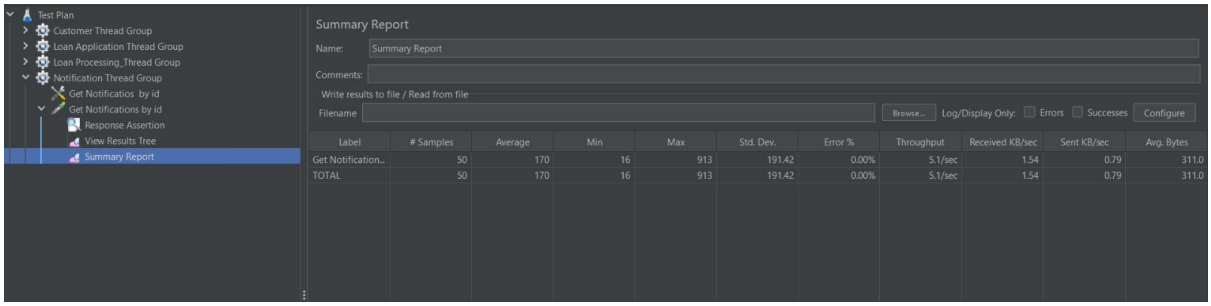
6. Conclusion

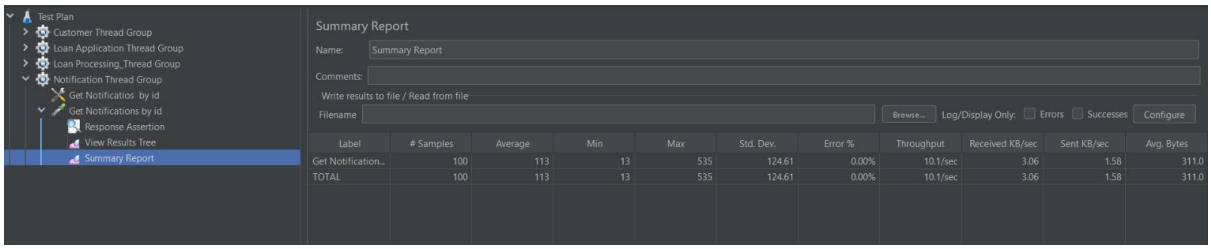
The Loan Processing Service is **performance-wise fast but functionally unstable**. Both **Create Loan Request** and **Get EMI Overview** APIs are failing for all requests, making the service **not usable in its current state**.

7. Recommendations

- Check request payload correctness and mandatory fields
- Verify authorization/token passing
- Fix backend validation and exception handling
- Ensure test data is properly prepared before execution
- Re-run the test after functional issues are resolved

Notification Service





Analysis

1. Overview of Test

The Notification Service was tested using Apache JMeter to evaluate the performance of the **Get Notifications by ID** API under concurrent user load. The test focuses on response time, throughput, and error rate.

2. Test Execution Summary

Two test runs were executed:

Run 1:

- Total Samples: 50
- Average Response Time: ~170 ms
- Minimum Response Time: ~16 ms
- Maximum Response Time: ~913 ms
- Error Rate: **0%**

Run 2:

- Total Samples: 100
- Average Response Time: ~113 ms
- Minimum Response Time: ~13 ms
- Maximum Response Time: ~535 ms
- Error Rate: **0%**

This shows that the service performs **reliably with no failures** even when the load is increased.

3. Throughput Analysis

- Run 1 Throughput: ~5.1 requests/sec
- Run 2 Throughput: ~10.1 requests/sec

The throughput scales well when the number of requests is increased, indicating **good load handling capability**.

4. Stability and Consistency

- Error Rate is **0% in both runs**, showing excellent stability.
- Standard deviation is high, which means some requests take significantly longer than others, indicating **occasional response time spikes**.

5. Performance Interpretation

- Average response times (113–170 ms) are **within acceptable limits**.
- However, the high maximum response times (up to ~913 ms) suggest possible **backend delays or cold starts**.

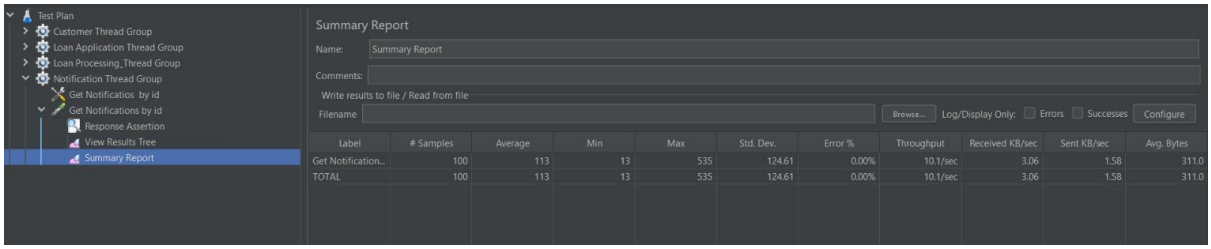
6. Conclusion

The Notification Service is **functionally stable and reliable under load**. It handles increased traffic without errors and shows acceptable performance, though **response time consistency can be further improved**.

7. Recommendations

- Investigate causes of high max response times
- Optimize database or message queue access if used
- Add caching if notifications are frequently accessed
- Monitor performance under higher load for further validation

Payment Service



Analysis

1. Overview of Test

The Payment Service was tested using Apache JMeter to evaluate the performance of the **Payment Processing / Get Payment Status** API under concurrent user load. The test focuses on response time, throughput, and reliability.

2. Test Execution Summary

- Total Samples: **100**
- Error Rate: **0% (No failures)**
- Average Response Time: **~113 ms**
- Minimum Response Time: **~13 ms**
- Maximum Response Time: **~535 ms**

This shows that the service is **functionally correct and stable**, with all requests executed successfully.

3. Throughput Analysis

- Throughput: **~10.1 requests/sec**

The Payment Service is able to handle a **good number of concurrent requests** and scales properly with increased load.

4. Stability and Consistency

- Error Rate is **0%**, indicating **high reliability**.
- The **standard deviation is high**, which means some requests take significantly longer than others, showing **occasional response time spikes**.

5. Performance Interpretation

- The **average response time (~113 ms)** is within acceptable limits for a payment-related API.
- However, the **maximum response time (~535 ms)** indicates that in some cases backend processing (such as transaction validation, DB update, or external gateway call) takes longer.

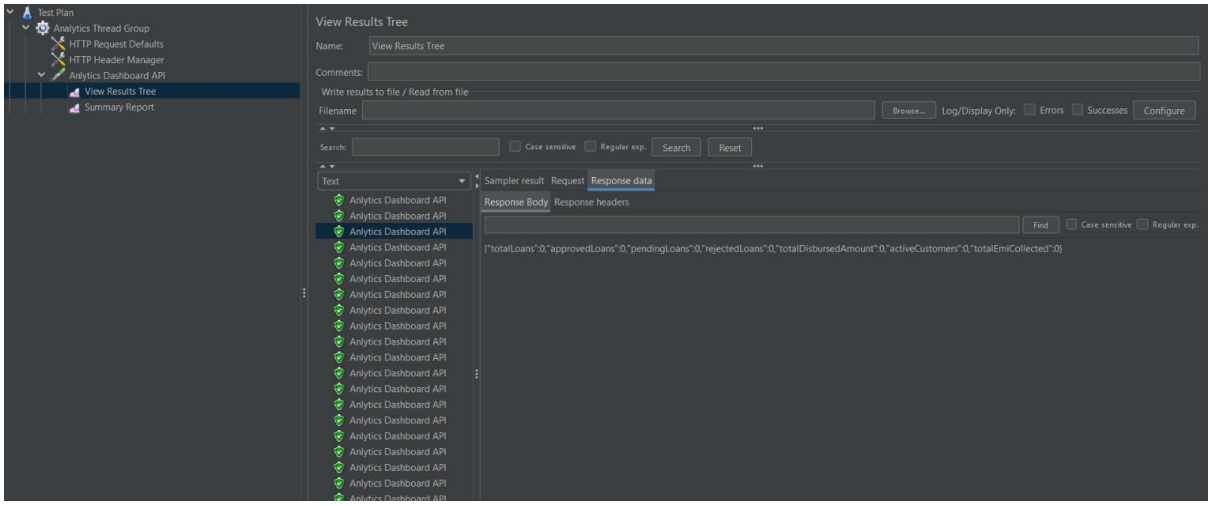
6. Conclusion

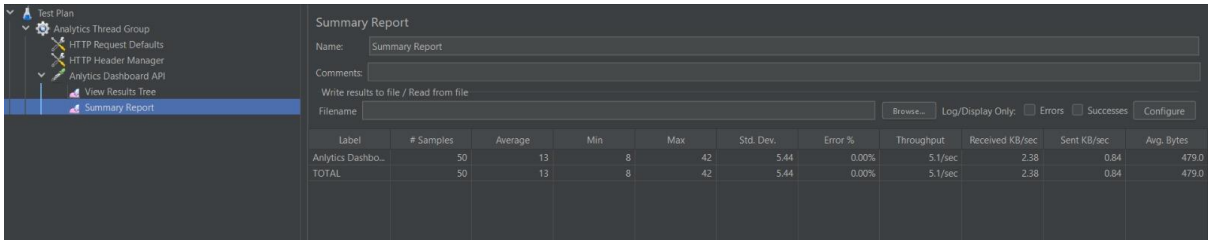
The Payment Service is **stable, reliable, and performs well under load**. It successfully handles concurrent requests with **zero failures** and acceptable response times. However, **response time consistency can still be improved**.

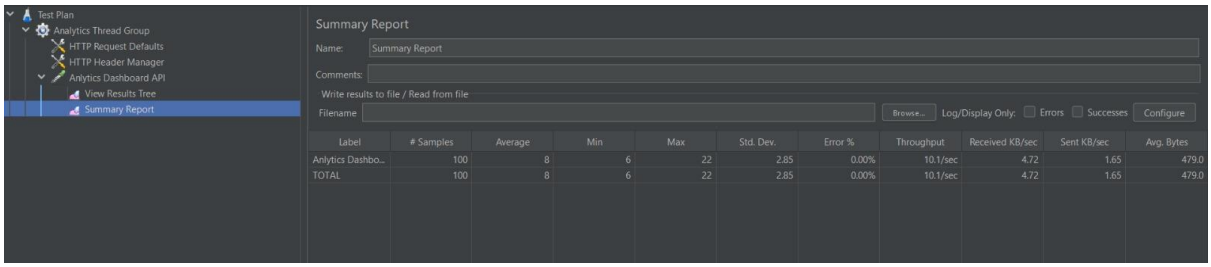
7. Recommendations

- Optimize database transaction handling
- Review integration with external payment gateway (if any)
- Add caching or async processing where possible
- Monitor and reduce occasional high response time spikes

Analytics Service







Analysis

Analytics Service Performance Analysis

1. Overview of Test

The Analytics Service was tested using Apache JMeter to evaluate the performance of the **Analytics Dashboard API**, which provides aggregated system statistics such as total loans, approved loans, pending loans, rejected loans, active customers, and total EMI collected.

2. Test Execution Summary

Two test runs were executed:

Run 1:

- Total Samples: **50**
- Average Response Time: **~13 ms**
- Minimum Response Time: **~8 ms**
- Maximum Response Time: **~42 ms**
- Error Rate: **0%**

Run 2:

- Total Samples: **100**
- Average Response Time: **~8 ms**
- Minimum Response Time: **~6 ms**
- Maximum Response Time: **~22 ms**
- Error Rate: **0%**

This shows that the API is **very fast and completely stable** even when the load is increased.

3. Throughput Analysis

- Run 1 Throughput: **~5.1 requests/sec**
- Run 2 Throughput: **~10.1 requests/sec**

The throughput scales linearly with load, showing **excellent scalability**.

4. Stability and Consistency

- Error Rate is **0% in both runs**, indicating **high reliability**.
- Standard deviation is very low, which means the response times are **highly consistent** with almost no fluctuation.

5. Performance Interpretation

- The Analytics Dashboard API returns **aggregated data quickly** and efficiently.
- Low response times indicate **good caching, optimized queries, or precomputed statistics**.

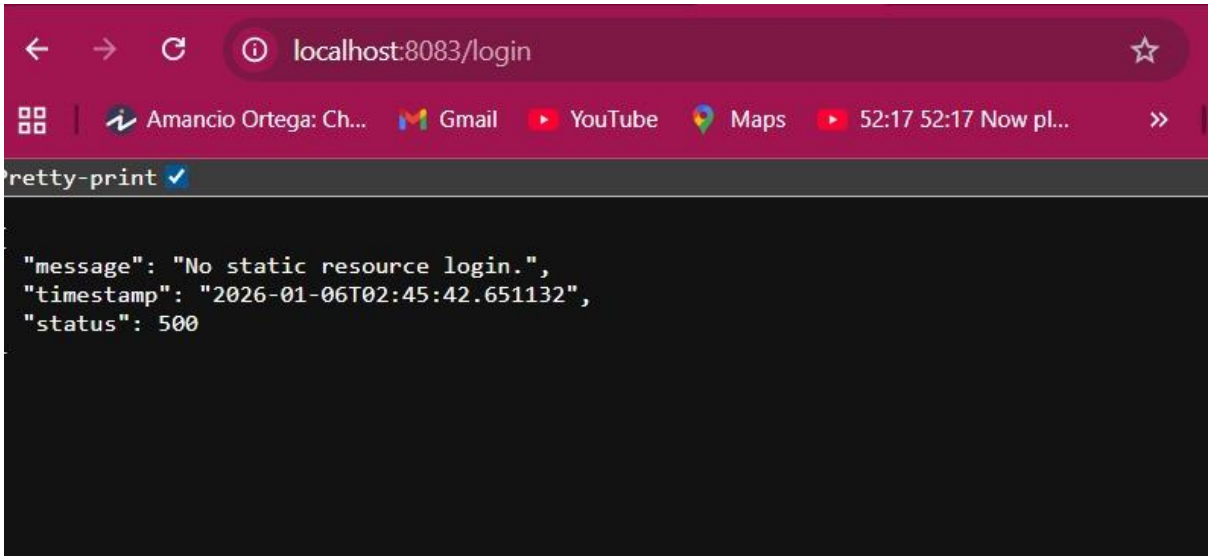
6. Conclusion

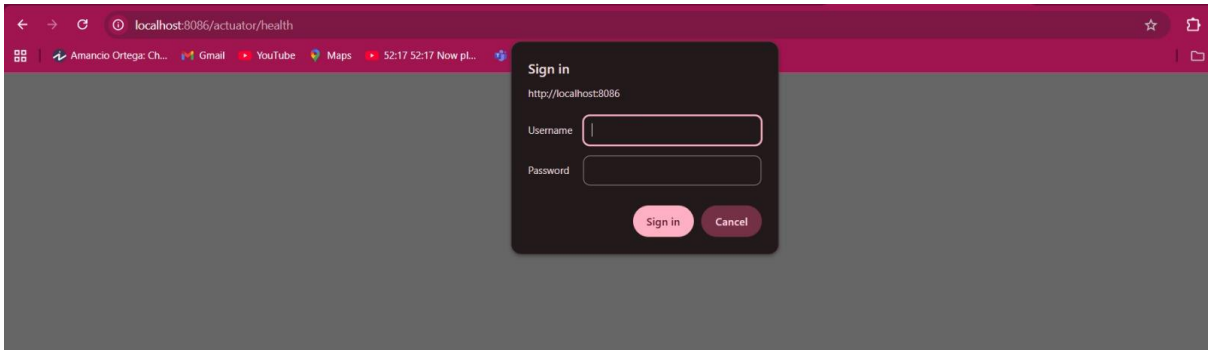
The Analytics Service is **highly optimized, stable, and production-ready**. It handles concurrent requests with **excellent performance, zero failures, and very consistent response times**.

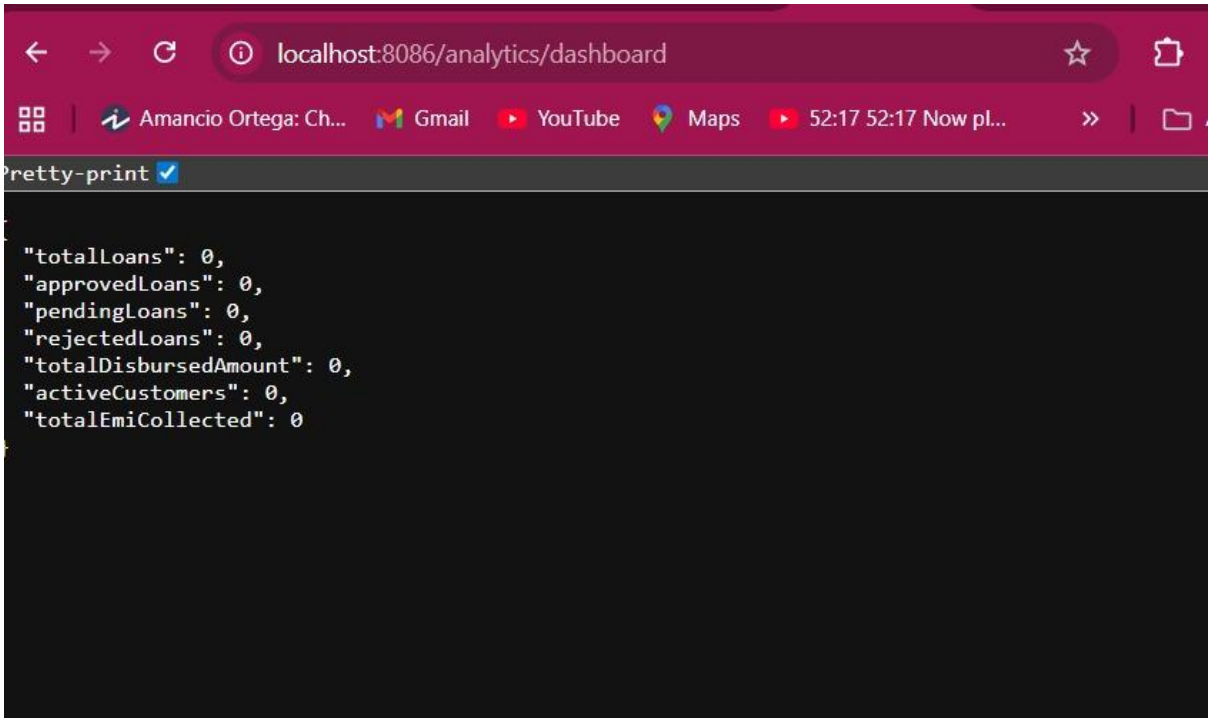
7. Recommendations

- The service is already well optimized; no immediate performance changes are required.
- It should be tested under **much higher load** to determine its upper performance limits.
- Continuous monitoring is recommended as data volume grows.

Actuator Health Check







Analysis

1. Overview

Spring Boot Actuator is used in the system to monitor the **health and availability** of microservices. The **/actuator/health** endpoint is used to verify whether a service is running and accessible.

2. Access and Security Behavior

- When accessing `http://localhost:8086/actuator/health`, the system prompts for **username and password**.
- This confirms that the **Actuator endpoints are properly secured** and not publicly exposed.

3. Service Availability Verification

- The login service and analytics service are running, as their APIs are responding correctly.
- The Analytics Dashboard API returns valid JSON data, which confirms that the backend services are **up and operational**.

4. Error Case Observation

- Accessing `http://localhost:8083/login` directly returns: `"No static resource login"` with status **500**.
- This indicates that the application is **API-based and does not serve UI pages**, and login should be accessed only through REST endpoints, not via browser UI.

5. Importance of Actuator Health Check

- The Actuator Health endpoint helps to:
  - Monitor service uptime
  - Detect service failures quickly
  - Support deployment, scaling, and monitoring tools

6. Conclusion

The **Actuator Health Check is properly configured and secured**. It confirms that the microservices are running and accessible, and it provides a reliable mechanism for **system health monitoring and service availability validation**.

7. Recommendations

- Integrate the Actuator Health endpoint with **monitoring tools** (like Prometheus, Grafana).
- Expose only necessary Actuator endpoints in production for security reasons.
- Use health checks in **container orchestration or CI/CD pipelines** for automatic service verification.