Lab 2-RabbitMQ and MongoDb

GitHub Repository: https://github.com/parthmupadhyay/cmpe273\_lab2

Credentials:

Email: [parthmukesh.upadhyay@sjsu.edu](mailto:parthmukesh.upadhyay@sjsu.edu)

Password: cmpe273\_temp

Introduction:

* Goal of this application is to demonstrate a “Simple Market Place”
* The eBay Application includes following functionalities:
* Sign Up: New users can register on ebay to buy or sell products
* Sign In: Users must sign in to ebay in order to view/buy/sell products
* Sign Out: User can sign out of ebay ,on sign out session is destroyed.
* Users can add advertisements for other users to purchase. User Advertisement includes product name,quantity,price ,seller information and product description.
* Users can buy products directly or bid for products which are available for bidding.

System Design:

* eBay Application is created using HTML5 , Bootstrap CSS, Angular JS and Node JS.
* Front-end of ebay is developed using HTML5 and bootstrap.
* Angular acts as a middleware controller that facilitates calling node JS API’s and rendering data returned from backend .
* The first page that a user can see on running eBay application is Sign In page.
* In order to view/buy or sell products user needs to sign in into application.
* The first page also has option to register a user which accepts following information:

First Name

Last Name

Email

Password

BirthDate

eBayHandle/User name

Address

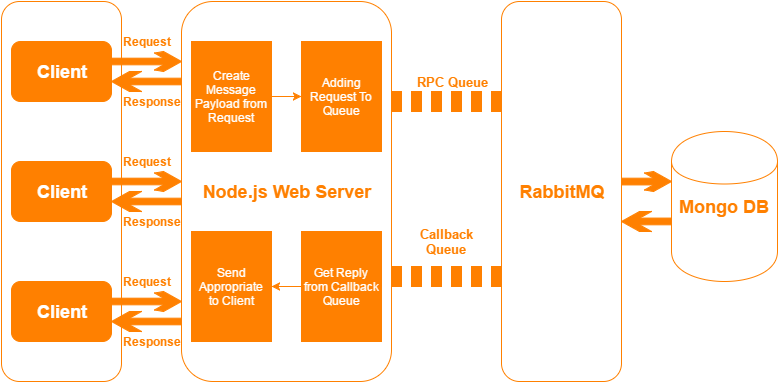
Contact info

* There are validations set for each input mentioned above to avoid any unexpected entries in user database.
* Once the user has signed in , user can see all the products available for buying/bidding other than products added by user itself.
* User can view its own profile which gives following information:
* Name, email, contact info ,address , Date of birth and Last Loggin time.
* User can add multiple products to cart. User can change product quantity in cart .User can delete products from cart.
* Checkout page contains VISA credit card validation and If the card is valid then it goes for processing the order.
* On successful order the product gets added to user’s order and the seller’s product quantity is updated.
* Server constantly checks for any running bids and checks whether bid end time has reached. Once bidEndTime matches current time or current time increases than bidEndtime , the bid product is added to highest bidder.
* Authentication is handled using local strategy provided by Passport JS.
* Encryption of passwords is done using bcrypt
* RabbitMQ is used in project to implement messaging queues.
* MongoDB NoSql database is used replacing Mysql.
* Sessions are stored within MongoDB.

RabbitMQ

RabbitMQ System Architecture:

* Below is the System Architecture showing RabbitMQ interaction with Client and Server in eBay.
* The Client makes request using HTML/AngularJS to node server.
* Node server adds requests to respective queues.
* Server.js running on the server picks up messages/requests from queue and executes them there by performing database interactions.
* After successfully handling request server.js returns results to node server which finally sends results to view.



Server.js

All the queues for handling requests are defined in server.js

Below is code snippet displaying some of the queue created.



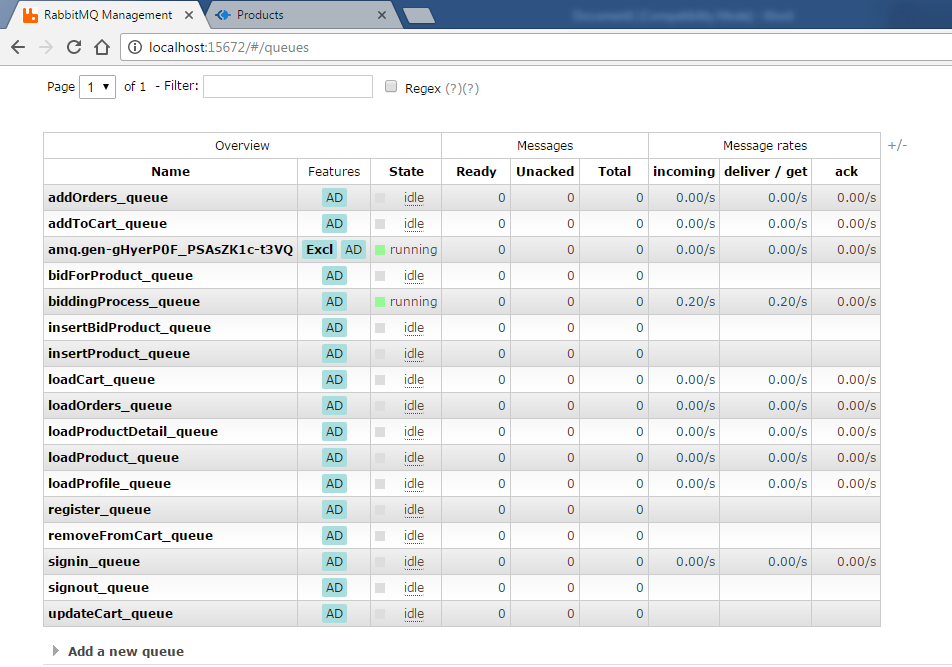


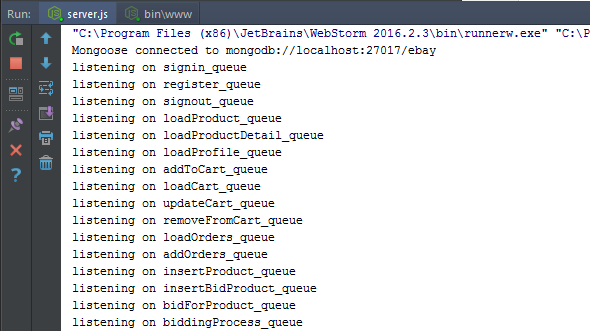


Rest of the queues are listed below in RabbitMQ queues section.

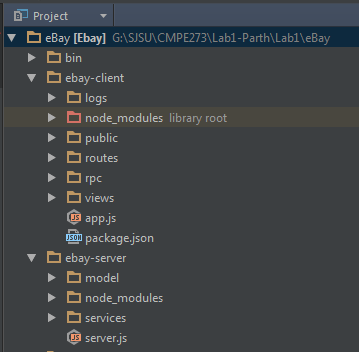
RabbitMQ queues:

Below is the list of RabbitMQ queues defined in ebay-Server





Project Structure:



The Project Structure is divided into eBay-client and eBay-server.

**eBay-Client:**

Views: Contains all *EJS* files required by client

Rpc : Configuration for RabbitMQ

Routes: Contains all REST API’s

Public\Stylesheets : CSS

Public\Angular : All angular files

Logs : Log files generated

**eBay-Server:**

model : mongoDB model

services : All Request handlers for queues.

Screen Captures of Application

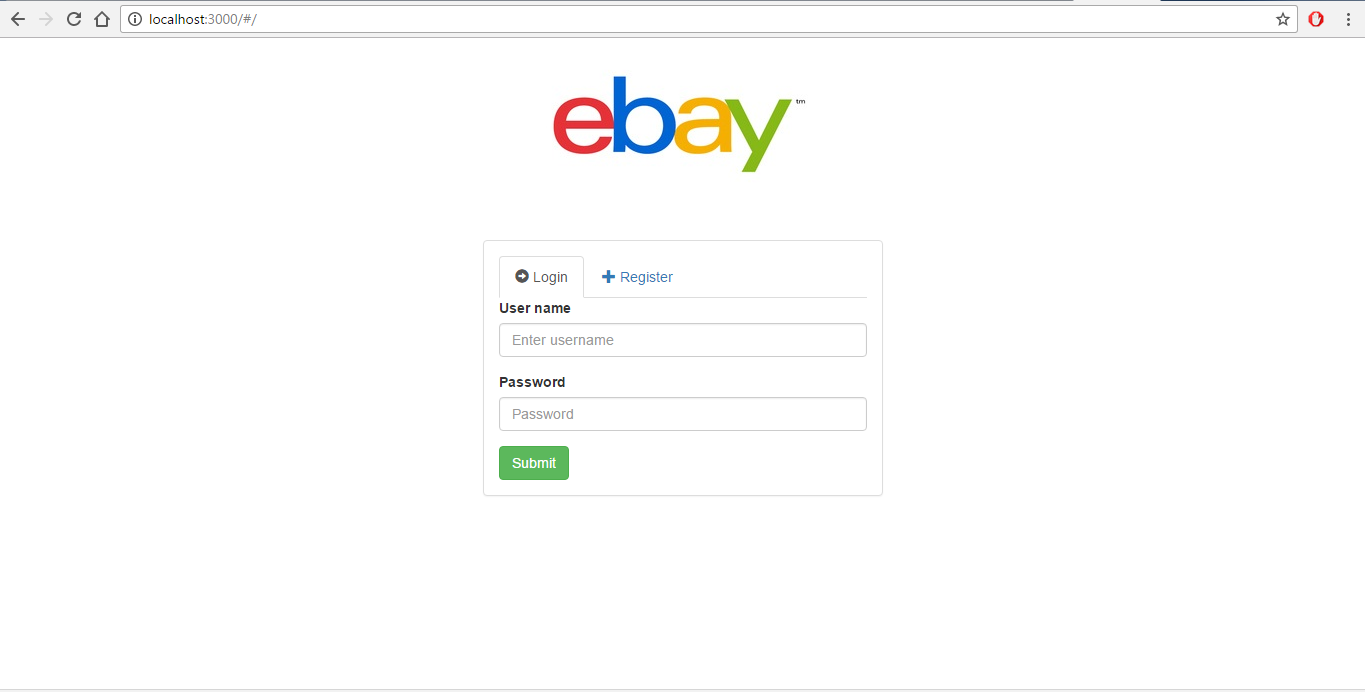
**Sign In :**

When we launch the application the first page loaded is Sign In page.

User can access all pages of the application only after successful login.

User must enter username/ebayHandle and password set while registering into application.

If user does not have a user account on application , it can create a user account by clicking on Register.



**SignIn Client Console:**

****

**SignIn Server Console:**



**Sign Up :**

If user does not have a user account on application , it can create a user account by clicking on Register.

Registration is 2 part process:

Part1 requires user to enter following information:

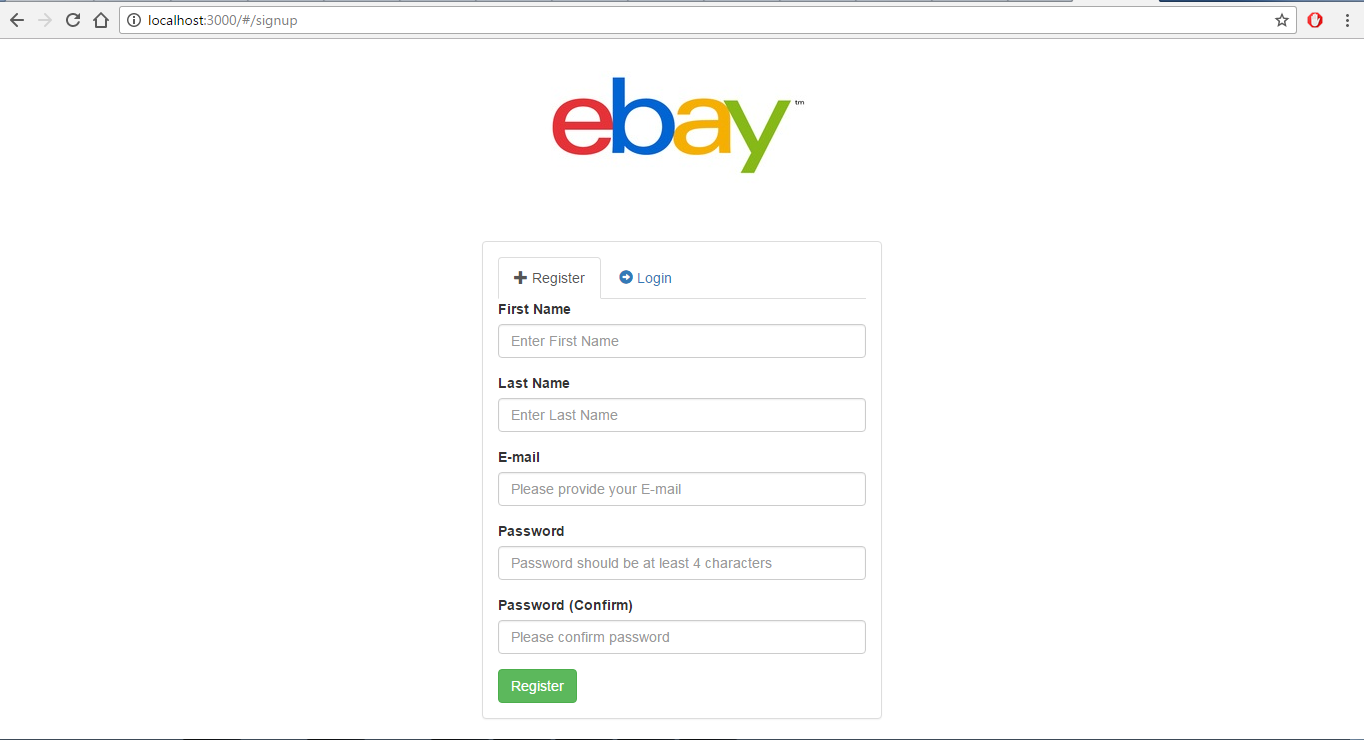
First Name

Last Name

Email : Unique

Password

Re-Enter Password

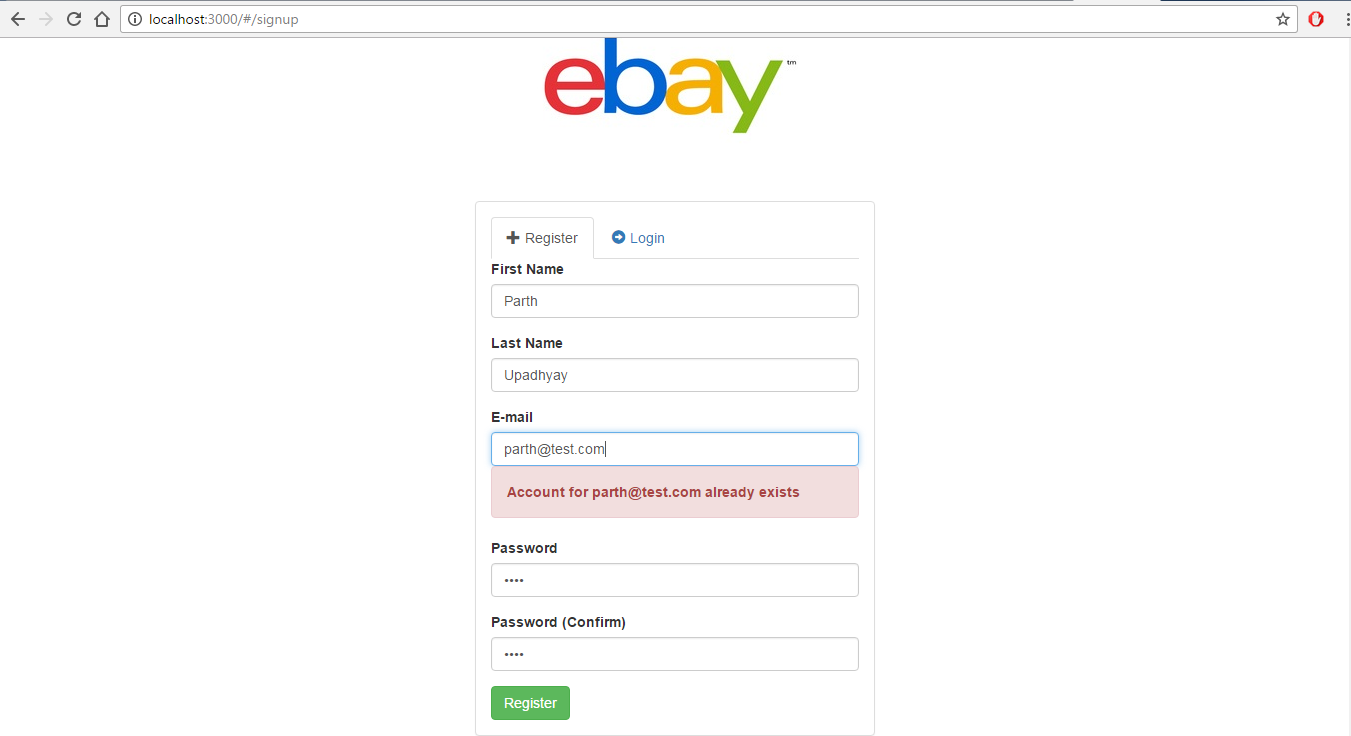


Email validation:

If user enters a already used email address in registration page then following message is displayed.

Angular ng-hide directive is used to display these message.

Angular ng-change event is used to call checkUsername API.

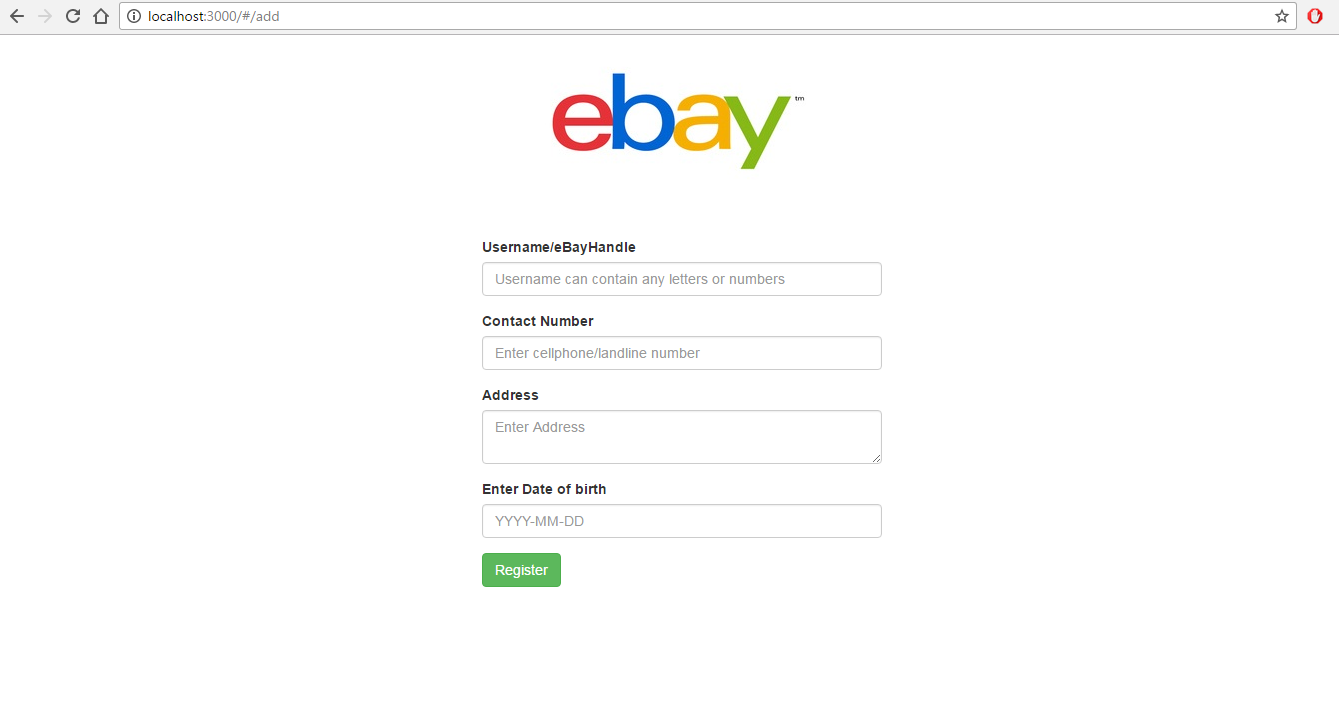


**Additional Sign Up Details:**

In this page additional details pertaining to new user account detail is asked.

User needs to choose a unique username/eBayHandle which will be used for user authentication.

Contact number , Address and Date of Birth are other details which are to be filled by user to complete registration process.

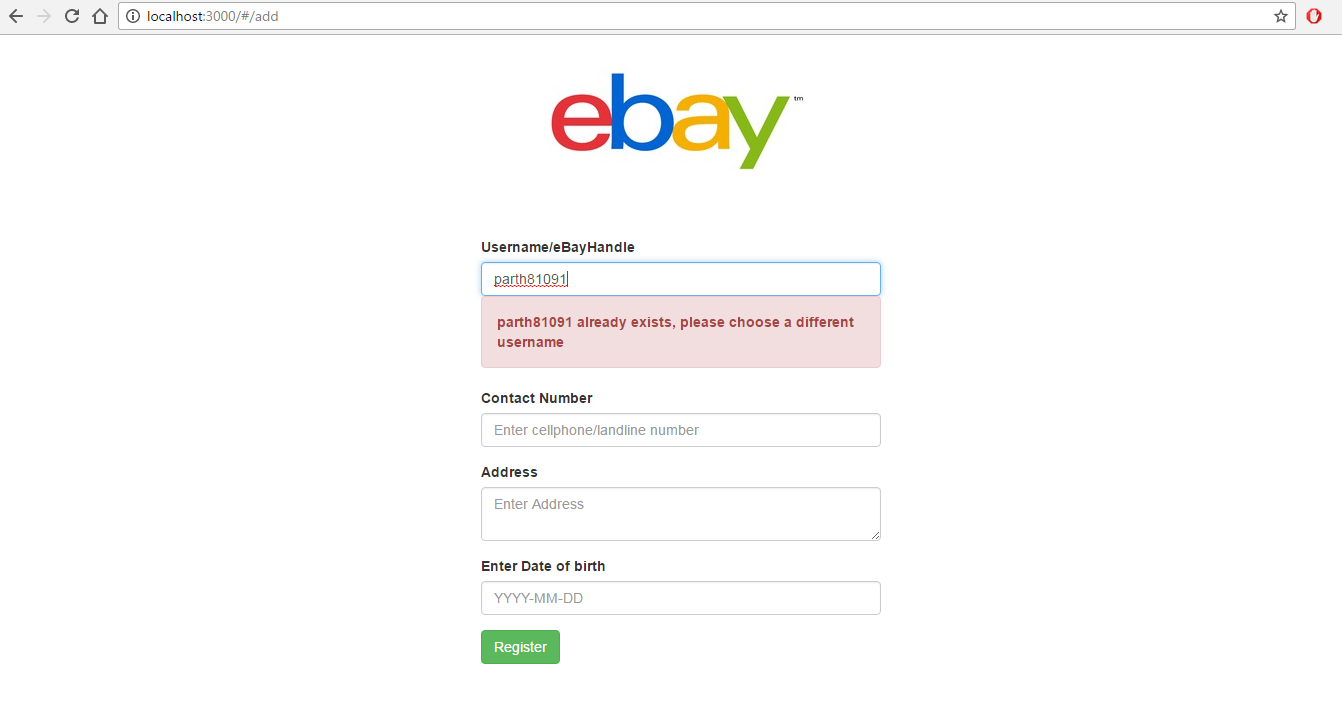


**Invalid Username:**

If user enters a existing username in registration page then following message is displayed.

Angular ng-hide directive is used to display these message.

Angular ng-change event is used to call checkUsername API.

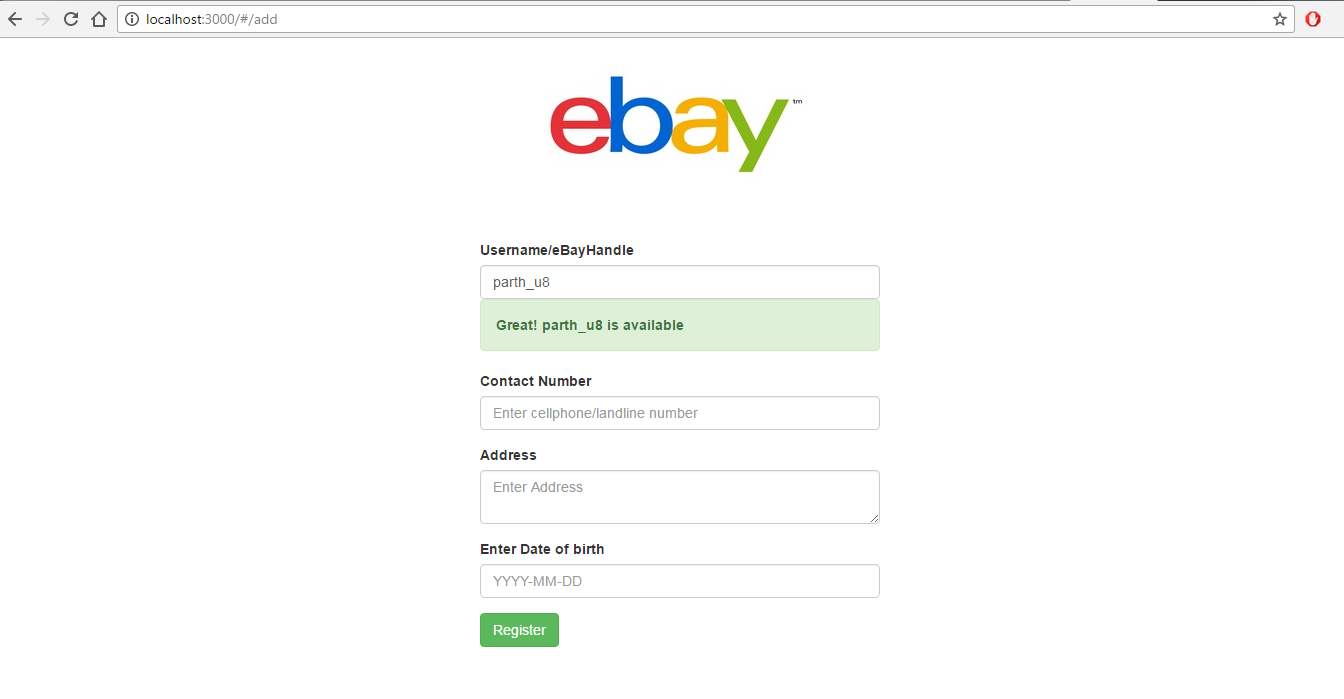


**Valid Username:**

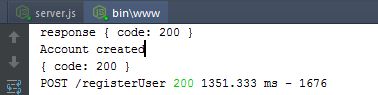
If user enters a valid unique username in registration page then following message is displayed.

Angular ng-hide directive is used to display these message.

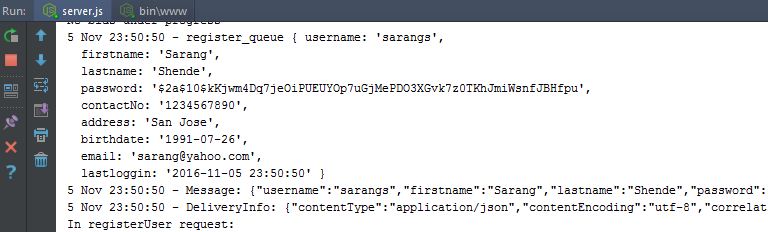
Angular ng-change event is used to call checkUsername API.



**Register Client Console:**

****

**Register Server Console:**

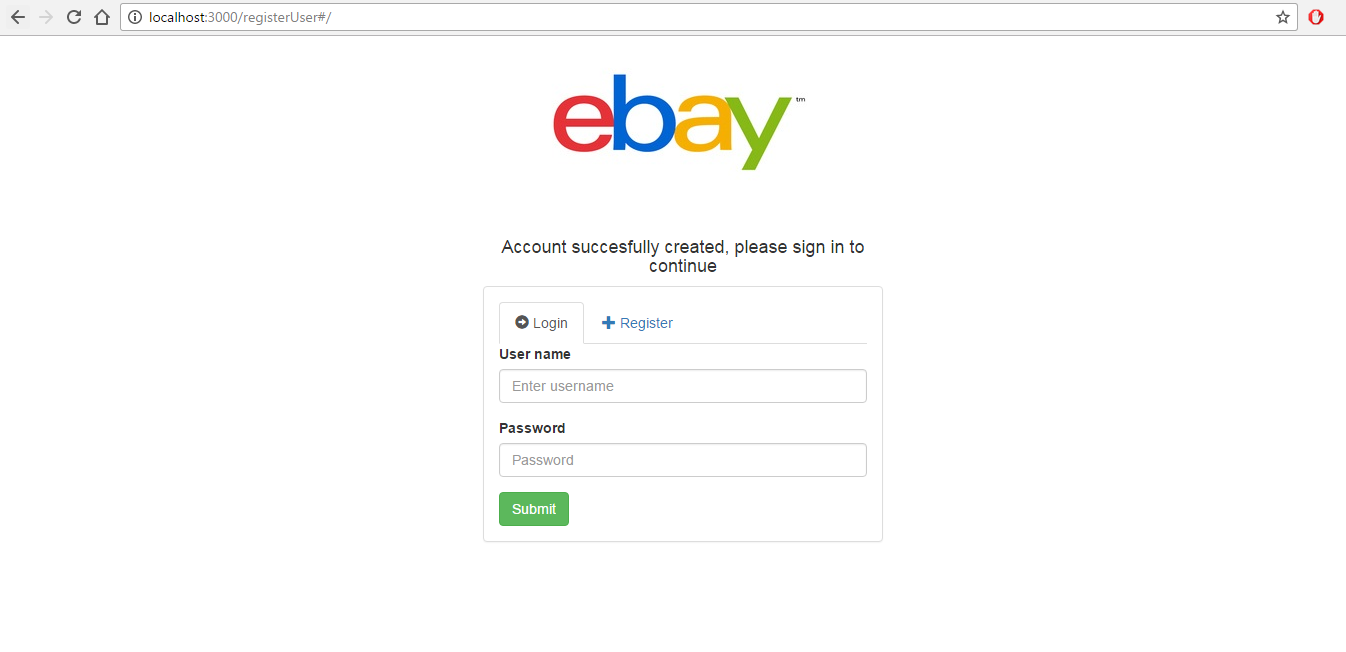
****

**On Succesful Registration:**

Once user completes registration, all the information entered by used is stored in Database.

User is then redirected to login page.

User can now login into application using the username/ebayHandle just created and password mentioned.



**After Successful sign in – Home page/Products Page**

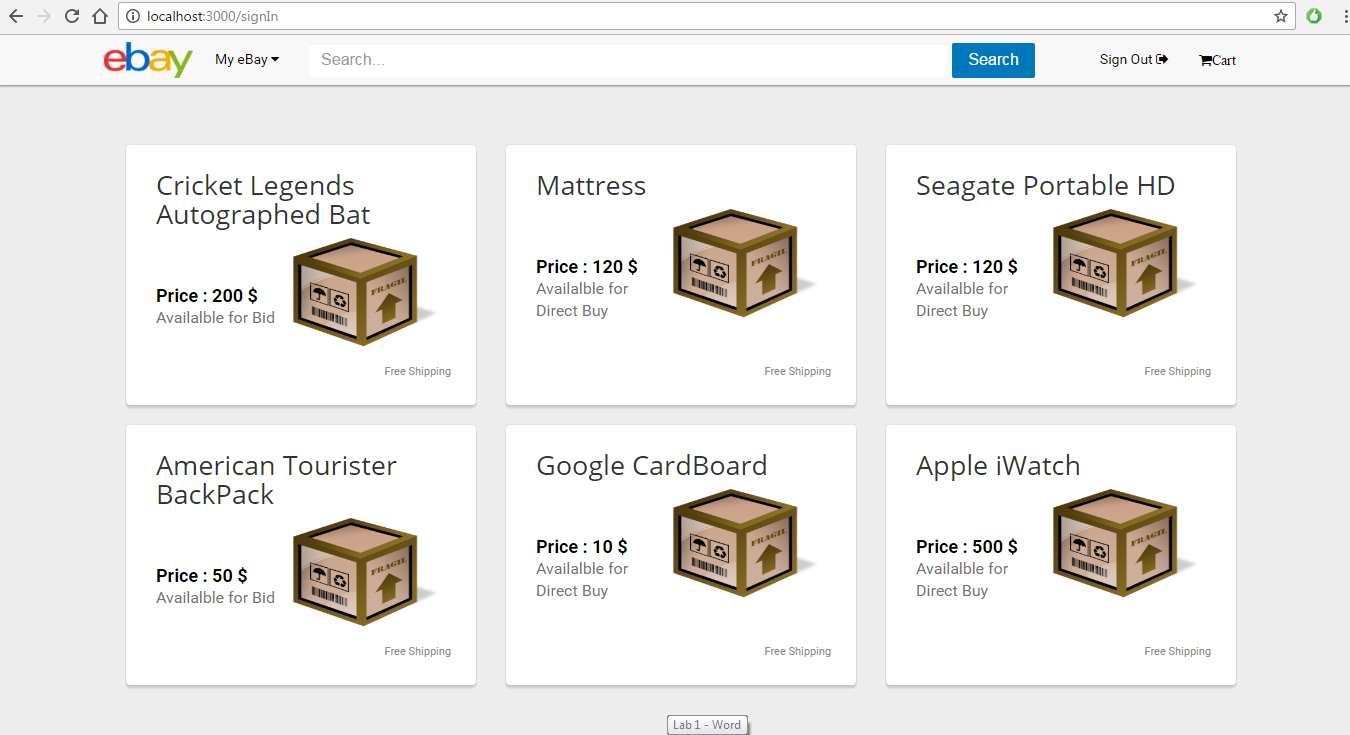
Once user enters correct username and password and clicks on login, ‘signin’ API is called.

It checks for authentication and if successful logs the user into application.

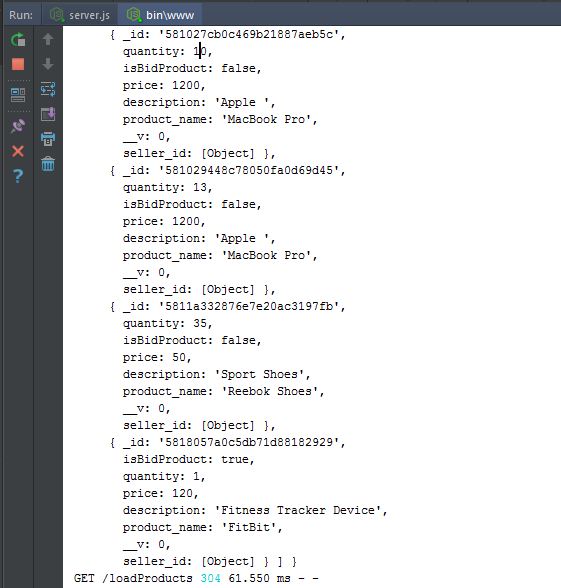
On successful sign user is redirected to Products page.

Products Page contains all the listed products for buying and bidding.

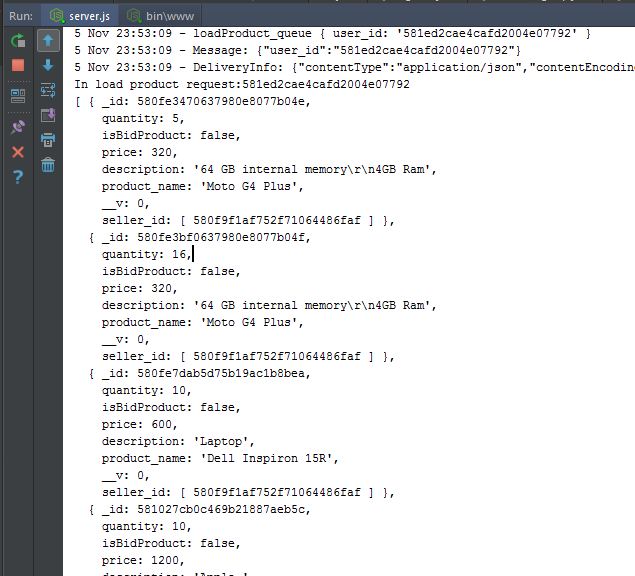
Each product shows its price and type of product: Direct Buy or Bid.



**LoadProducts Client Console:**



**LoadProducts Server Console:**



**Product Detail :**

When a user clicks on any product, application redirects to Product Detail page.

User can view all the information about the product like:

Name

Description

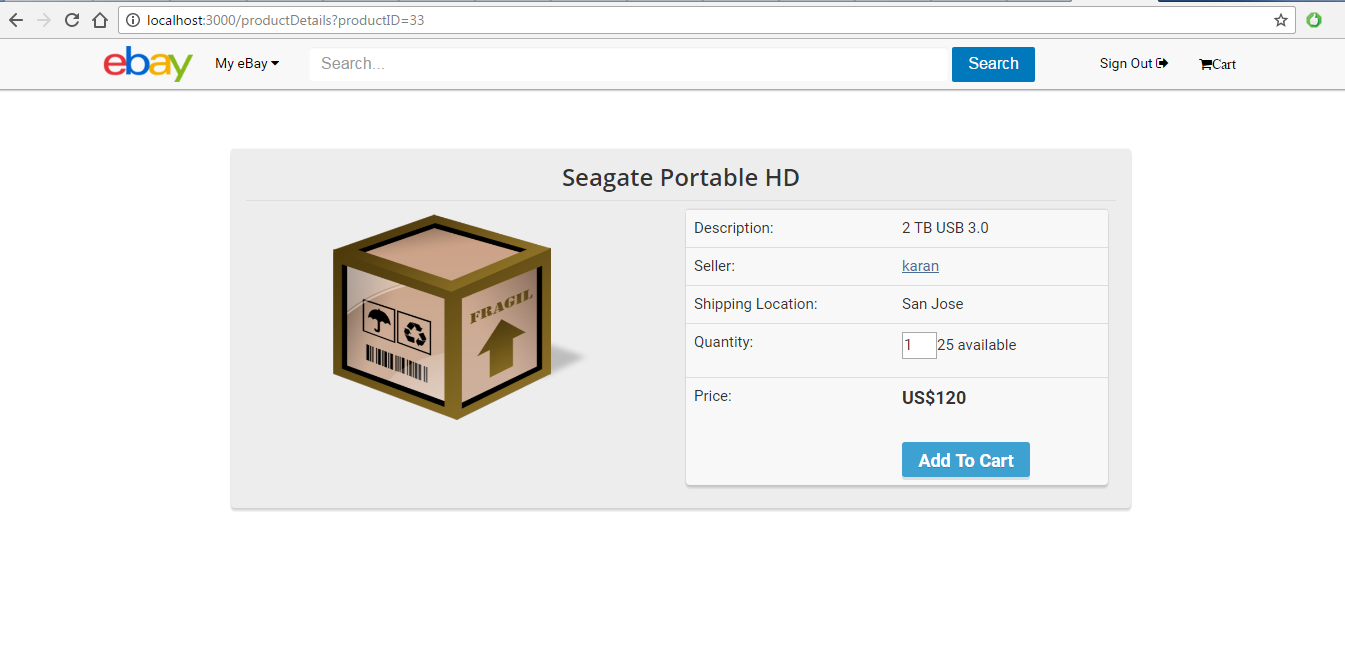
Seller

Shipping Location

Quantity

Price

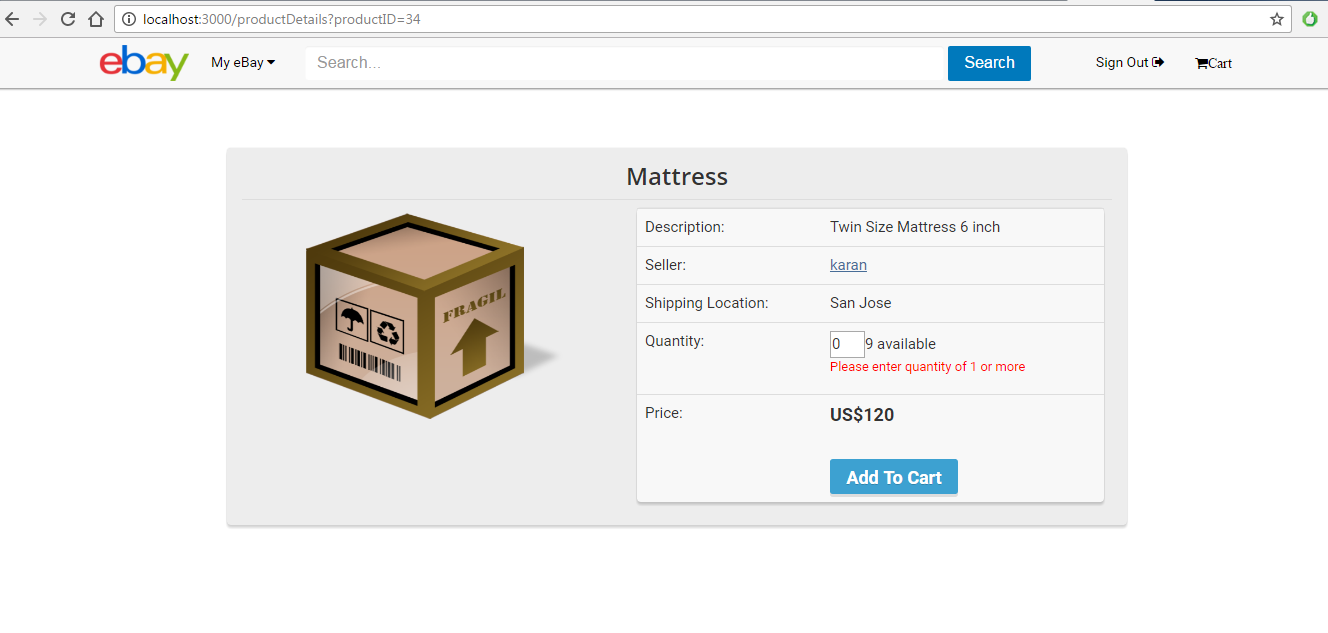
User can change quantity and add product to cart for purchase.



**Product Detail Quantity Validation:**

While changing the quantity for purchase user cannot enter 0 as quantity

If user enter 0 in Quantity Box , following message is displayed.



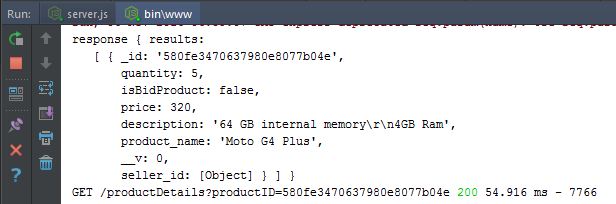
**Product Detail Quantity Validation:**

While changing the quantity for purchase user cannot enter quantity greater than available quantity.

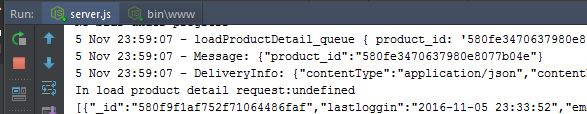
If user tries to enter quantity greater than available quantity, following message is shown to user.



**Product Detail Client Console:**

****

**Product Detail Server Console:**

****

**Seller Information from Product Detail :**

Product Detail contains a hyperlink to Seller information.

When a user clicks on seller hyperlink , user is redirected to seller’s profile page

Which includes following information:

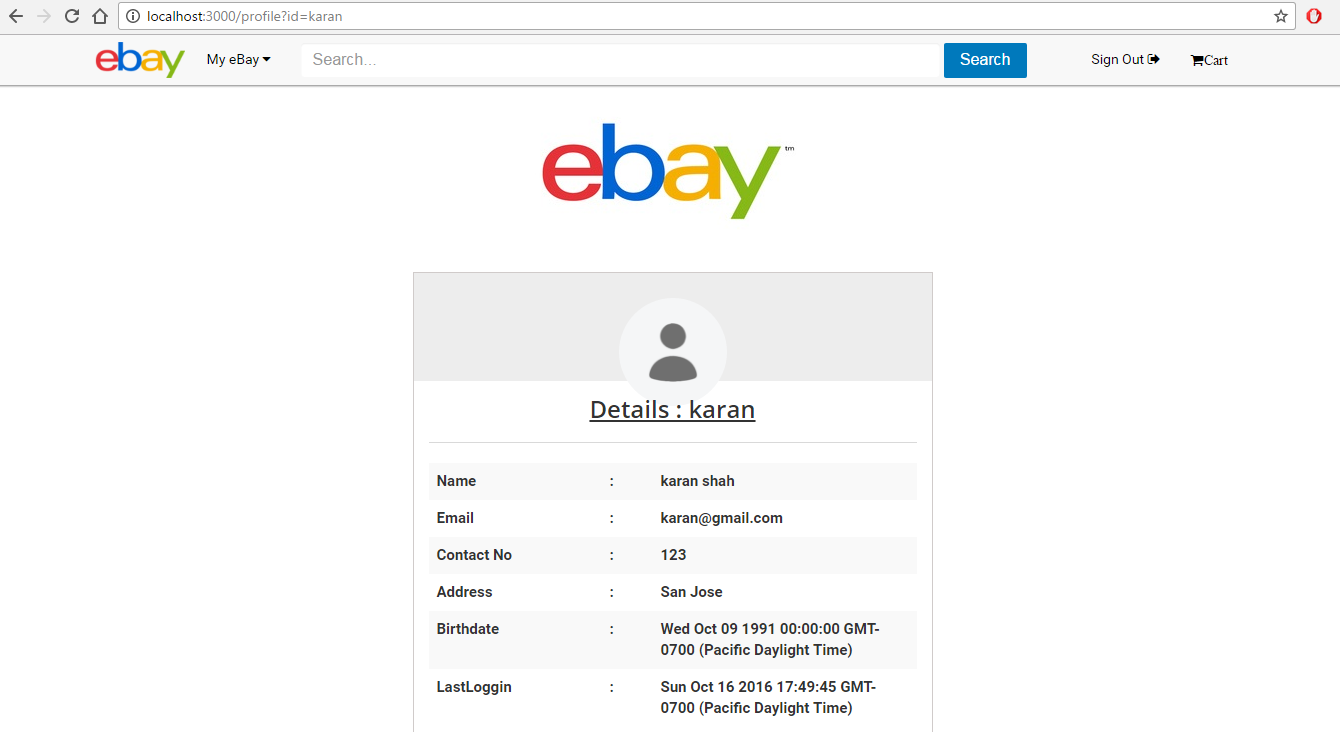
Name

Email

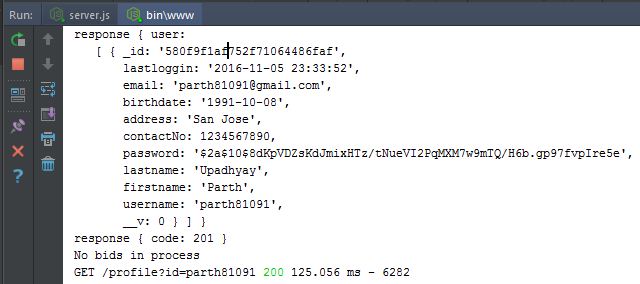
Contact No

Address/Shipping Location

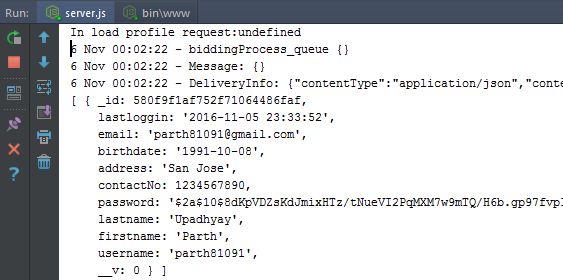
Birthdate



**Profile Client Console:**

****

**Profile Server Console:**



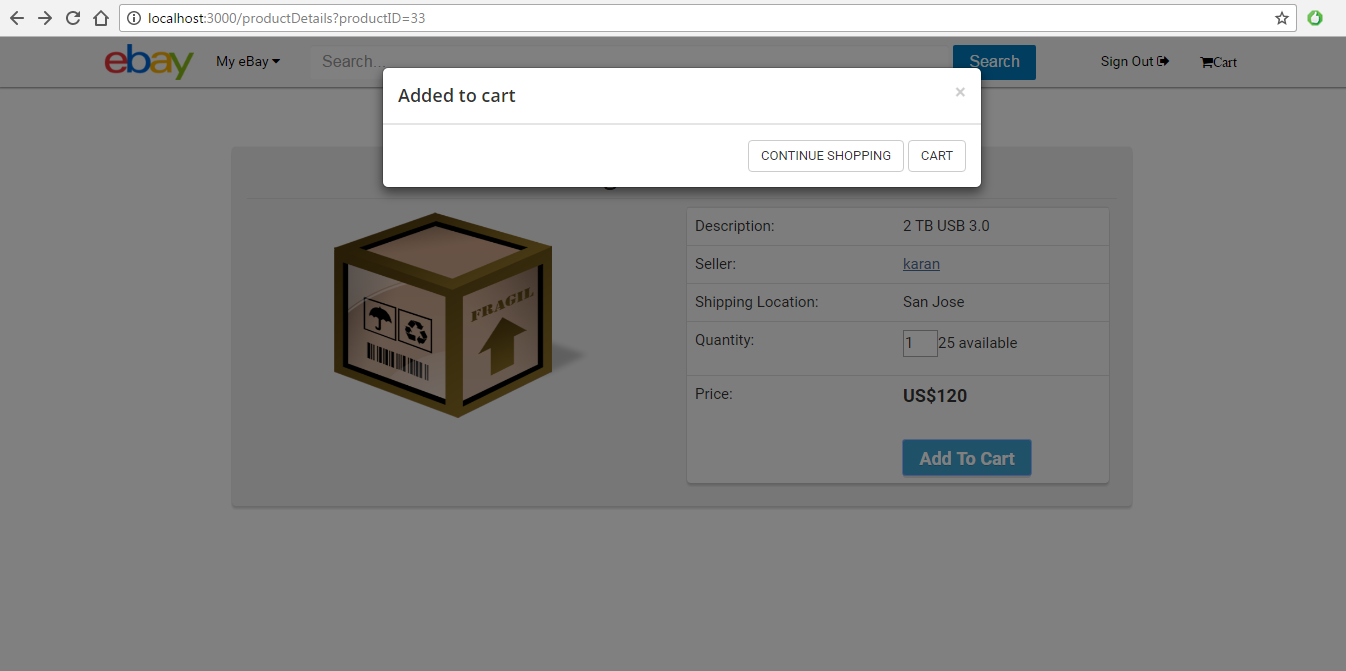
**On Add to Cart Click:**

When user clicks on Add To Cart , new entry is created in Cart collection with specified quantity.

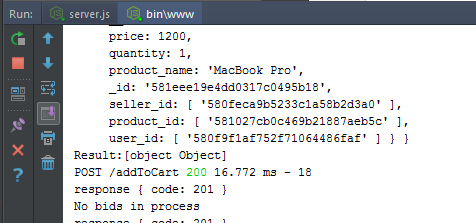
On successful cart entry a modal gets opened with options to continue shopping or open cart.

If user clicks on continue shopping then it is redirected to products page

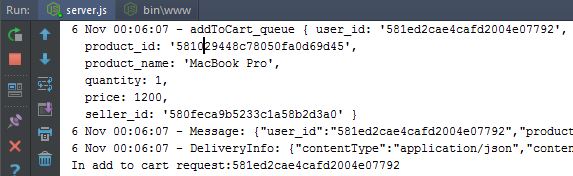
And if user clicks on Cart then it is redirected to User Cart.



AddToCart Client Console:



AddToCart Server Console:



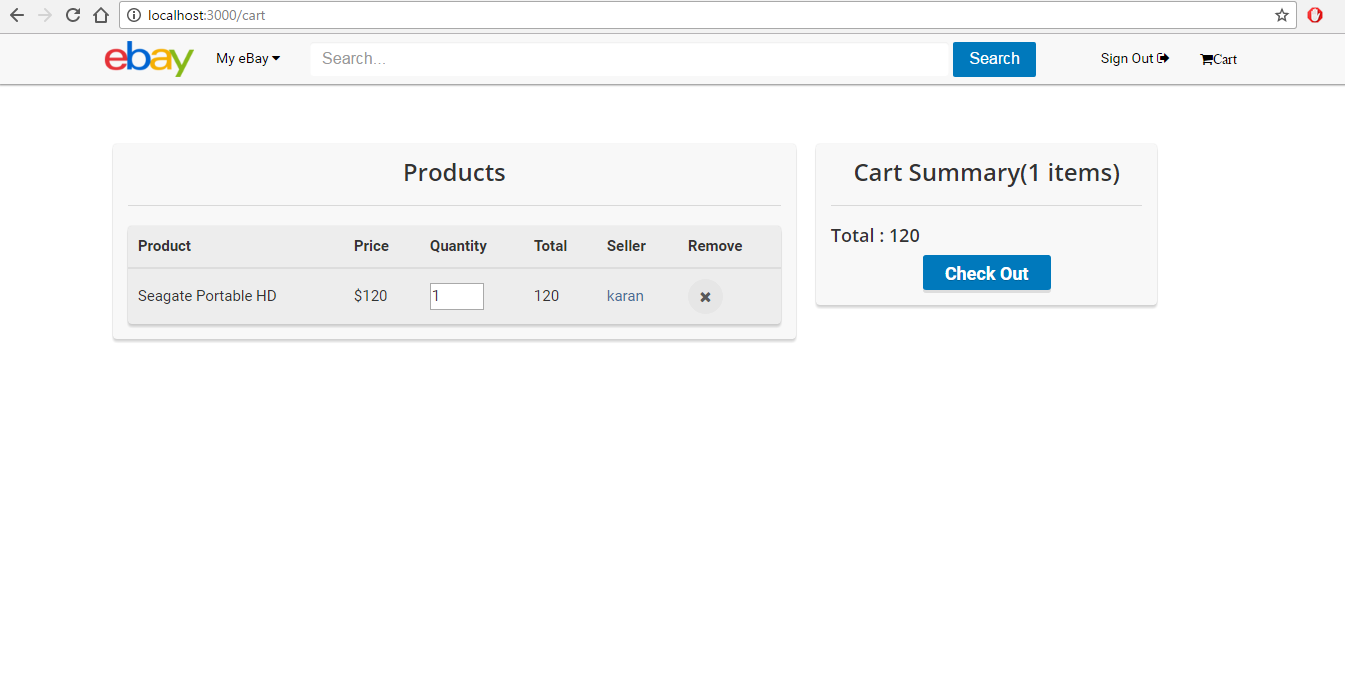
**Cart:**

When user clicks on Cart from header navigation it is redirected to User Cart Page.

It loads all the products added by user to the cart with their respective quantities.

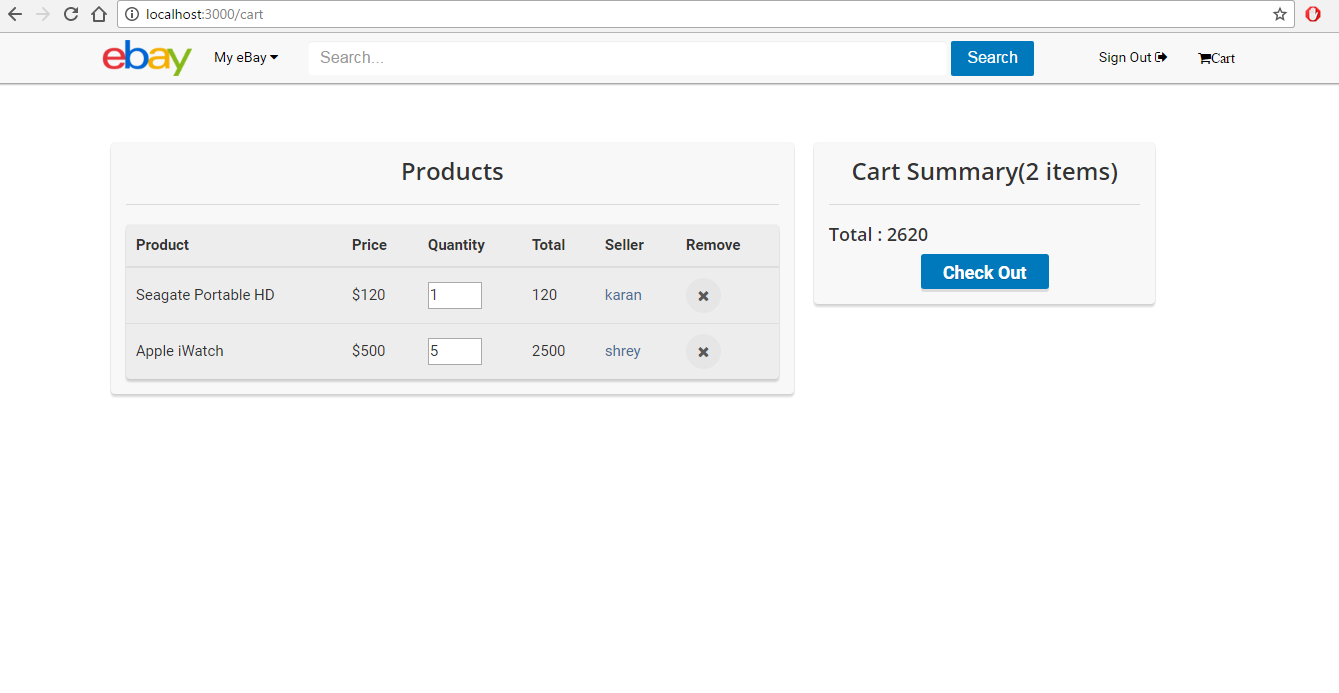
User can remove all the products from Cart

User can update quantities of Product in cart.

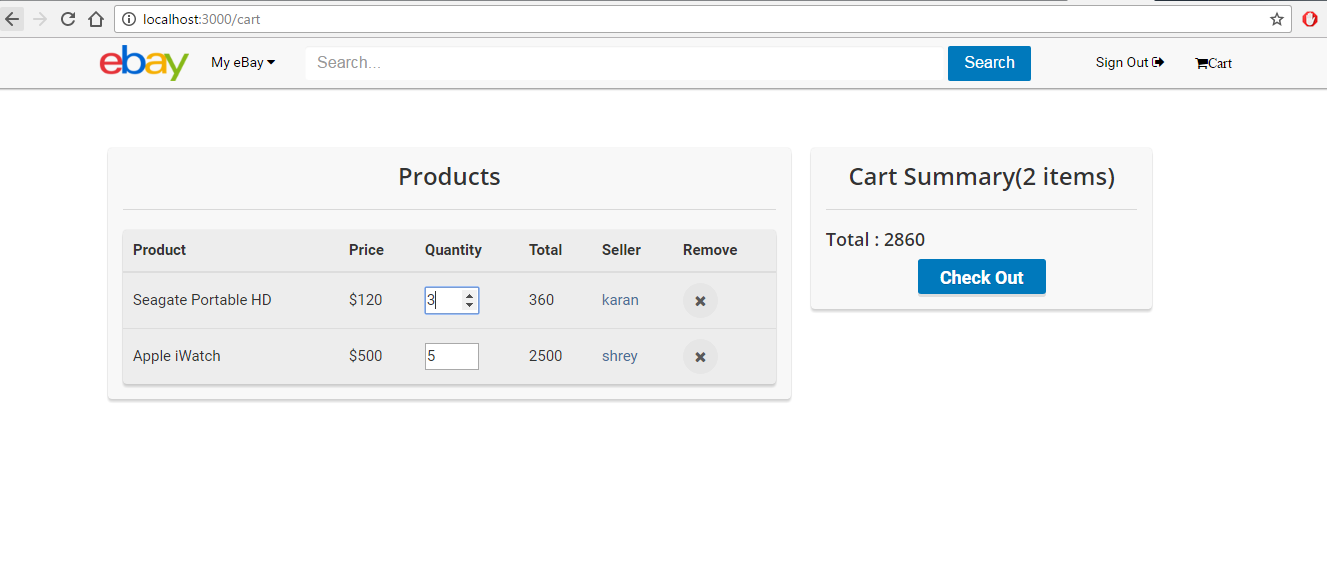


**Multiple Products in Cart:**

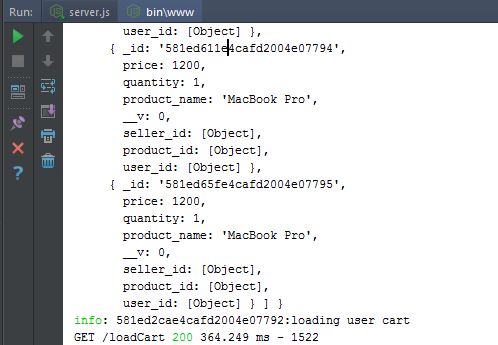
User can add multiple products to cart.



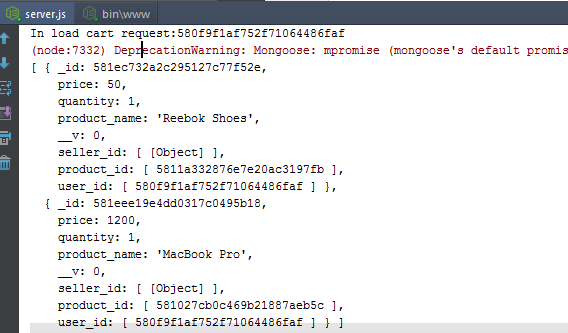
**Products in cart can be deleted from cart , Quantities can be changed from cart. Cart Summary and total is updated dynamically.**



**LoadCart Client Console:**

****

**LoadCart Server Console:**



**CheckOut Page/Credit Card Validations:**

Once user clicks on Checkout in cart page , user is redirected to CheckOut Page.

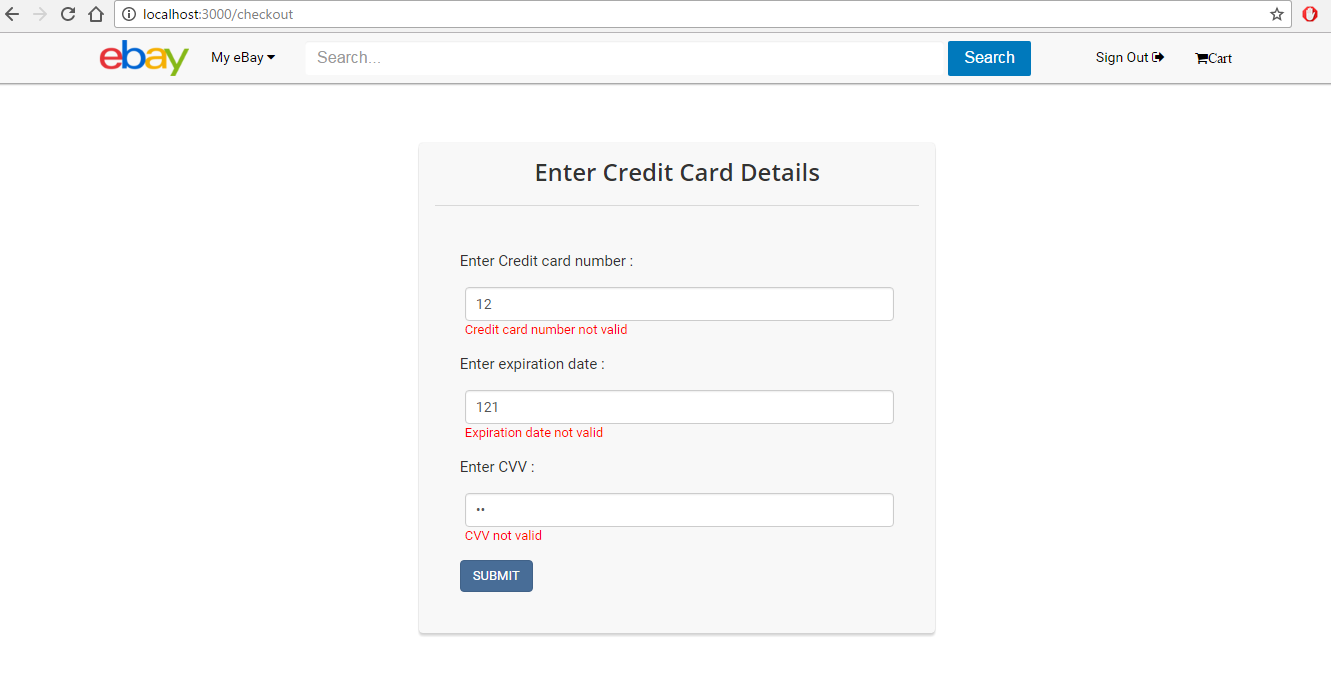
Check out page contains a form that accepts credit card information for payment.

Credit Card validations a performed for:

Credit card Number

Expiration Date

CVV



**Valid Credit Card Details:**

Once user clicks on Checkout in cart page , user is redirected to CheckOut Page.

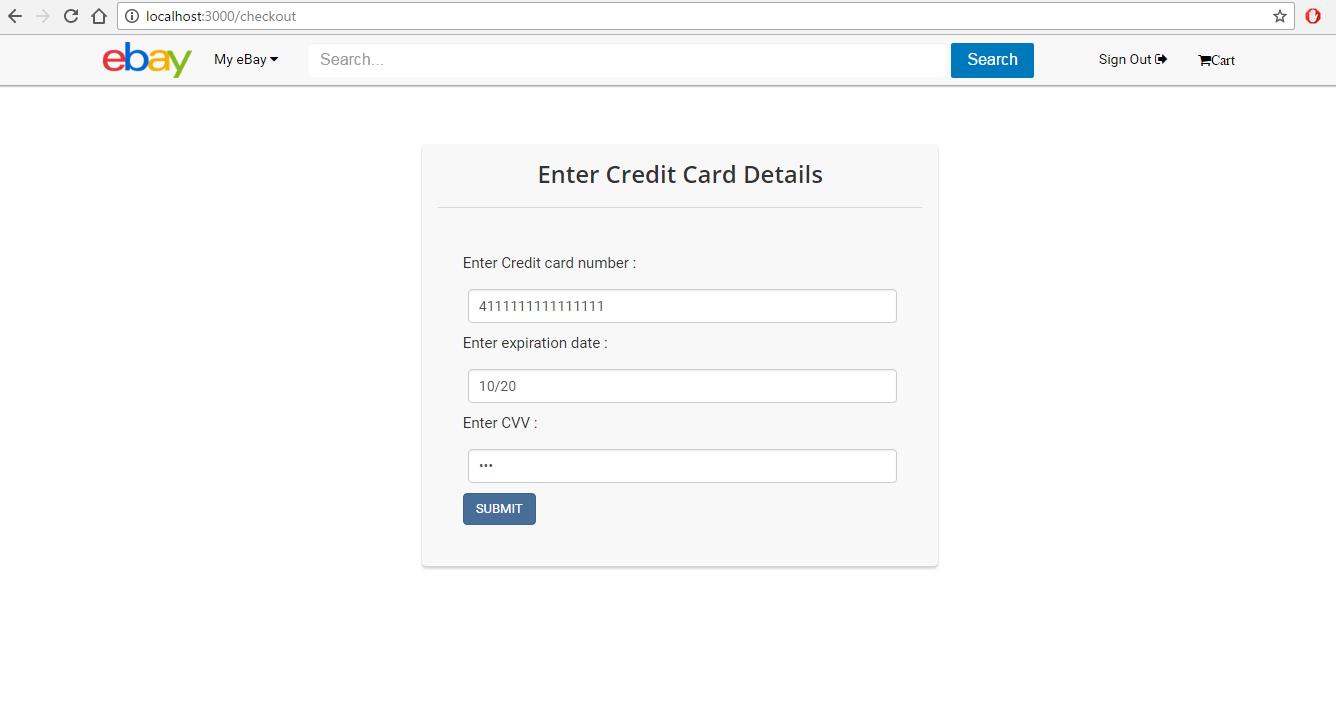
Check out page contains a form that accepts credit card information for payment.

Credit Card validations a performed for:

Credit card Number

Expiration Date

CVV



**Processing Payment:**

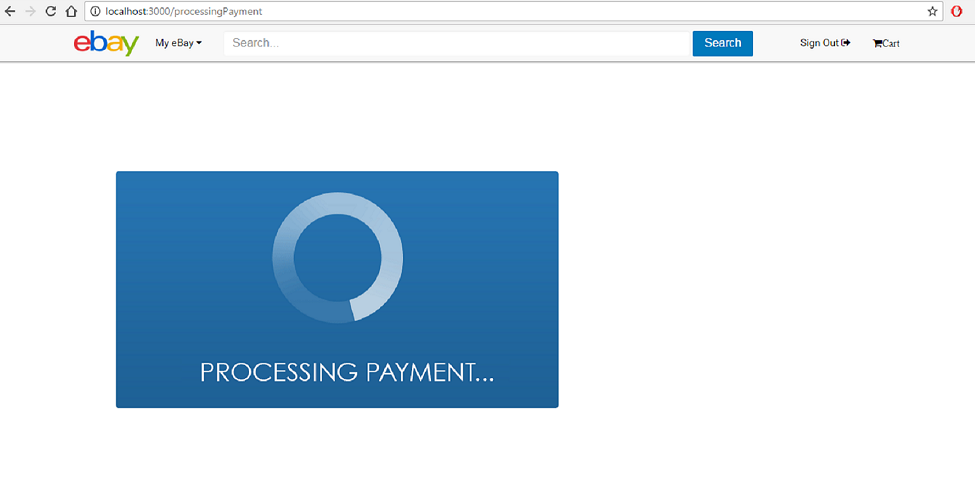
Once user enters correct credit card details and clicks on submit it is redirected to Process Payment Page.

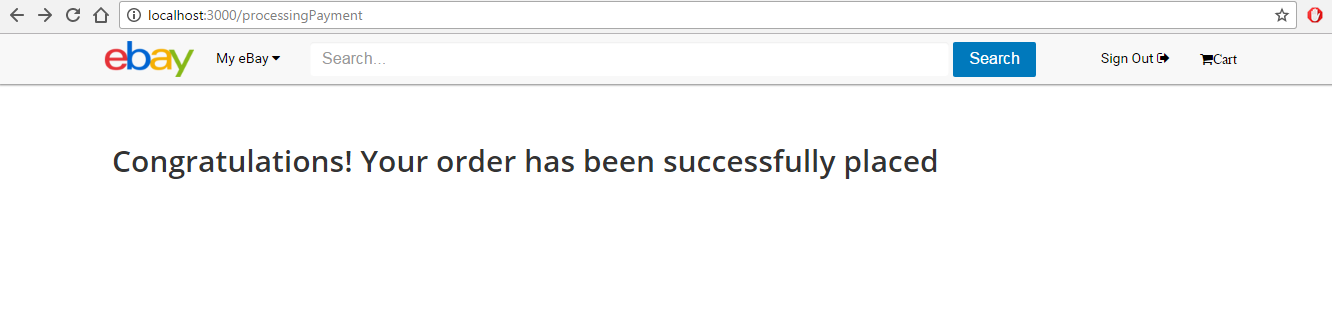
On Submit Click following things happen in server:

Add all Products in user’s Order

Update Seller’s Product Quantities

Empty Cart





**Empty Cart and My eBay Menu:**

If there are no items in cart or if user removes all items from cart then following page is loaded.

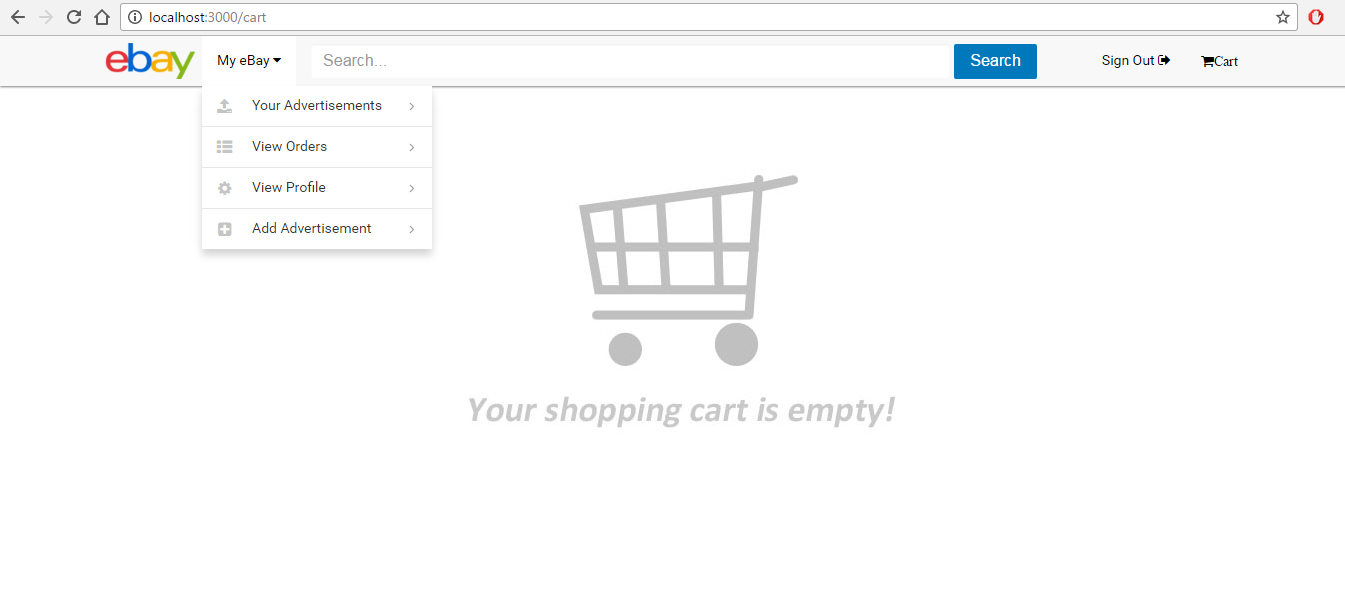
eBay Menu comprises of:

Your Advertisements: Advertisements added by User

View Orders : View Successful purchases

View Profile : Load user profile

Add Advertisement : Add new advertisement on eBay.

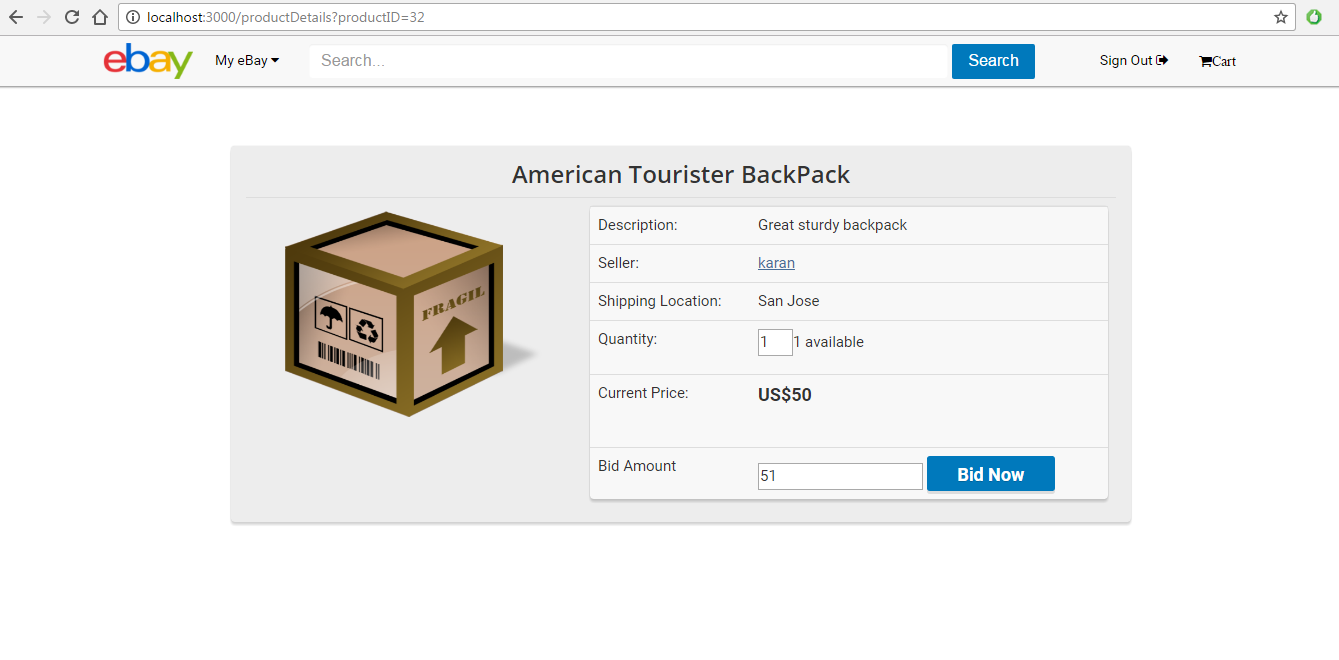


**Product Detail for Bidding:**

Product Detail page for Bidding products is same as that for Direct Buy products.

Only difference is that quantity is always 1 and user can enter amount for bid.

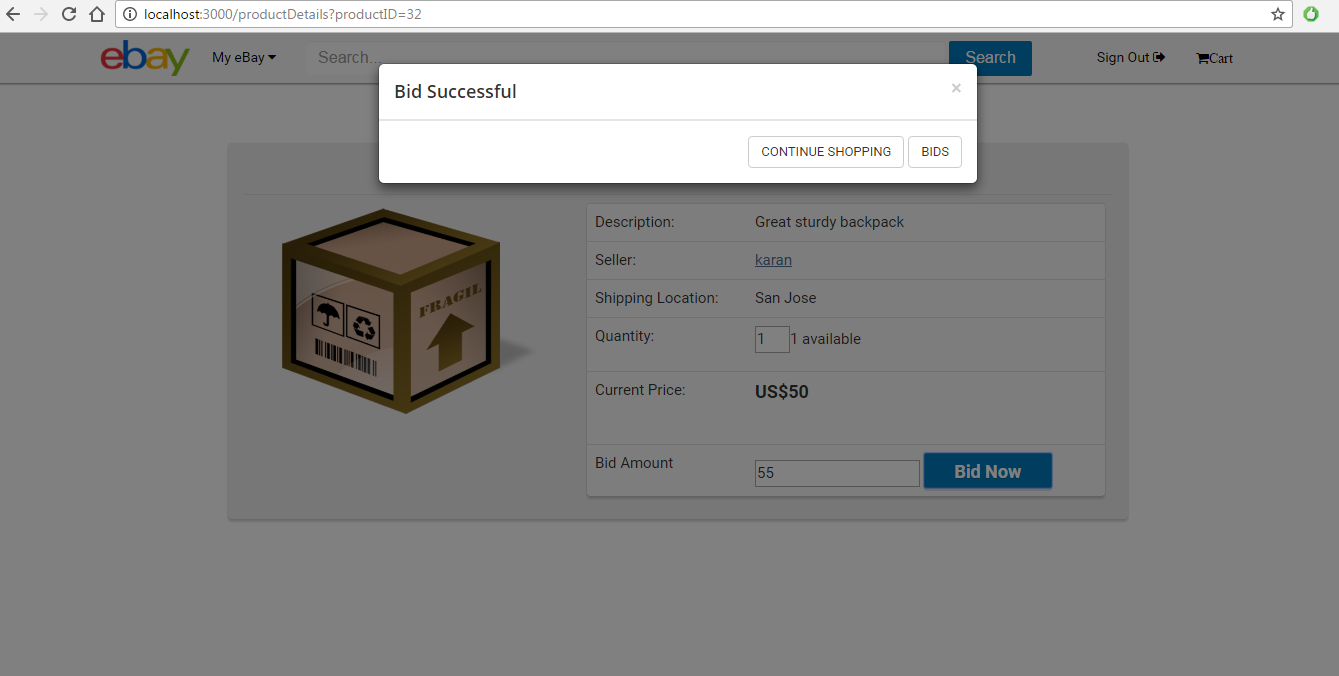
On click on bid now current price for the product updates to highest bid.



**Successful Bid:**

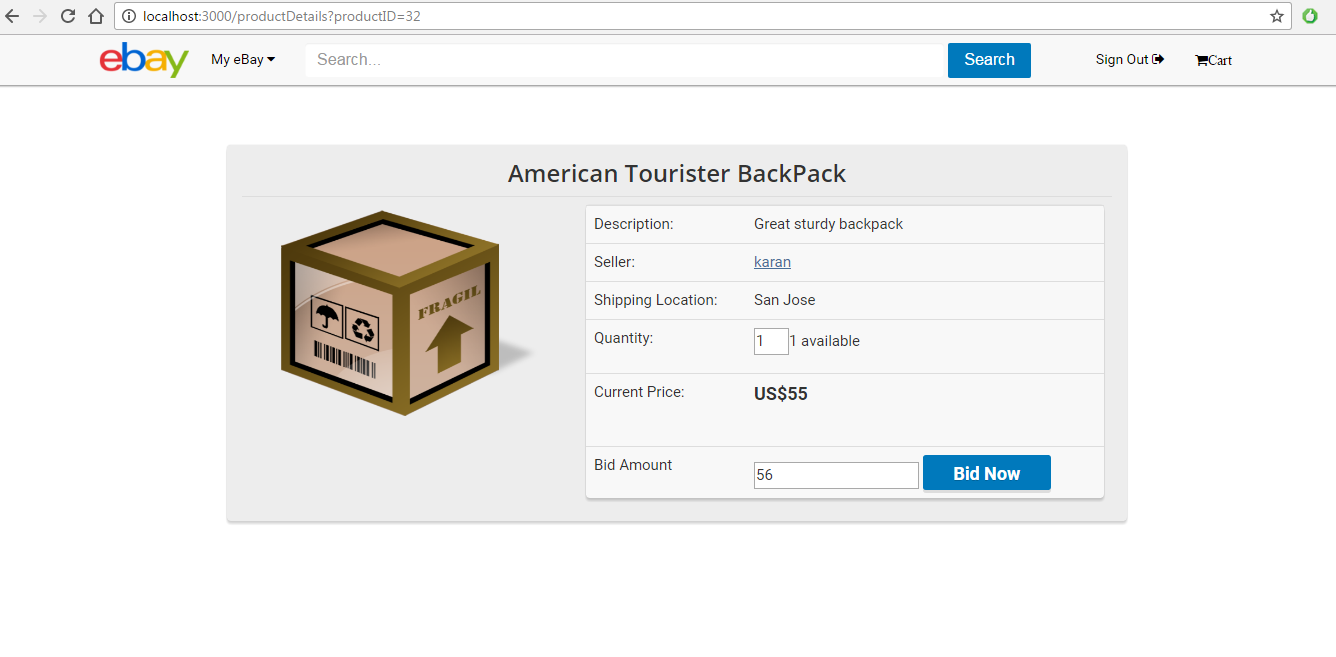
When user enters a valid bid amount and clicks on bid now, a new bidding entry is created in BidDetail collection for the user and Current Bid is updated for the product.

When the Bidding duration for the product ends , the product is added to highest bidder’s order list and seller’s account is also updated on successful bid end.



**Current Price Updated:**

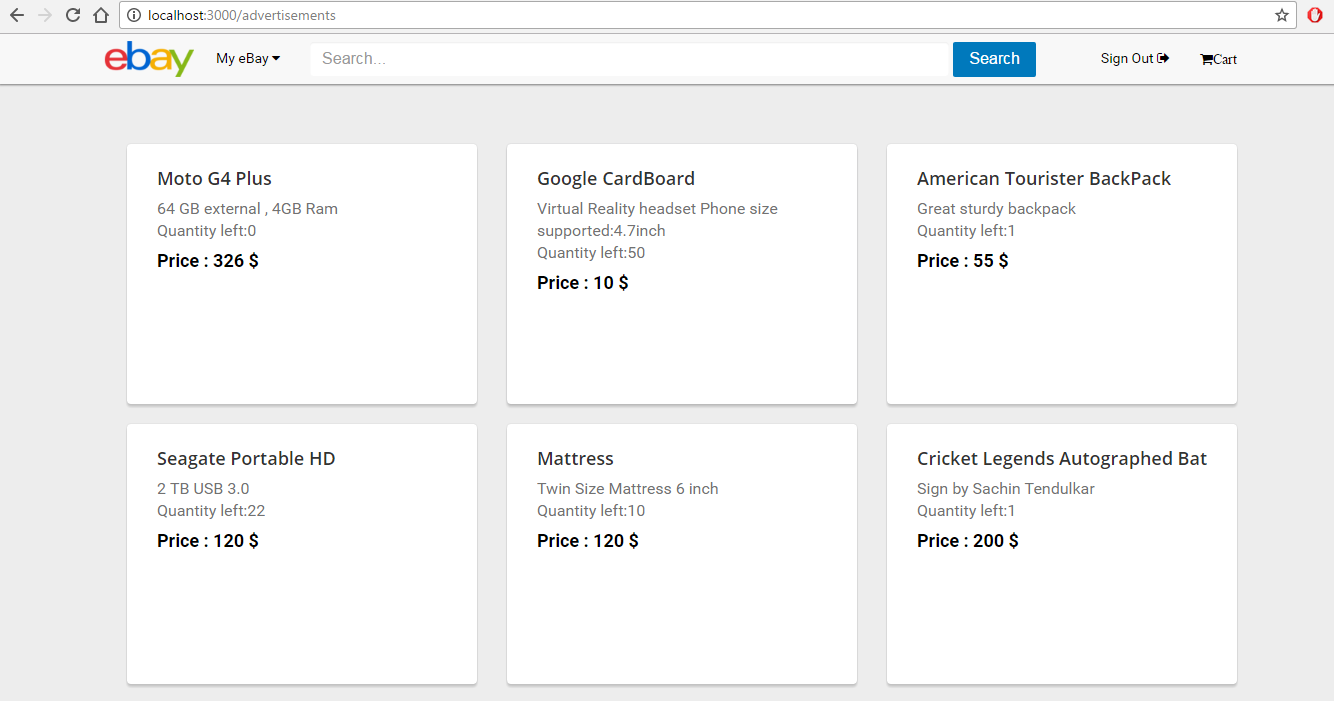
When user enters a valid bid amount and clicks on bid now, a new bidding entry is created in BidDetail collection for the user and Current Bid is updated for the product.



**Products added by user for selling:**

All the advertisement that are added by user for selling or bidding on eBay are visible to user on this page.

User can view the available quantities and the current highest bid of the bidding product added by user.

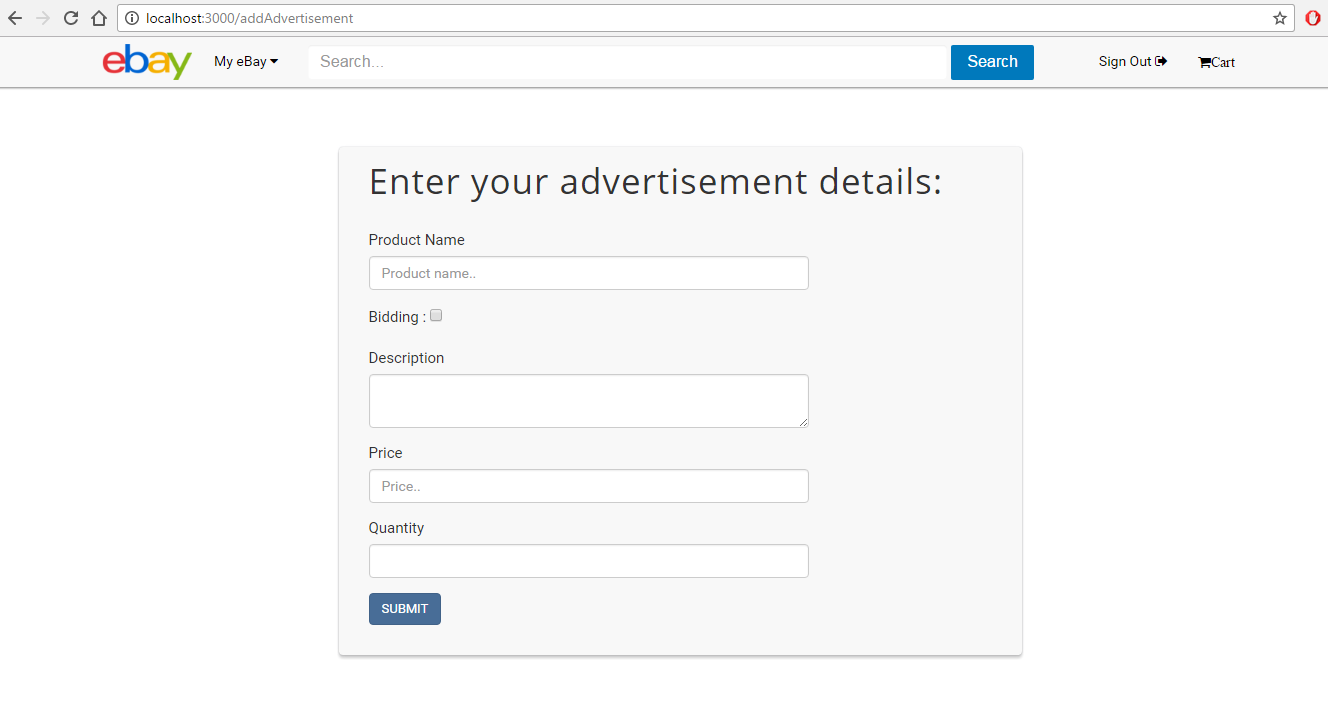


**Adding a new product advertisement:**

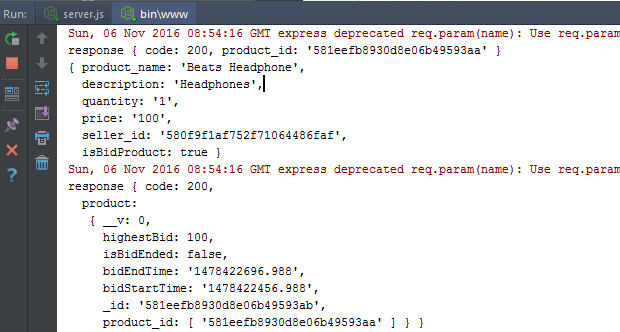
User can add new products for direct buy or bidding using these page.

Below form is to add a product for direct buy.

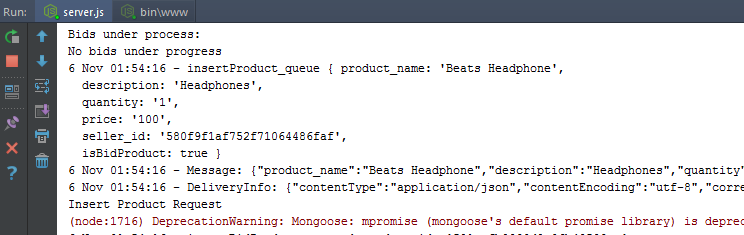
User can set price and quantity.



**InsertProduct Client Console:**

****

**InsertProduct Server Console:**

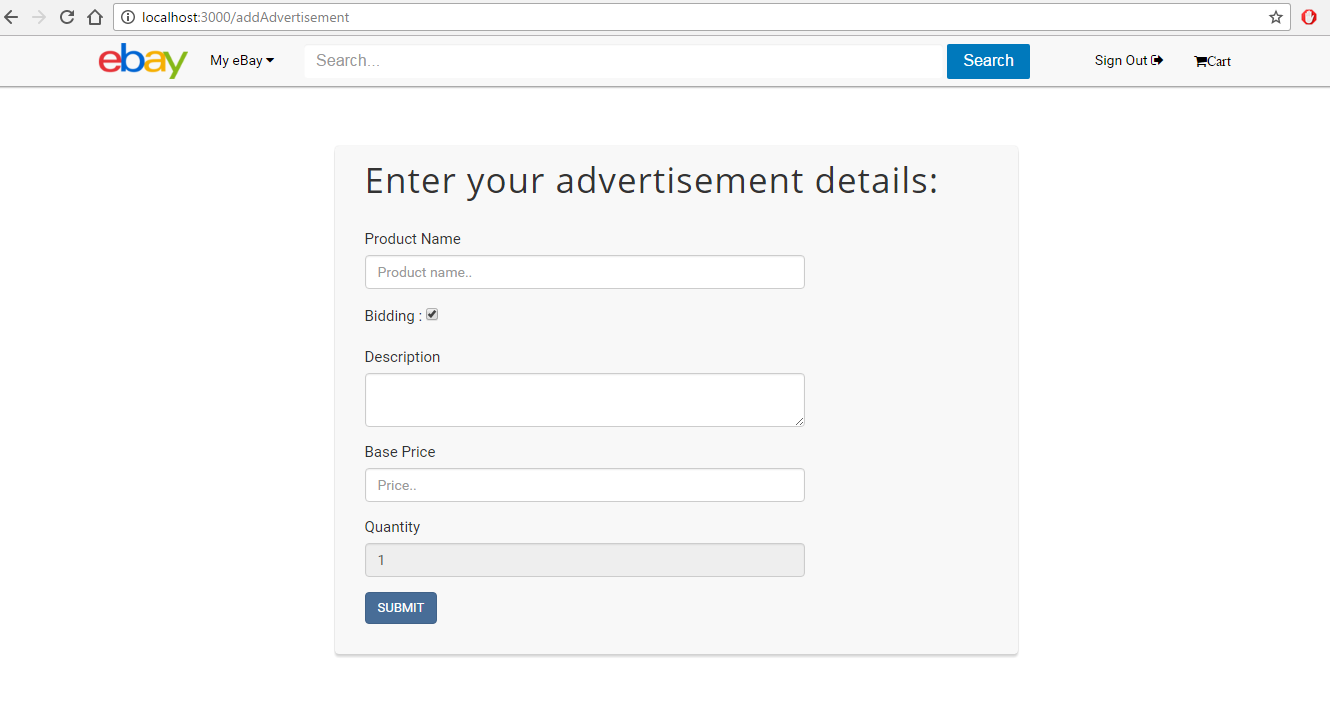


**Adding a new bidding product**:

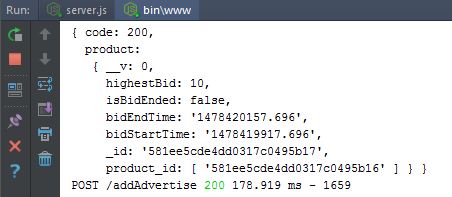
User can add new products for direct buy or bidding using these page.

Below form is to add a product for bidding.

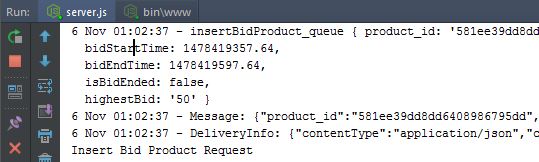
User can set base price and quantity is fixed to 1 for bidding products.



**InsertBidProduct Client Console:**

****

**InsertBidProduct Server Console:**

****

**View Orders:**

All the successful purchases done by user are available in user’s Order History.

For each order following information is available:

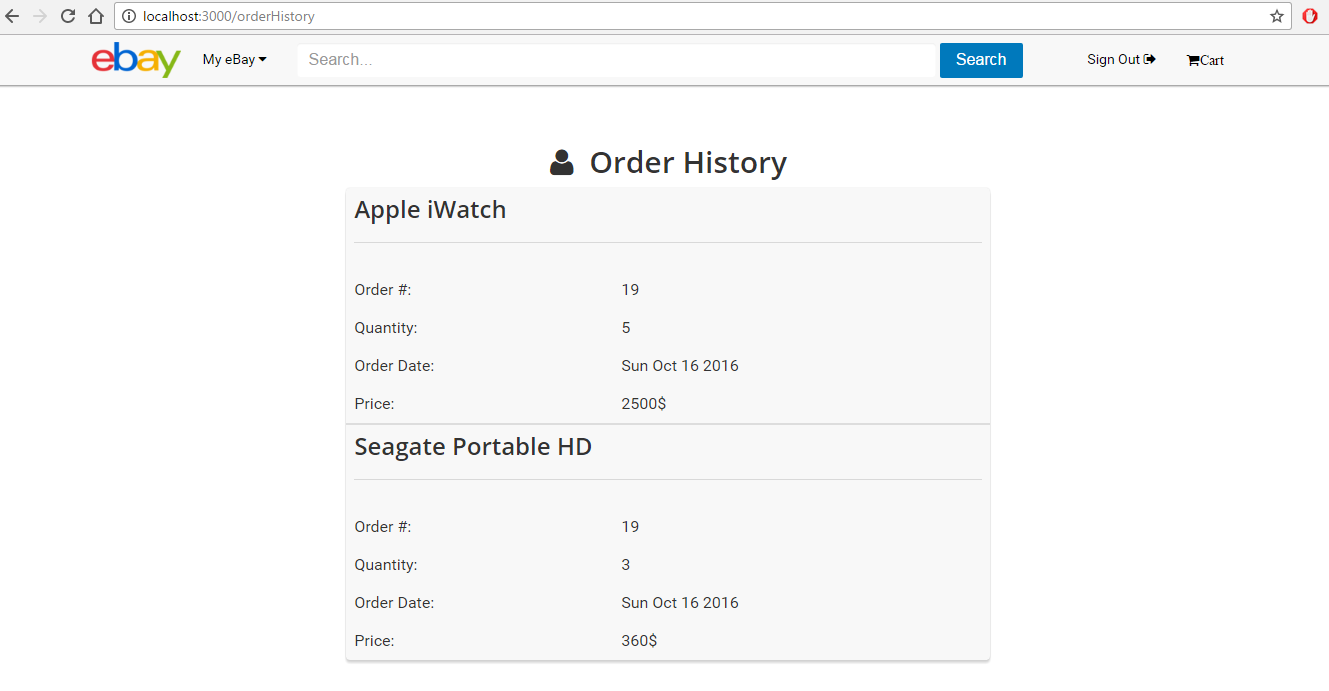
Order number

Quantity

Order Date

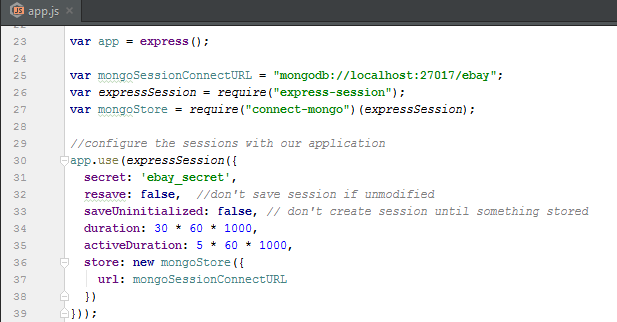
Product Name

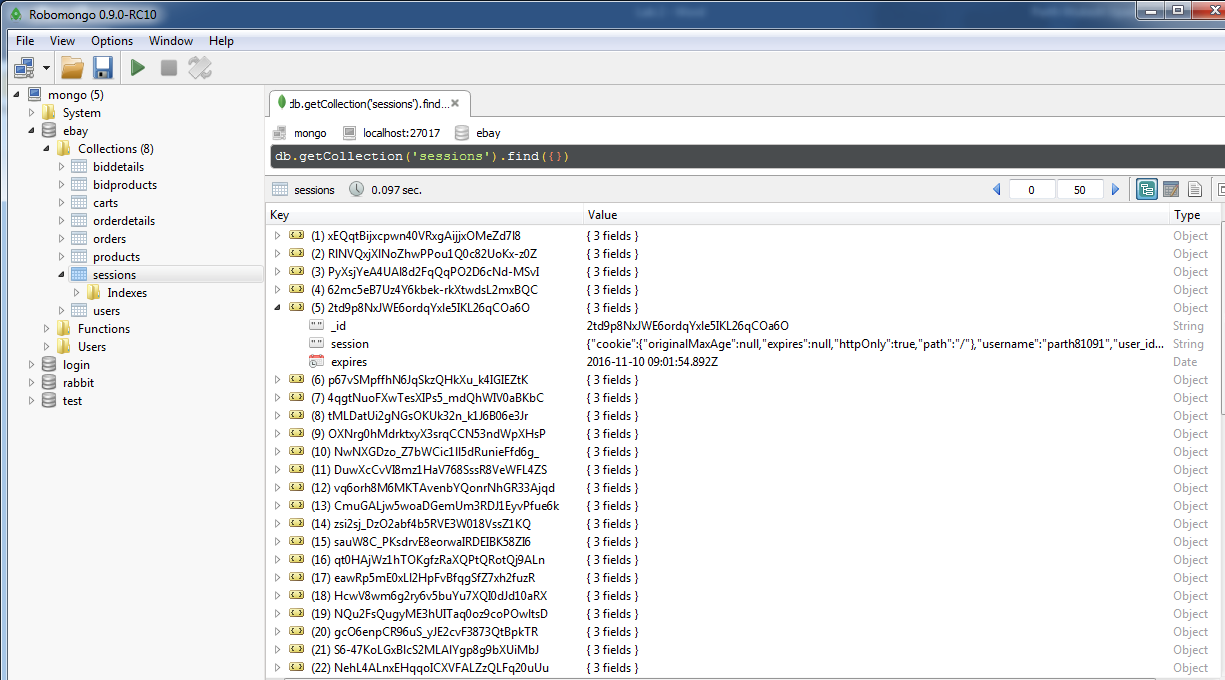
Price



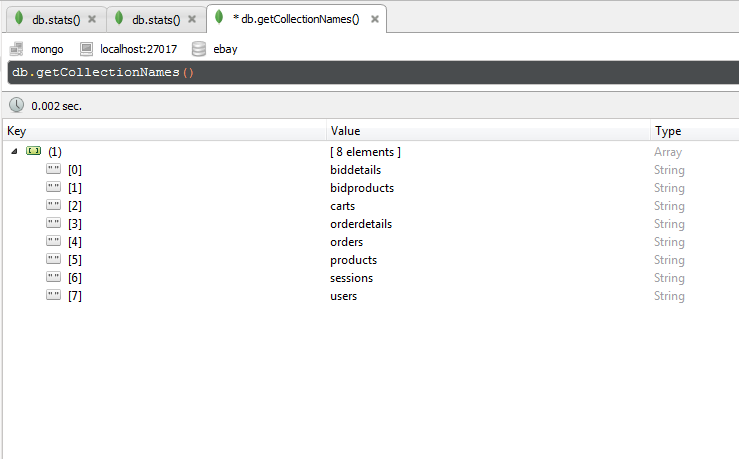
Mongo DB

Mongo Session:

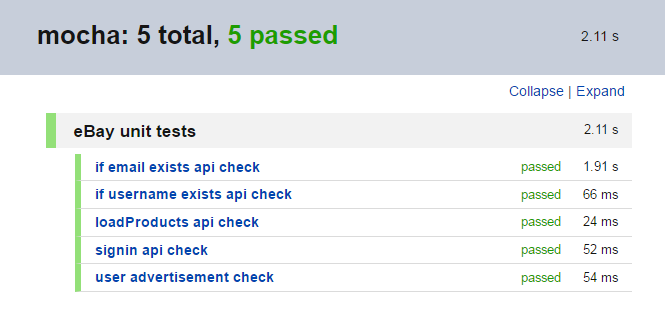




Mongo Collections:



Mocha testing





Passport JS – User Authentication

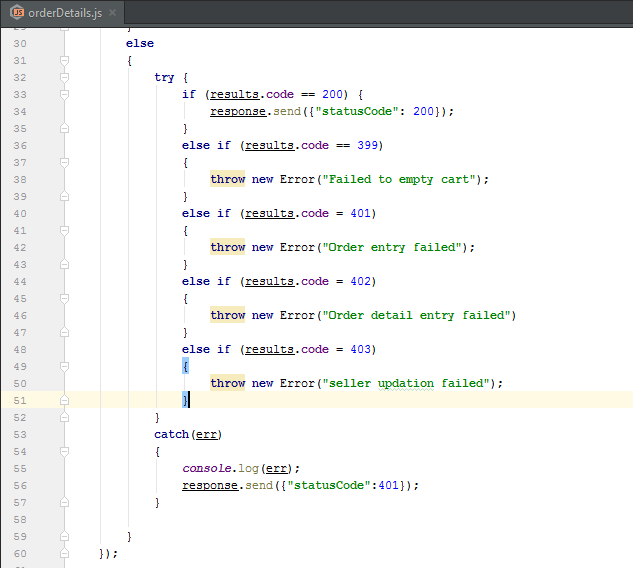
Passport.js



App.js



Exception Handling

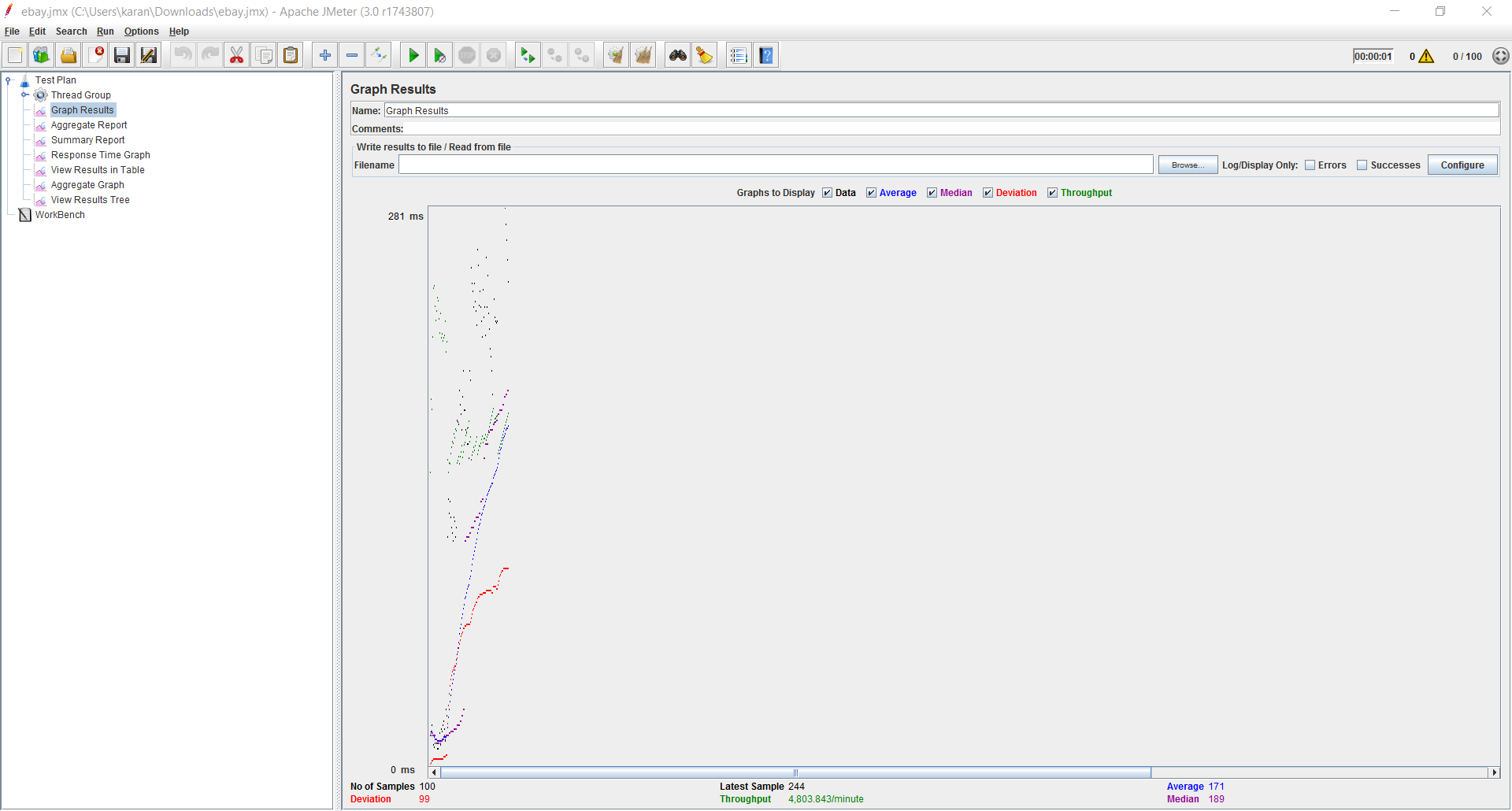




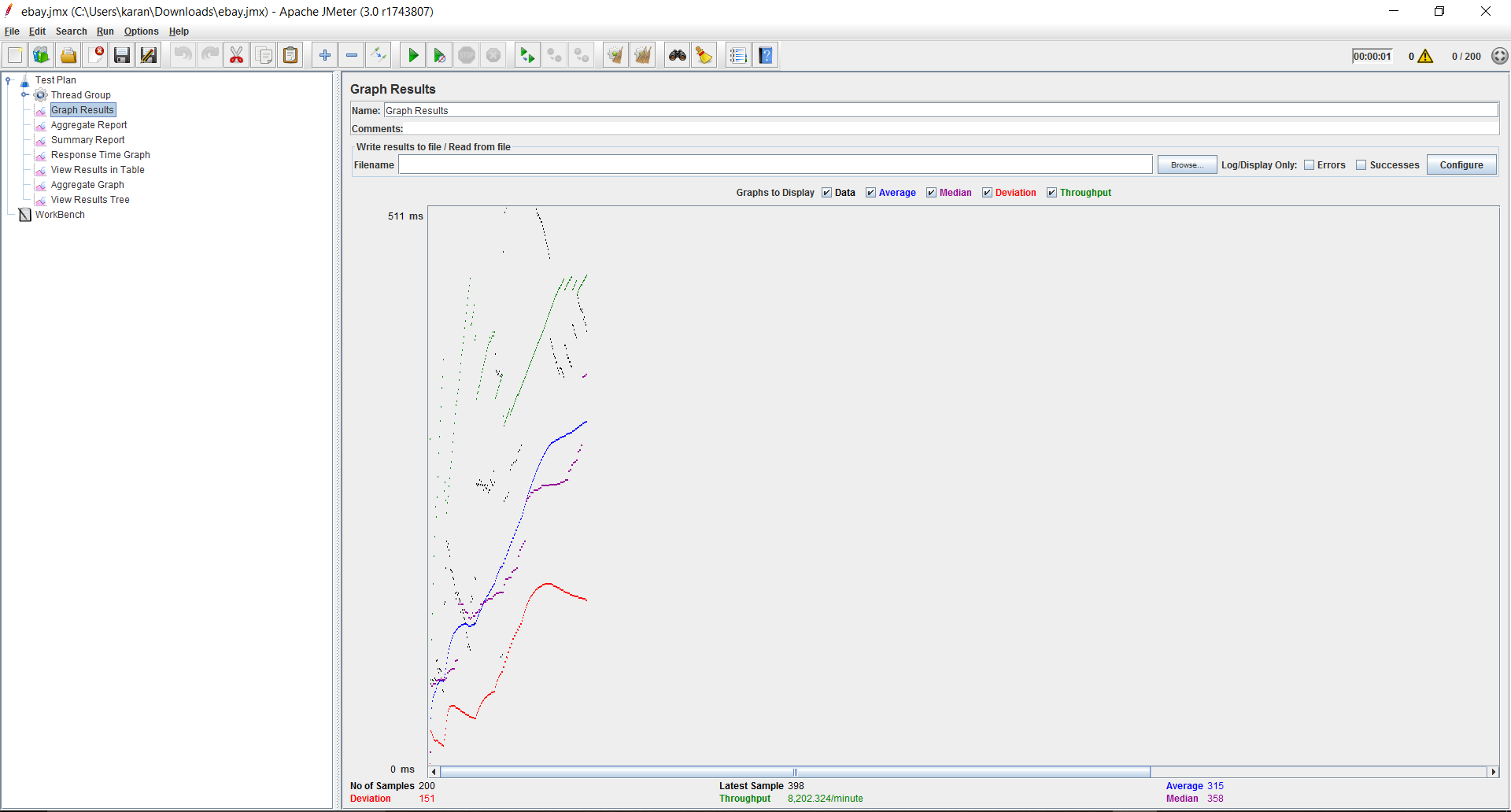
JMeter Testing

1. Without Connection Pool

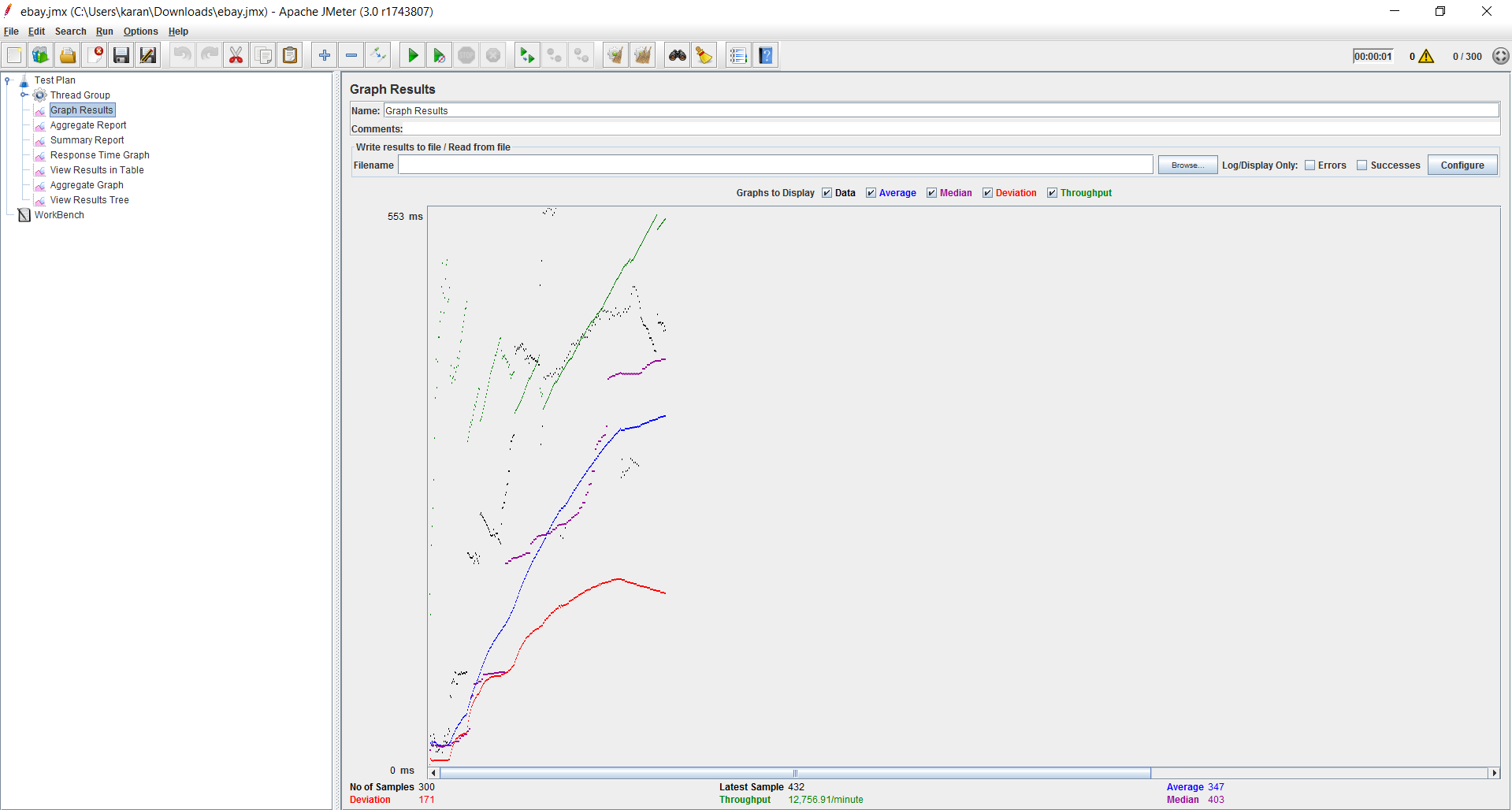
* 100 Concurrent Users



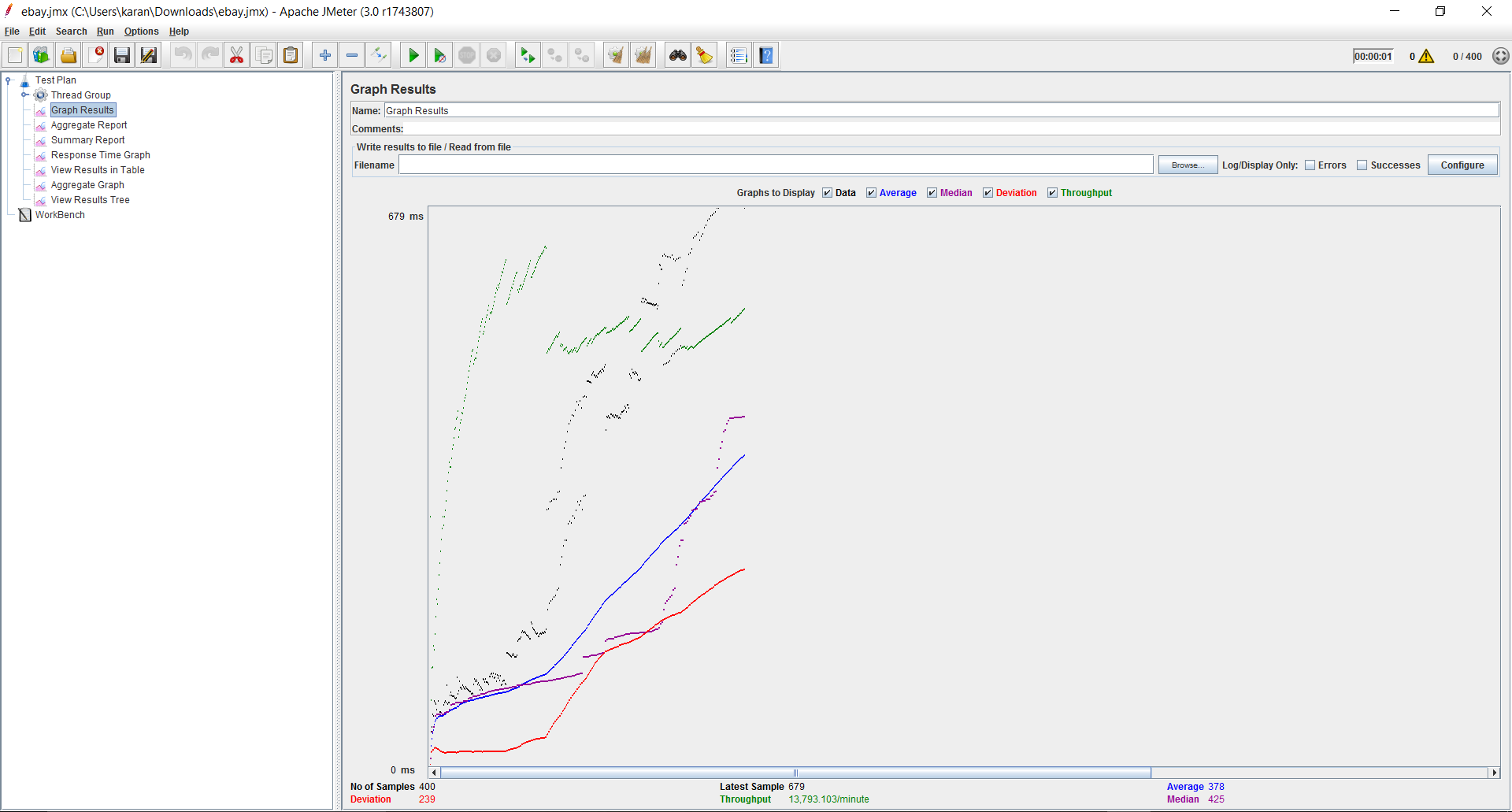
* 200 Concurrent Users



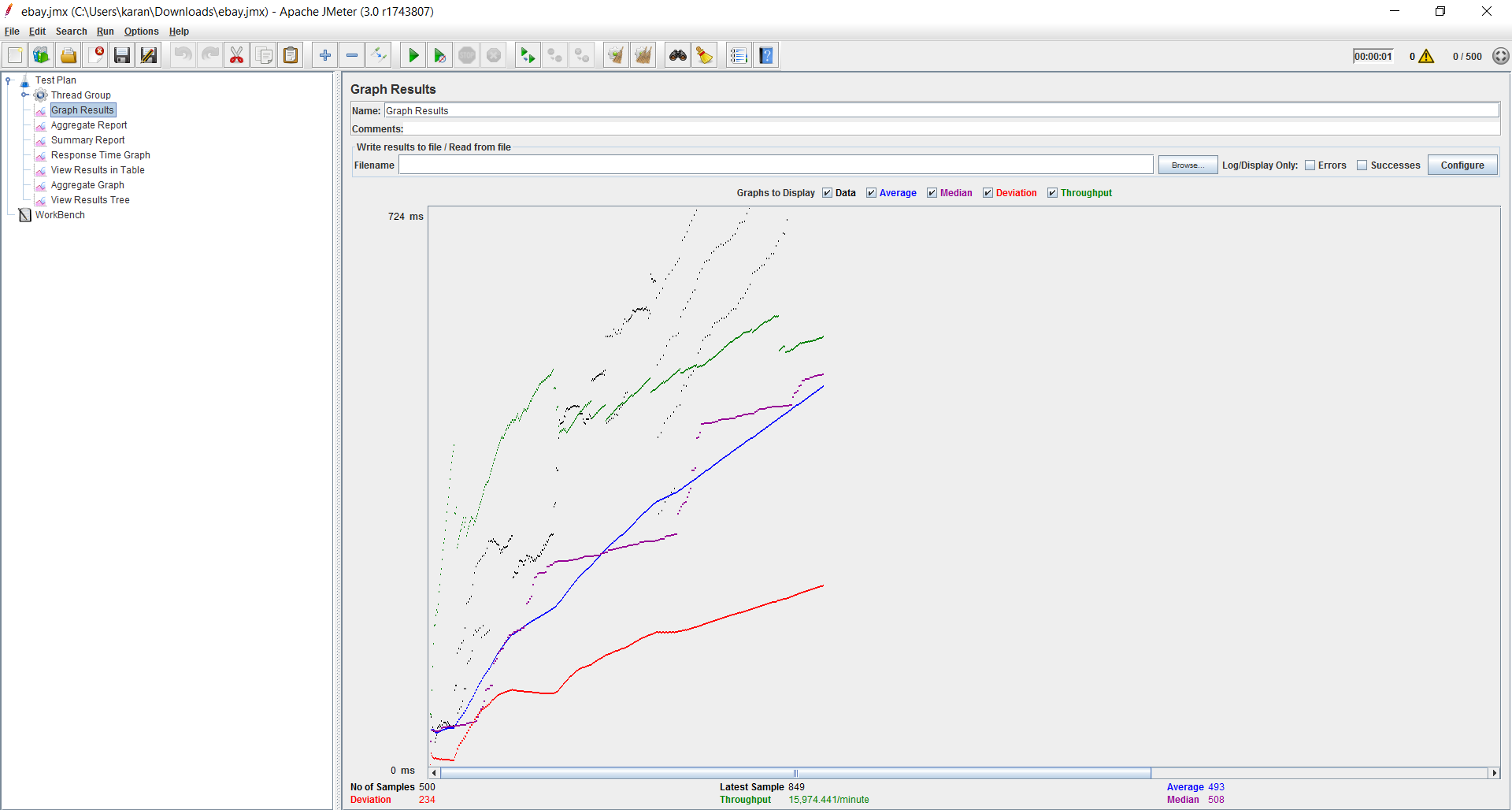
* 300 Concurrent Users



* 400 Concurrent Users

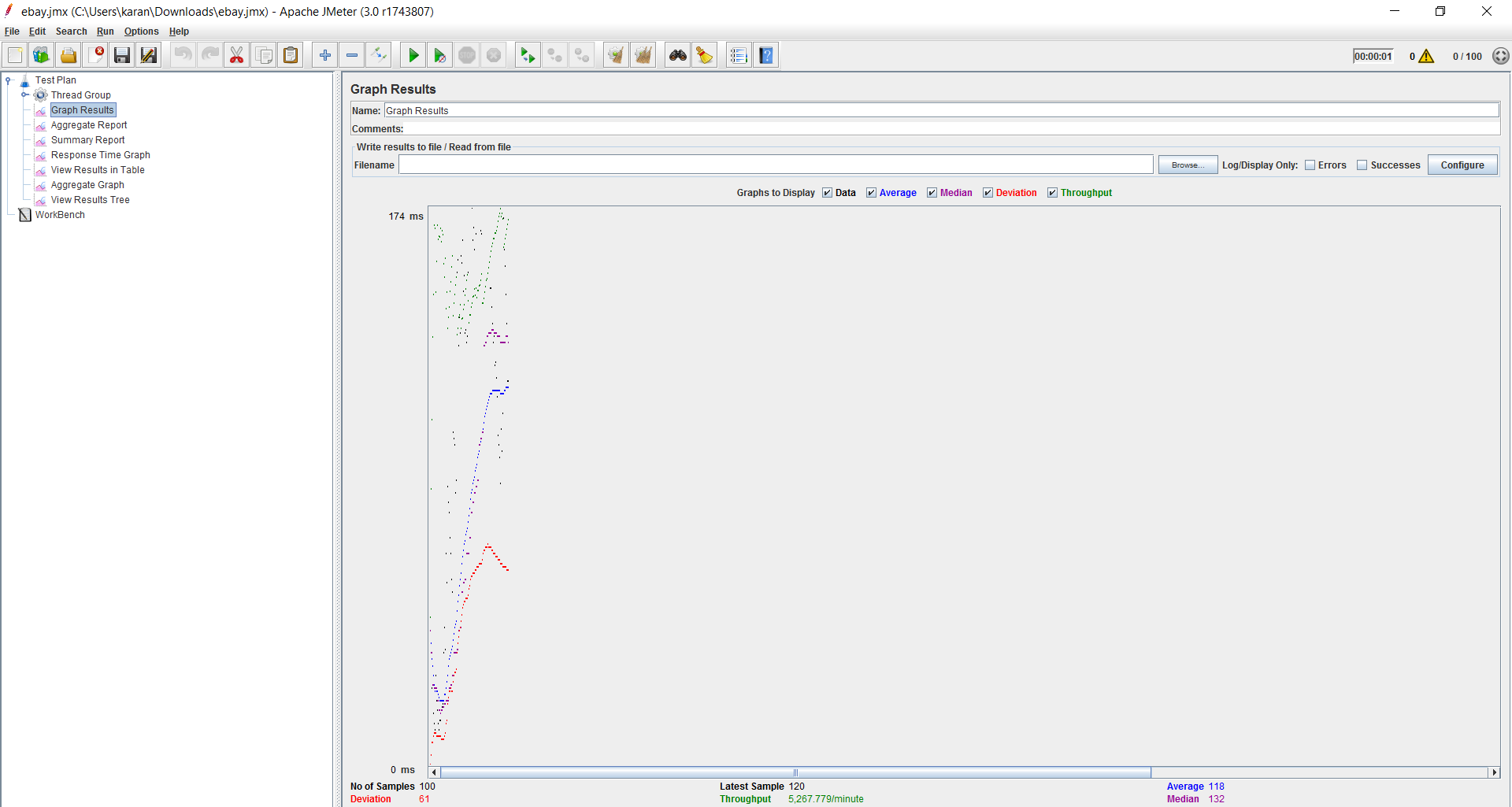


* 500 Concurrent Users

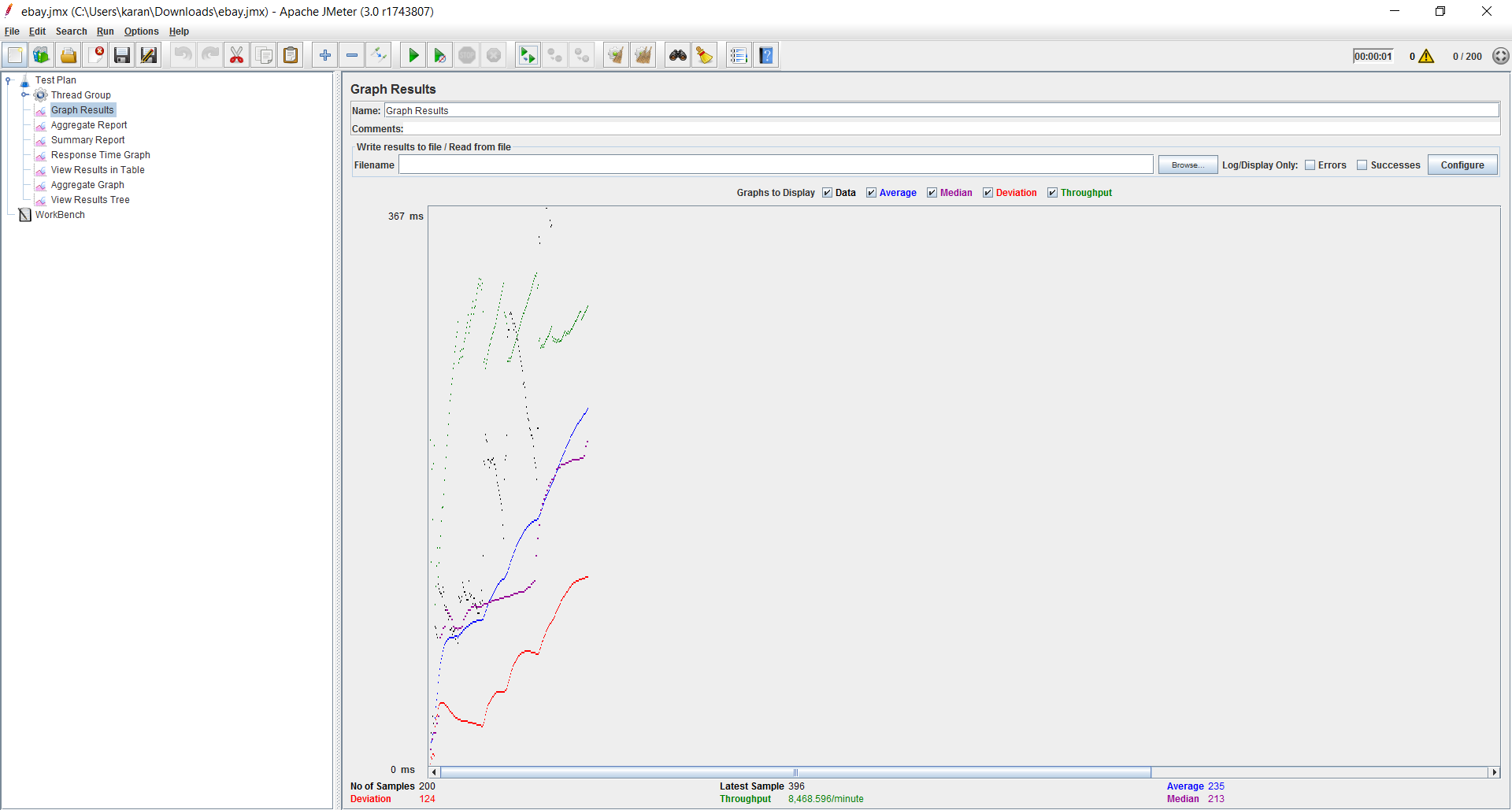


1. With Connection Pool

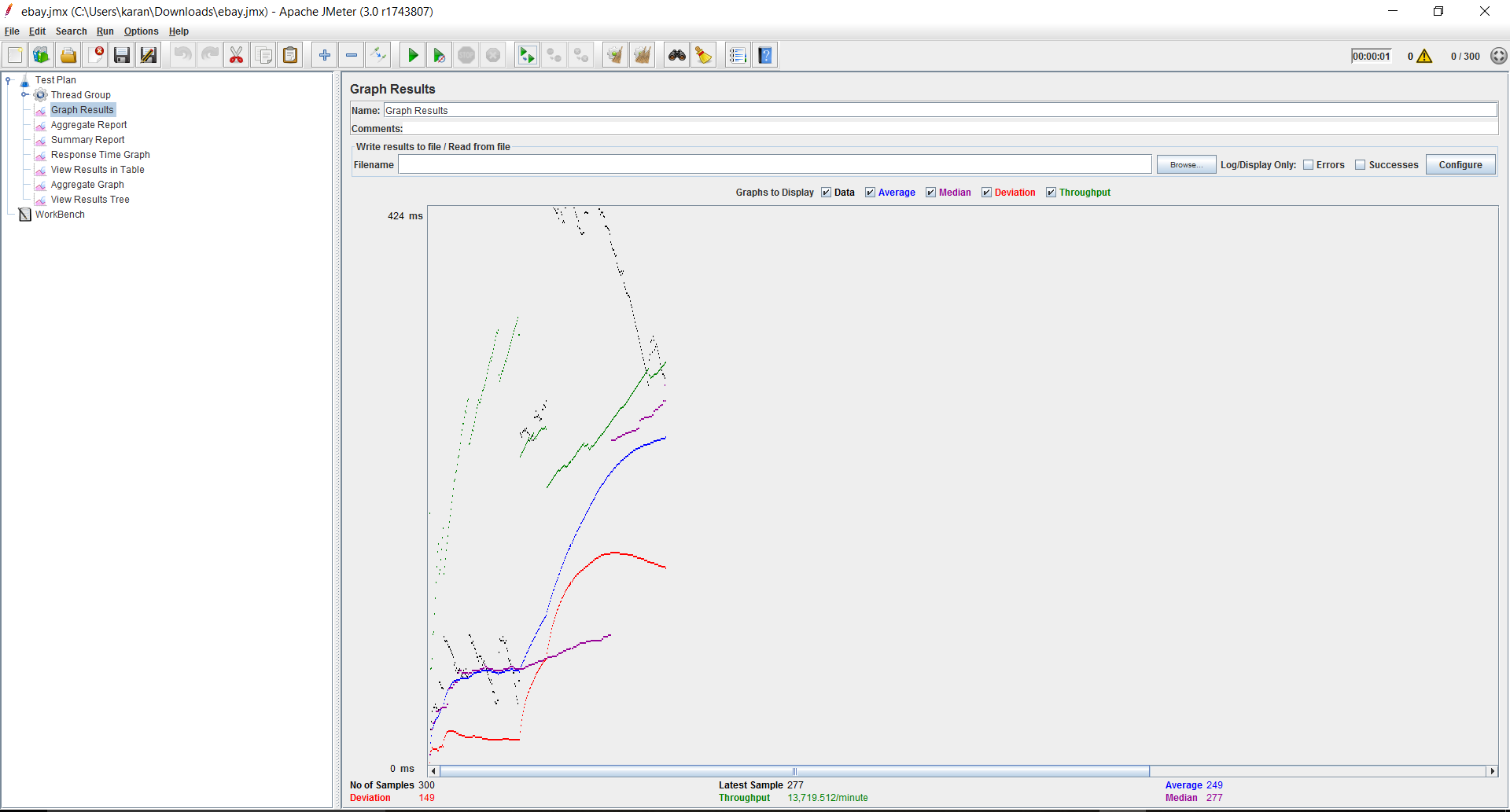
* 100 Concurrent Users



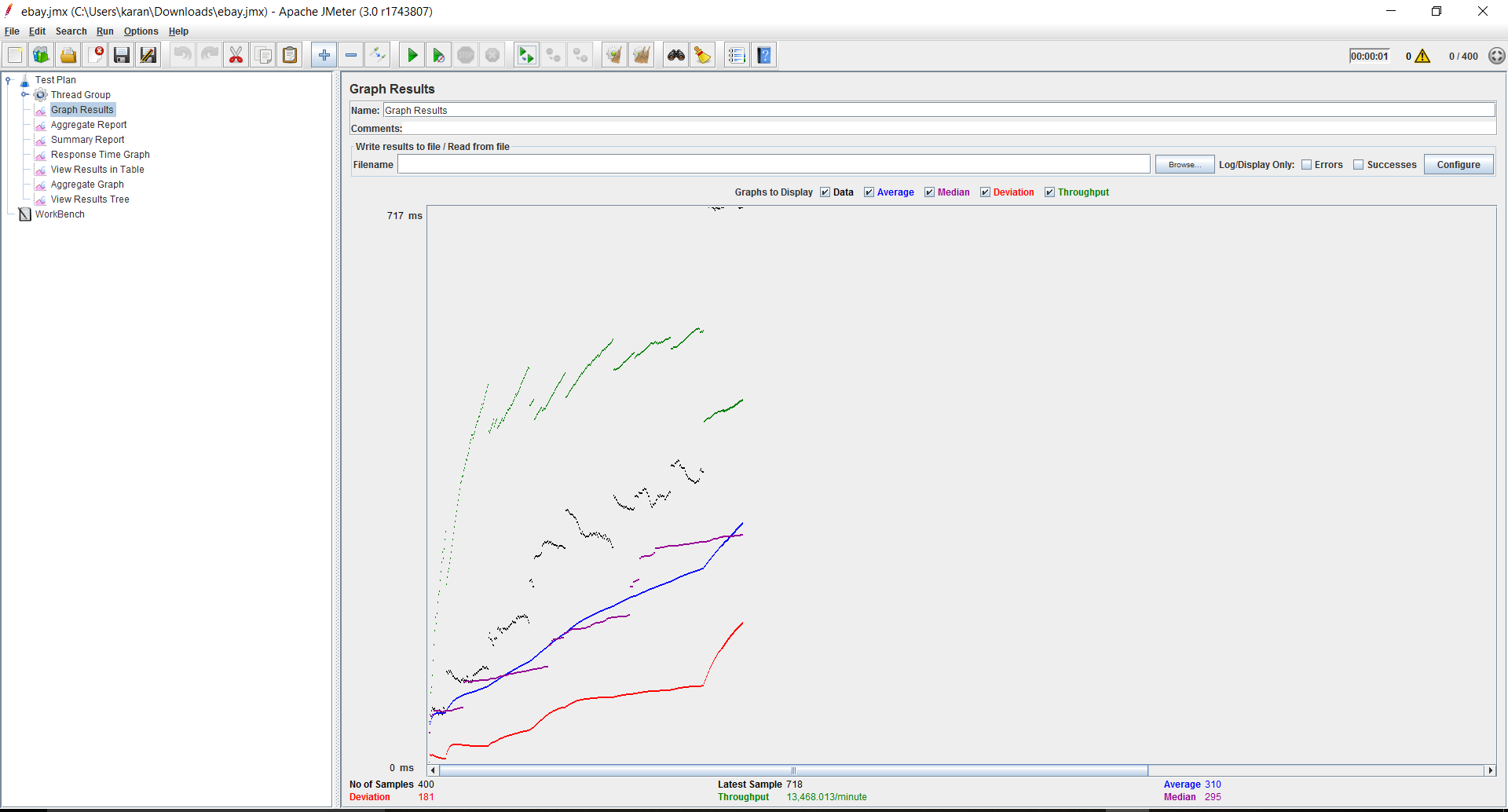
* 200 Concurrent Users



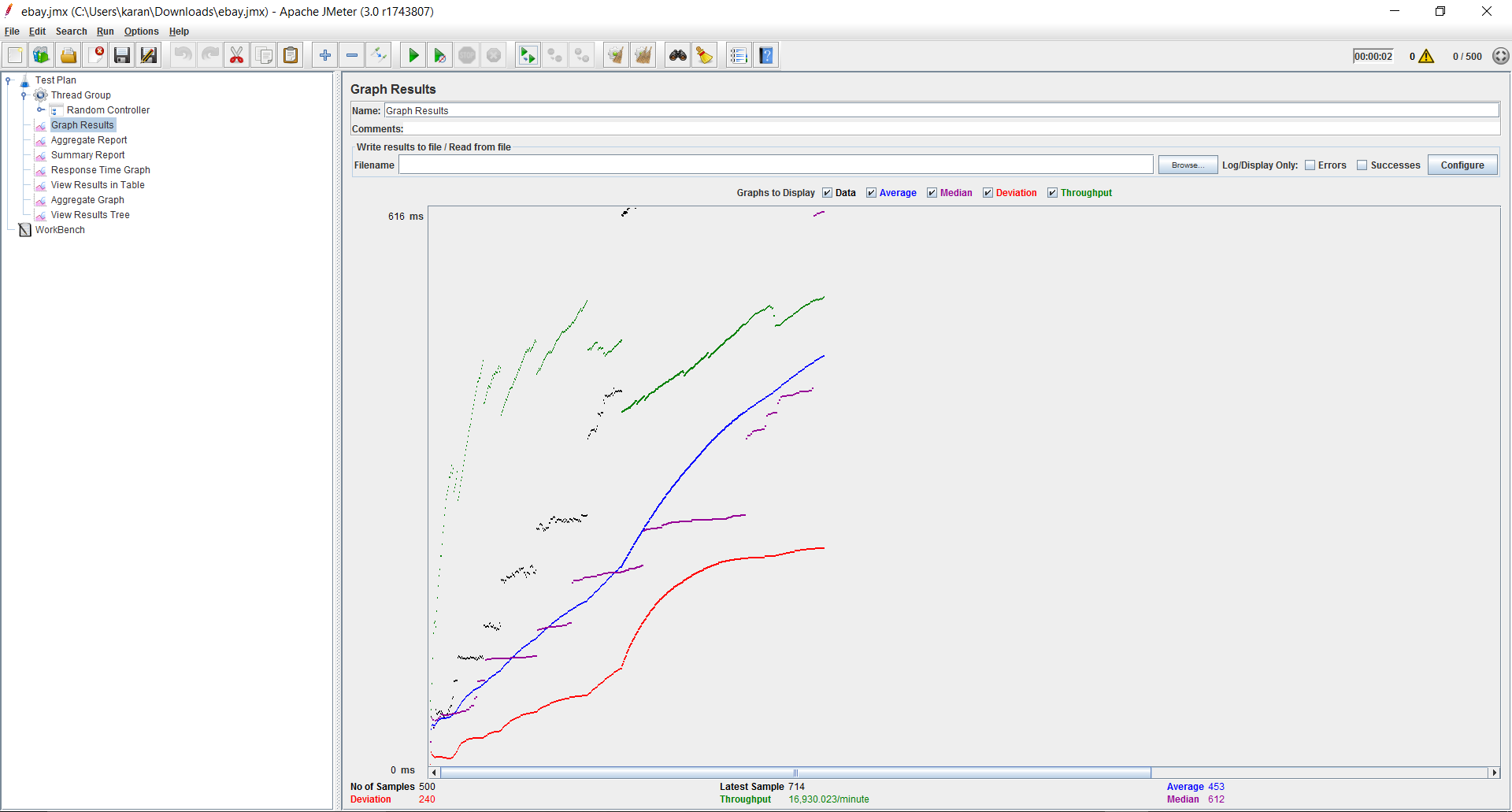
* 300 Concurrent Users



* 400 Concurrent Users



* 500 Concurrent Users



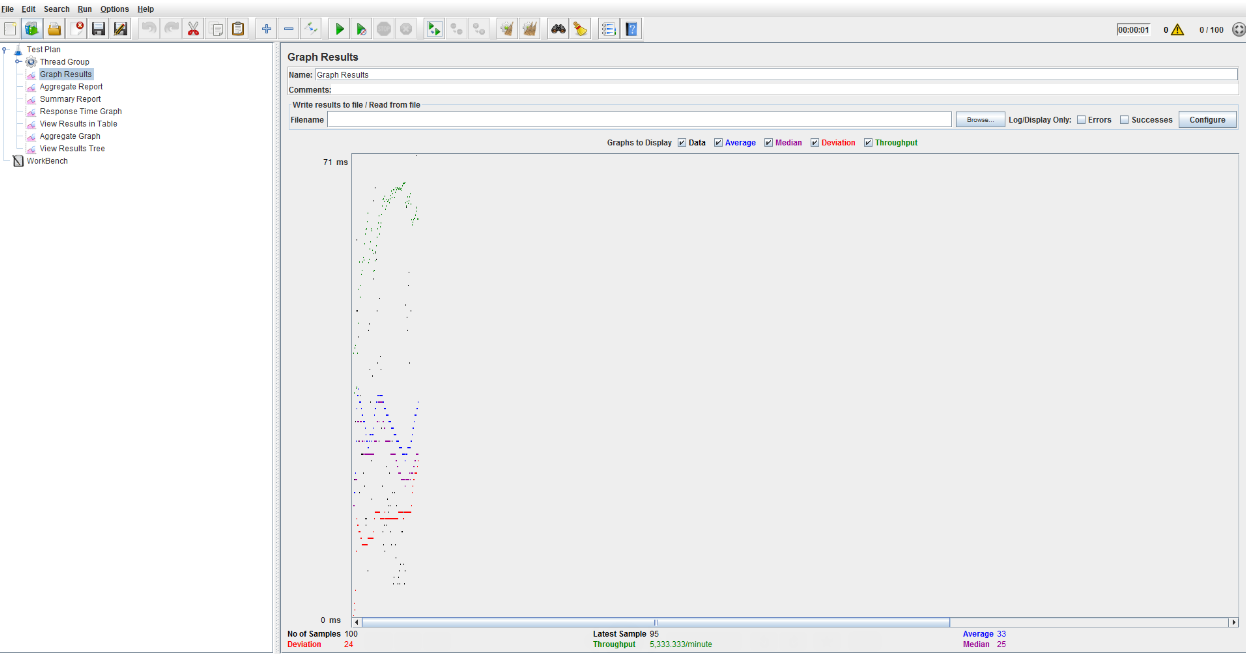
Average Time Comparison (Connection pool vs Without Connection pool):

Above you can see chart showing average time comparison between “With Connection Pool” and “Without Connection Pool”.

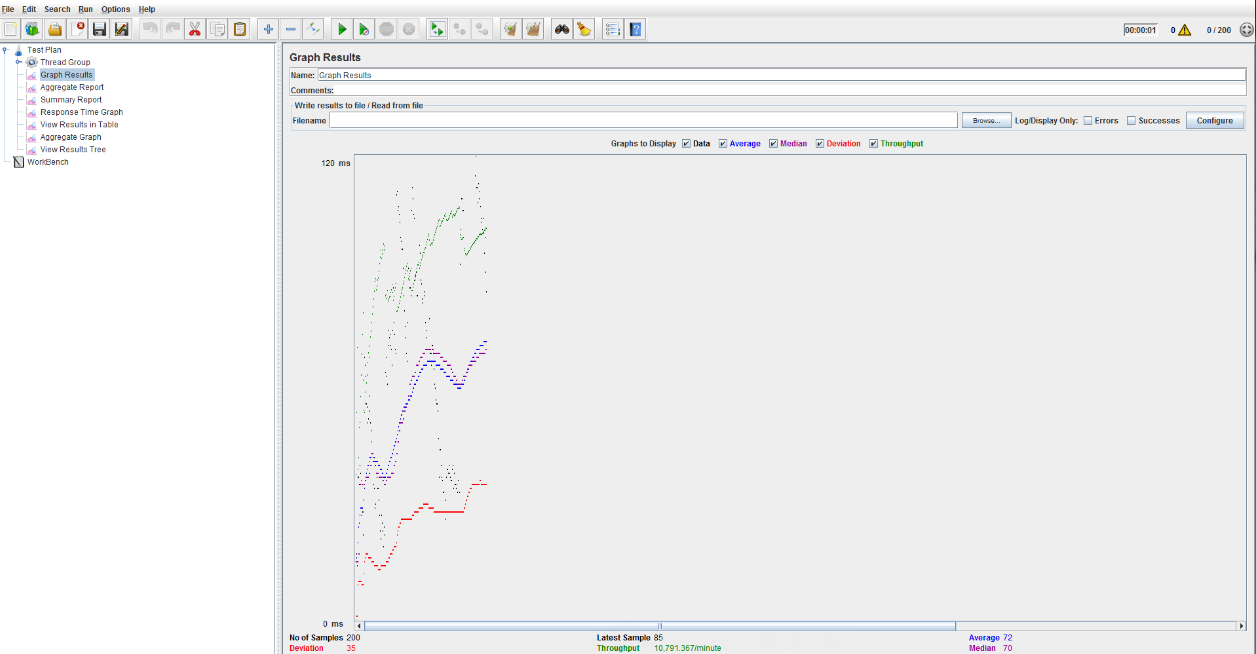
It is evident from above chart that Average time taken by api calls with connection pool is less as compared to without connection pool.

1. Without RabbitMQ

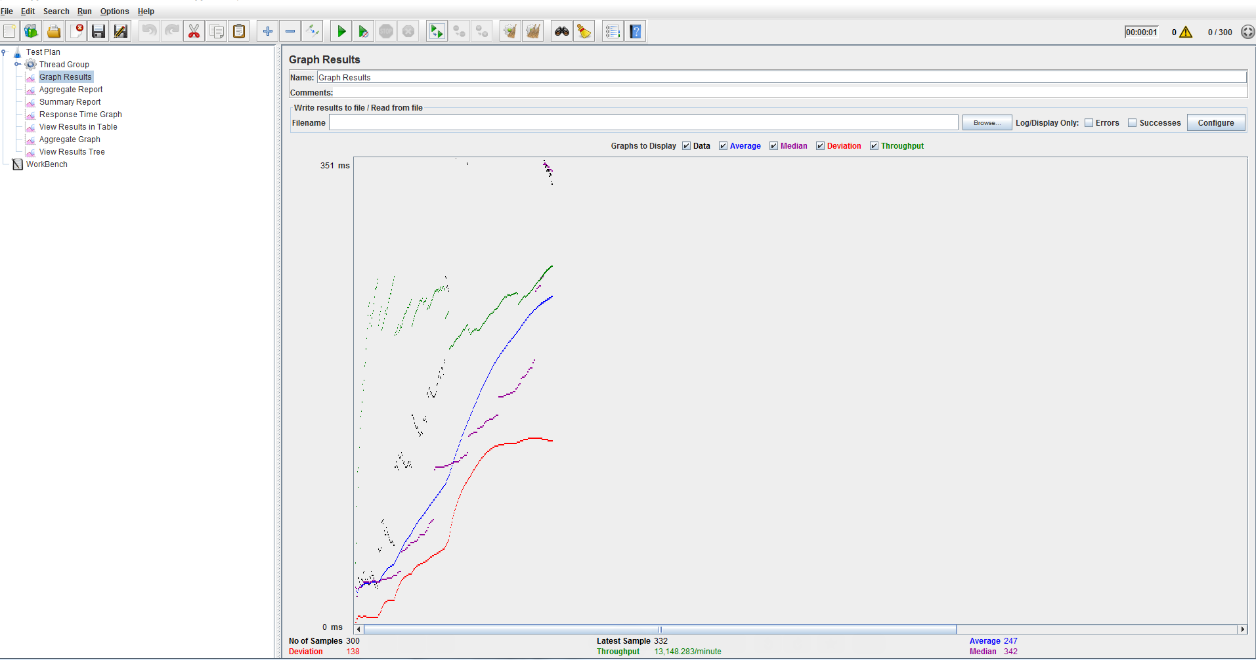
* 100 Concurrent Calls



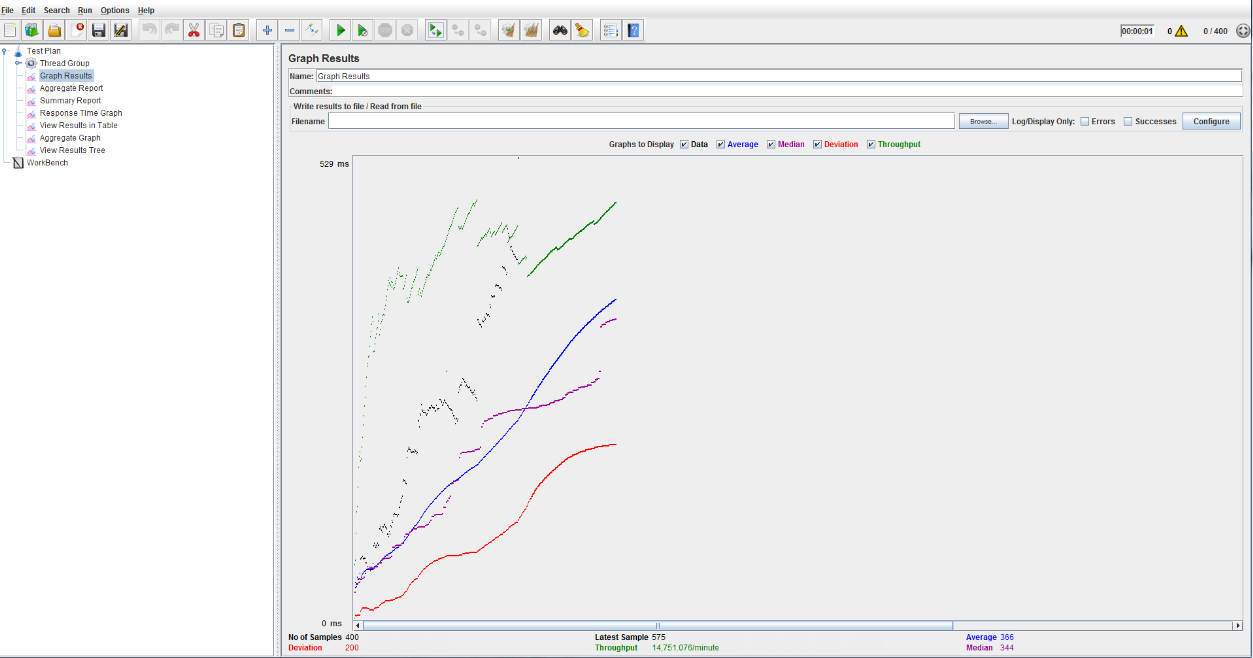
* 200 Concurrent Calls



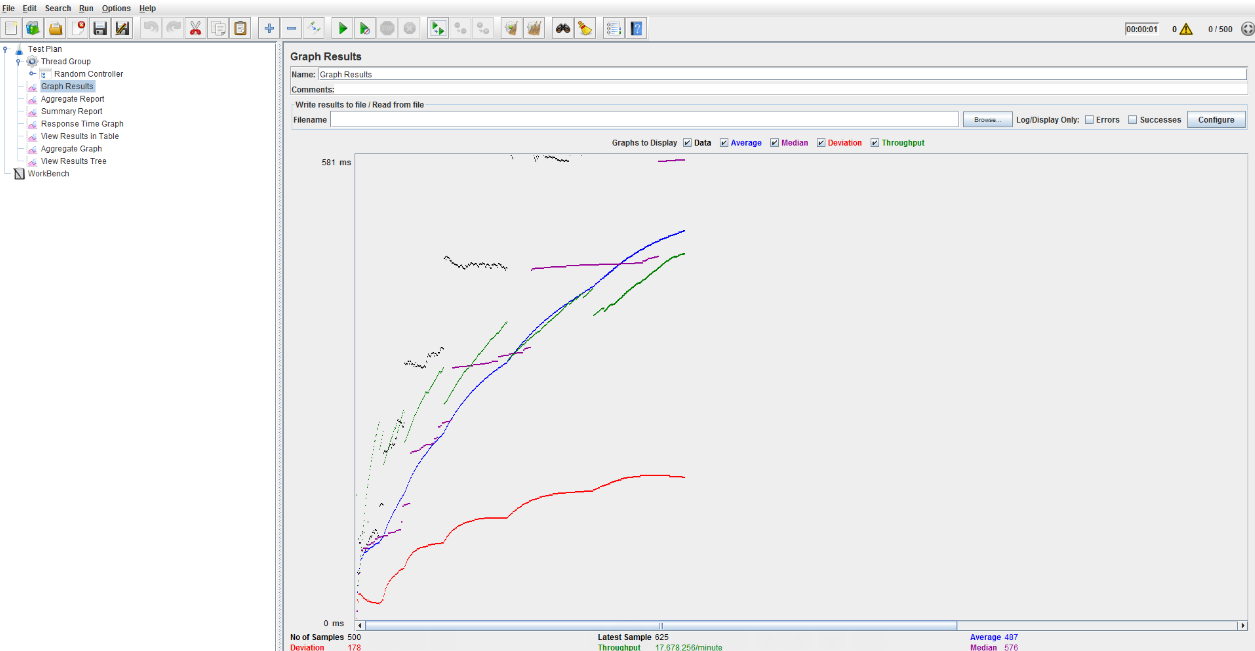
* 300 Concurrent Calls



* 400 Concurrent Calls

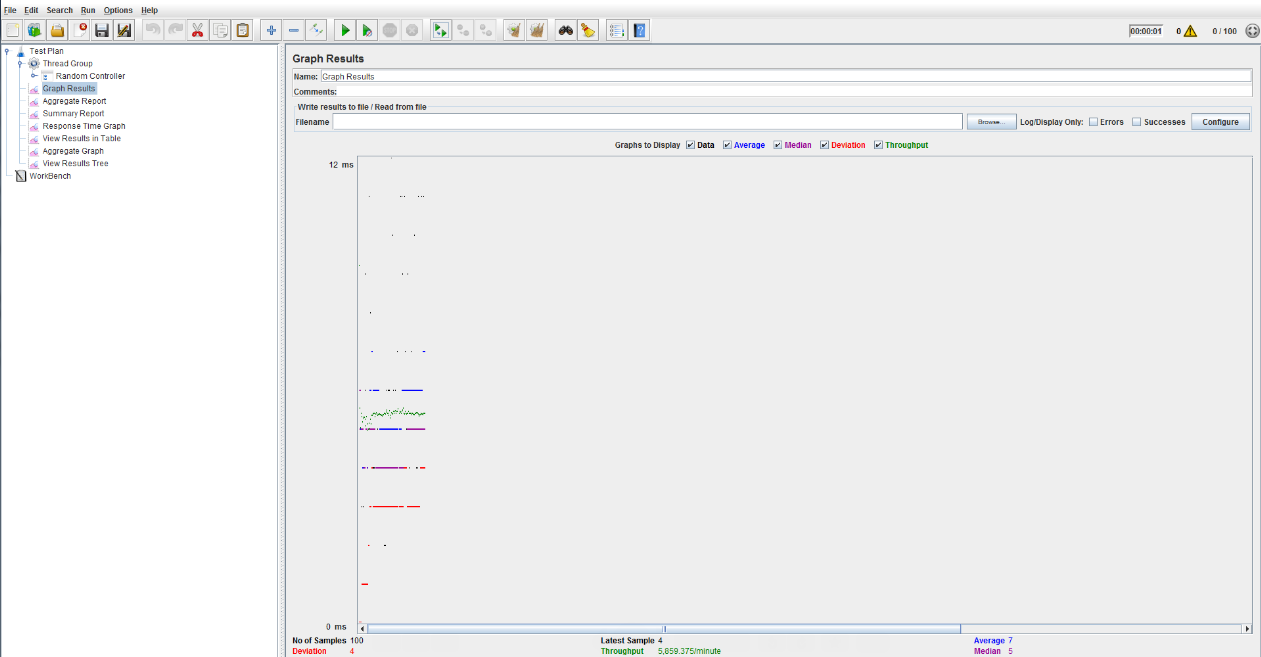


* 500 Concurrent Calls

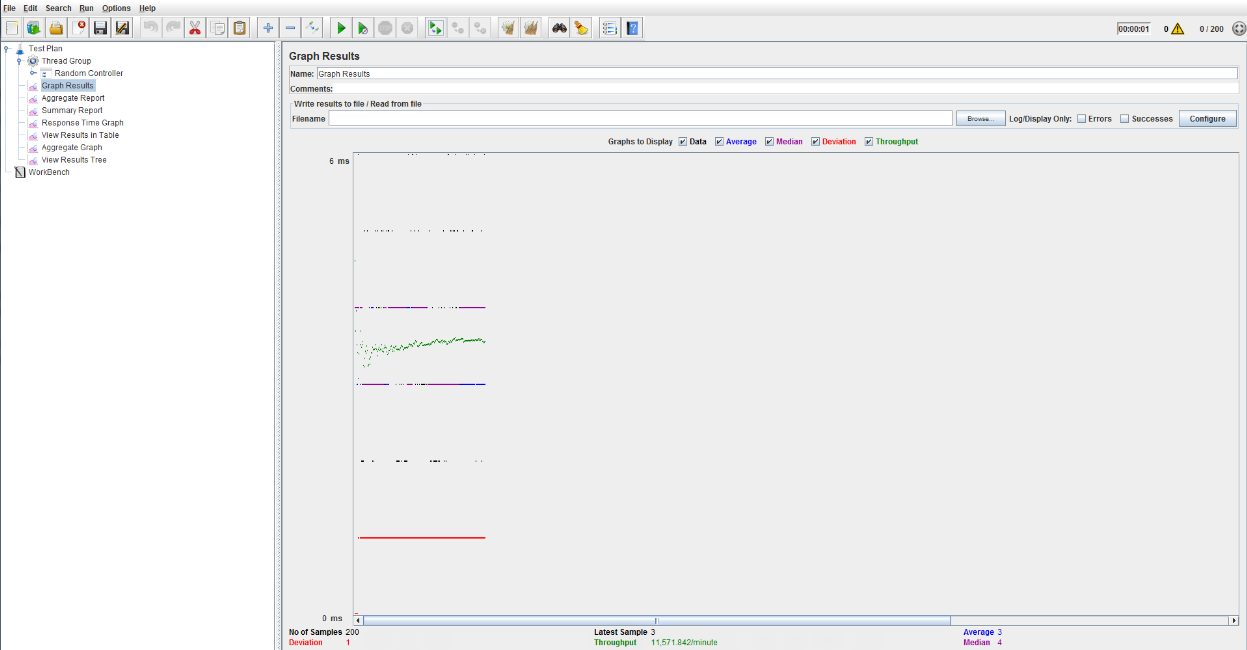


1. With RabbitMQ

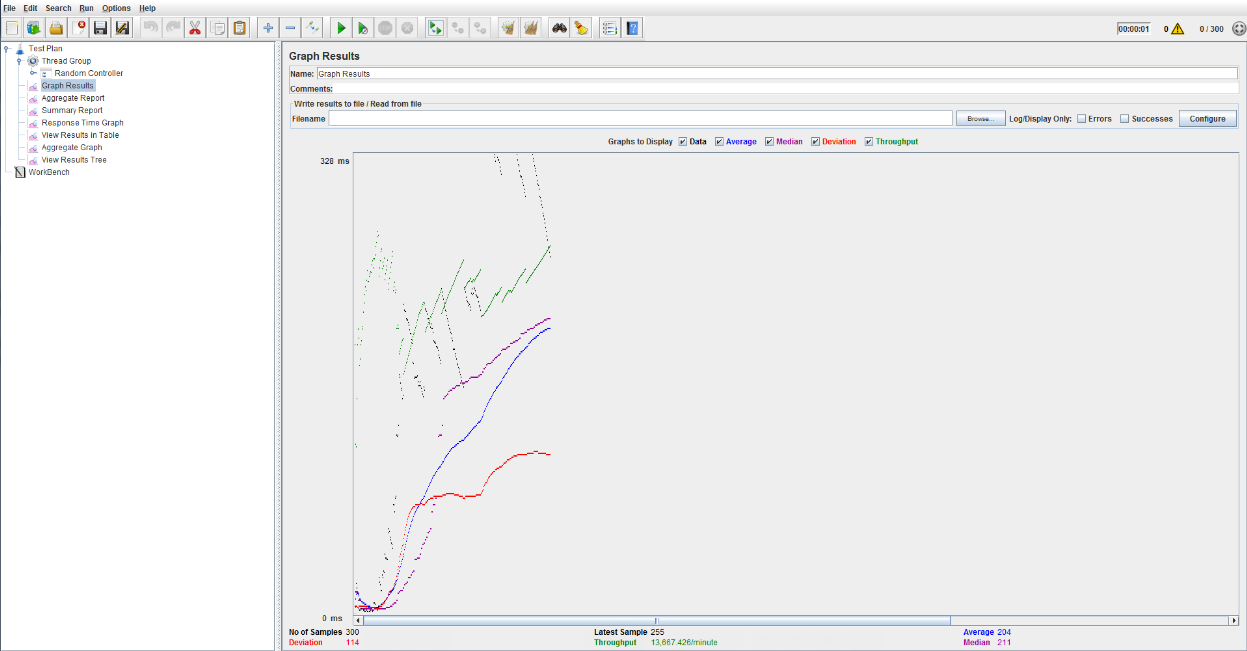
* 100 Concurrent Calls



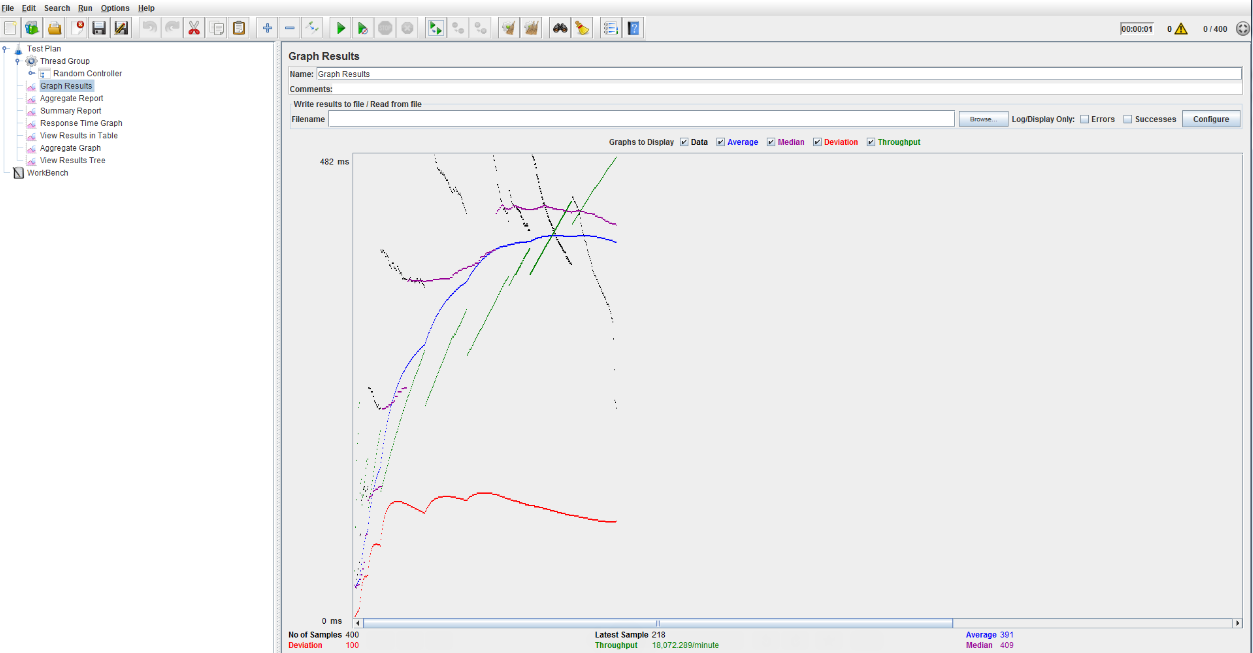
* 200 Concurrent Calls



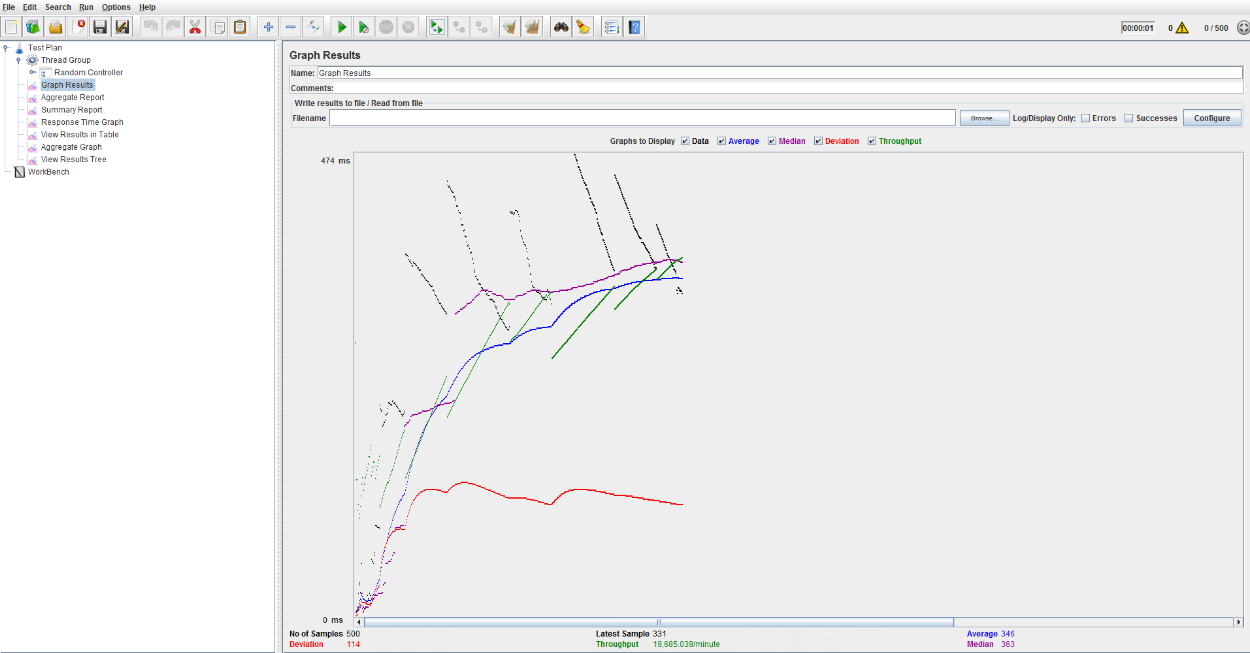
* 300 Concurrent Calls



* 400 Concurrent Calls



* 500 Concurrent Calls



Average Throughput Comparison(With RabbitMQ vs Without RabbitMQ)

Part 2 : Questions

1. Explain what performance change RabbitMQ provides? Elaborate on the results of throughput with and without using RabbitMQ. If you find any increase/decrease in the throughput, explain the reason for the same

Answer:

Average Throughput comparison chart for Performance with RabbitMQ and Performance without RabbitMQ is available above.

Y- Axis shows throughput per minute

X- Axis shows concurrent users.

RabbitMQ provides several advantages like reliability, Highly Available Queues etc to let you trade off performance.

As seen in the above chart if we use RabbitMQ then the throughput to process the requests increases than the throughput without using RabbitMQ.

RabbitMQ has a significant impact on performance/throughput of the system. For normal operation, there is not much difference in performance.

The requests are buffered heavily in message queue of the RabbitMQ. So the processing time for each request reduces and thus the throughput increases with RabbitMQ than what it was without RabbitMQ.

1. Compare passport authentication process with the authentication process used in Lab1.

Answer:

For Lab1 authentication process used was bcrypt.

For Lab2 we used local strategy of Passport JS.

Local Strategy of Passport JS lets you authenticate using “username” and “password” in your node js application.

Passport JS allows user to manage encryption in whichever way user wants.

Using passport JS we can modularize and organize entire authentication code.

We can use passport.authenticate() and specify which strategy to use. authenticate() function is used as standard connect middleware which can be used as route middleware in Express applications.

In Lab2 along with Passport JS I have used bcrypt encryption module to encrypt the passwords.

1. If given an option to implement MySQL and MongoDB both in your application, specify which data of the applications will you store in MongoDB and MySQL respectively.

Answer:

MongoDB database provides fast lookup as compared to MySql. Also MongoDB allows to create flexible Schemas.

If my application has to deal with data which needs flexible schema management and the size of the data is comparatively large for the processing, then I will use MongoDB over MYSQL.

One of the important disadvantage of Mongo is that there are no joins available like MySQL. So if my application has high transactional data then I will prefer using mysql database over MongoDB nosql database.

Mongoose package allows users to populate feature which doesn’t work like SQL joins but allows user to get required data as in joins.