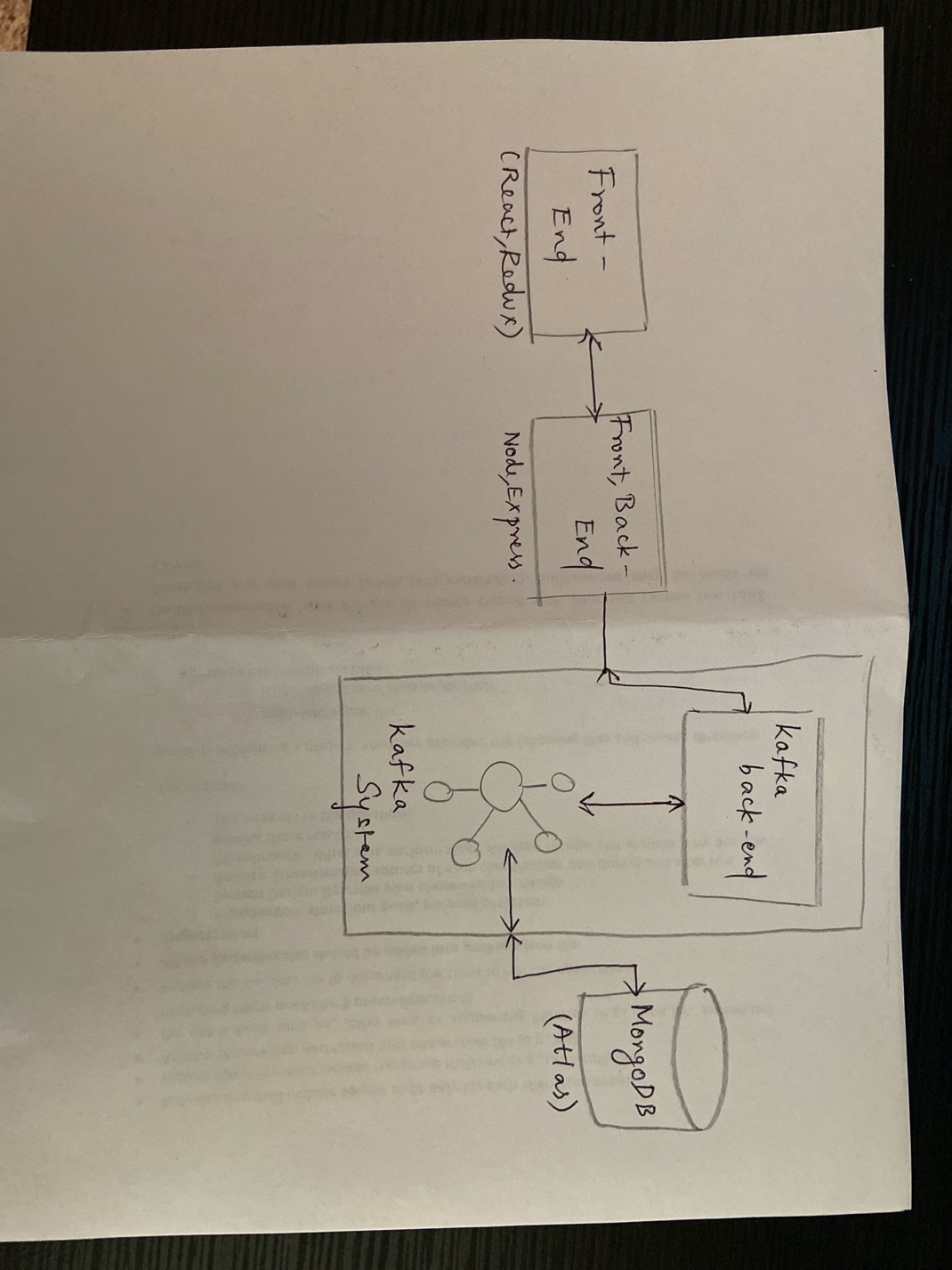
Grubhub Lab2 – Aswin Prasad

# Introduction

We have implemented the skeleton of Grubhub in first lab, but it lacked in many areas like No authentication to API’s, User details stored in front-end un-encrypted and we also used Relational database at first, for our application, which has a fairly varying data structure NoSQL works the best. So we chose MongoDB as data store. And as a mechanism to streamline the data pipeline of the application, we have implemented Kafka at its started stages.

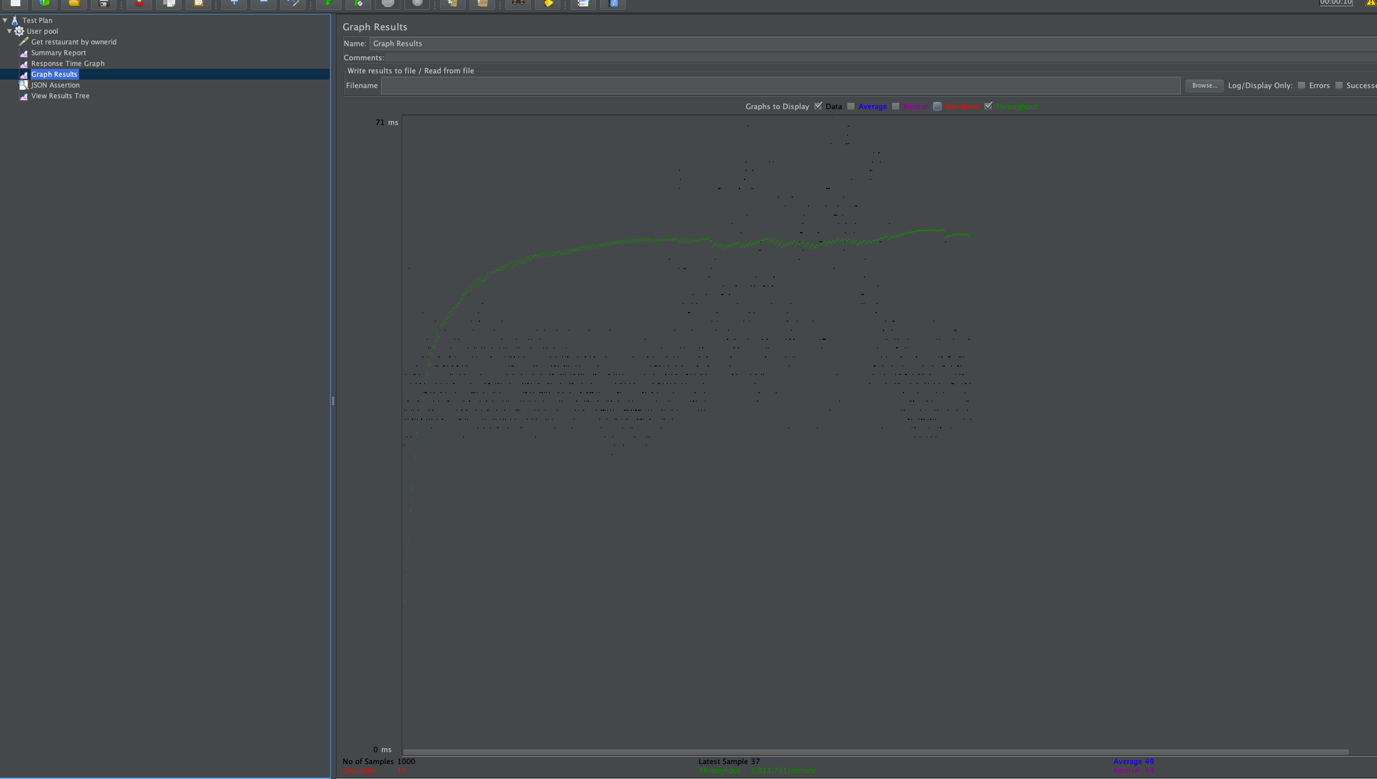
# System design



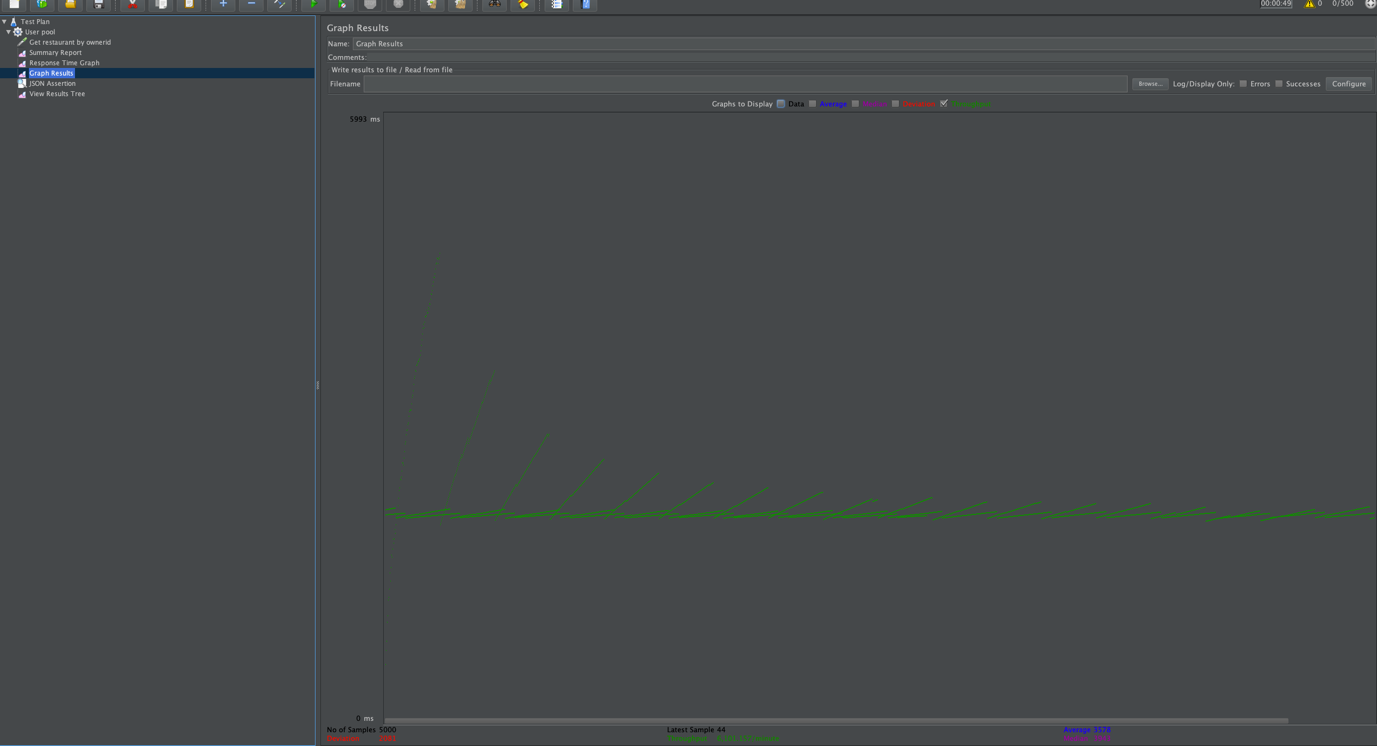
# Performance Testing

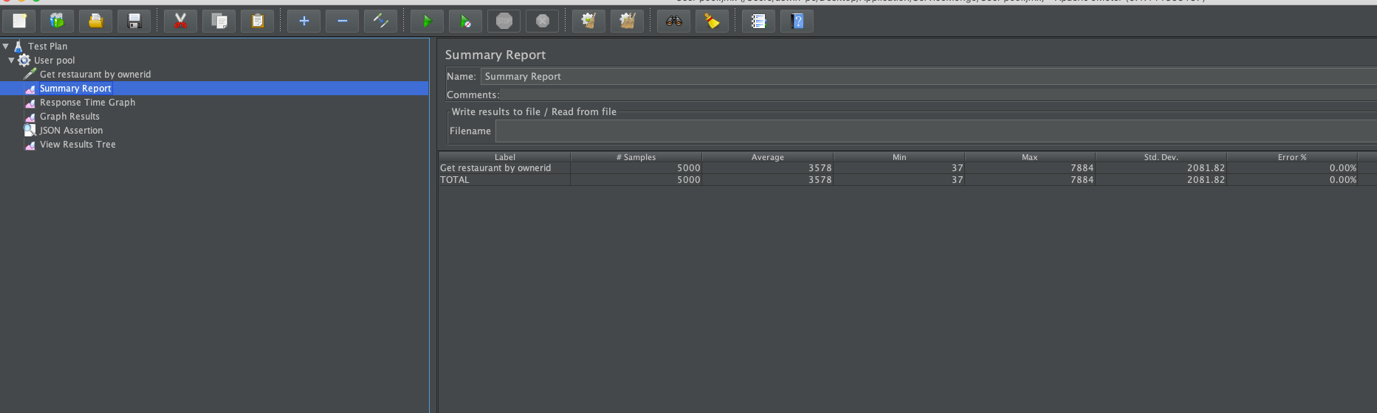
### JMeter

Without pooling, Without Kafka - 100 Users

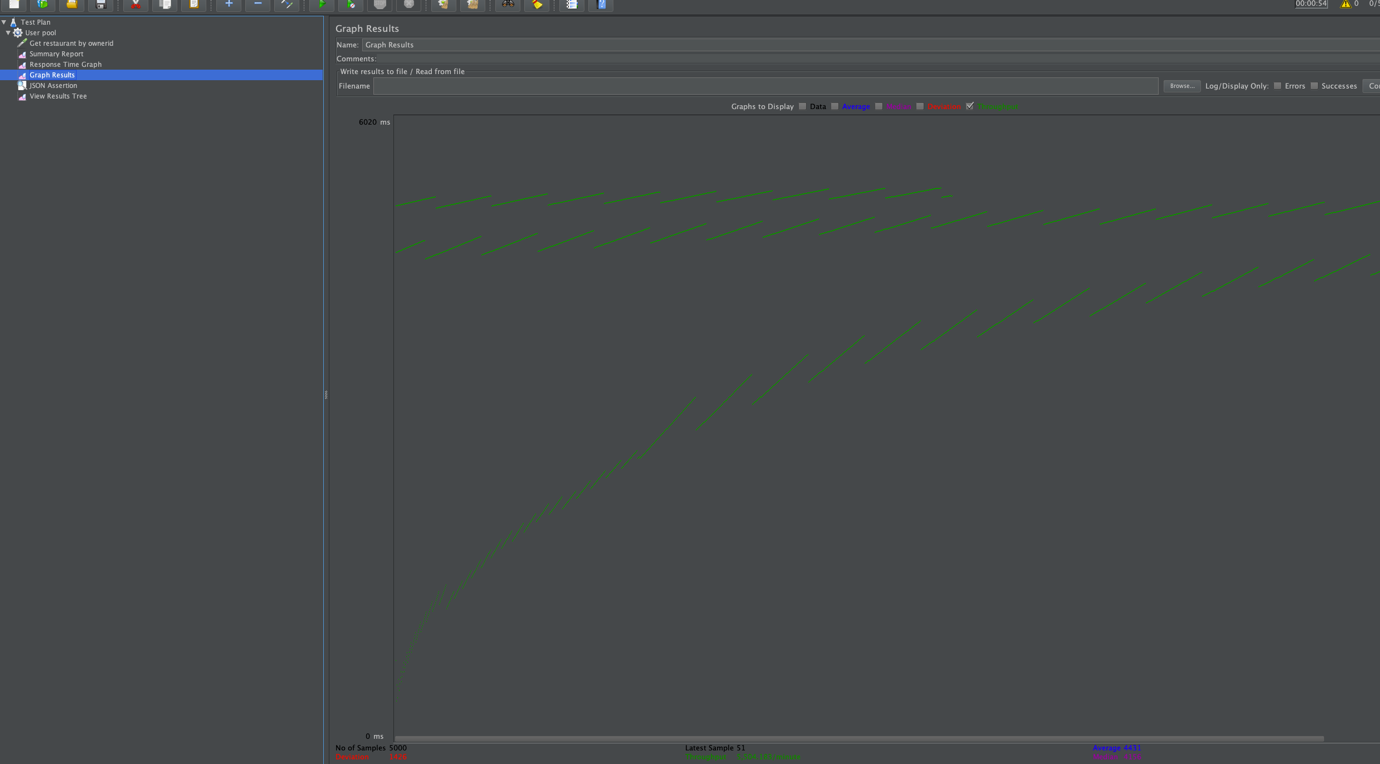


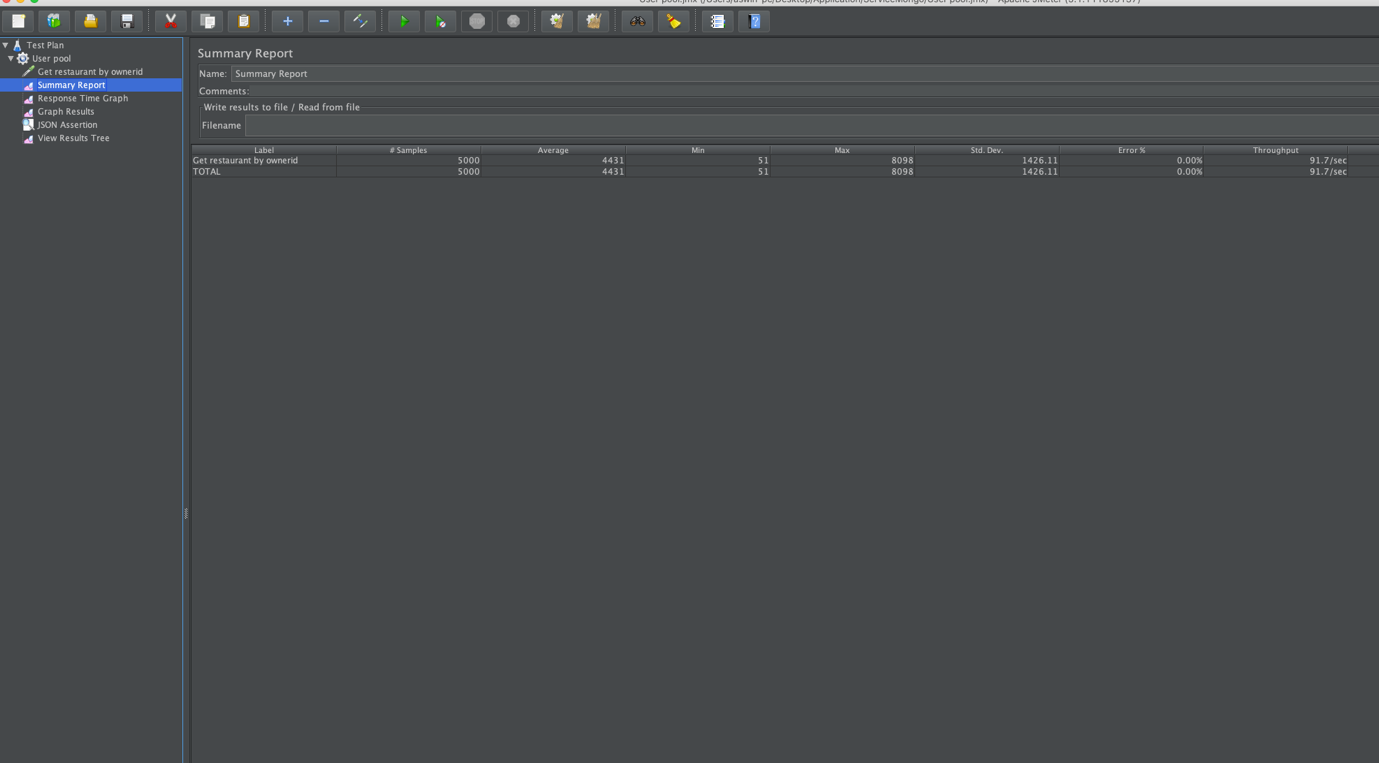
Without pooling, Without Kafka – 500 users





With Pooling (30 connections), Without Kafka – 500 Users





**Explanation: Why pooling of 30 connections are not showing improvement over default ones.**

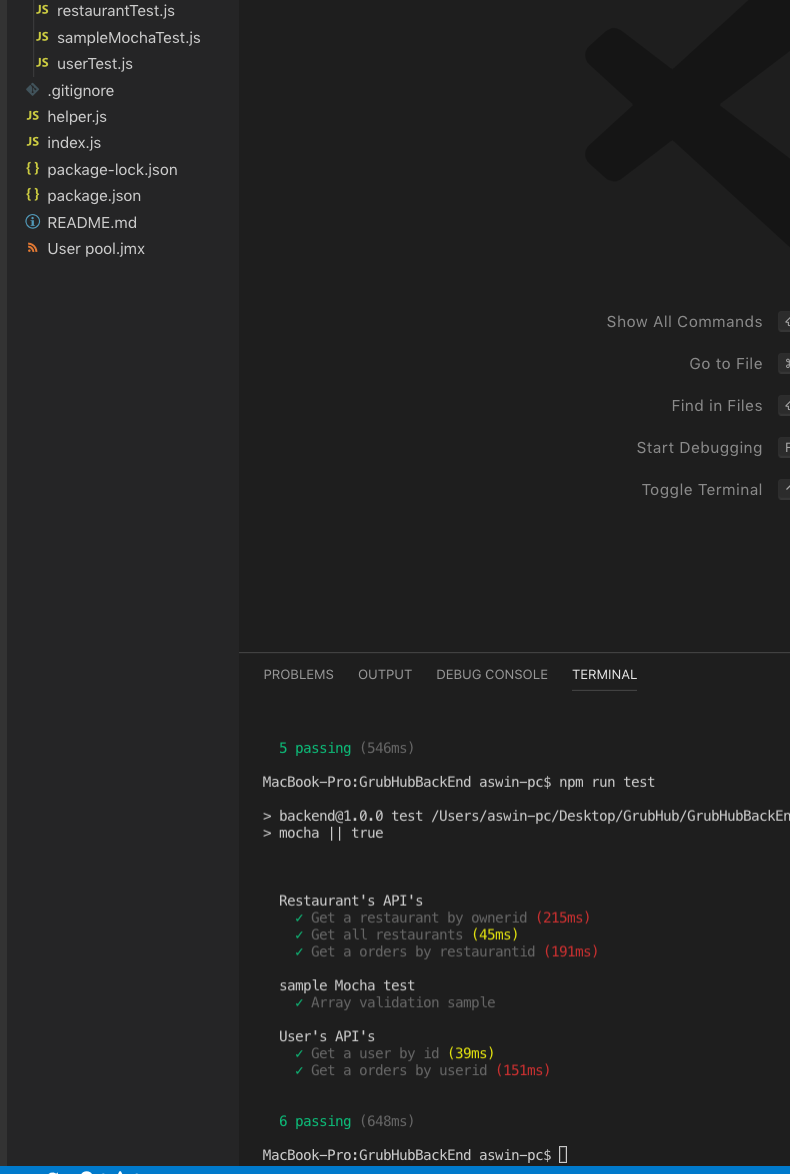
From the official document of mongoose.

poolSize - The maximum number of sockets the MongoDB driver will keep open for this connection. By default, poolSize is 5.

Our pool of 30 connections is not increasing performance and works same as poolSize of 5 because, the node’s throughput(Capacity) has been reached in this scenario and pool size doesn’t make a difference.

One approach to improve performance is to increase the instances of node to cope up with the user demand.

Mocha testing (Output)



# Questions

1. **Compare authentication in first and second lab**

In first lab, I stored cookie in front end un encrypted, so any one can take the cookie and use them to have interaction with my server and get all the data.

In lab2, I have Used passport authentication method, and used JWT strategy.

Now, the encrypted data only will be stored in cookie. Anyone who gets access to the data is going to hate it as they will not be able to decode it without the secret from backend.

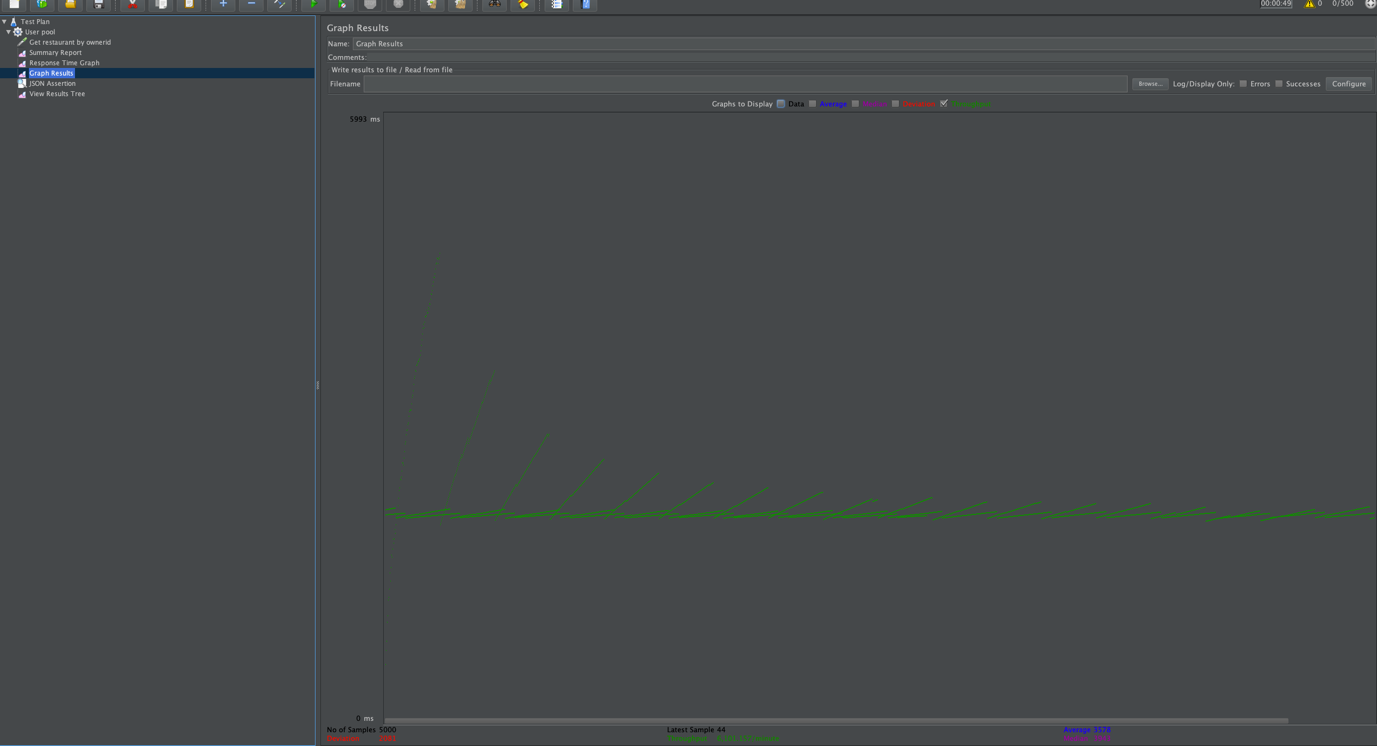
Also each time when one logs in, The time stamp also is included in the data so the code keeps varying when one logs in each time.

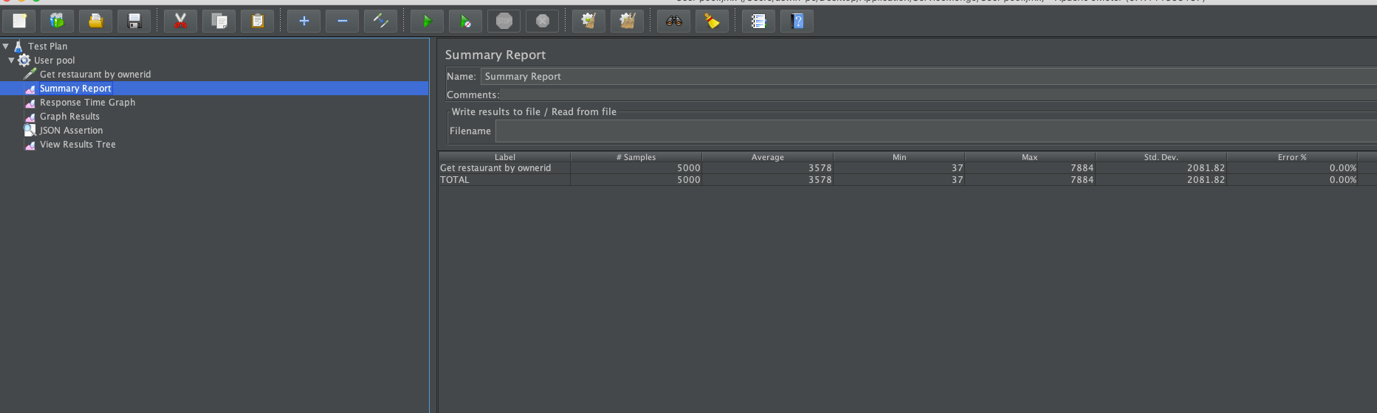
The best part of using passport is that, we not only have JWT strategy now and can easily include various other strategies like Facebook, Google sign-in to our applications.

We just need to write a strategy for them and then connect them as a middle ware to each request.

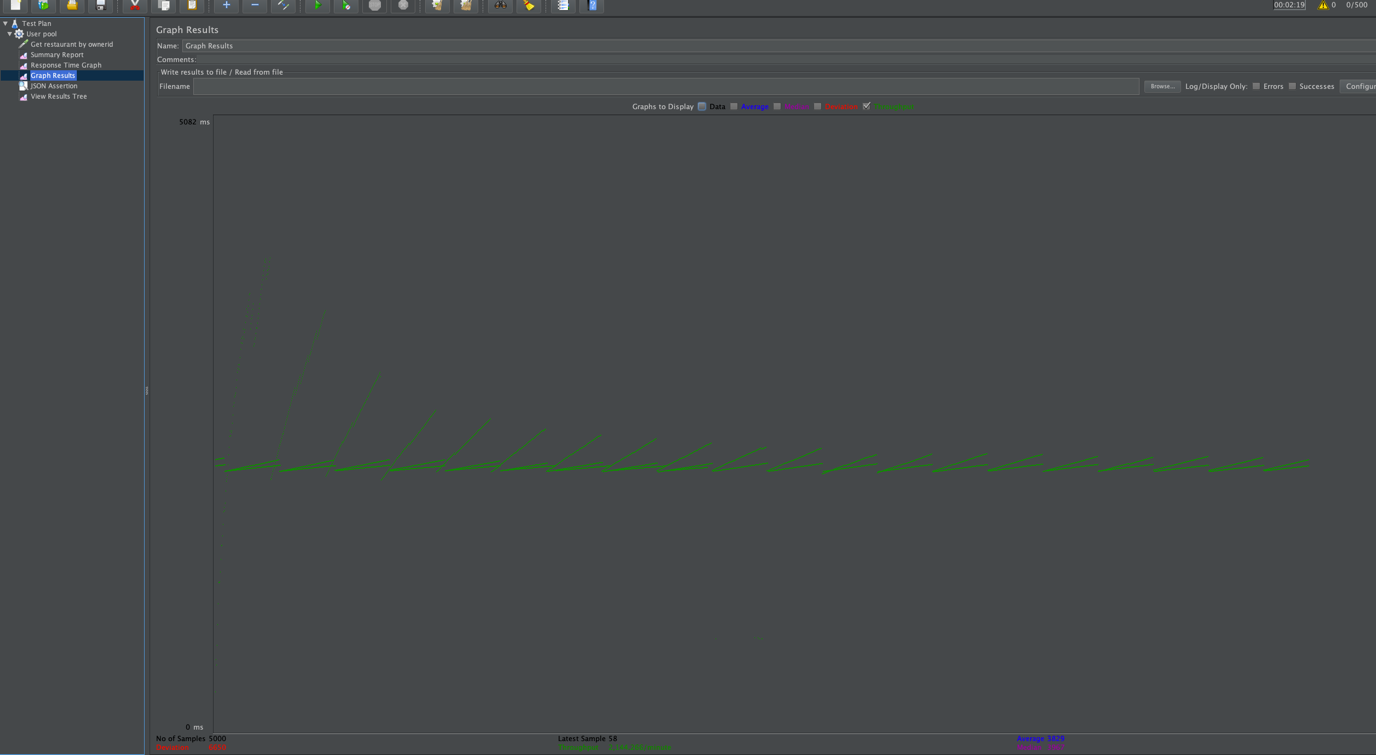
1. **Compare performance with and without Kafka.**

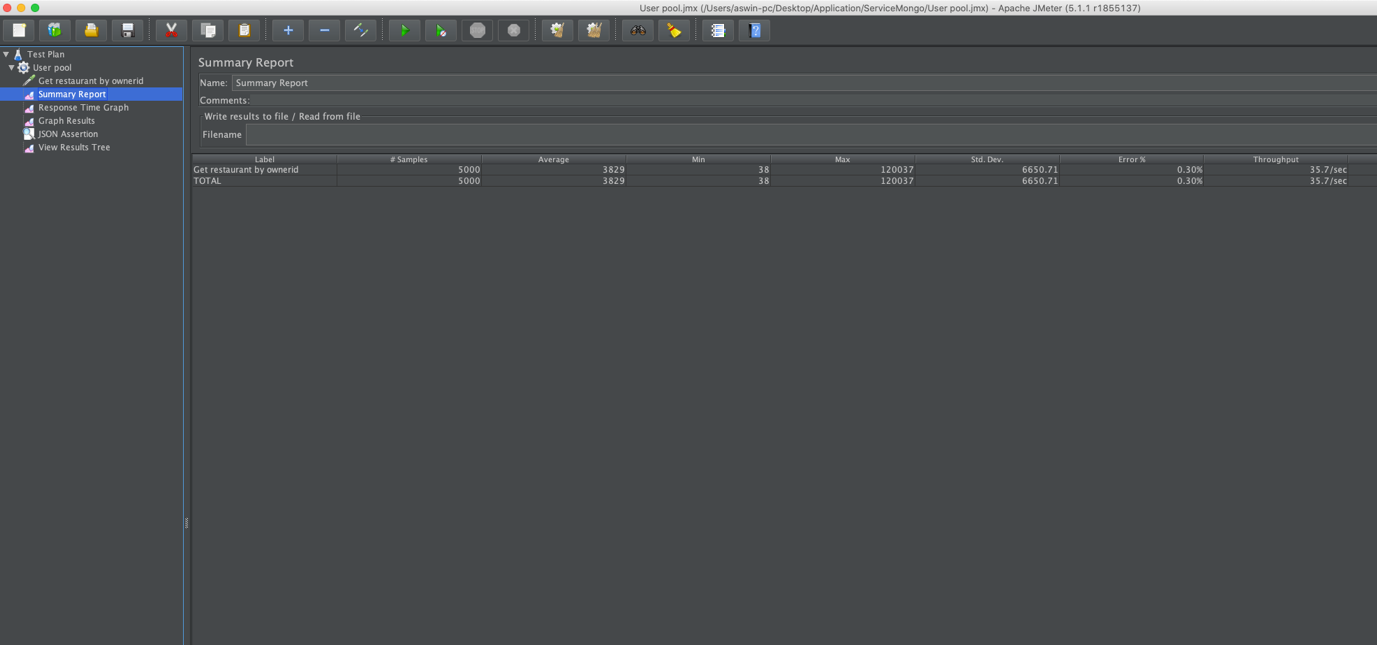
Without pooling, Without Kafka – 500 users





Without pooling, With Kafka – 500 users





**Explanation: Of why there is no performance improvement when using Kafka.**

The main reason here for not getting any improvement and actually a slight lag when using Kafka is the use-case and how it is used. Kafka is mainly used for a delayed processing mechanism, where we collect data for analytics and future use. Here we are just putting Kafka as a middleware to pass our data and just do the same process with MongoDB in the back-end. The performance of MongoDB remains the same in both case, which is a bottle neck thus hitting a cap of speed.

Fire and forget works the best when using Kafka, he we are putting in Kafka queue, and wait for response from MongoDB this we actually have a slight Kafka overhead.

There are not much complex interactions in our applications as of now, this the use-case for Kafka is limited in our application.

1. **Usage of MySQL and MongoDB in our application.**

Let’s consider our application in scale now (Thousands of users), as then only the difference between the different databases can be exemplified.

Our schema is going to be fairly varying, as each day when a new feature comes, we have to modify the database slightly. We need to identify the areas that are not going to vary much in coming times, those are candidates for MySQL. As MongoDB can handle schema changes easily.

Relation between tables, whenever there is a strong relation between tables, we need to join them using Joins in MySQL and when using MongoDB, we combined order and orderDetails so as to avoid joins. We have an ability to put array of data that MongoDB supports.

There is a problem of data growth in our application that will come when user base grows very big, like a person would place multiple orders each day, out order table will keep growing and as there is no concept of horizontal scaling in MySQL, we have to do a vertical scaling and move to a big server, which will have downtime and so on, also this can continue till an extent and have a hard stop someday. So, User and Restaurant details can be kept in MySQL and Order details can be kept in MongoDB as that will grow humongous.