**Ebay**

Github URL:

https://github.com/gaurangmhatre/Lab1\_Ebay

**Introduction: Goal and purpose of the system**

This website is a site where user can buy product sold by other user. Also users can auction their products where other users can bid on it. After bidding slot of 4 days, the auction gets over and highest bidder can get a message to pay for the item in Auction tab.

**Goal:** The goal is to provide portal for online shopping and selling.

**Purpose:** The client will send operation request to server and server will respond with the result of the operation.

**System Design:**

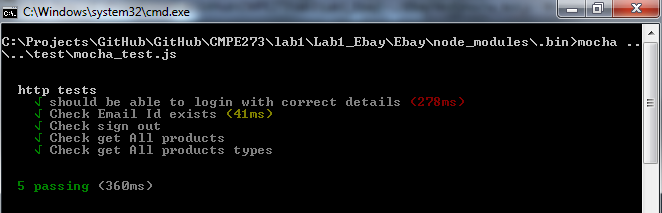
The Ebay web site is coded in Angular JS with UI in bootstrap. The calls are handled in NodeJS. Also on server other node packages used are Express, Winston, client-sessions.

For testing NPM packages used are mocha, chai.

Testing:

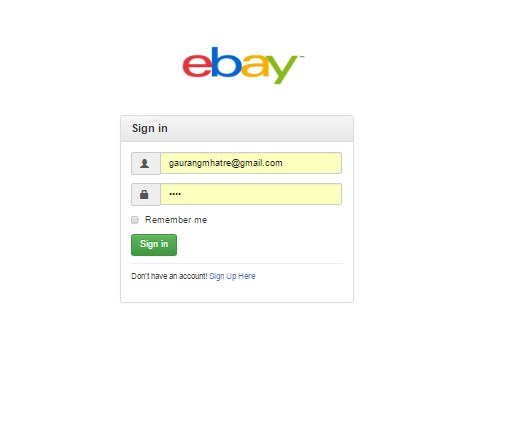
1. Testing with **J-Meter** is added in the answer of Question 2 bellow.
2. **Mocha:** <code added in solution test folder>

Five API Calls

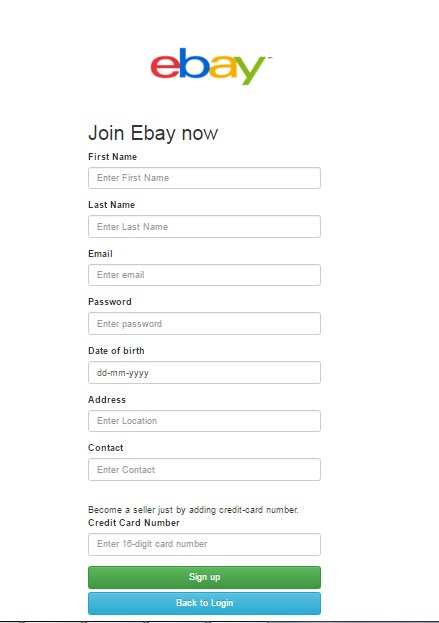


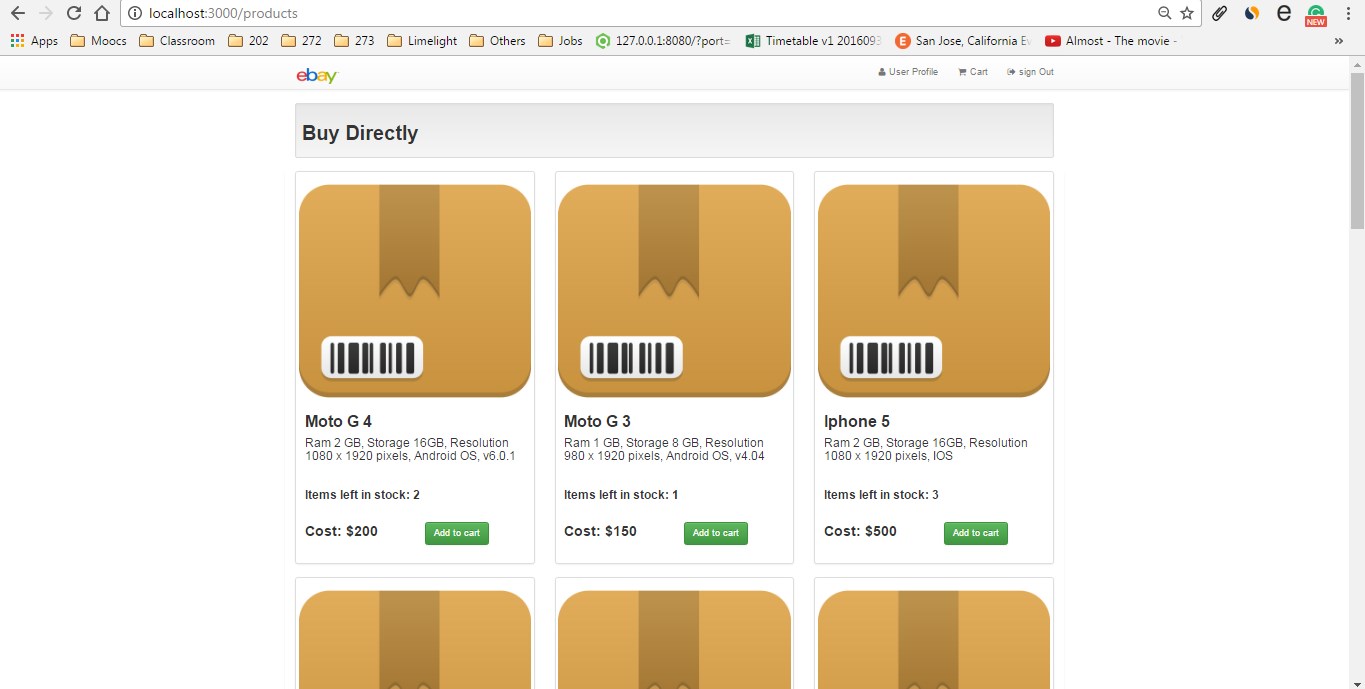
**Results and Screen Capture for Client:**

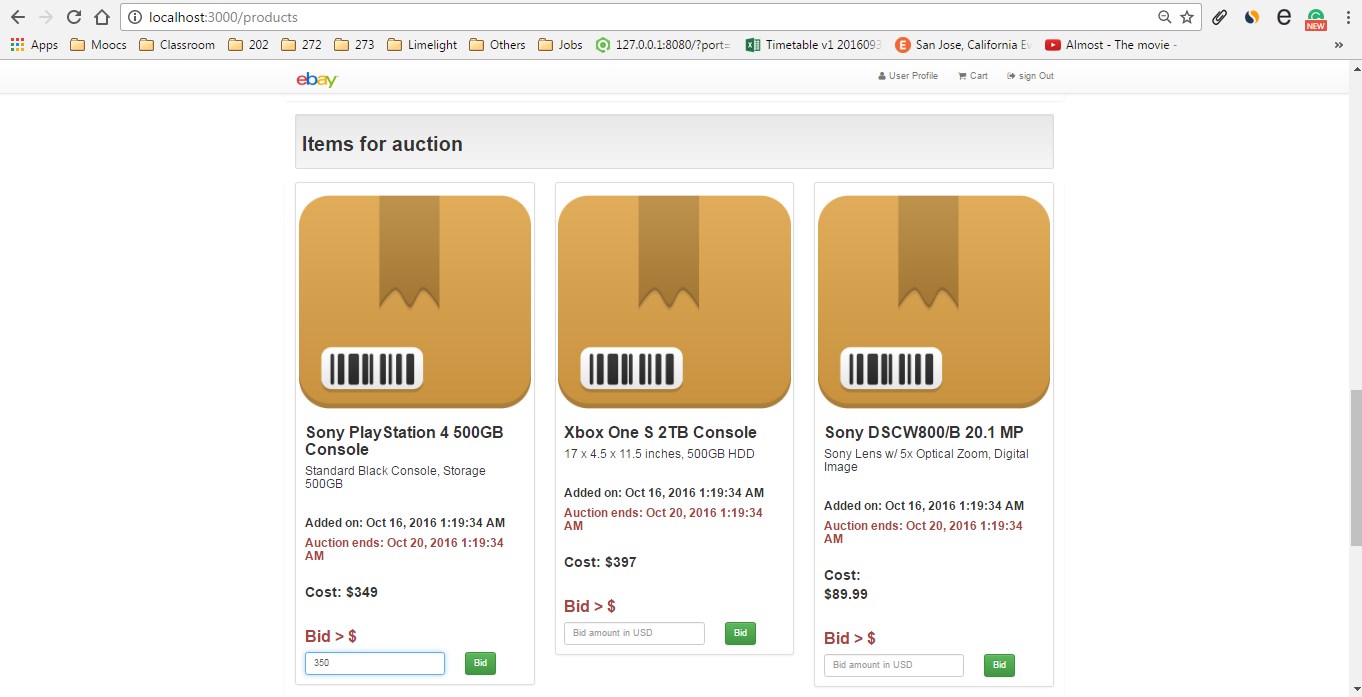
1. **Login Screen**

****

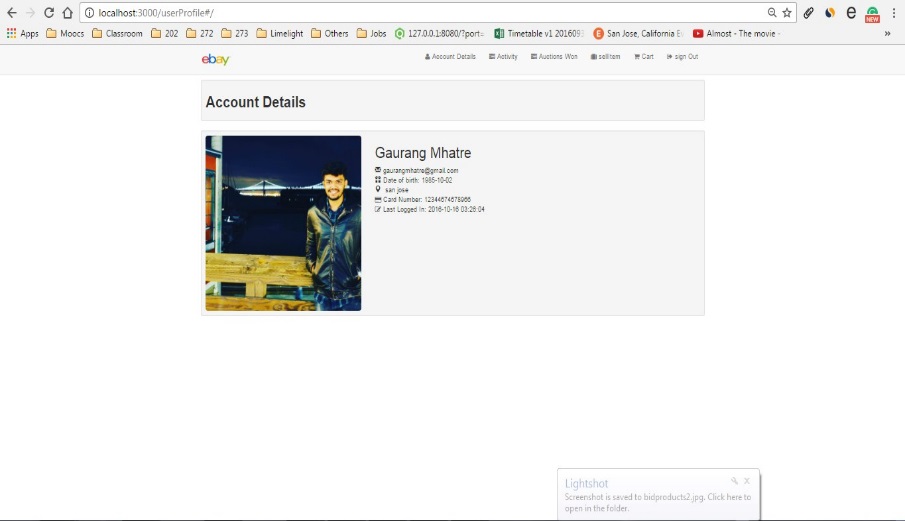
1. **Signup page**

****

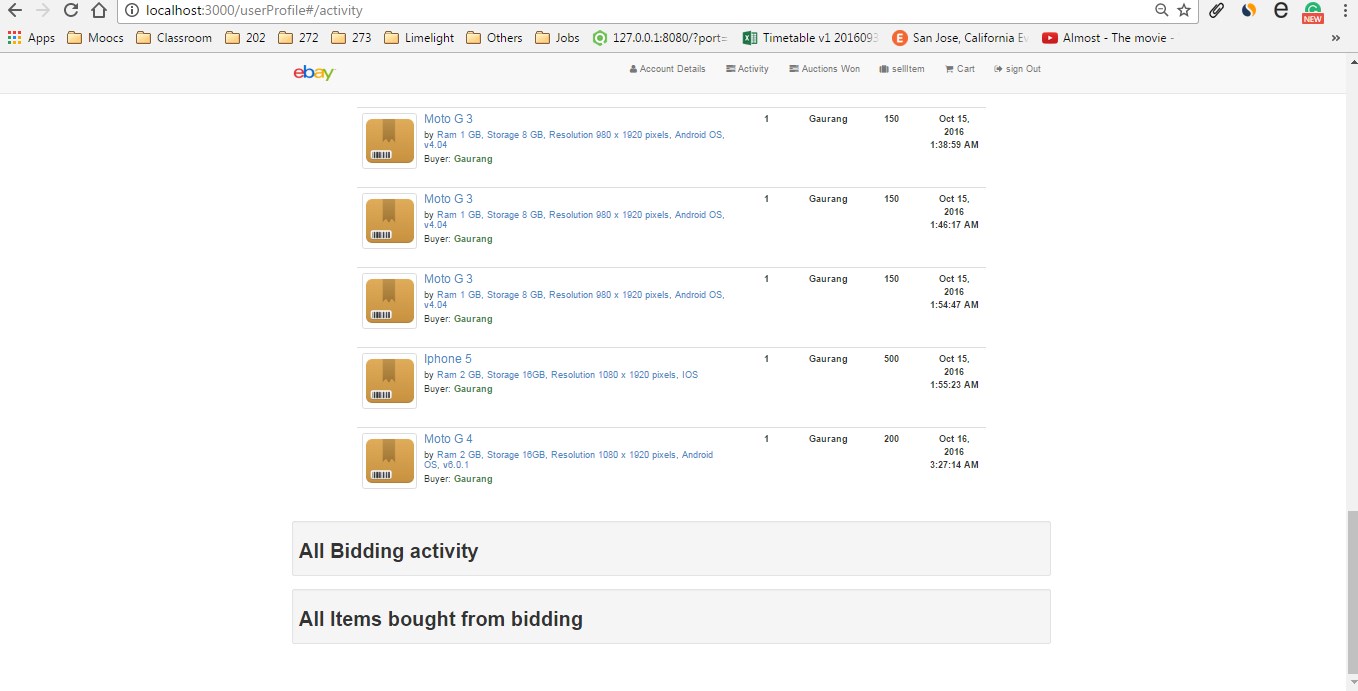
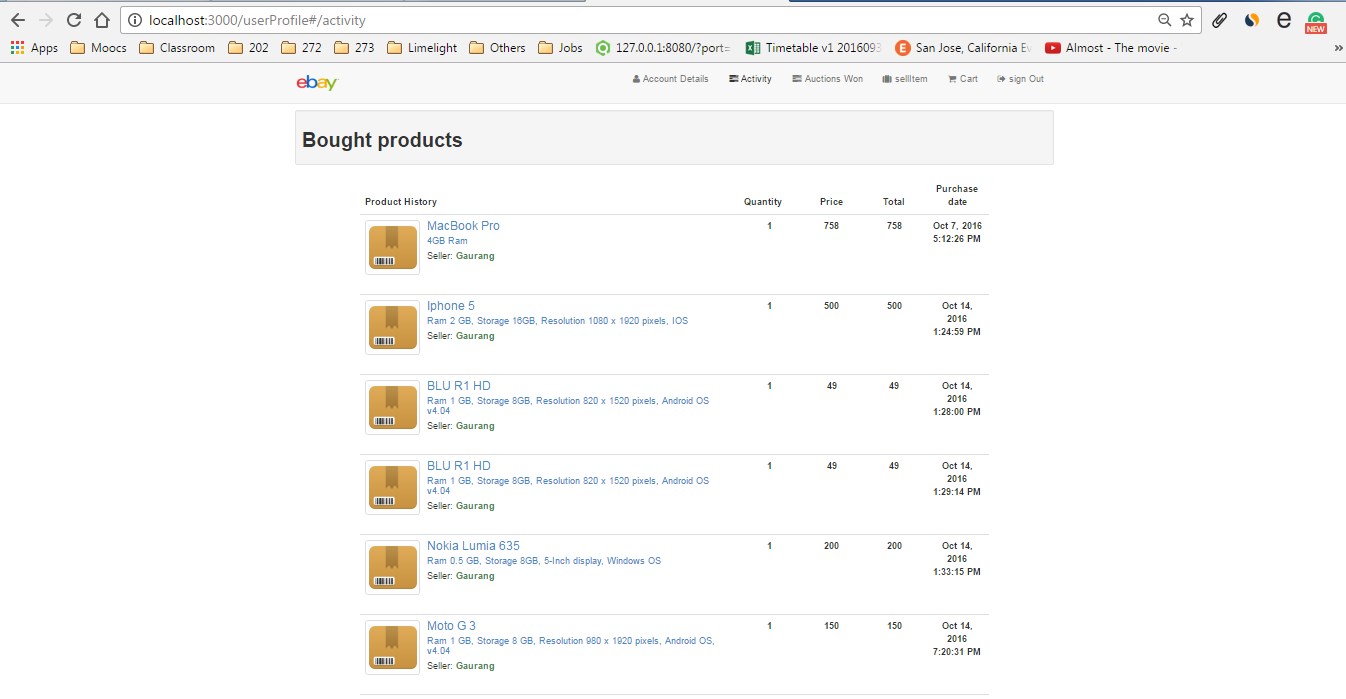
1. **Products page (direct buy/Bid)**

****

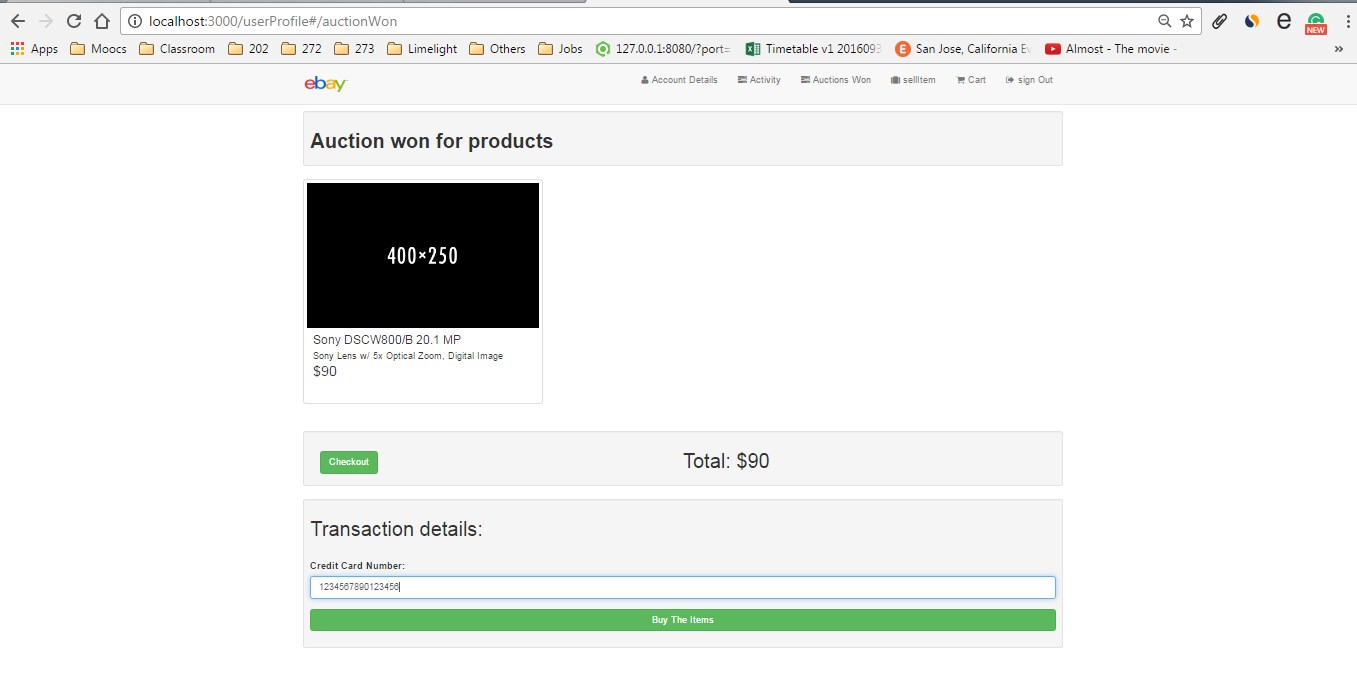
1. **User Account**

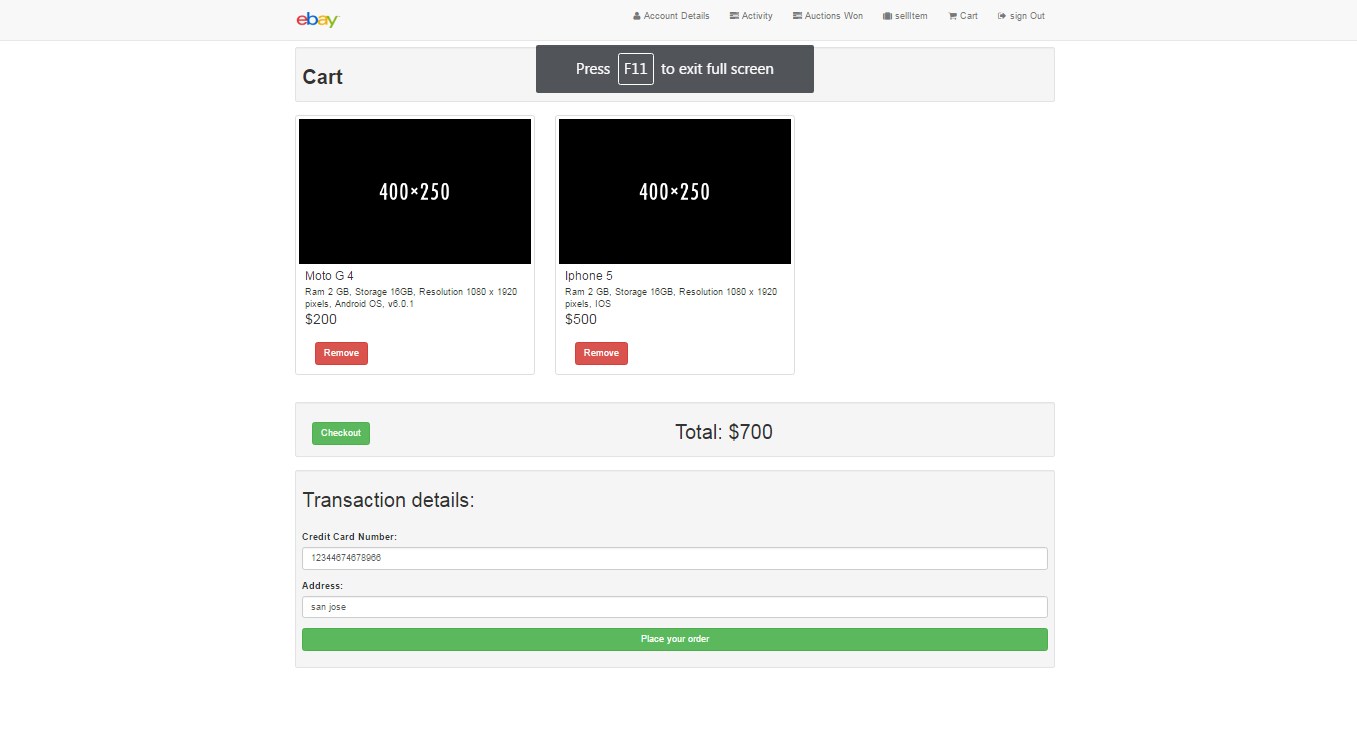
****

1. **User Activity**

****

1. **Auction won (Payment screen)**

****

1. **Cart Items and payment**

**Questions,**

**Q1: Explain the encryption algorithm used in your application. Mention different encryption algorithms available and the reason for your selection of the algorithm used.**

In my application for Encryption I have used **bCrypt.** It is basedon a has a password hashing function.

Other available algorithms from encryption are,

1. simple-encrypt
2. node-rsa

The reasons for choosing bCrypt were,

1. It is based on Eksblowfish setup.
2. It stores the extended key with salt making it harder to break.

**Q2: Compare the Results of the graphs with and without connection pooling. Explain the results in detail. Describe the algorithm of connection pooling used in your application**

**Answer:**

In Ebay, I have added methods getUserAccountDetailsWithoutConnetionPool and getUserAccountDetailsWithConnetionPool for testing connection pool.

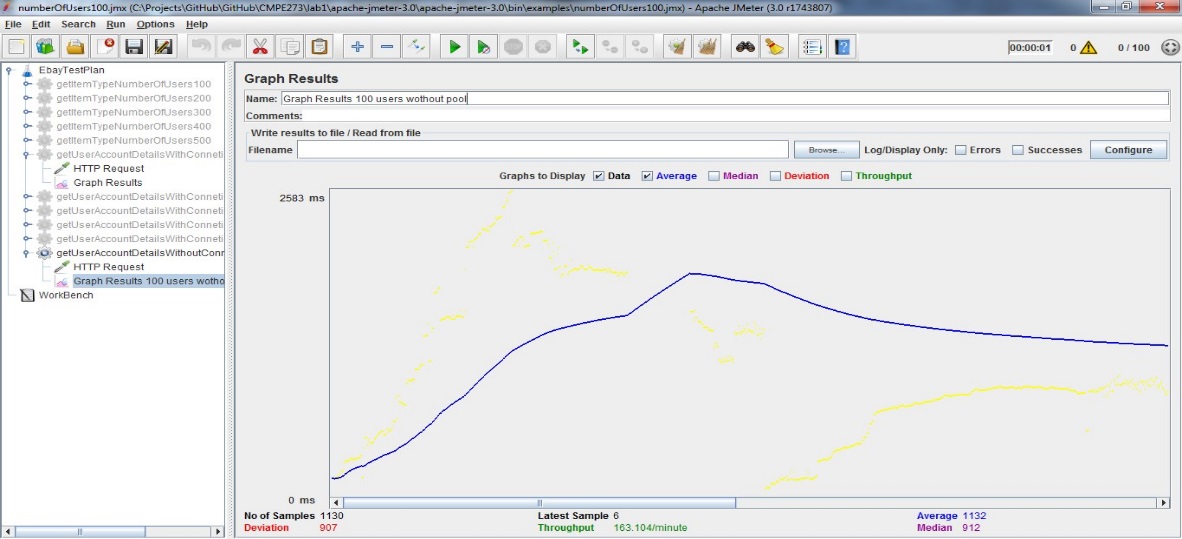
In getUserAccountDetailsWithoutConnetionPool method, we can use a pool of 500 connections which were initialized on the start of the server.

Algorithm of connection pool,

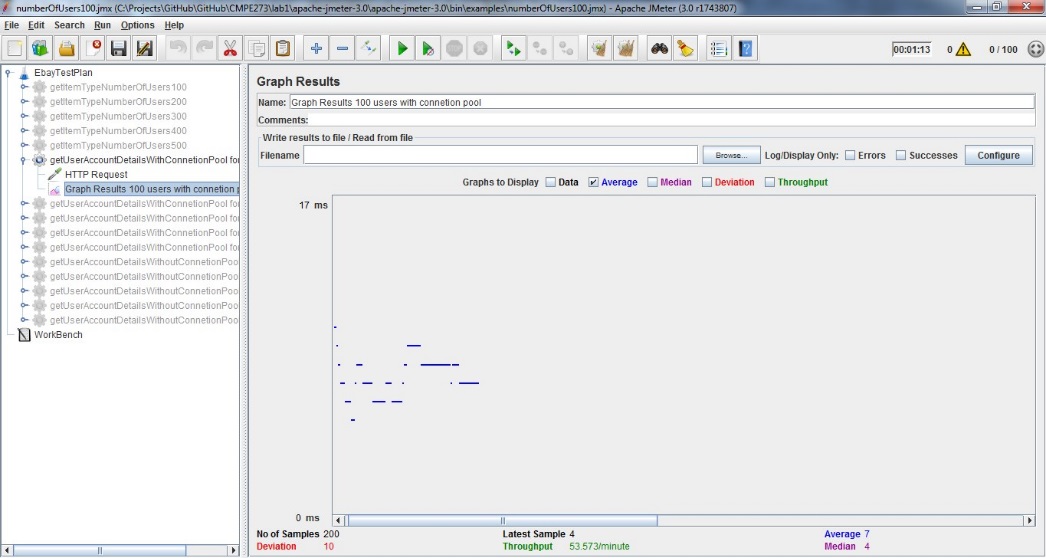
1. At the start, we will push the connections in stack with, createConnectionPool(numberOfConnections)
2. When a method requests for the connection, we check if we have connection available in stack. If yes, we pop the connection from the stack and pass it. If not, we will add the request in queue.
3. We also have a setInterval method, which checks for any available connection in stack and assign it to the waiting items in the queue.

**Observations:**

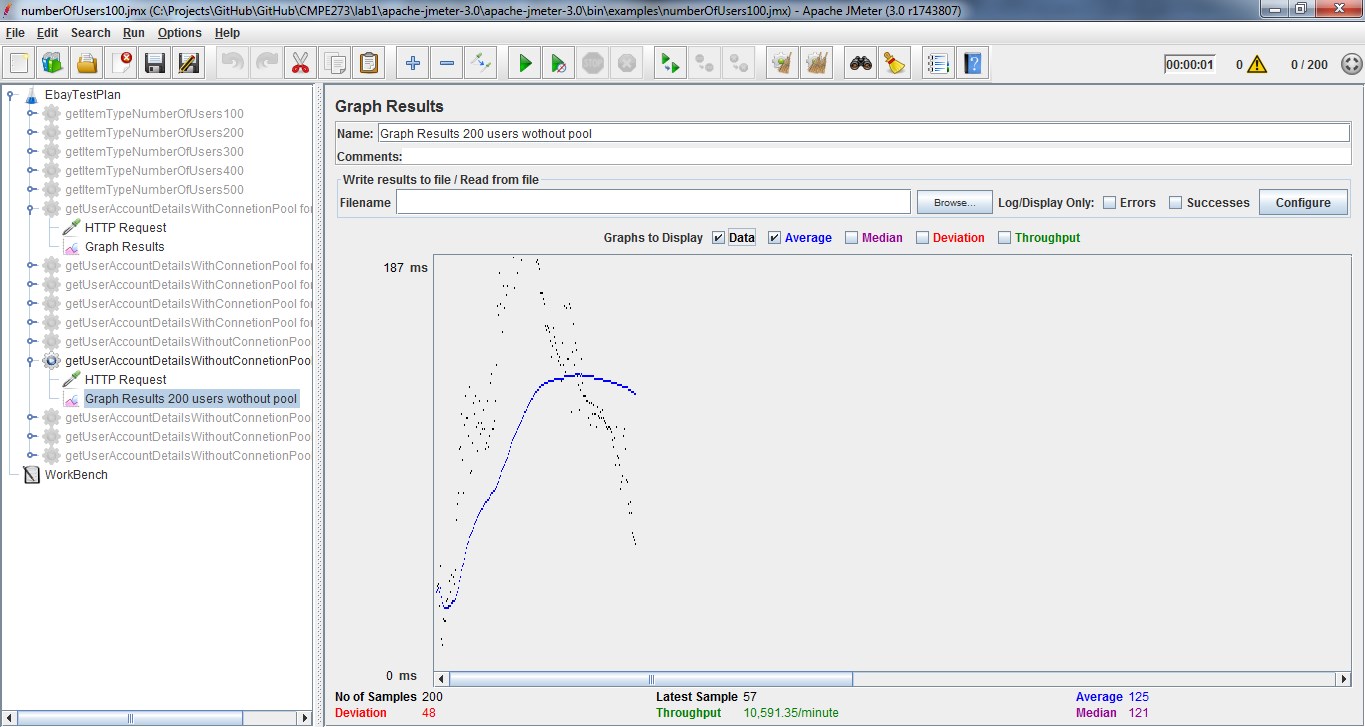
1. For 100 users
   1. Without connection pool (Avg:1132)



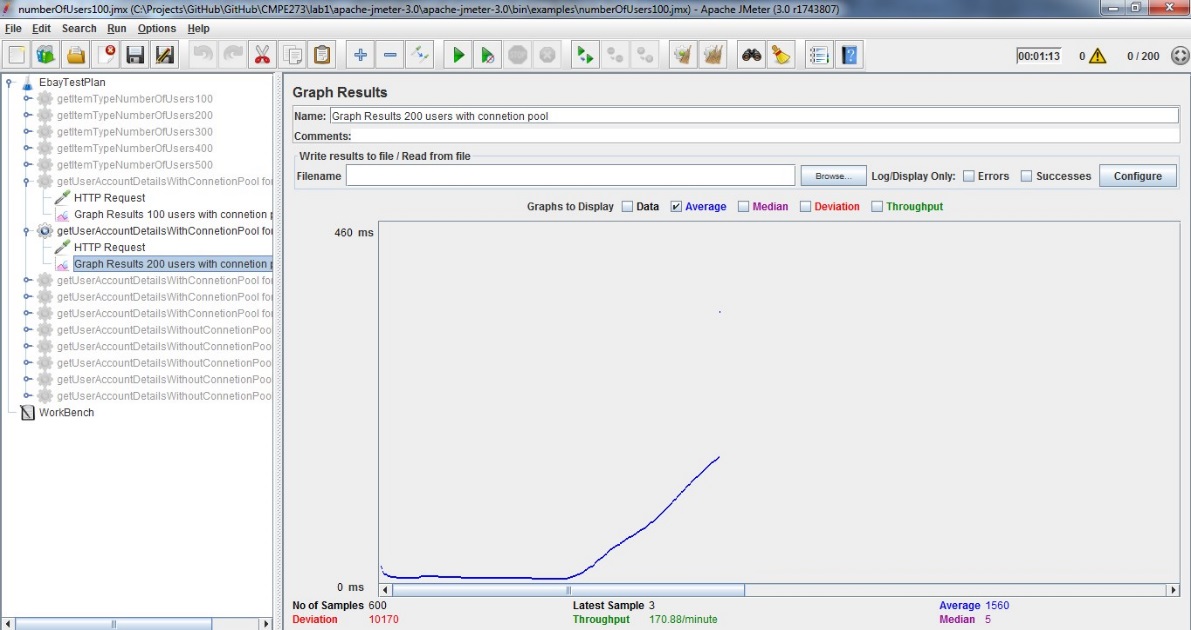
* 1. With Connection pool(Avg:7)



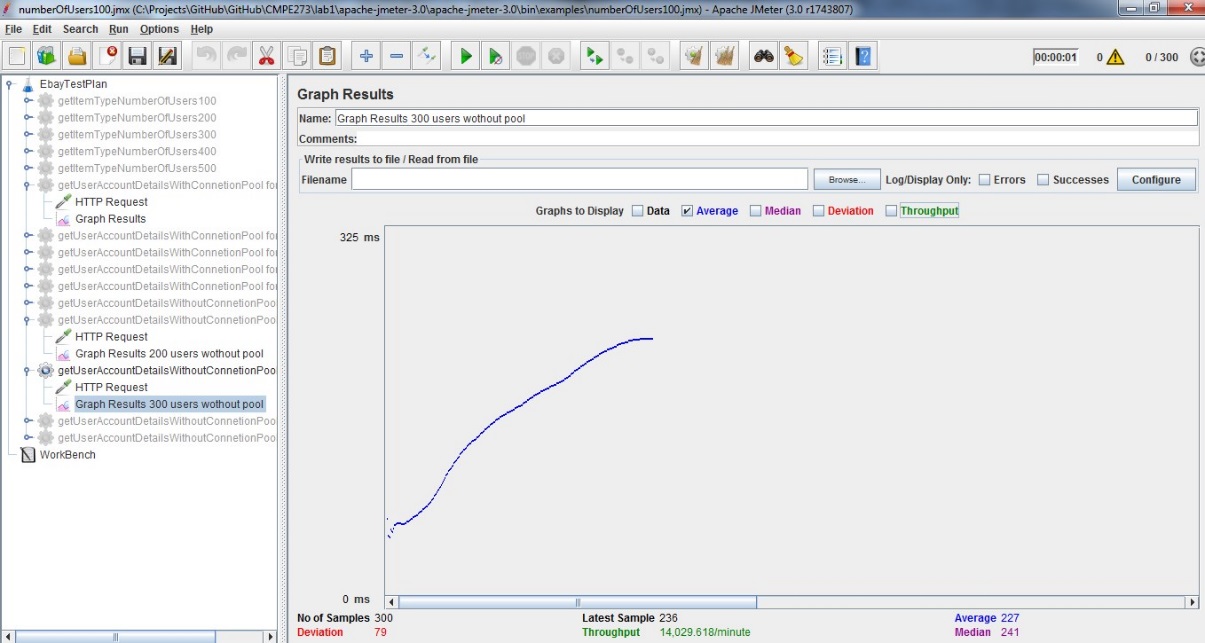
1. For 200 Users
   1. Without connection pool(Avg:125)



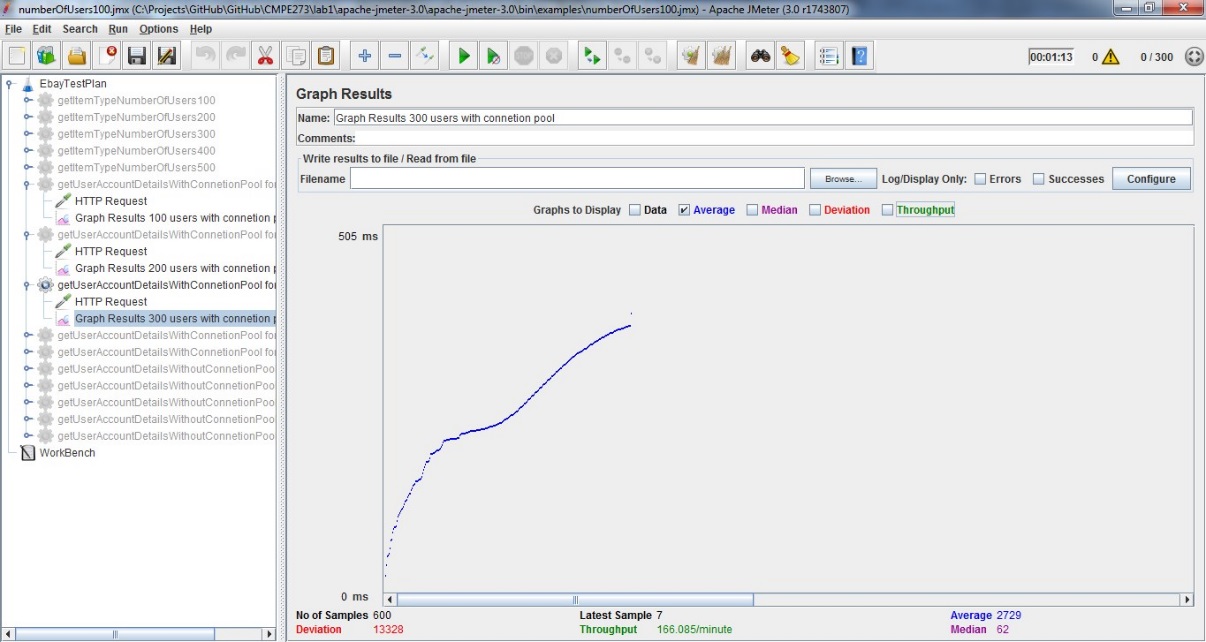
* 1. With connection pool (Avg:1560)



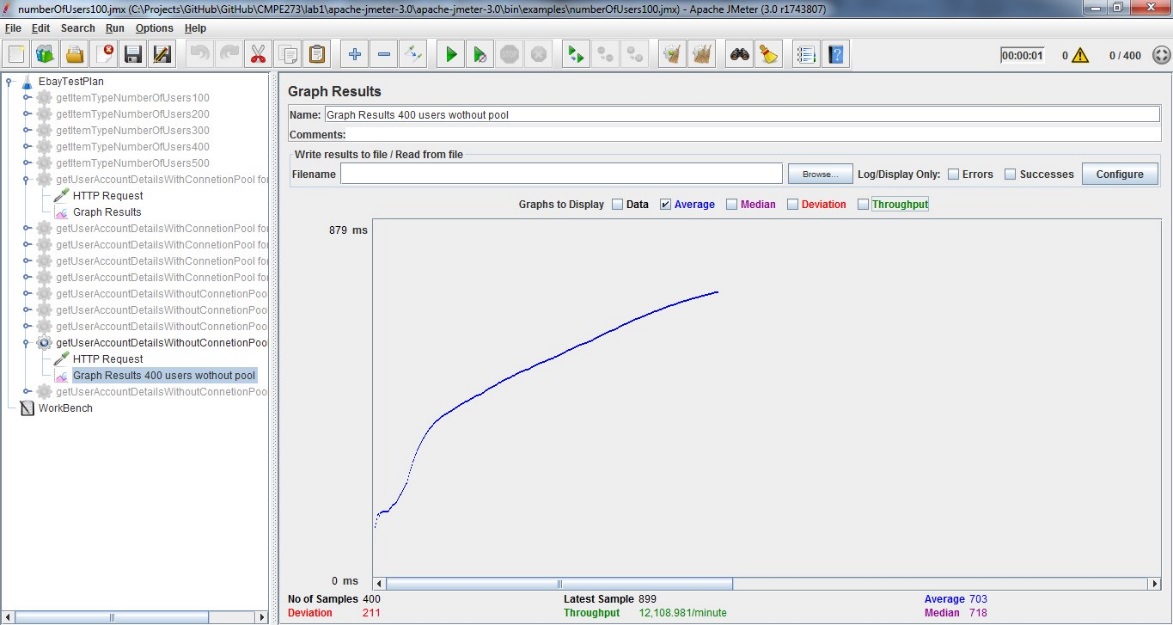
1. For 300 Users
   1. Without connection pool(Avg:227)



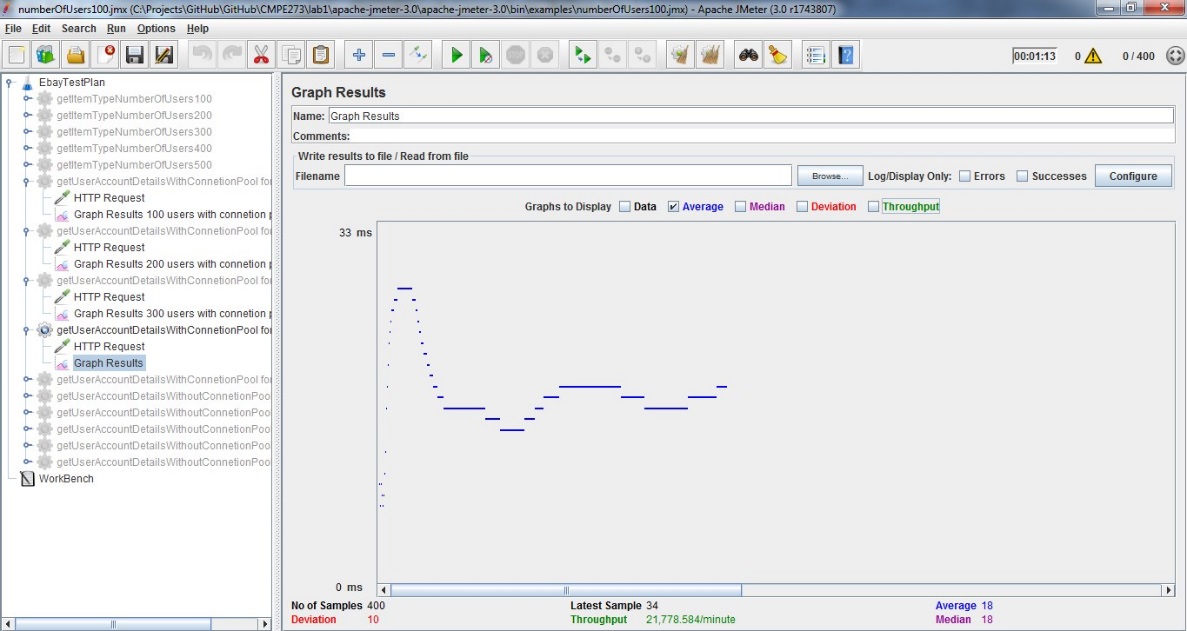
* 1. With connection pool(Avg:2729)



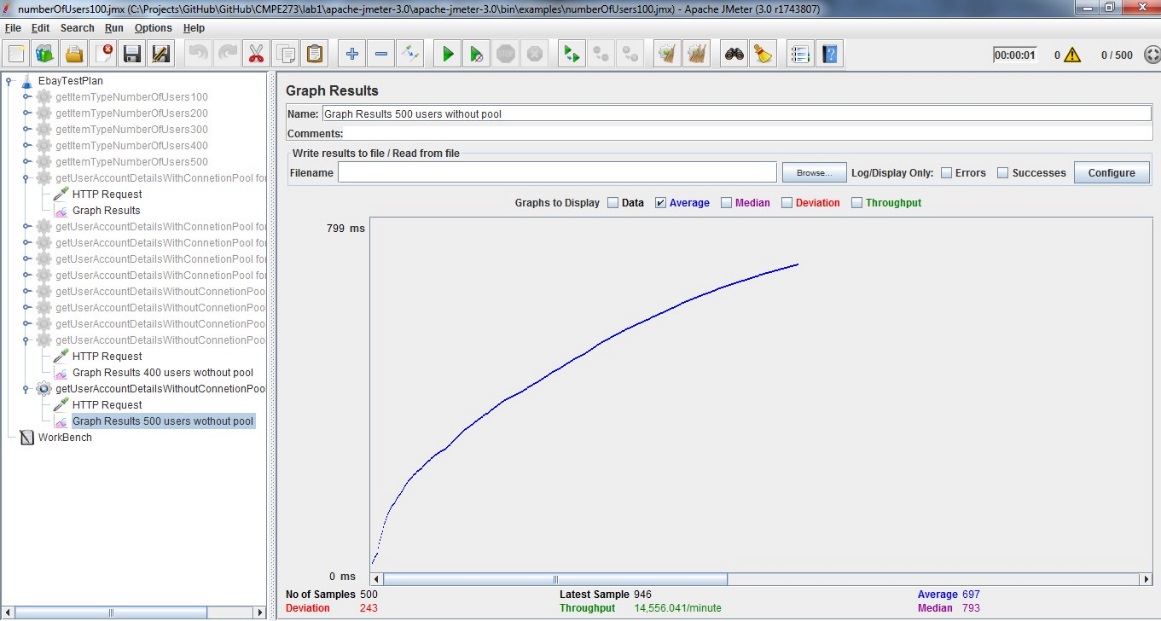
1. For 400 Users
   1. Without connection pool(Avg:703)



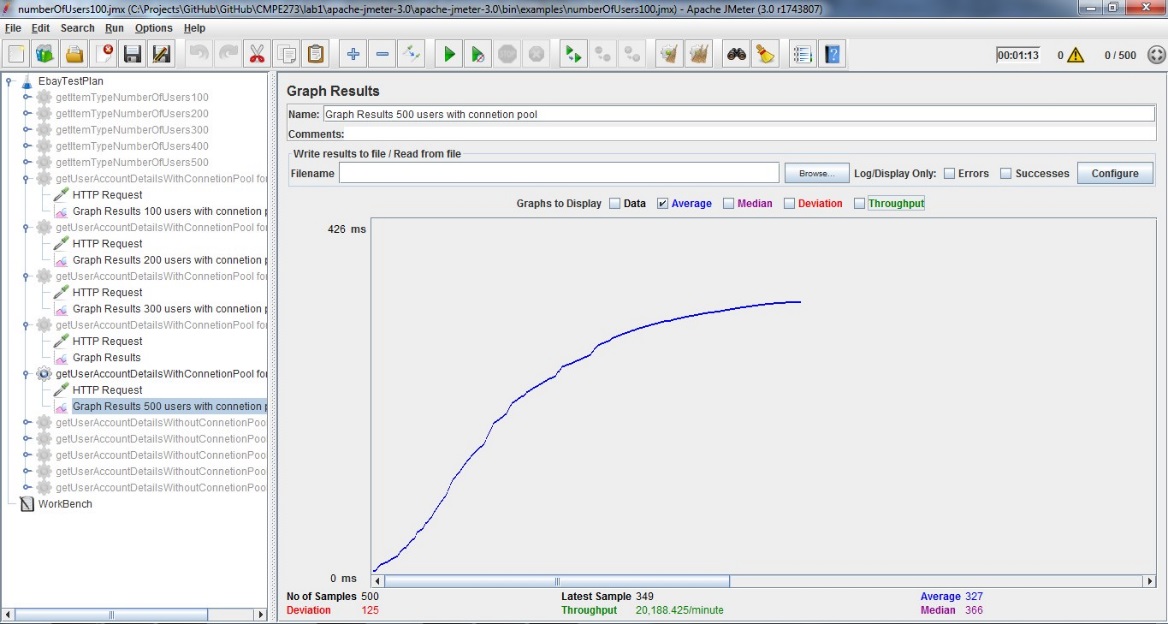
* 1. With connection pool(Avg:18)



1. For 500 Users
   1. Without connection pool(Avg:697)



* 1. With connection pool(Avg:327)



By above results, we can see that for 100,400 and 500 users, Connection pool has reduced average time per request by substantial amount.

**Q3: How would you implement Request Caching? Explain in detail. No need to implement a function – use pseudo code or detailed explanation**

Express provides mem-caching. For that we have to include memory-cache in node file.

The goal of server side cache is responding the same content for the same request independently of the client’s request. The client requests would hit the cache instead and the server would be able to send the response in a few milliseconds.