CMPE-273  
**LAB-1**

horizontal line

# 

Github: <https://github.com/parthpandya17/Lab1-Parth/tree/master/Lab2-eBay>

password: abc@12345

# Introduction:

*Goals and purpose of your system*

# This report consists of eBay a simple market place:

# EBay, a simple market place, developed based on restful architecture. Server is divided into two part. The front end part of the server is implemented in nodejs and express framework which receives the request from the client over the web, making and the back end part is a rabbitMQ server that performs all the database operations. mongodb used is the database which is light weight and nosql. client is implemented in angularjs and bootstrap. Most of all the functionalities have been implemented, including signin, buying items, add to cart, checkout, etc. Jmeter and mocha testing has been used to perform testing and analysis. Various encryption algorithm and techniques have been used to improve performance and features.

# System Design: Express and Nodejs are used to develop the front part of server implementing the restful services. The back end server consists of the rabbitMQ server which is based on asynchronous message queueing protocol. MongoDB database has been used to persist the data, which is light weight and NOSQL. Various validations have been implemented to provide better usage. Different views have been developed to signin, signout, view profile, etc. Handler page has been developed to view all the user information using dynamic routing. Logs have been printed to track the user. Session has been maintained to ensure that only logged in user can perform actions. All the transactions and related information is available to user.

Bootstrap + HTML5

NodeJS + RabbitMQ

Server

NodeJS + Express

Server

MongoDB

AnguarJS

System Block Diagram

# RabbitMQ RPC strategy

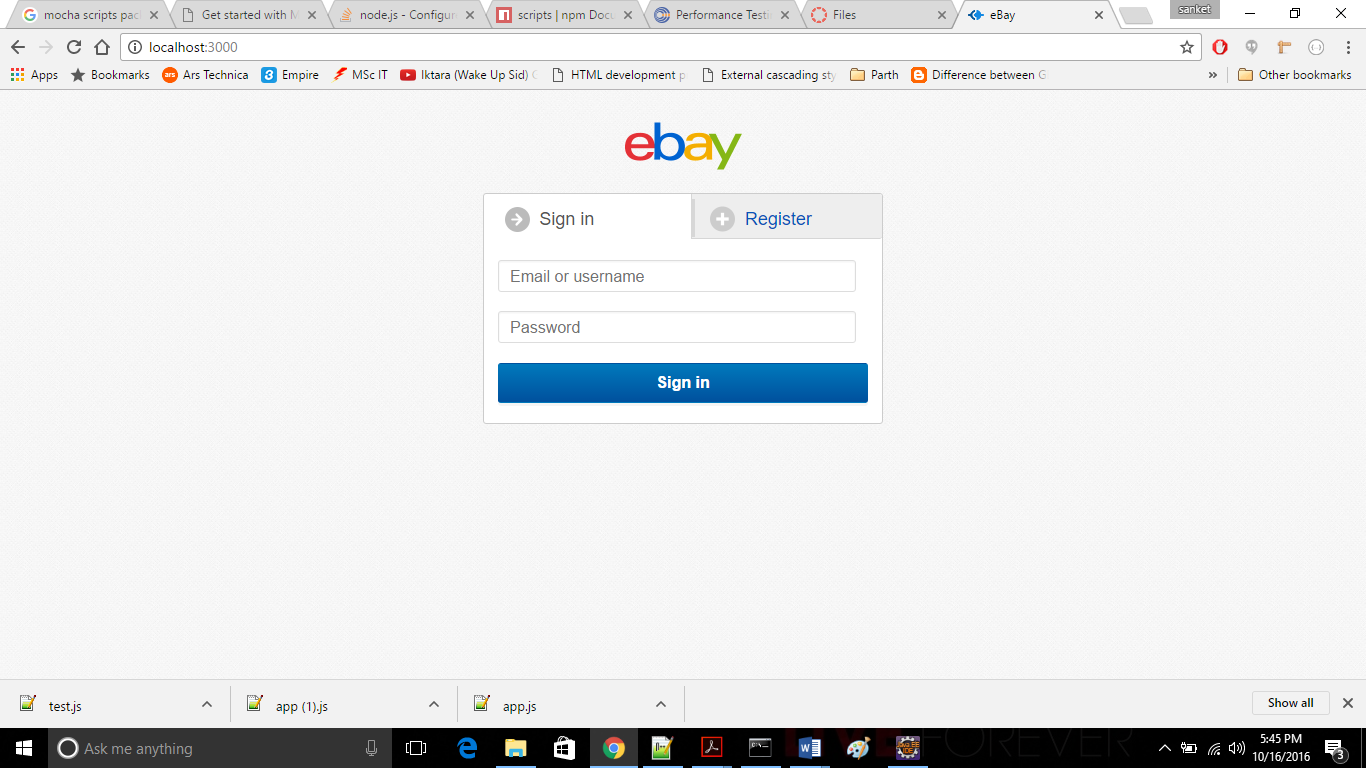
# https://www.rabbitmq.com/img/tutorials/python-six.png

# Results:

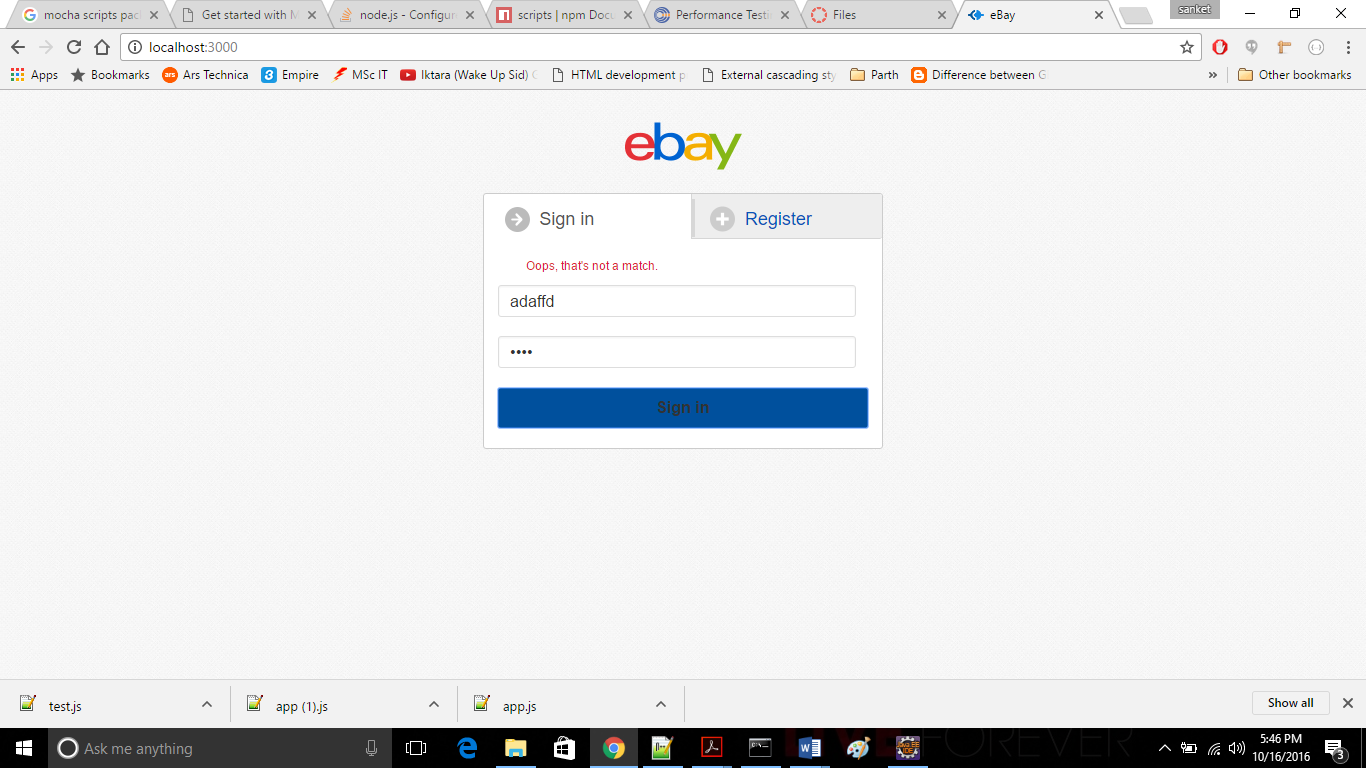
*Screen captures of Calculator*

***eBay***

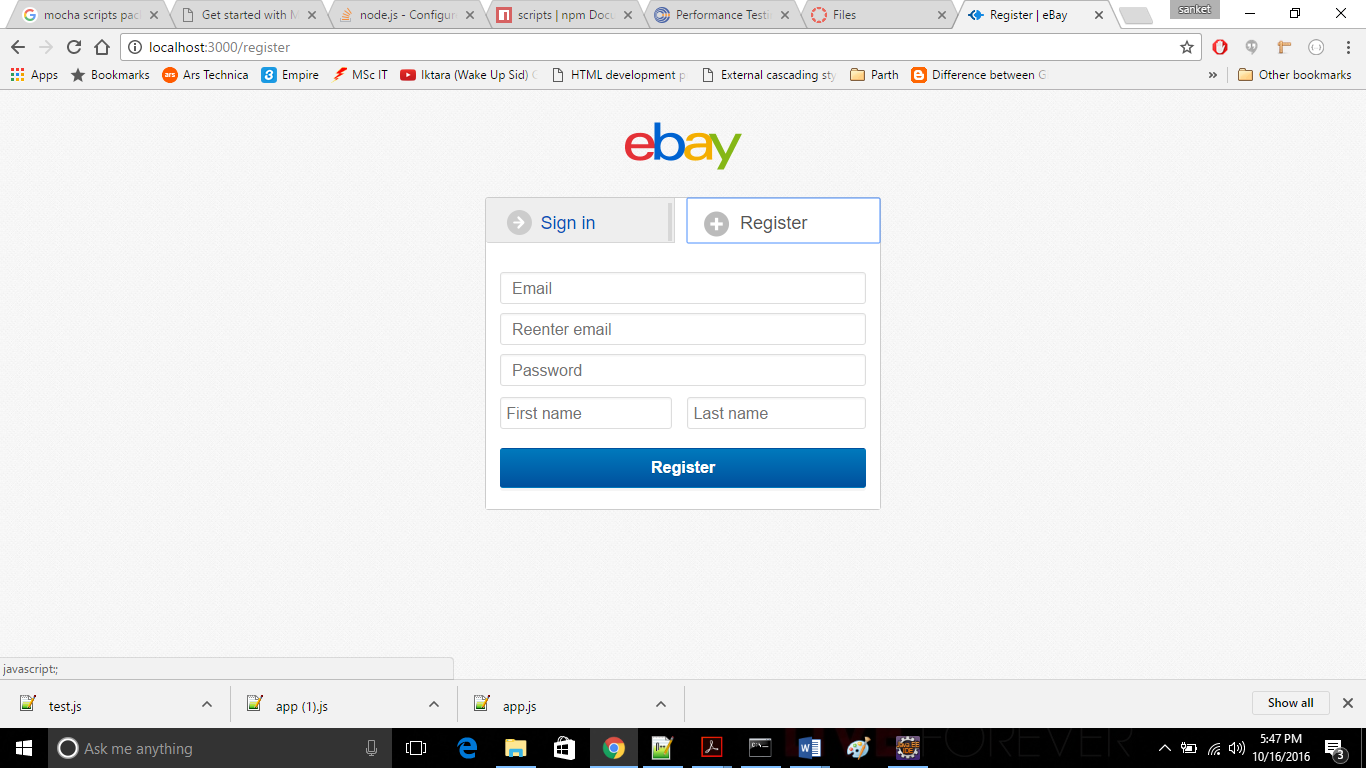
**Signin**

****

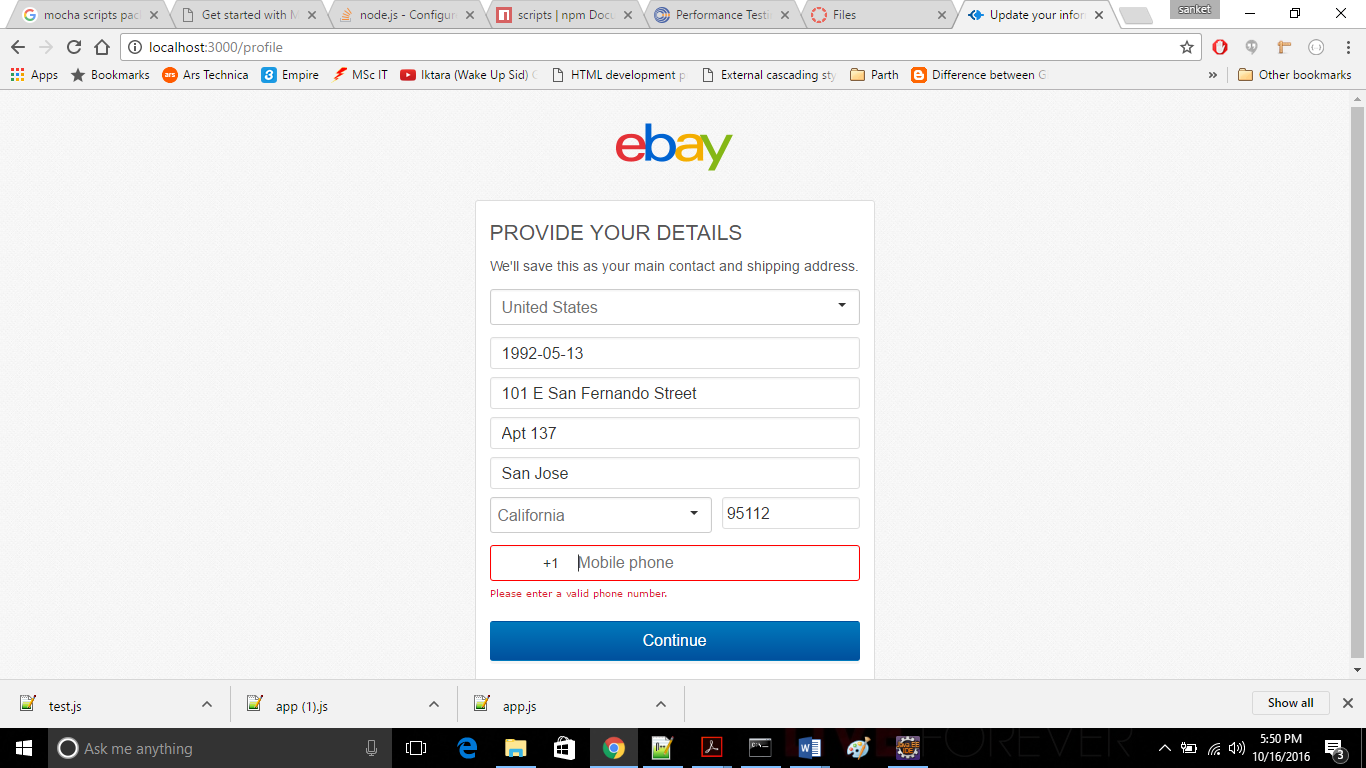
**Signin Error**

****

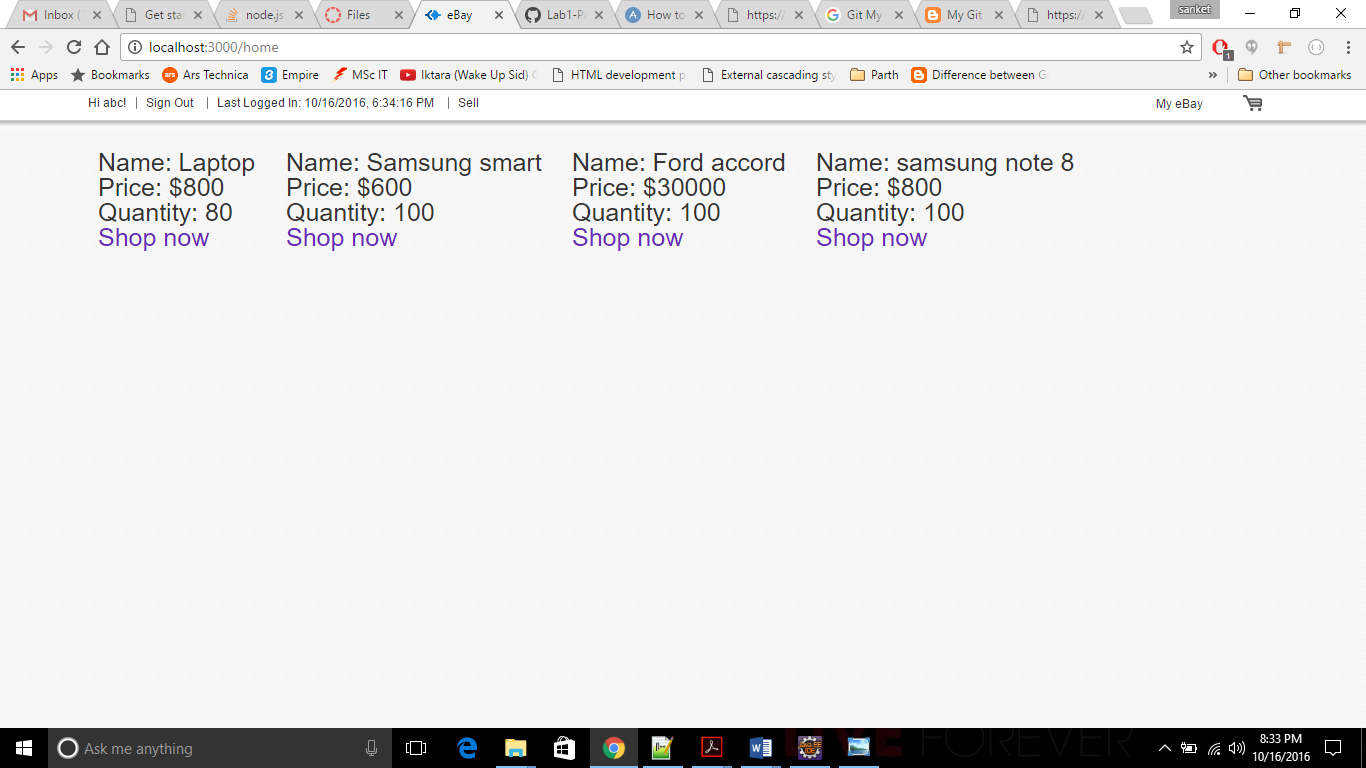
**Register**

****

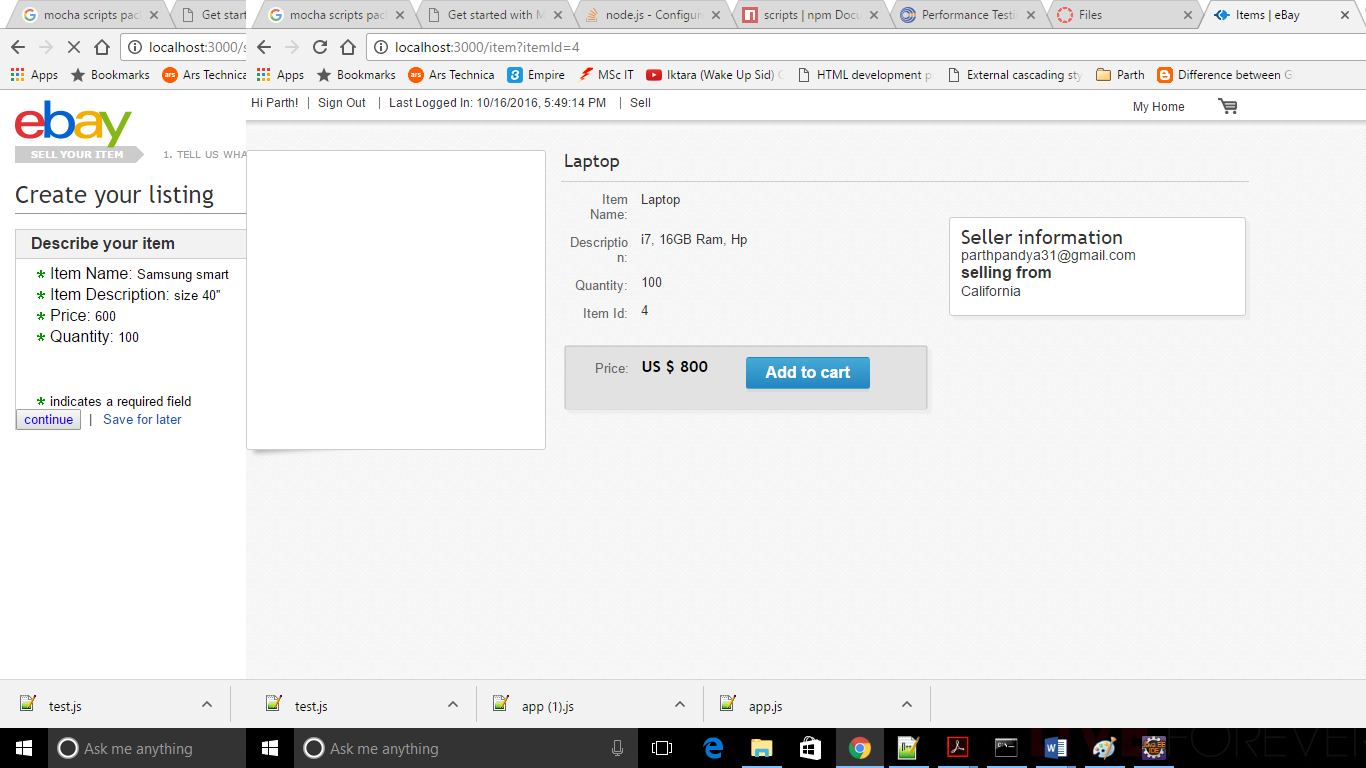
**Register Error**

****

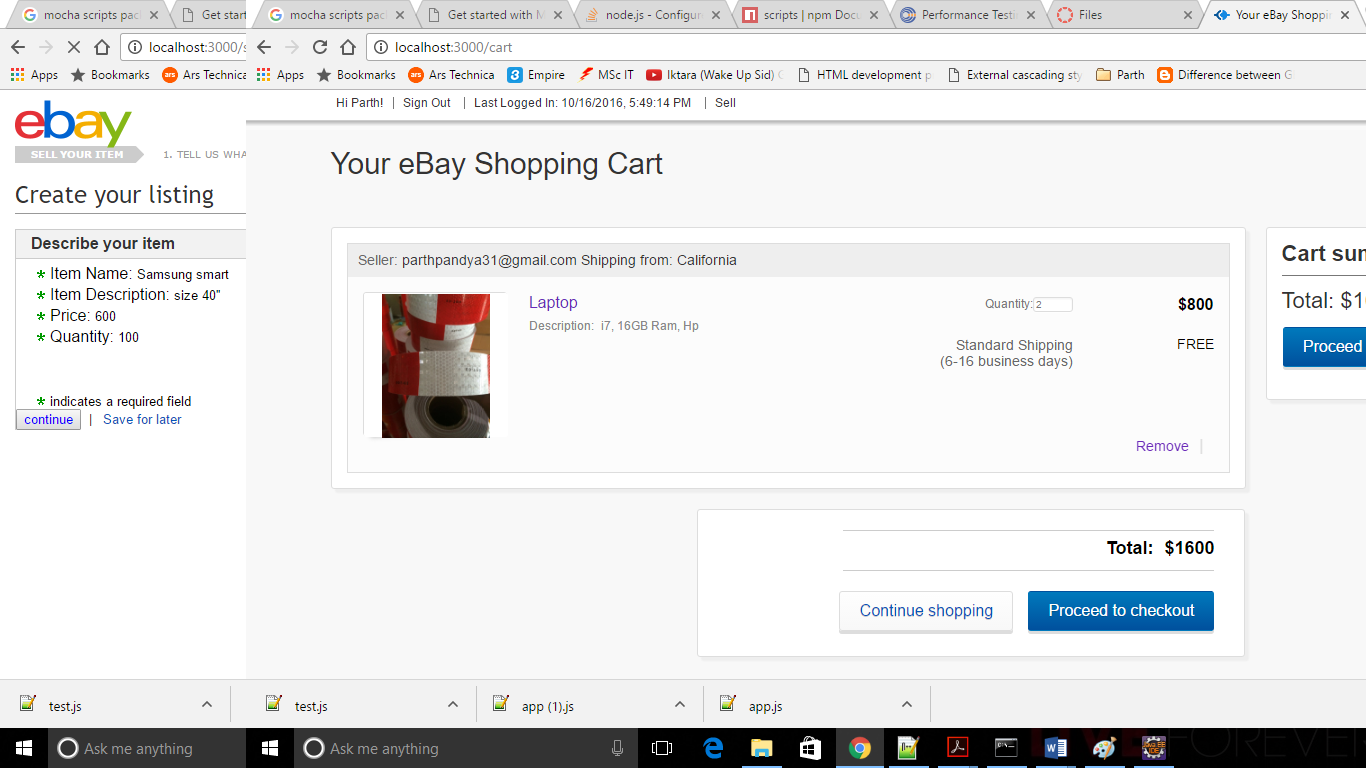
**Home Page**



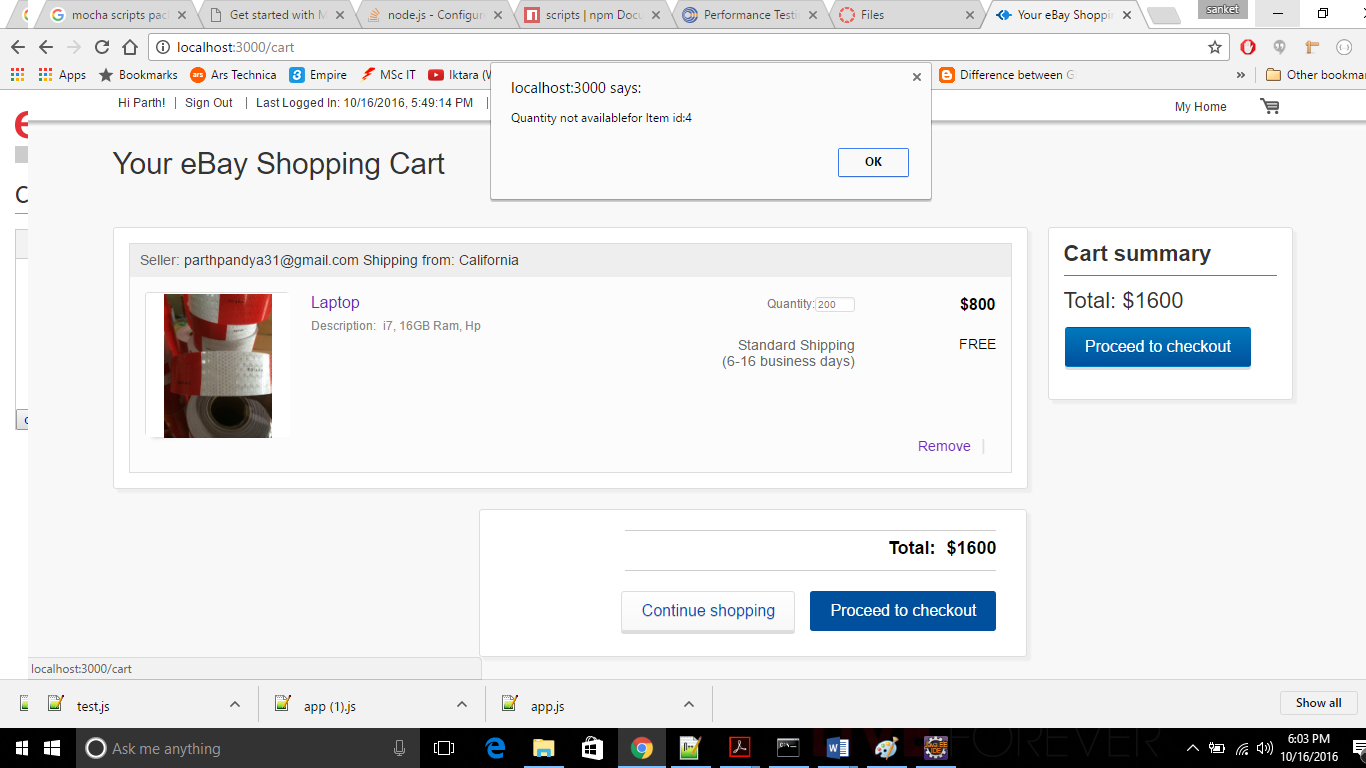
**Items**

****

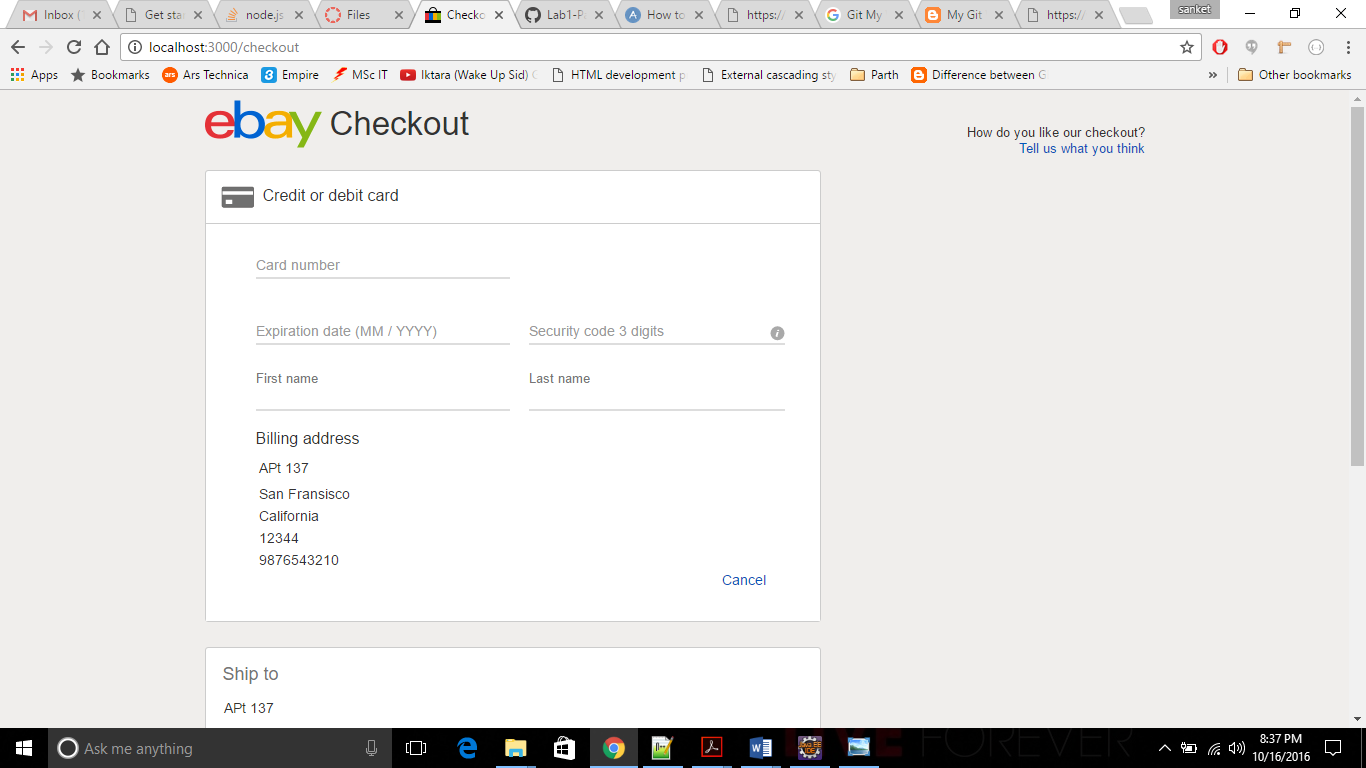
**Cart**

****

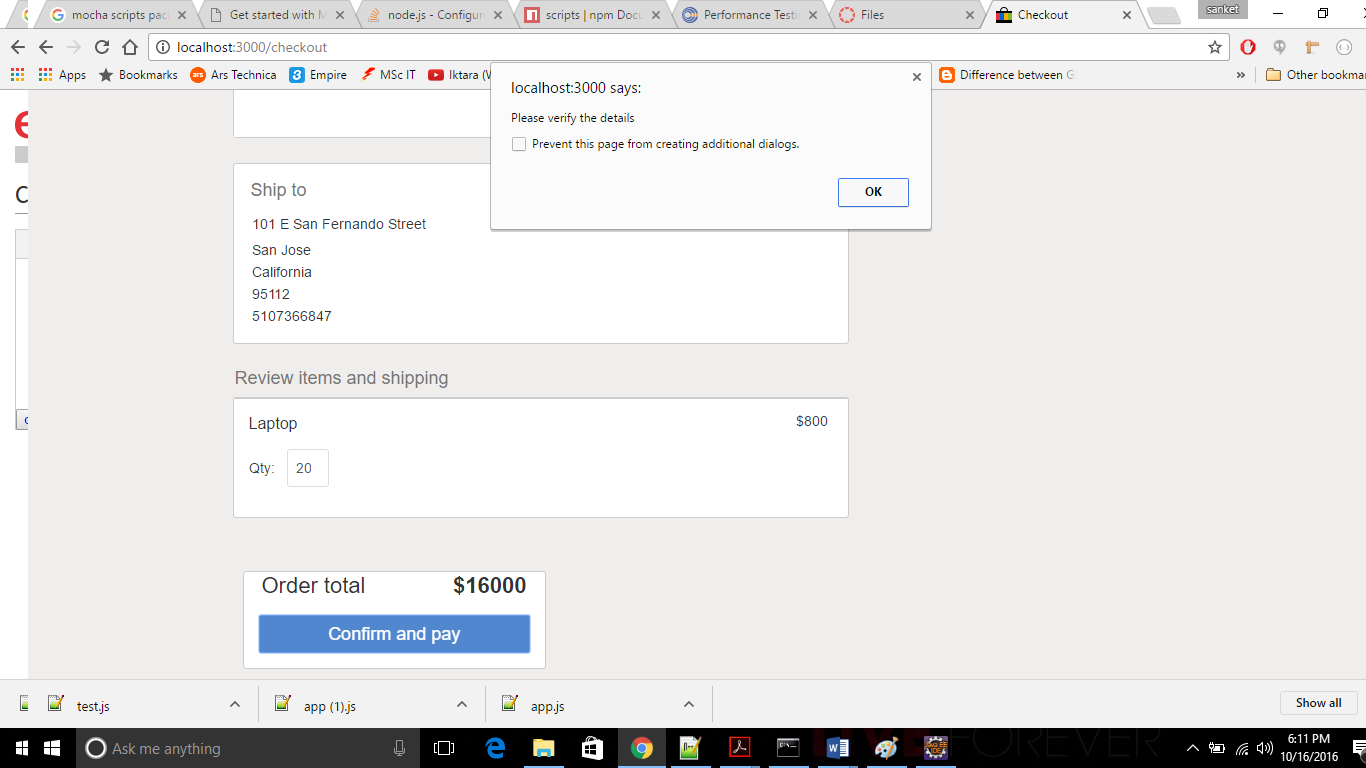
**Cart Validation**

****

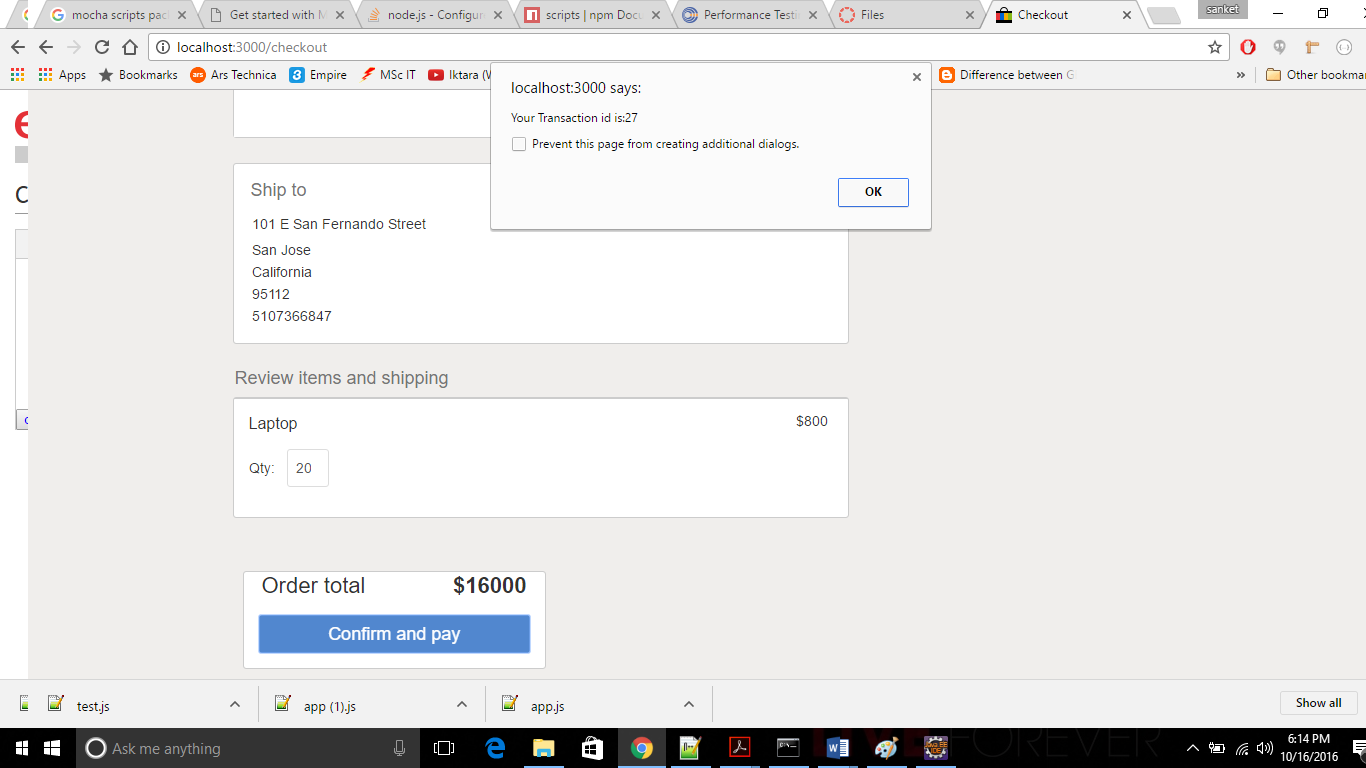
**Checkout**



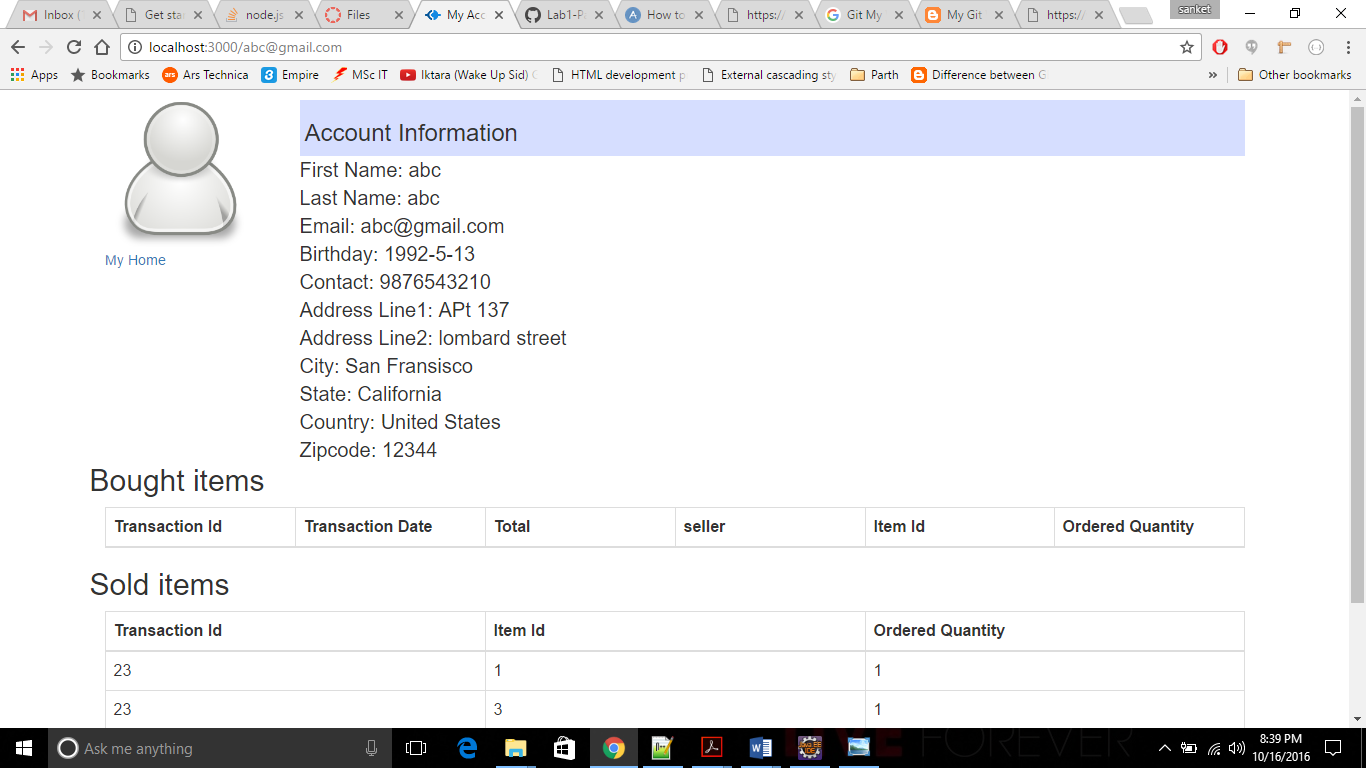
**Checkout Validation**

****

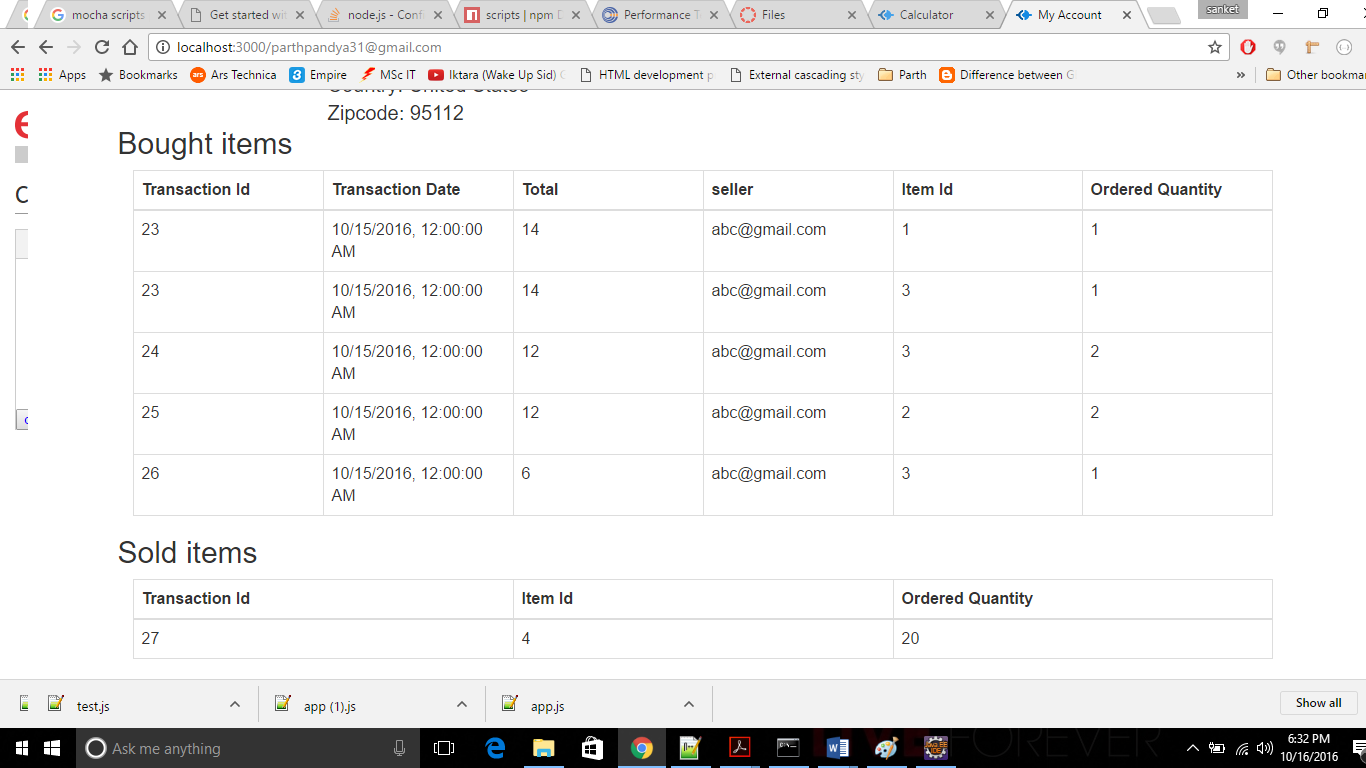
**Successful checkout**

****

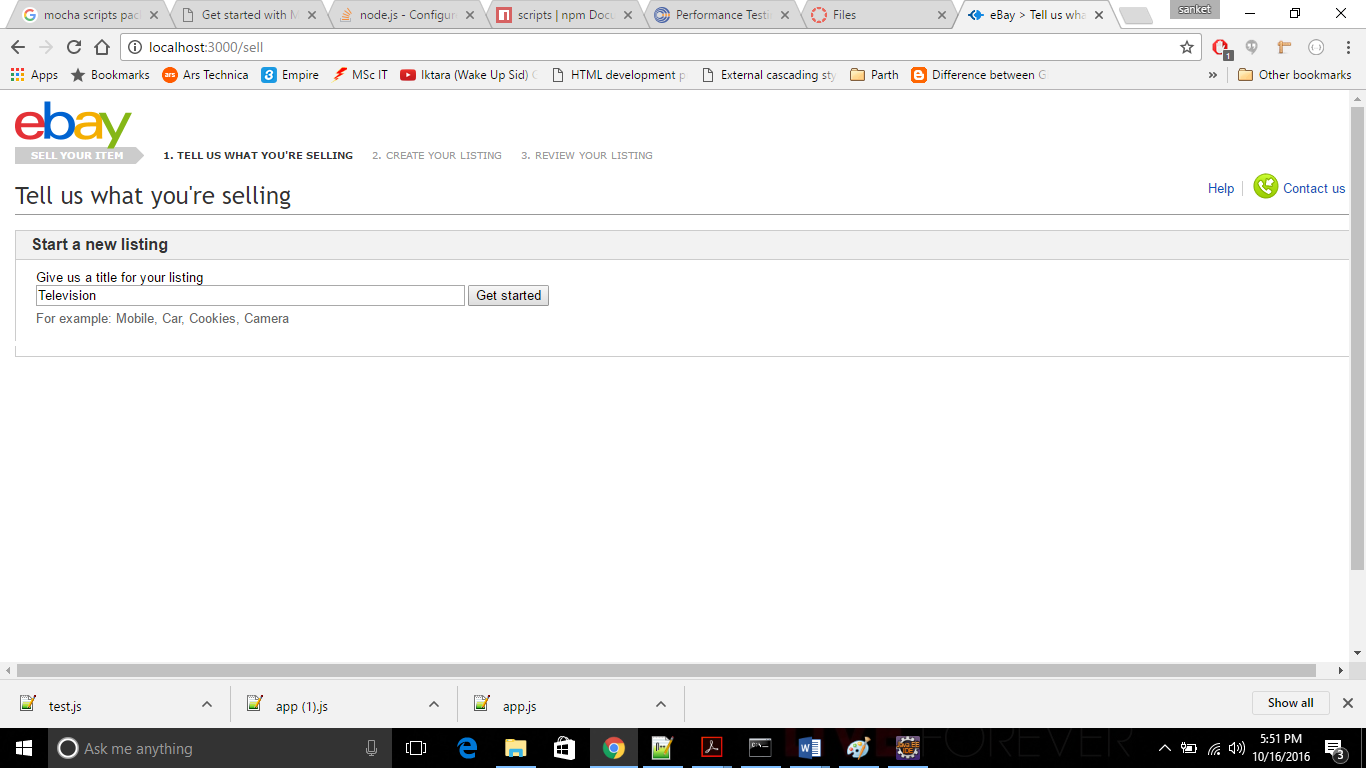
**Handler Page**



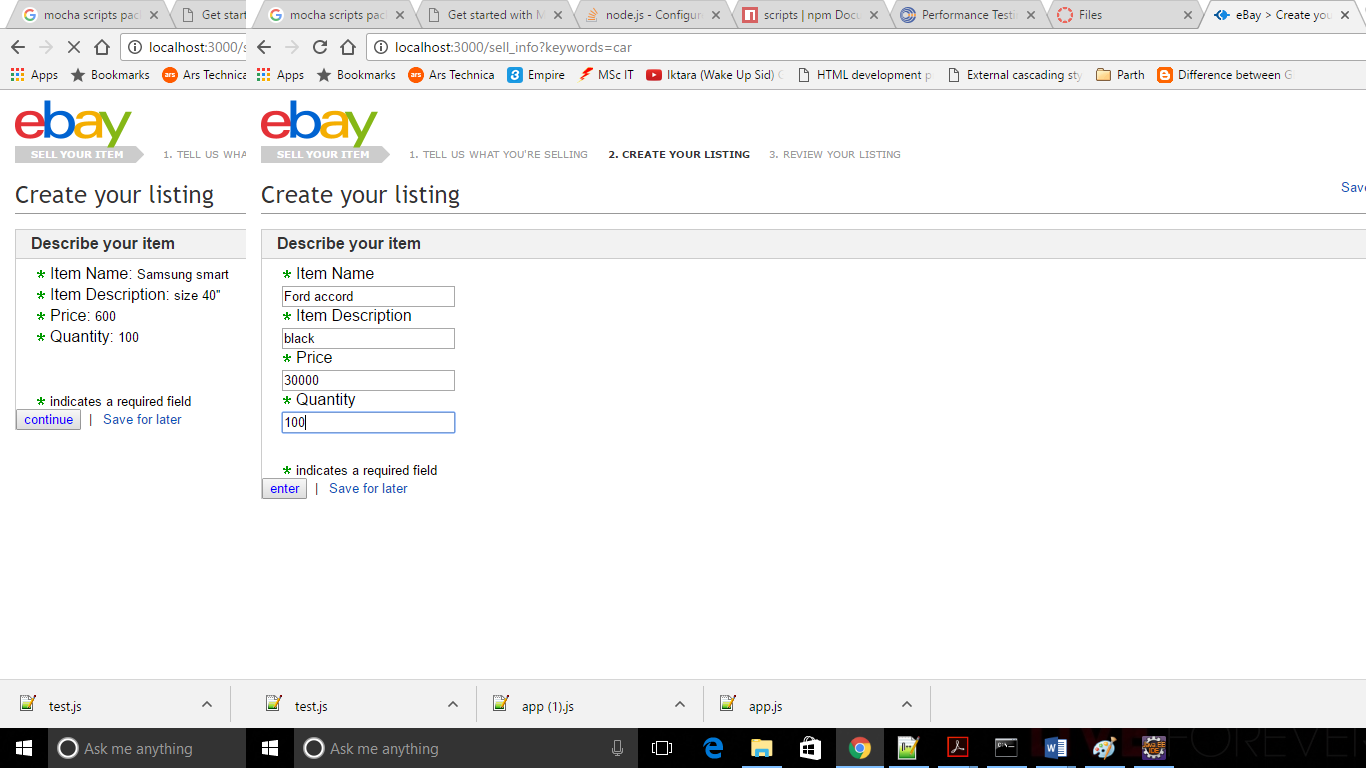
**Bought and Sold Items**

****

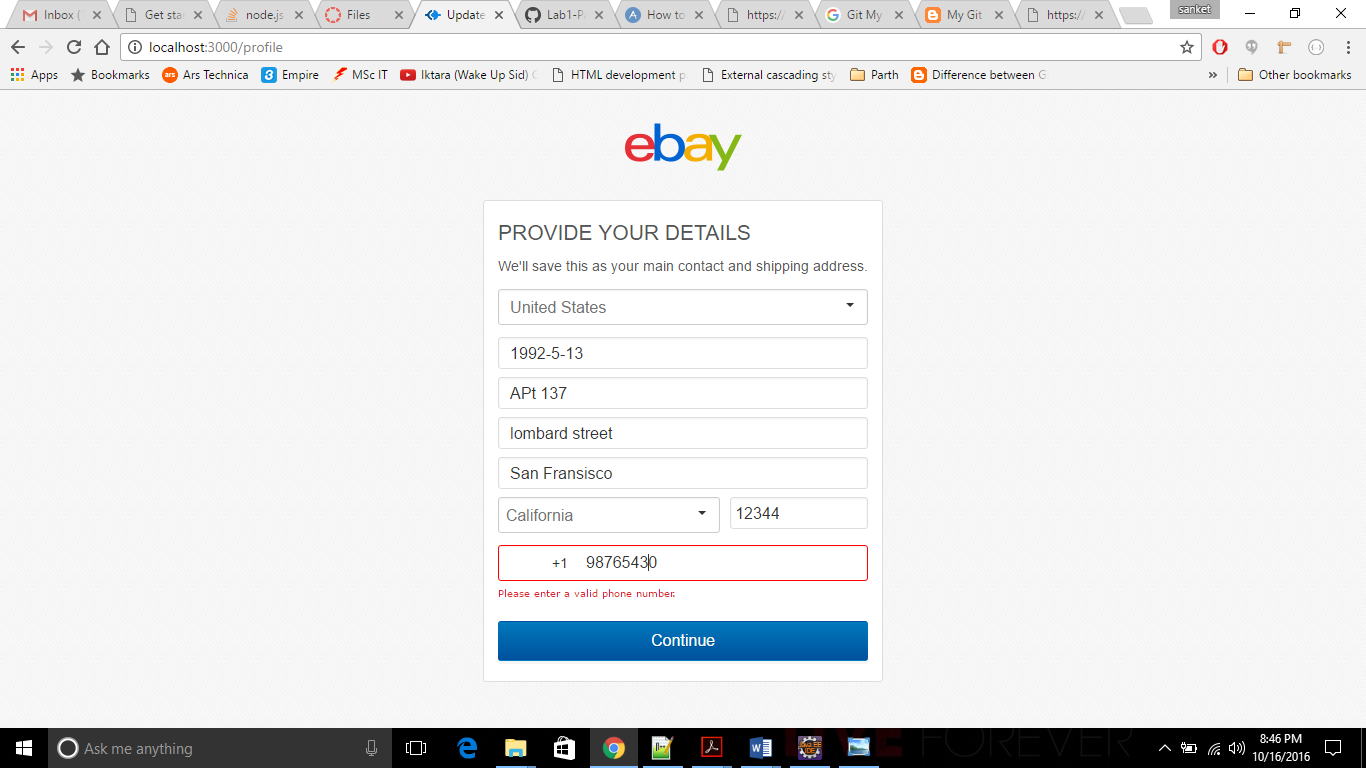
**Sell item**

****

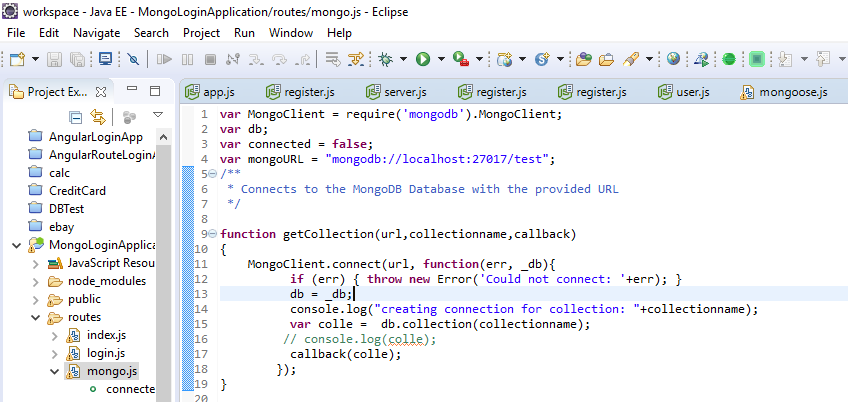
**Item Details**

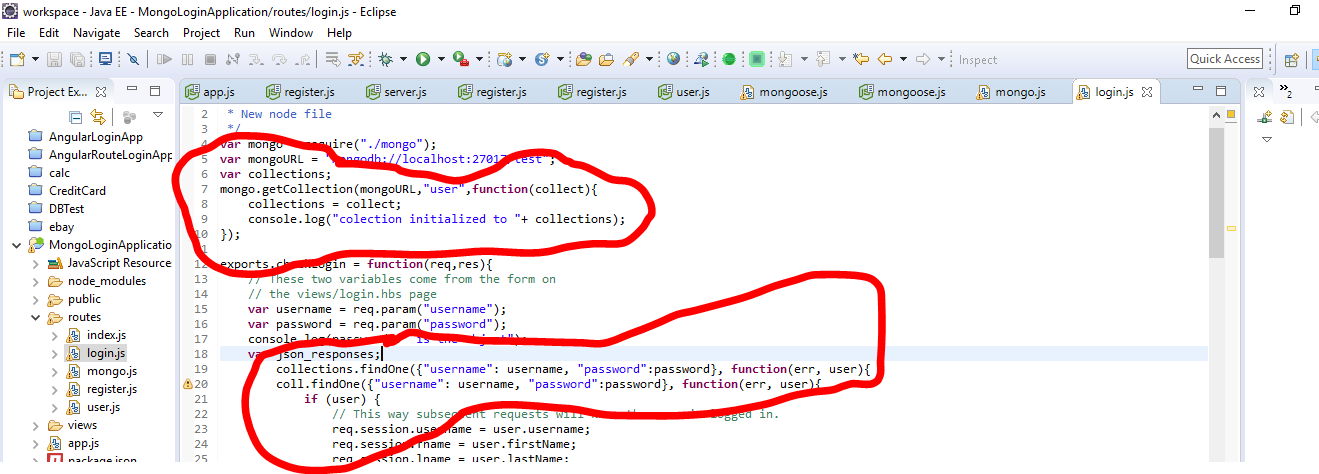
****

**Update Profile**

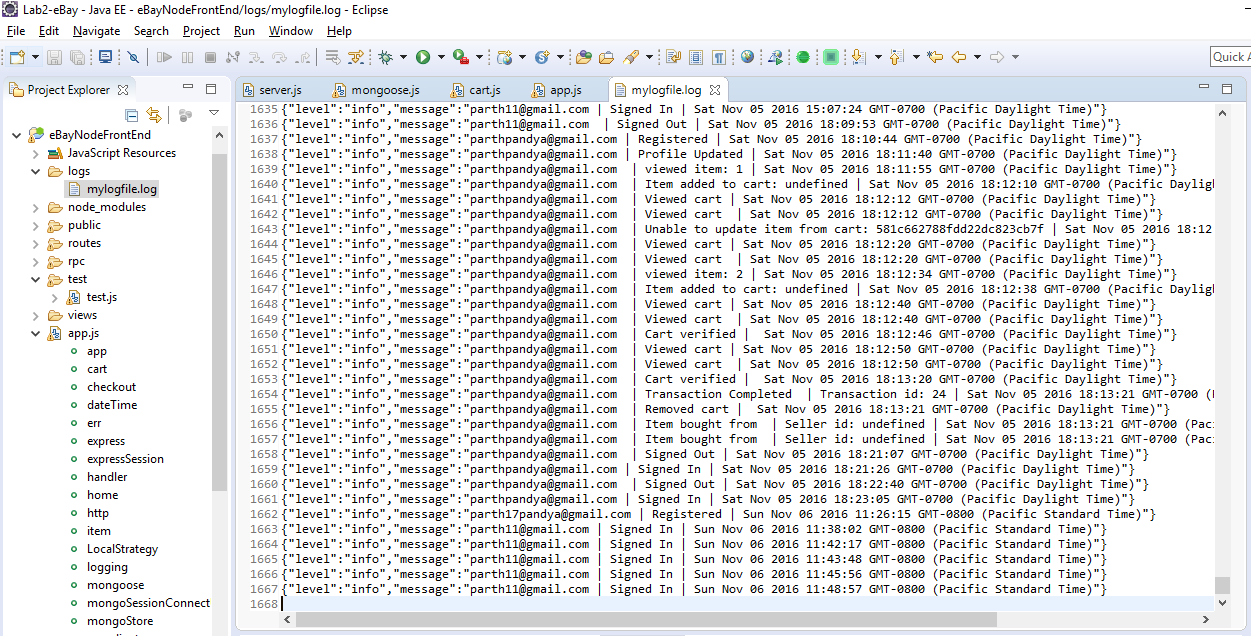


**Connection Pooling**

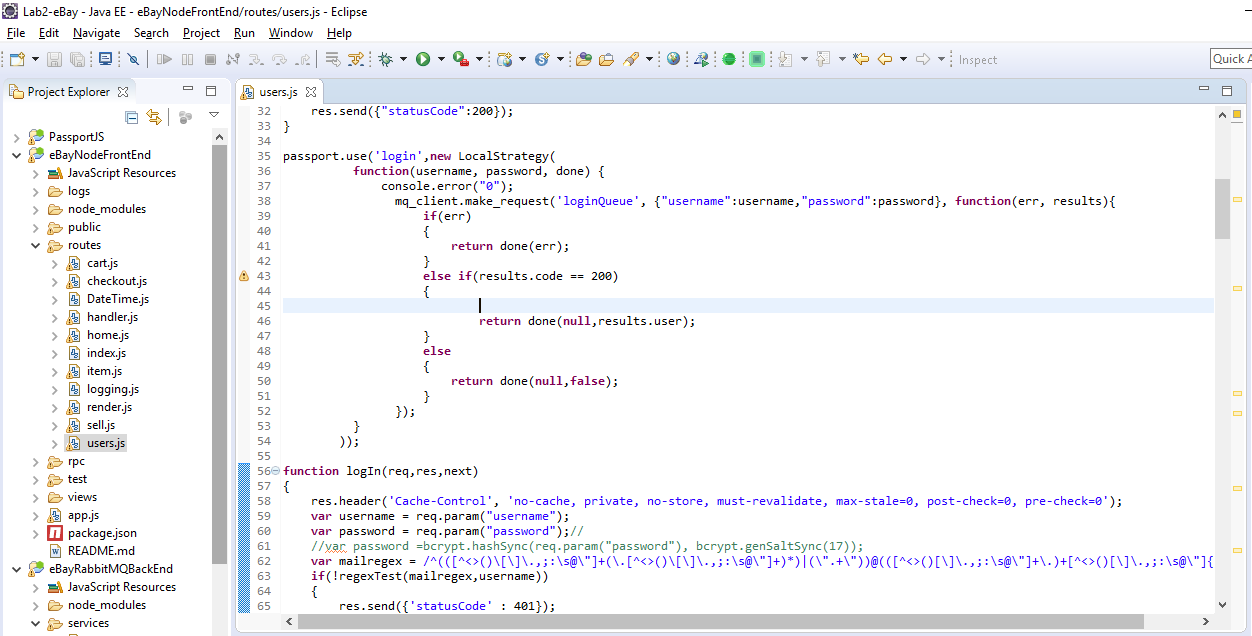




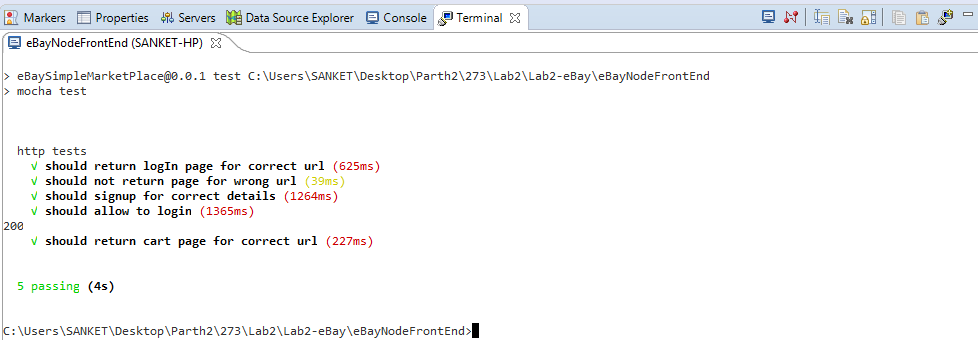
**User tracking**



**Passport authentication**



**Mocha Testing**

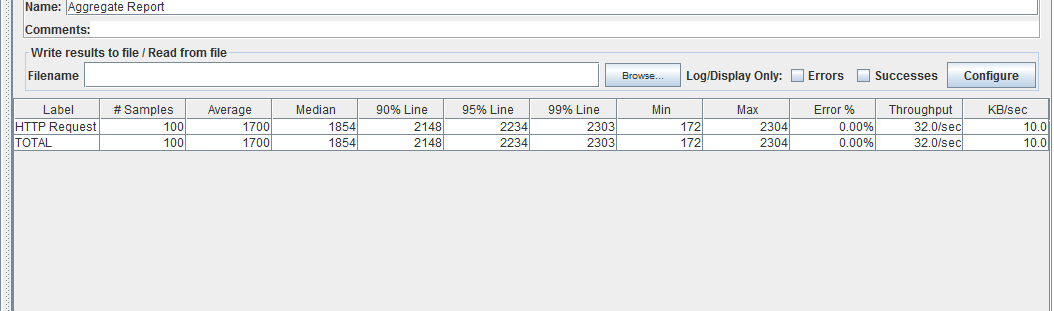
****

**eBay Jmeter Testing**

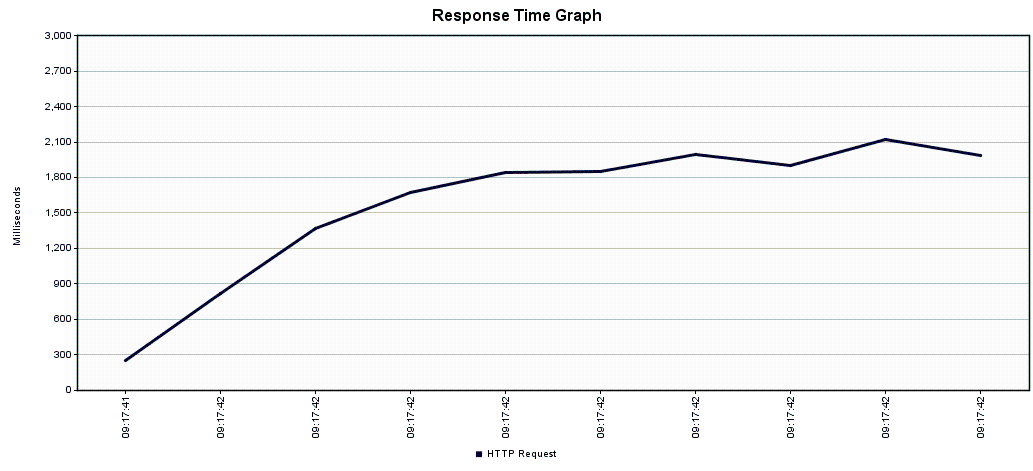
**100 users**

**Without Connection Pooling**

**Aggregate Report**

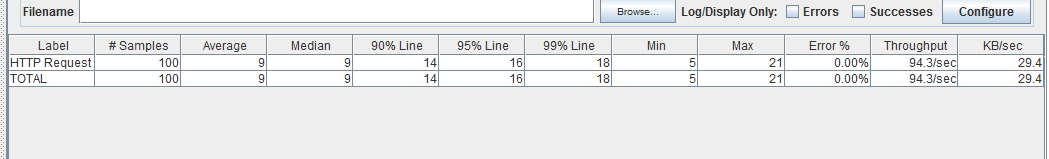
****

**Graph**

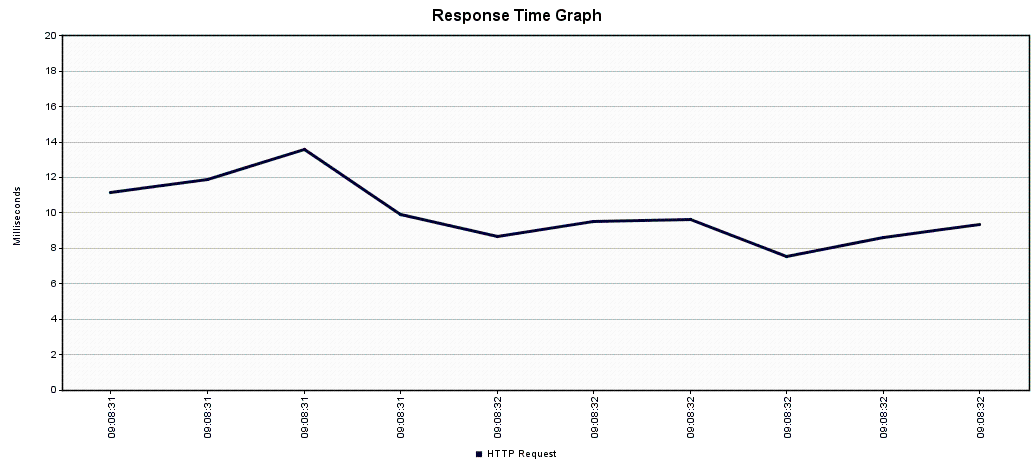
****

**With Connection Pooling**

**Aggregate Report**

****

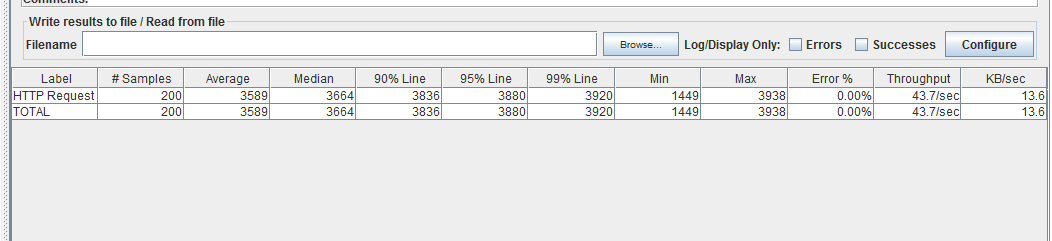
**Graph**

****

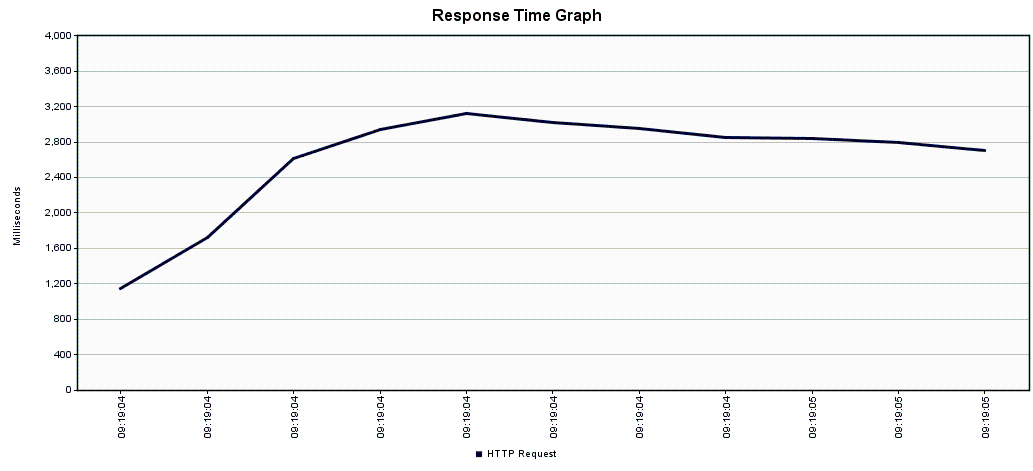
**200 users**

**Without connection Pooling**

**Aggregate report**

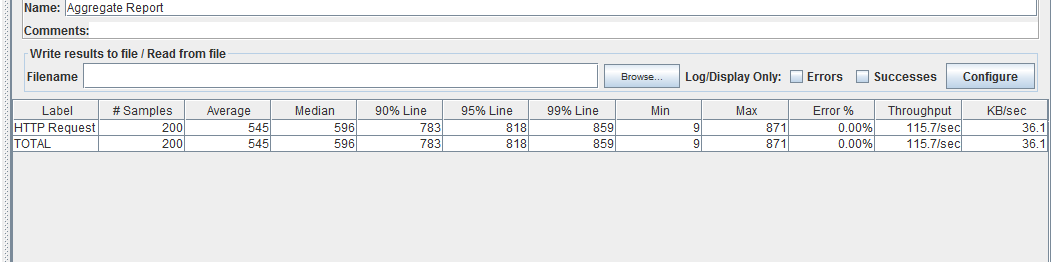
****

**Graph**

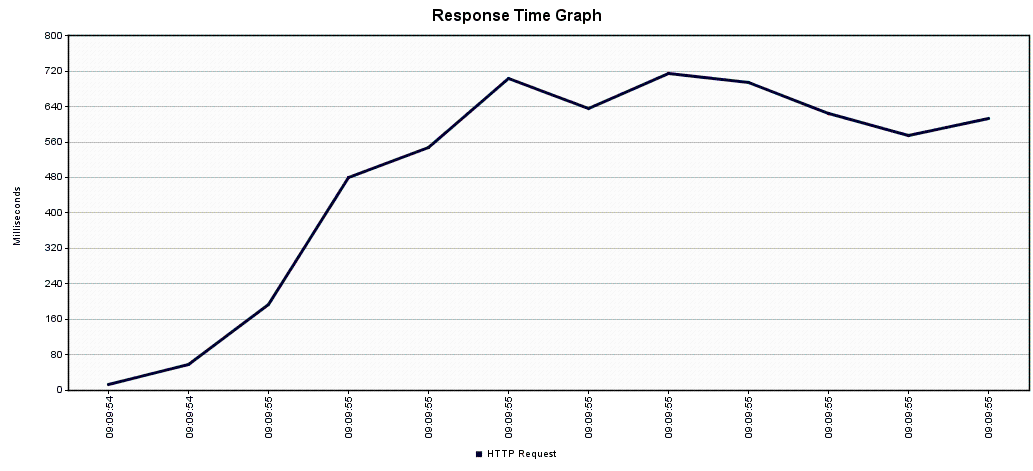
****

**With Connection pooling**

**Aggregate report**

****

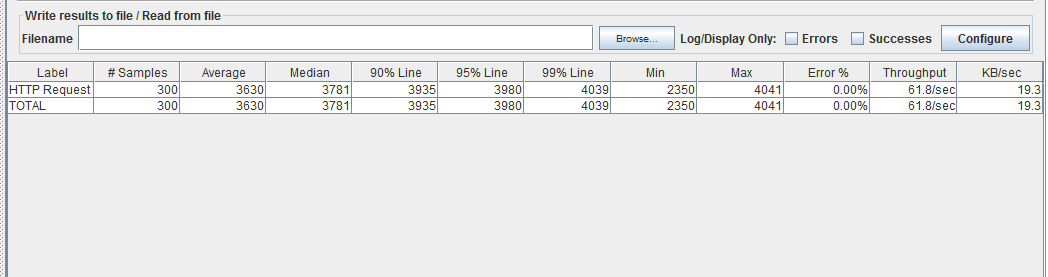
**Graph**

****

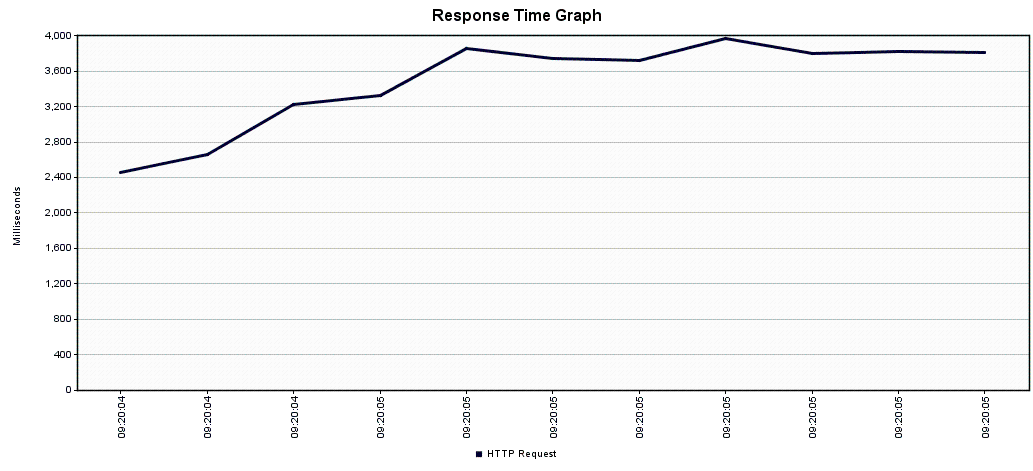
**300 users**

**Without connection pooling**

**Aggregate report**

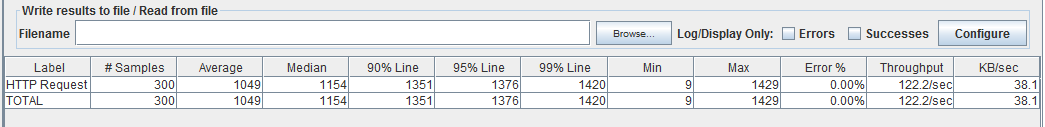
****

**Graph**

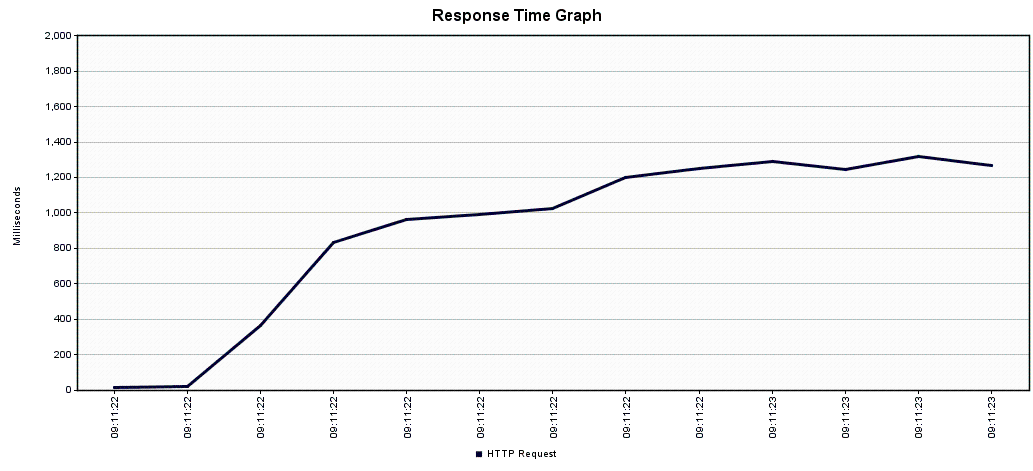
****

**With connection pooling**

**Aggregate report**

****

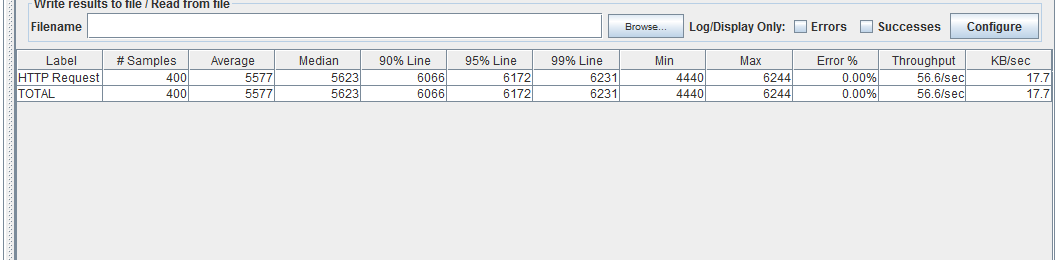
**Graph**

****

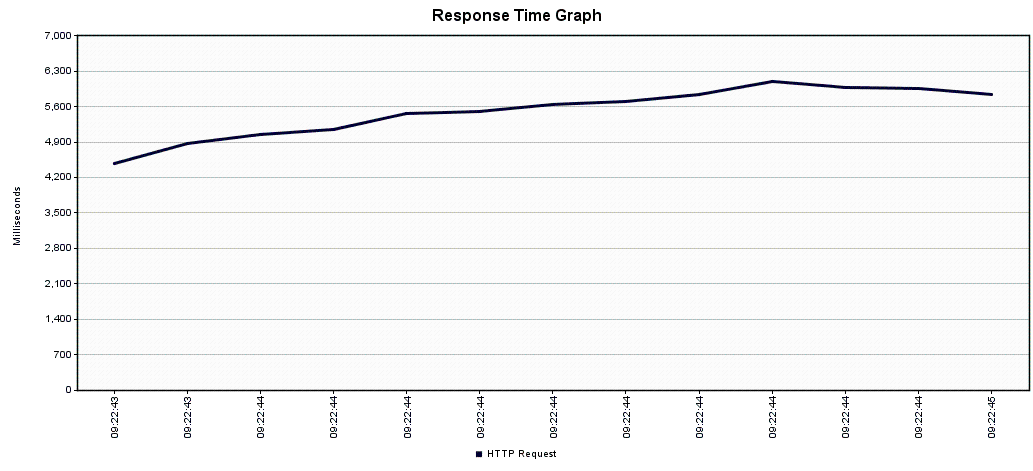
**400 users**

**Without connection pooling**

**Aggregate report**

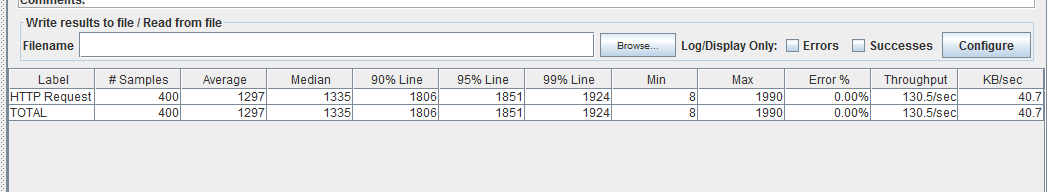
****

**Graph**

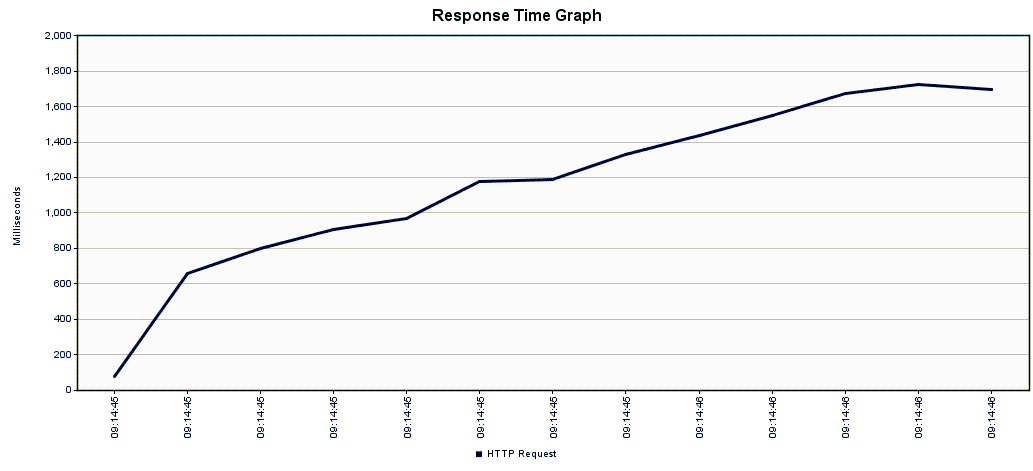
****

**With connection pooling**

**Aggregate Report**

****

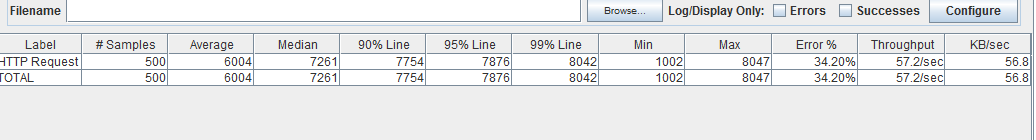
**Graph**

****

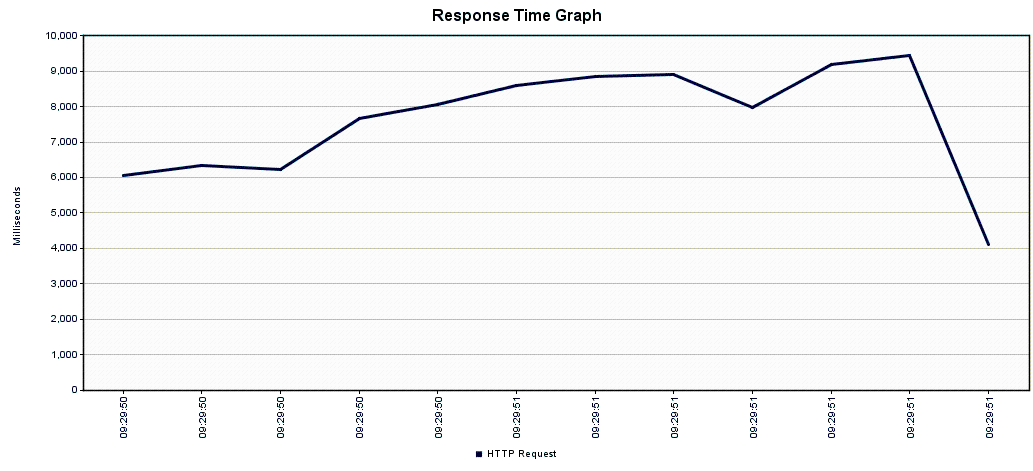
**500 users**

**Without connection pooling**

**Aggregate Report**

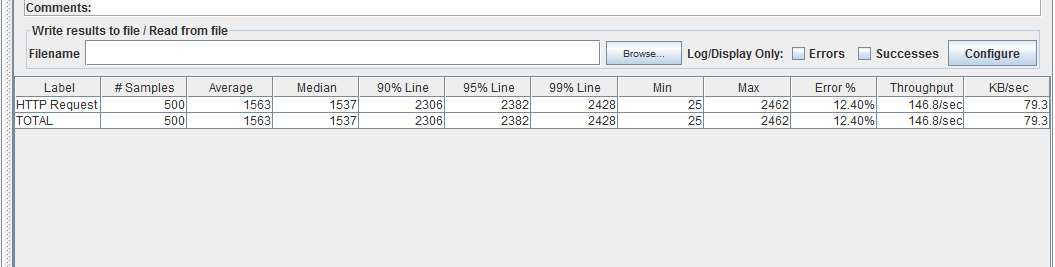
****

**Graph**

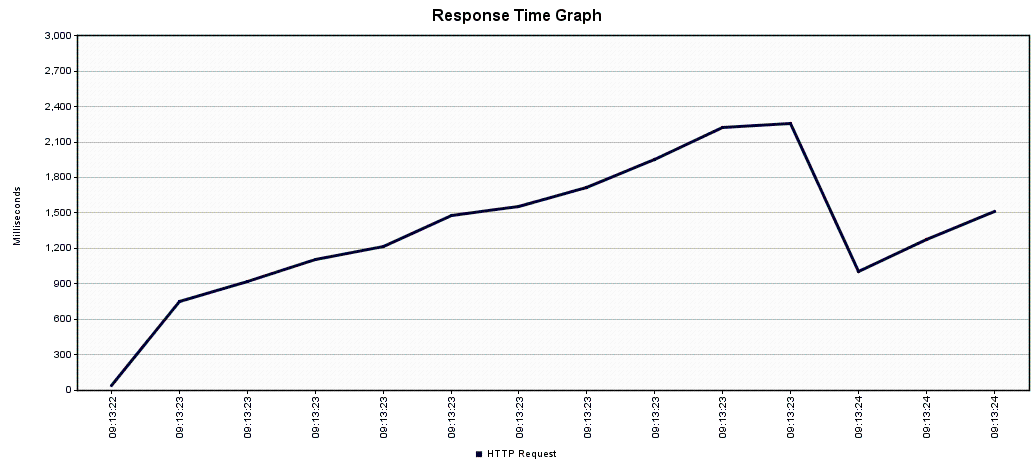
****

**With connection pooling**

**Aggregate Report**

****

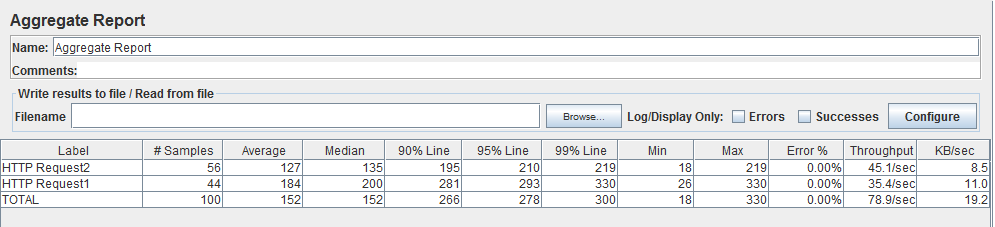
**Graph**

****

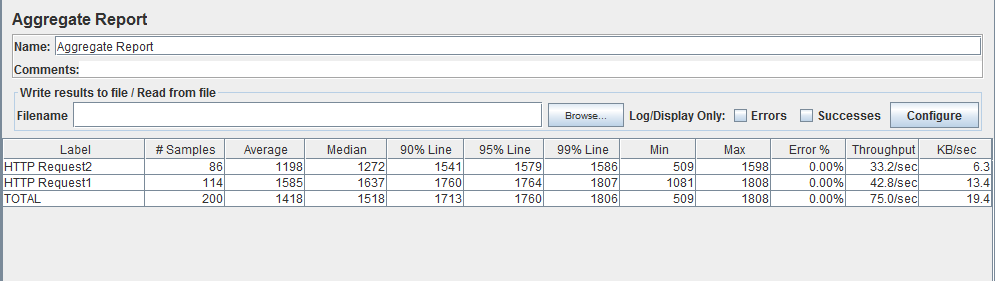
***Part 2***

**Performance without RabbitMQ**

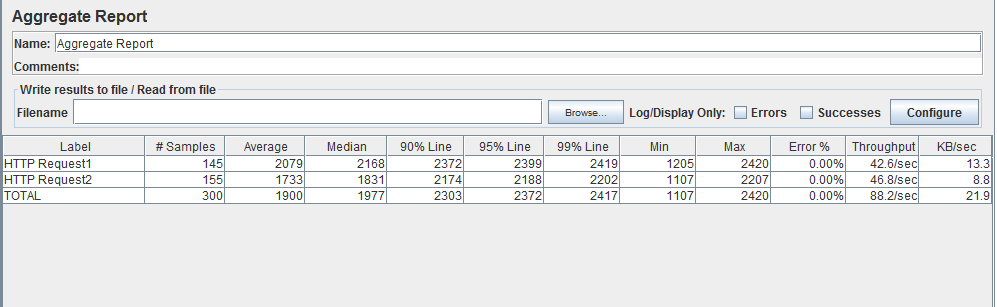
**100 users**



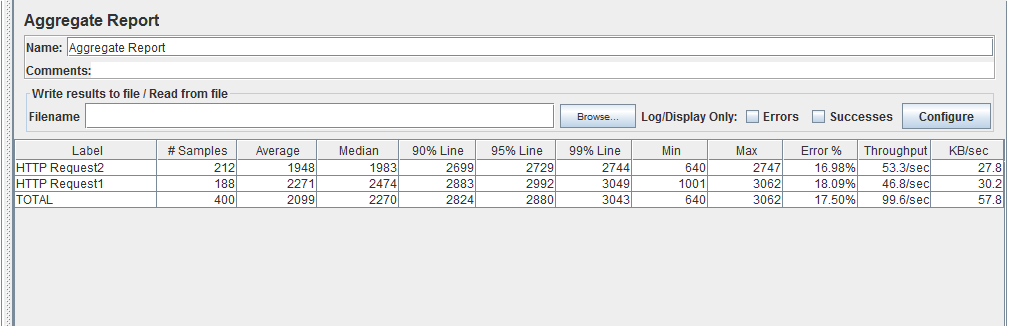
**200 Users**

****

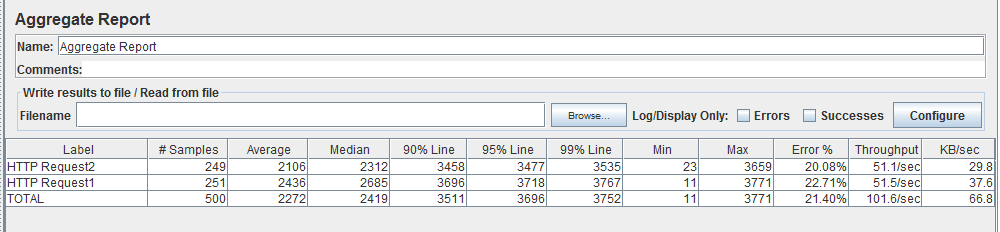
**300 Users**

****

**400 Users**

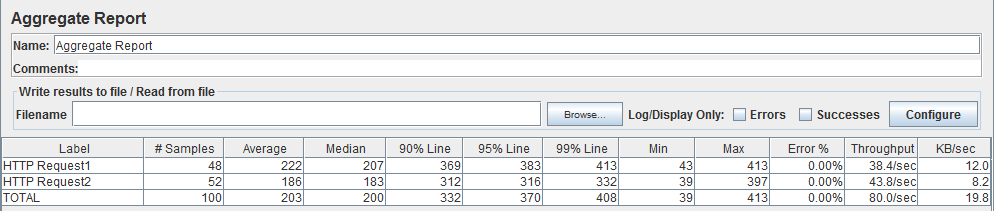
****

**500 Users**

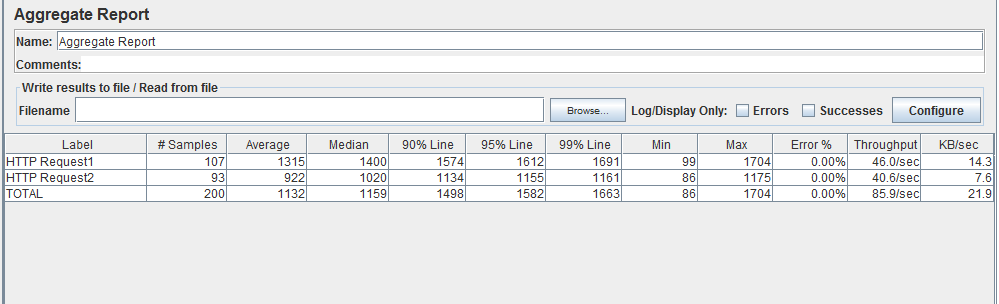


**With RabbitMQ**

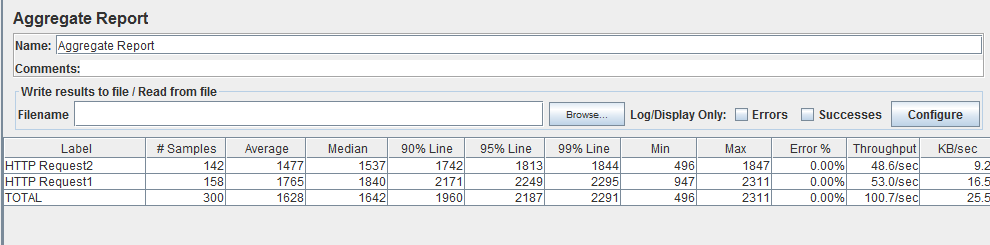
**100 User**

****

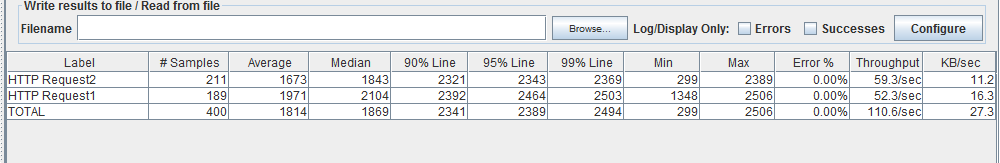
**200 User**

****

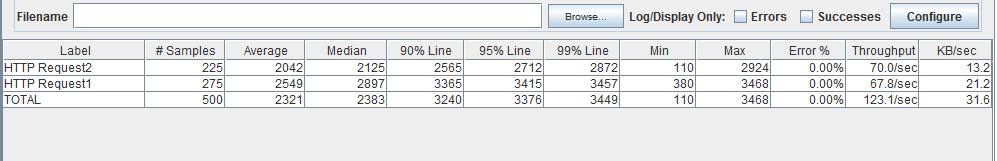
**300 User**

****

**400 User**

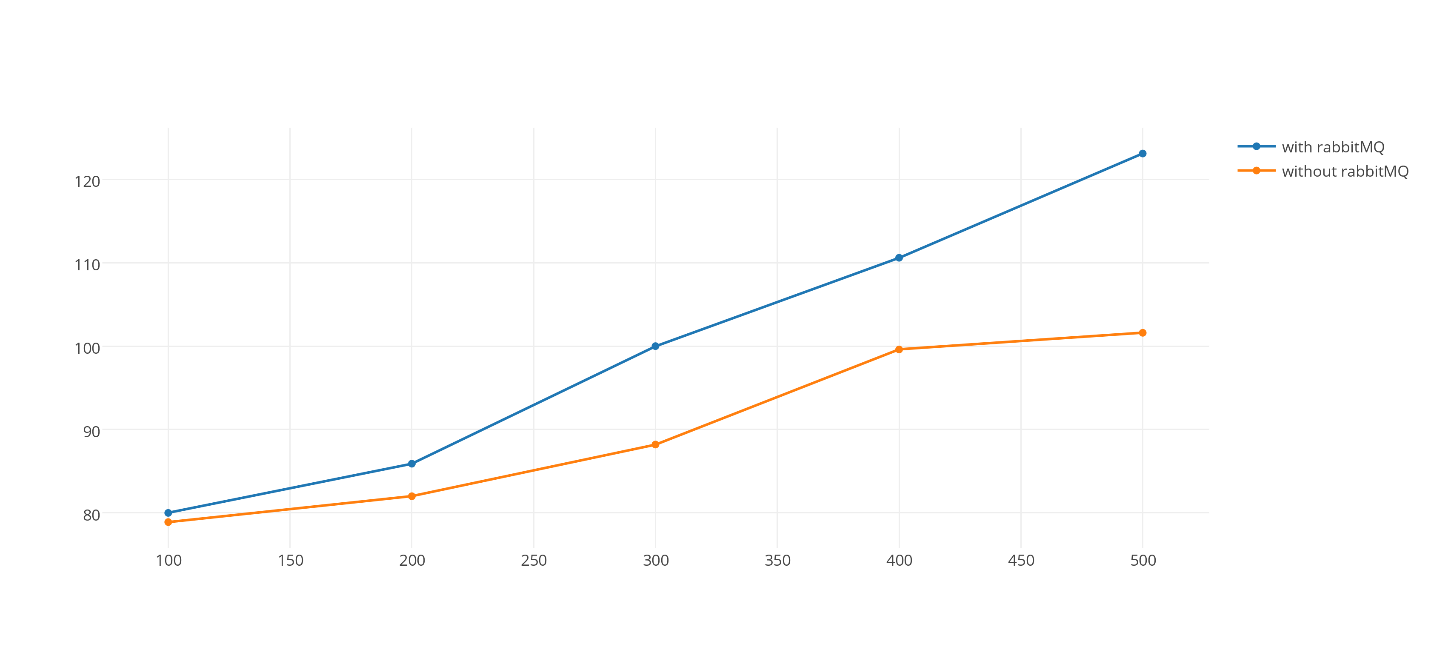
****

**500 User**

****

|  |  |  |
| --- | --- | --- |
| **Users** | **Throughput w/o rabbitMQ** | **Throughput with rabbitMQ** |
| 100 | 78.9 | 80.0 |
| 200 | 82.0 | 85.9 |
| 300 | 88.2 | 100.7 |
| 400 | 99.6 | 110.6 |
| 500 | 101.6 | 123.1 |

**Graph**

****

**Conclusion**

As the number of users increases, the throughput increases with rabbitMQ as compared to the strategy employed without rabbitMQ. This is because of the asynchronous message queueing protocol which allows multiple request to be queued and process them as soon as possible.

2)

Bcrypt

Brcypt encryption as a part of passport authentication is used in lab2 which employs **adaptive hash algorithm**. It is better as compared to tradition encryption techniques as it uses **derived key for encryption** of known plain text. It comprises of five parts.

1. $2a or $2b or $2b - which indicates bcrypt encryption is used.
2. Cost factor c- 2c iterations of key derivation function are used.
3. 22 character salt
4. 31 character of base 64 encoded hash.

Brcypt is far more resistant to brute force attack as compred to tradition hashing like MD5. With a cost factor of 10, it can take up to 210 hashing, hence it is difficult to break the encrypted password. This comes at the cost of large number of CPU cycles for encryption, as a slow function is slow for both, an attacker and defender.

3)

MySQL follows ACID properties. Hence it is better to be used for those data of application where multiple transactions need to be performed and any failure in performing one of the operation should cause rollback of all previous operations.

Updating quantity, performing payment, etc. should be done using mysql.

NoSQL should be used when need to handle large amount of unstructured data. Mongo is light weight, scalable, allows unstructured and de-normalized storage of data in simpler format like JSON. Hence it is better to be used when data is unpredictable, large and requirement loose or evolving. Data such as comments, posts, messages, different types of data, etc should be stored using Mongodb.