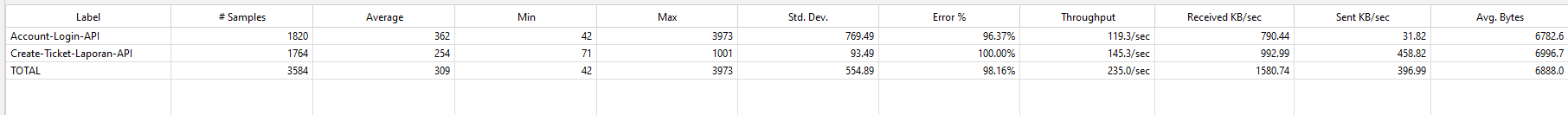
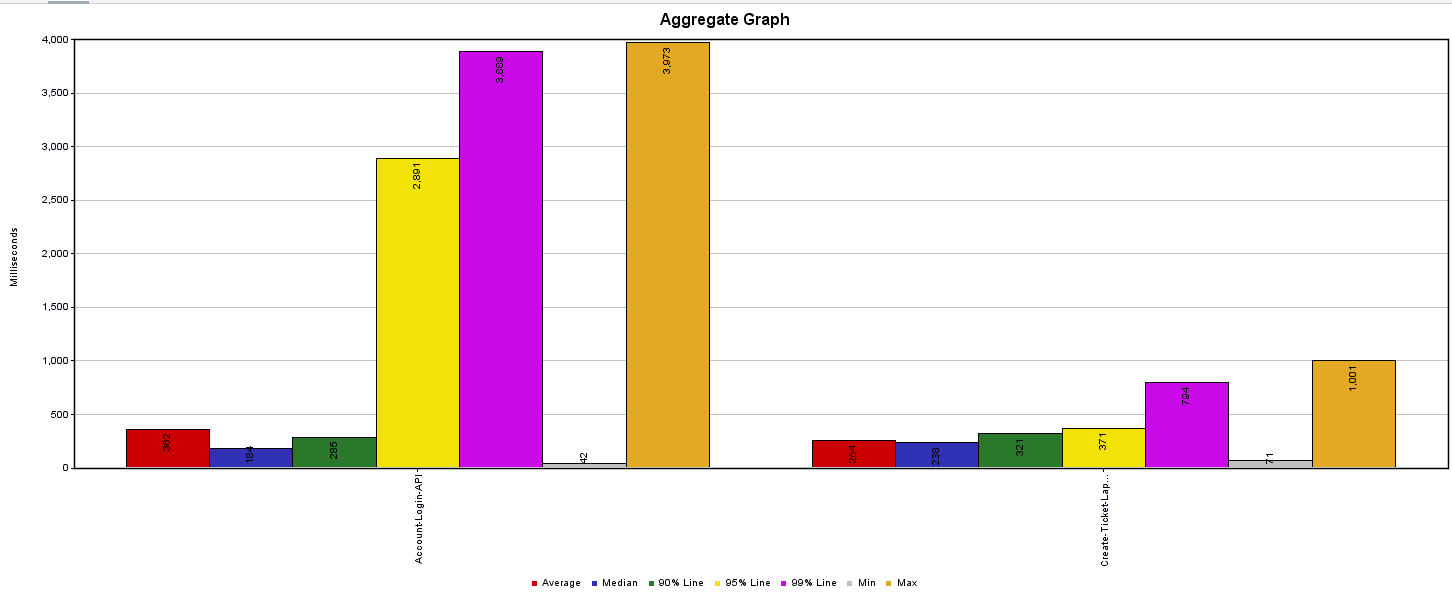
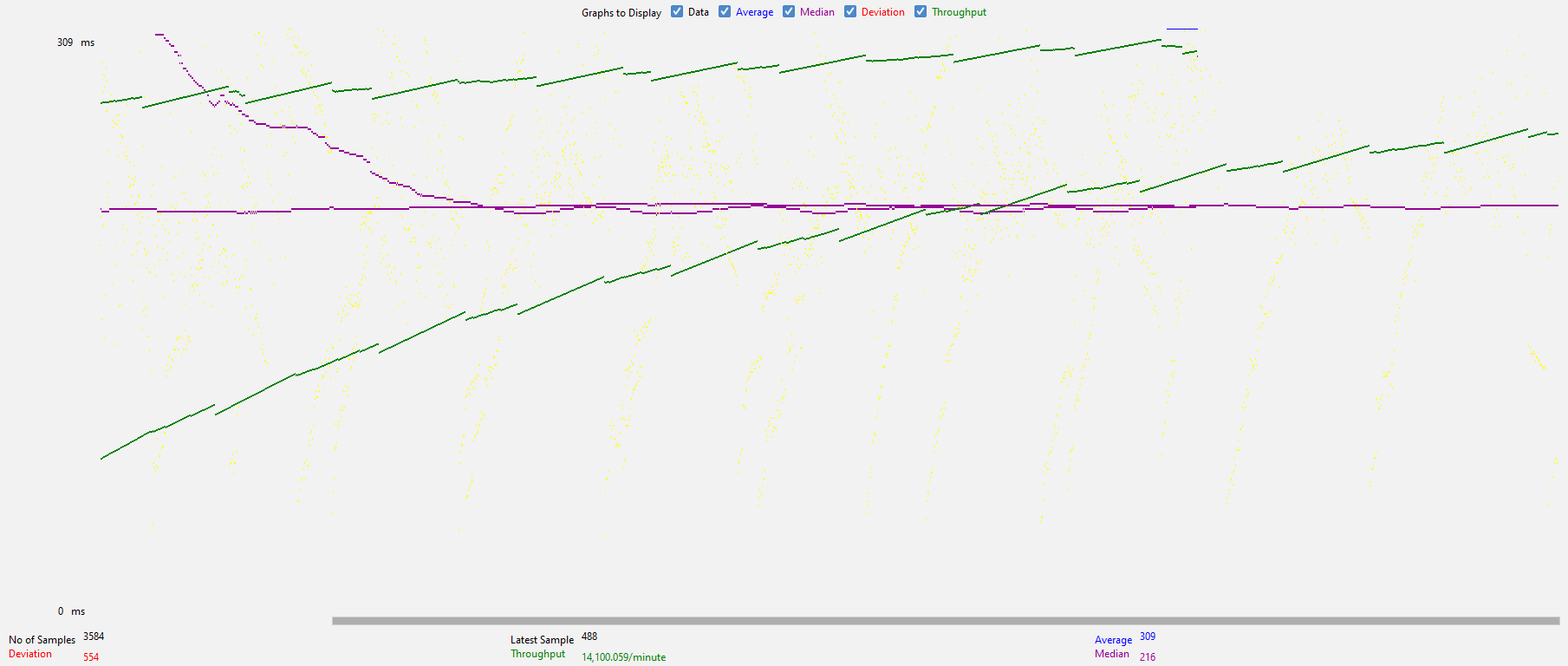
1. **Summary Result:**



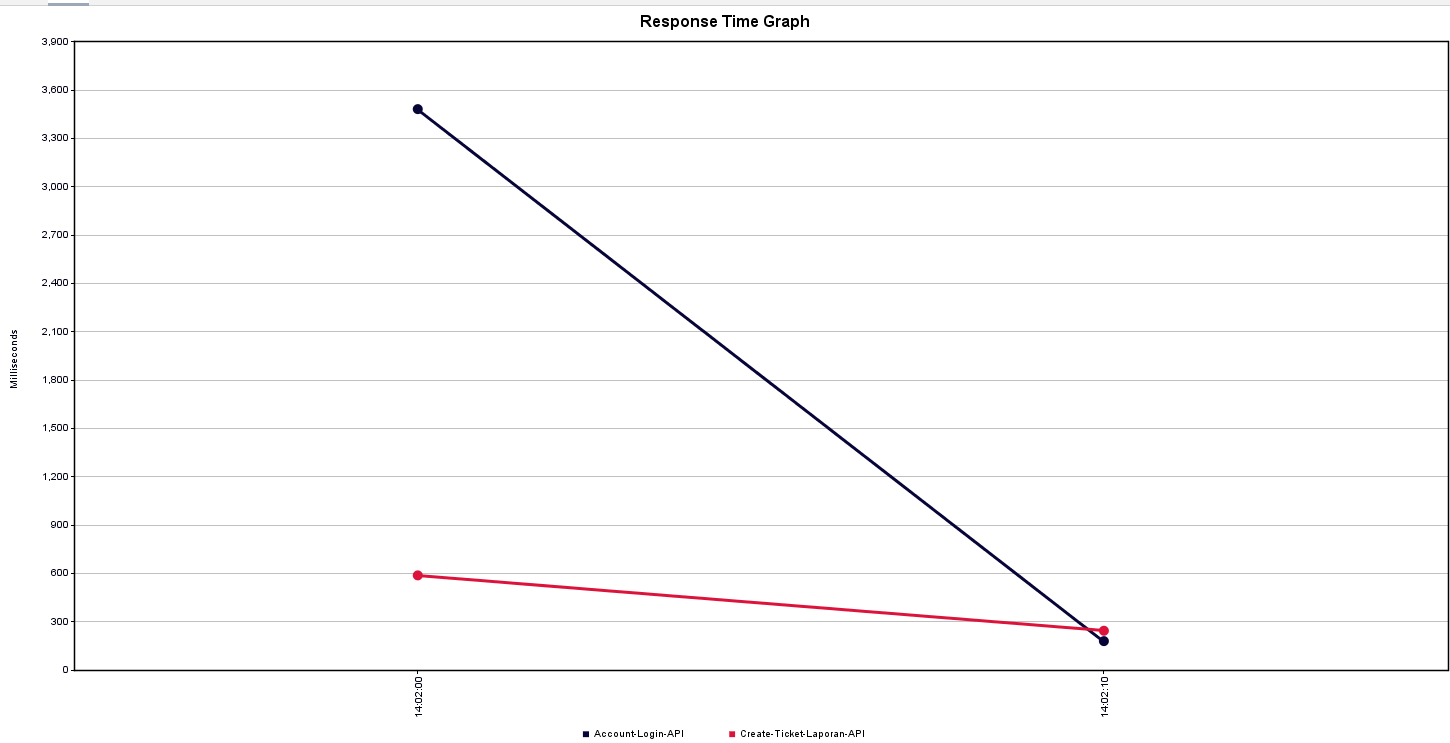
1. **Aggregate Graph:**



1. **Graph Result:**



1. **Response Time Graph**

****

1. **JMeter result Analysis:**
2. **High Error Percentage:**

The error percentage for both APIs is significantly high (96.37% for Account-Login-API and 100% for Create-Ticket-Laporan-API), indicating that most requests are failing. This might be due to system constraints, such as insufficient RAM, causing the JMeter server (my PC) to struggle under the load.

1. **Standard Deviation:**

The standard deviation values are large for both APIs (769.49 and 93.49), especially for the Account-Login-API. This means there is significant variation in response times, likely due to resource contention. Insufficient RAM may cause the system to frequently swap between memory and disk, leading to such variability.

1. **Throughput:**

The throughput values of 119.3/sec and 145.3/sec are decent but could be bottlenecked due to hardware limitations. If the RAM is maxed out, throughput may be lower than it could be on a better-optimized system.

1. **Response Times and Received KB/sec:**

The average response times are 362ms and 254ms, which aren't too high, but combined with the high error rates, they suggest that the system is often unable to process requests successfully. You can also observe a higher rate of data received per second (790 KB/sec and 992 KB/sec), which implies high data handling despite the errors.

1. **Conclusion:**

The results indicate that my PC, acting as a JMeter server, is likely overwhelmed due to RAM constraints. The combination of high error percentages, large response time variations (standard deviation), and relatively low throughput for the workload suggests that increasing the available RAM could help mitigate these issues by reducing memory swapping and improving overall system stability during the test.

1. **Recommended Action:**

* Increase the RAM on jmeter server machine or offload the test to a more powerful machine (e.g., a dedicated JMeter server or cloud-based instance).
* Monitor the resource usage (CPU, RAM, disk I/O) during the test to confirm the bottleneck is indeed the RAM.
* Lower the test load or split it into smaller chunks to reduce resource pressure on personal PC.