**Lab 2 Assignment**

Nodejs/Express FrameWork/RabbitMQ/MongoDB/AngularJS

Hema Begur

**Abstract**

This lab assignment covers developing REST services using Node.js (Express), Angular JS, MongoDB and RabbitMQ.

**Facebook Prototype**

**Server / Client**

**Server**:

a) Basic **Users** functionalities:

1. Sign up new user (at least first name, last name, Email and password)

2. Sign in existing user

3. Sign out. Sign Up should have first name, last name, Email and password.

In order to use the system, a user must sign in first to the system.

b) **Users account** should provide basic details such as:

1. **About**: User overview, Work and education, contact info and life events.

**2. Friends list**: Send friend request, accept friend request.

3. **Interests** like music, shows and sports.

c) Provide **news feed** functionality.

d) Provide **Groups** functionalities:

1. Create group

2. Add member in a group

3. Show members in group

4. Delete member from a group

5. Delete group

e) Should perform **connection pooling** for database access.

**Client**:

Develop a client application. It should include Node.js as a server. The GUI should include AngularJS and Bootstrap. Make GUI as similar to the Facebook as possible.

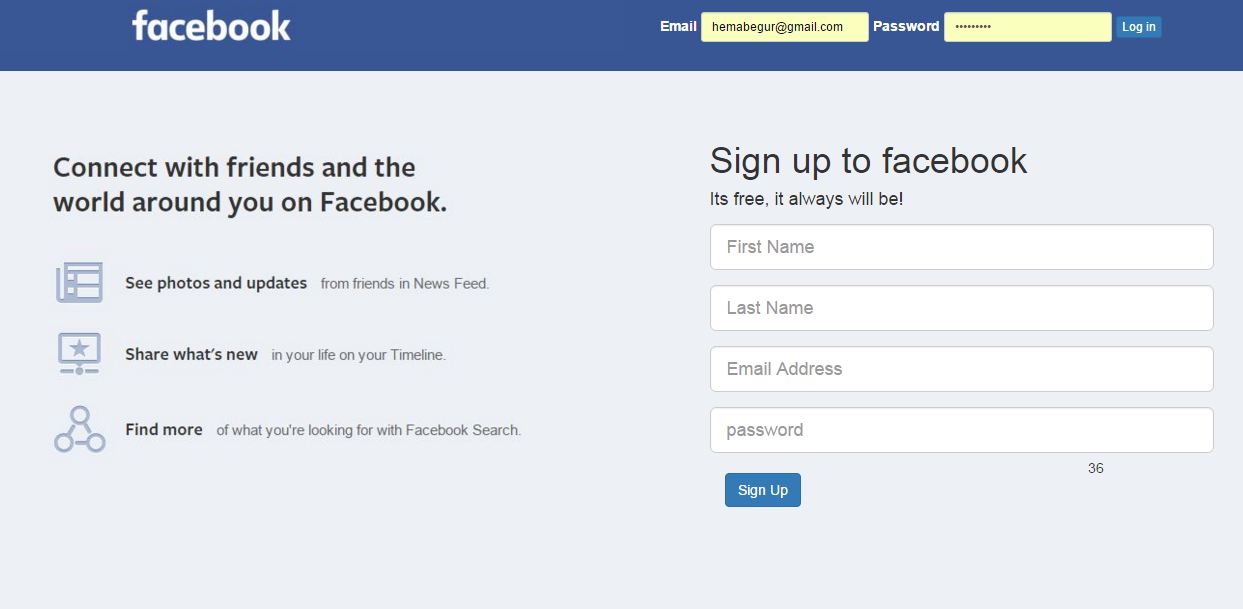
**System Design:** The system design for Facebook prototype is as follows:

* Front end developed using Angular JS, Bootstrap, HTML and CSS.
* **RAbbitMQ** is used for messaging to interact with server and client.
* **Node Js** is used as the server side technology. Express module is used.
* **MongoDB** – Document type database MongoDB is used to persist all the data as to achieve scalability.

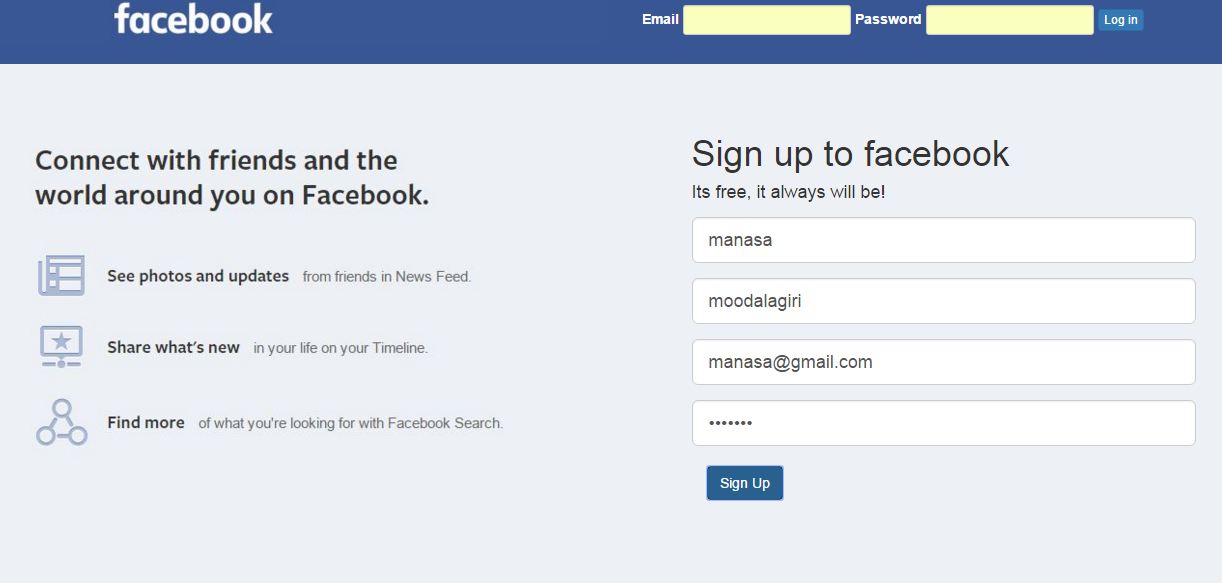
**Results**

**Login/ Sign Up Module/ Last Login Tracking**

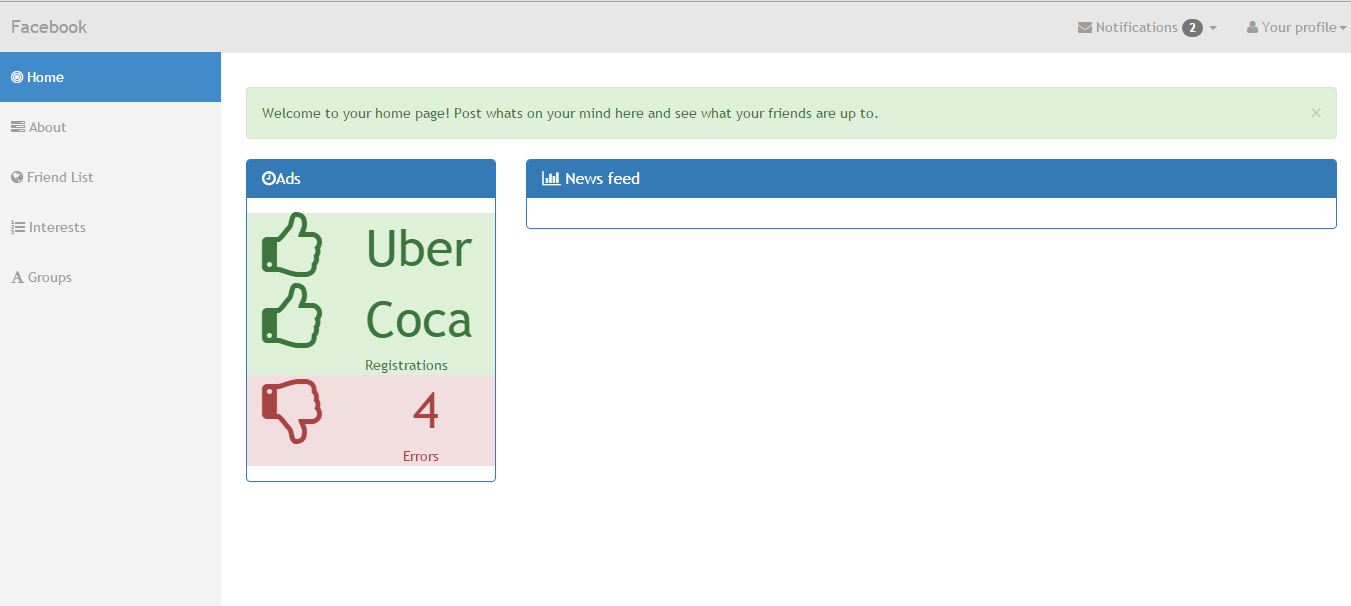
Login Page



**Signup page**

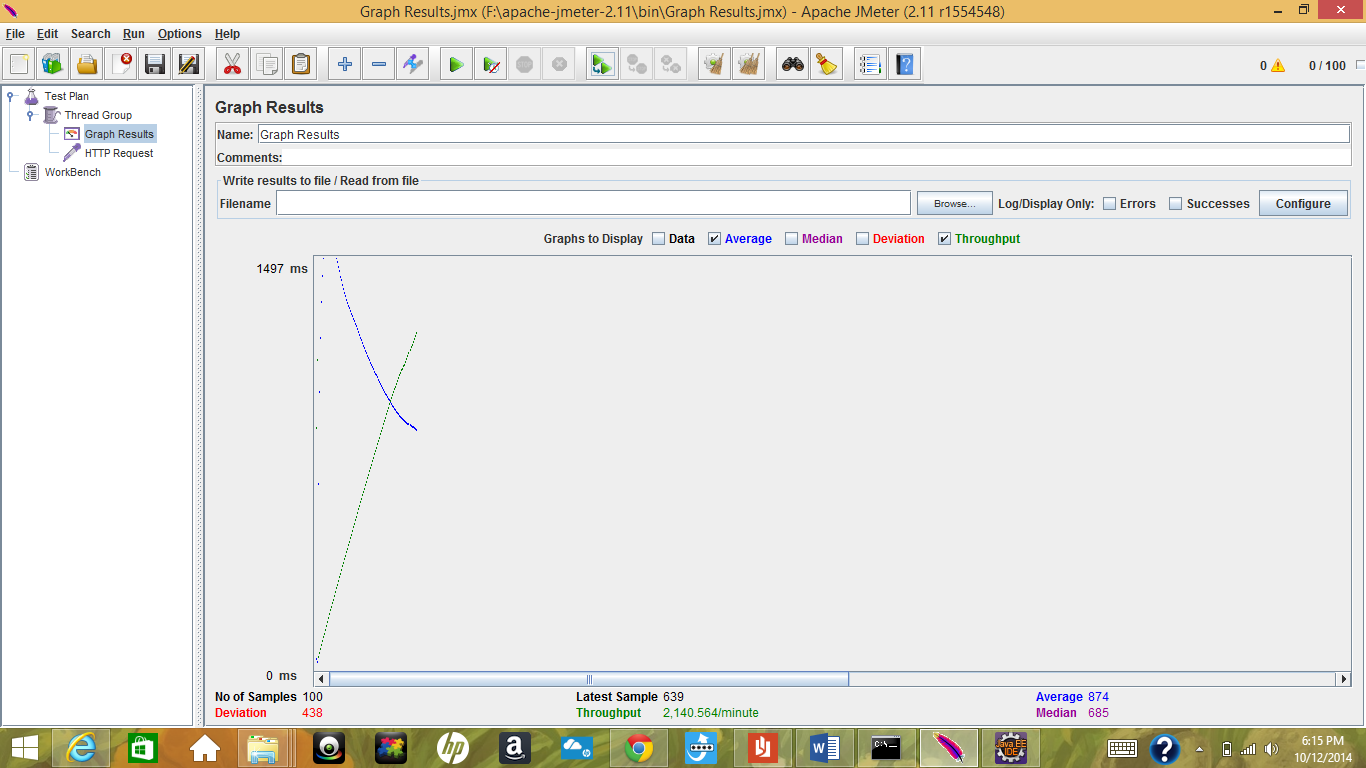
****

**Home page**

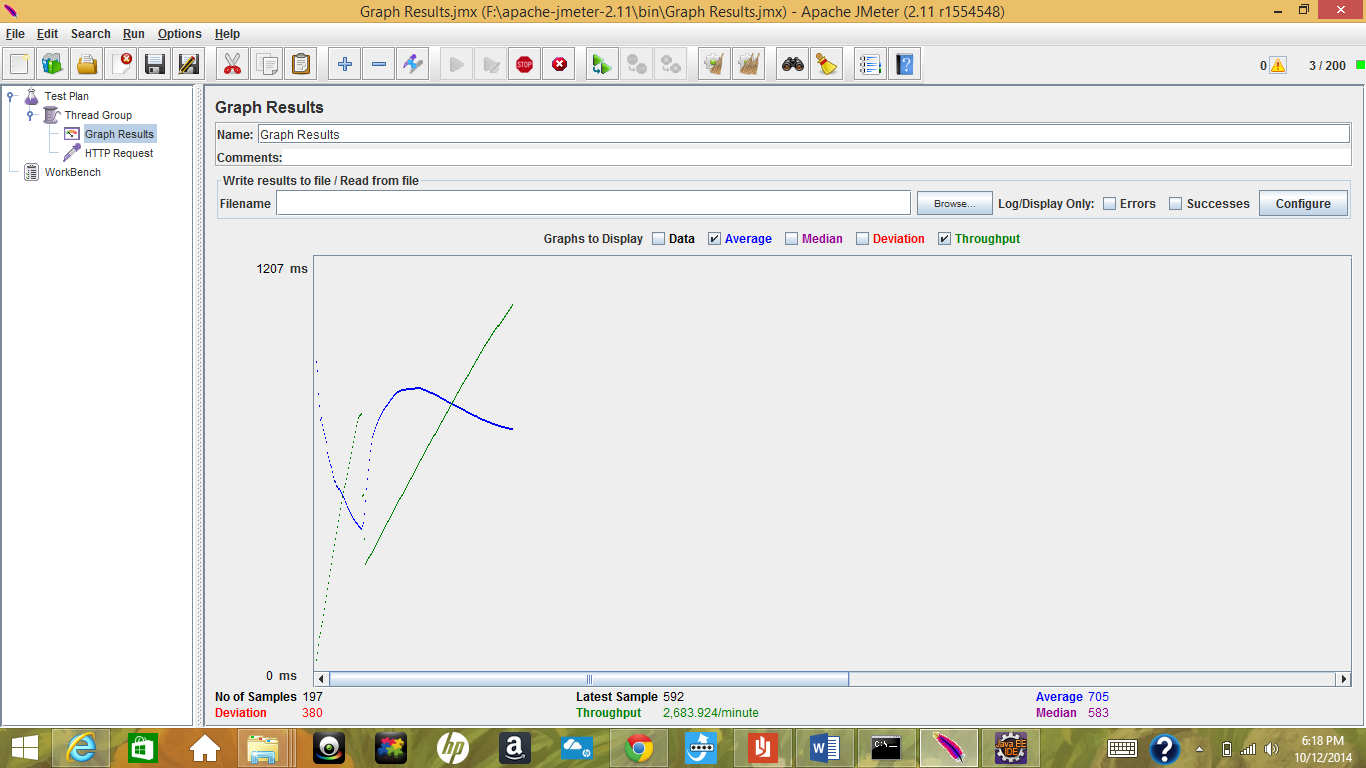


**Performance characteristics WITH & WITHOUT RabbitMQ**

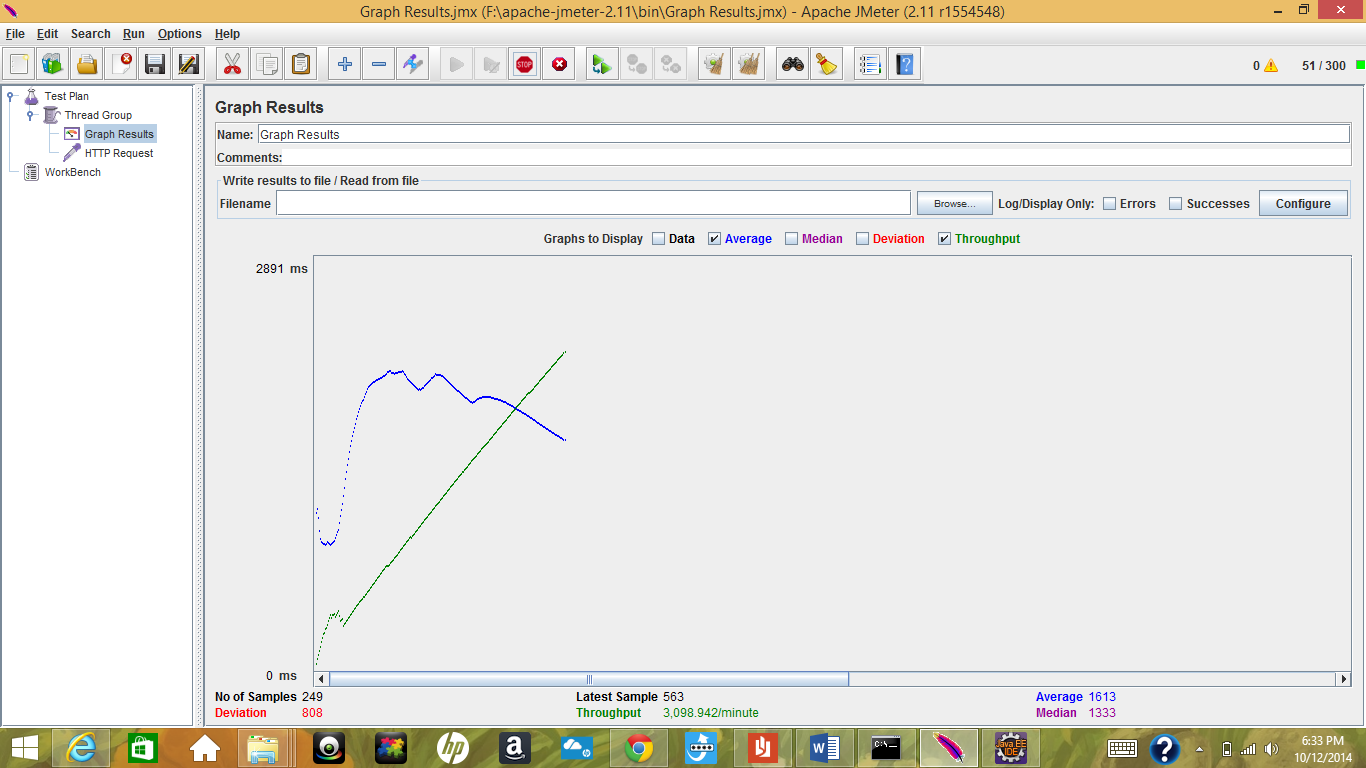
**Without RabbitMQ**

Users : 100

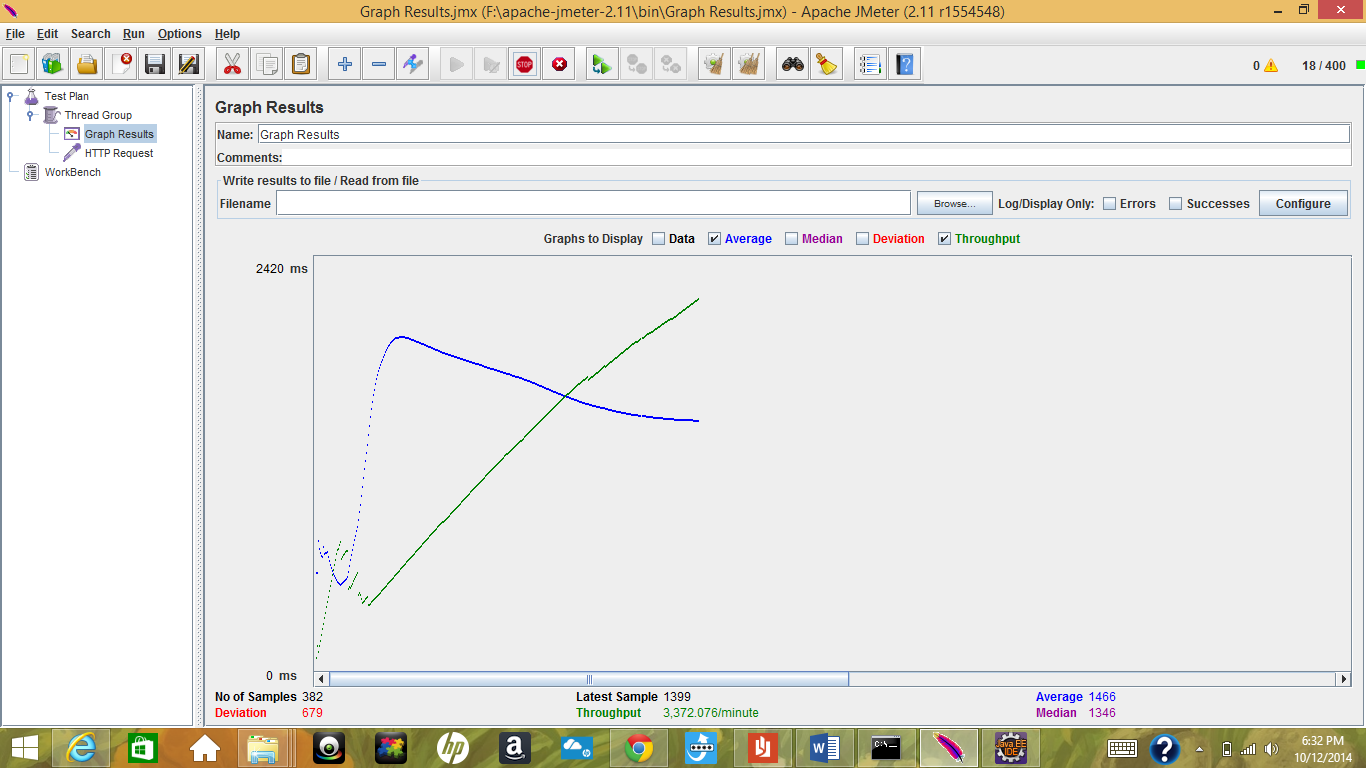
Users : 200



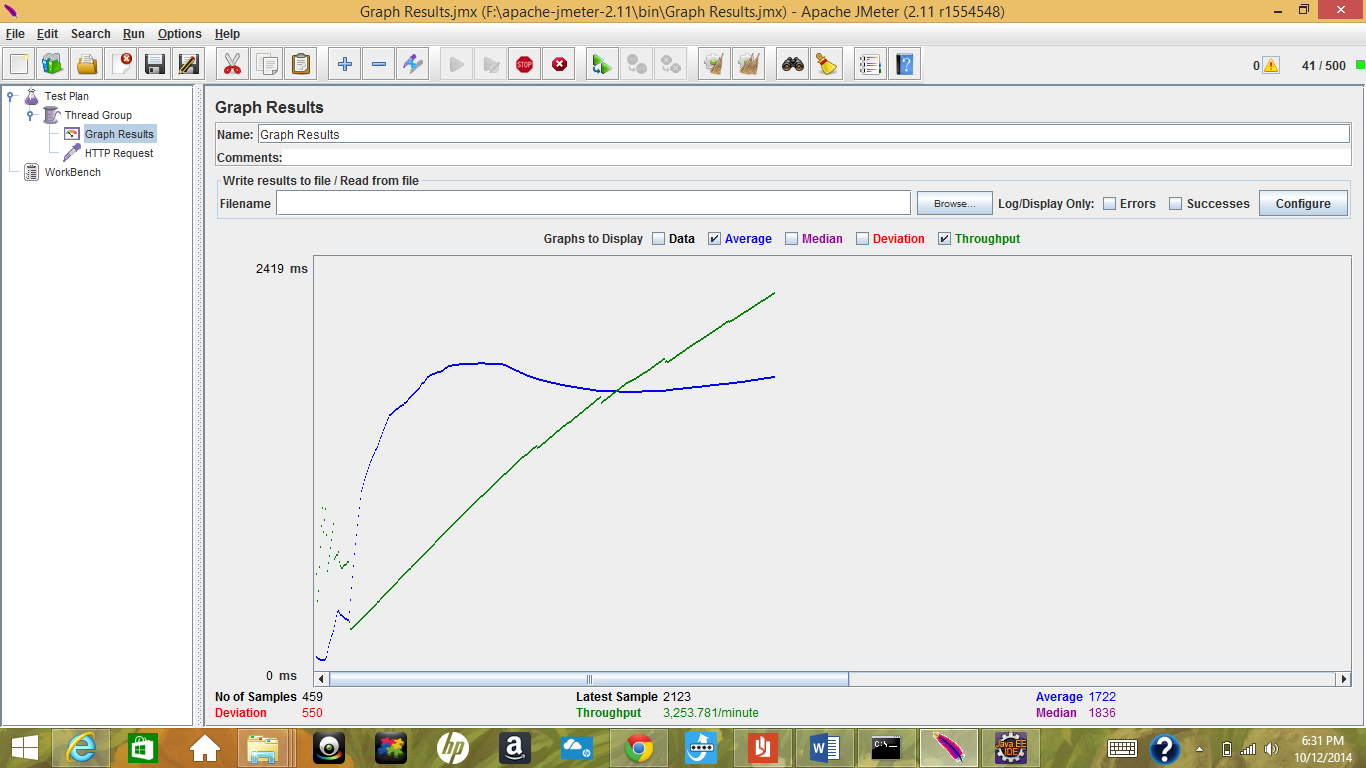
Users : 300



Users : 400

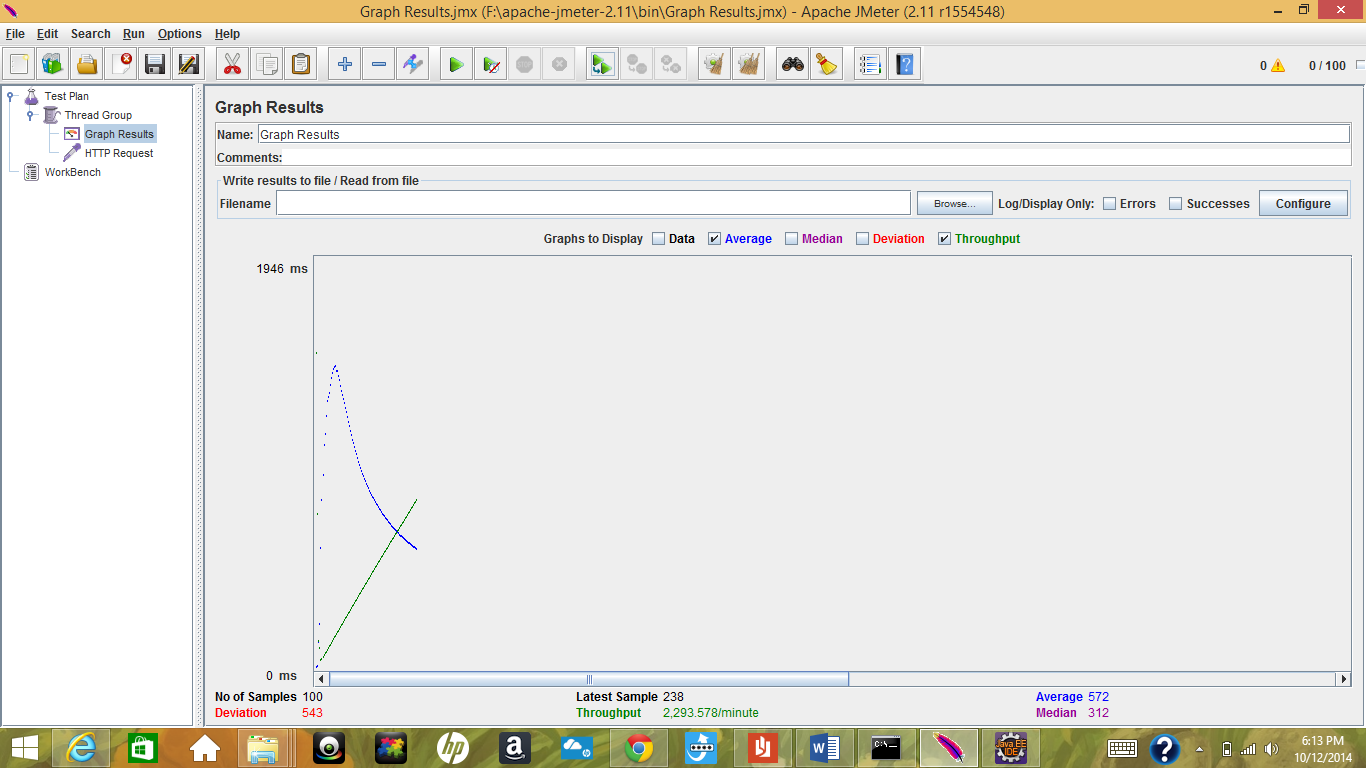


Users : 500

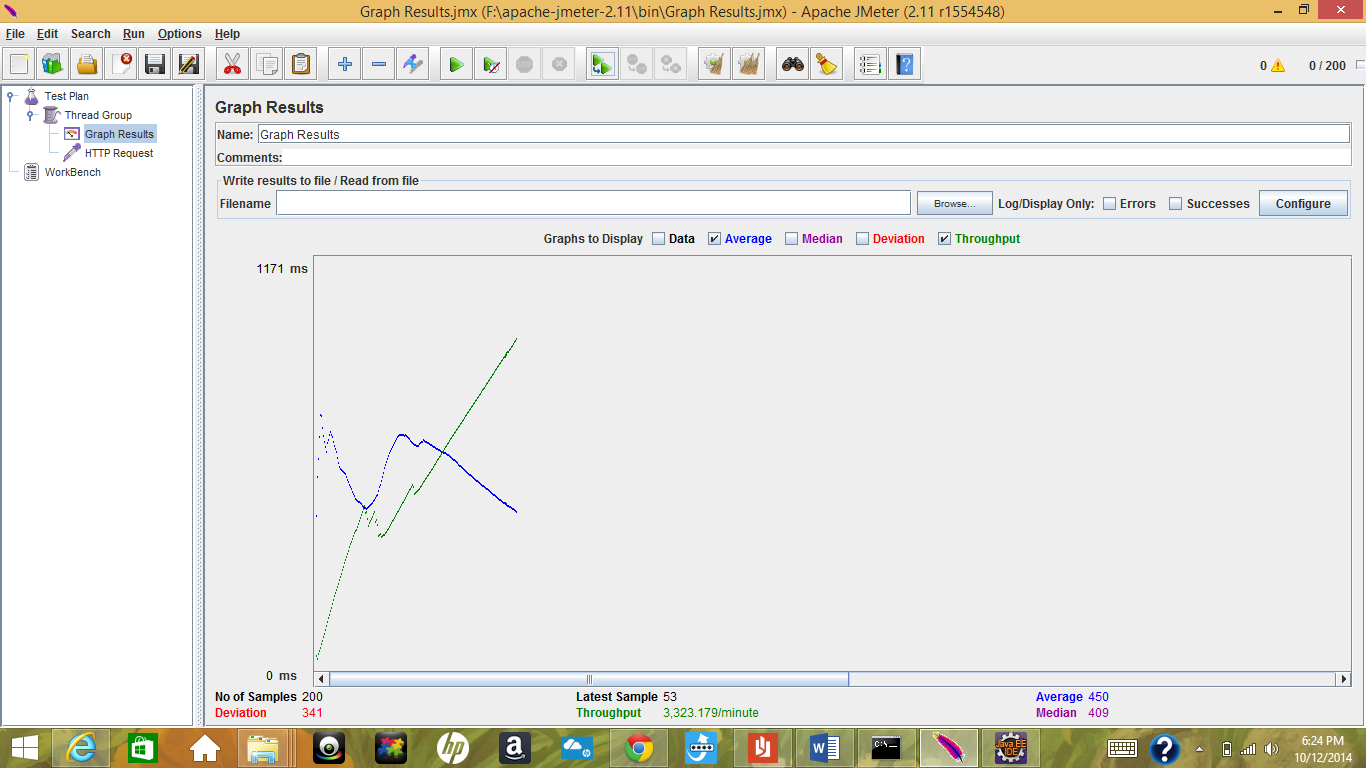


**With RabbitMQ**

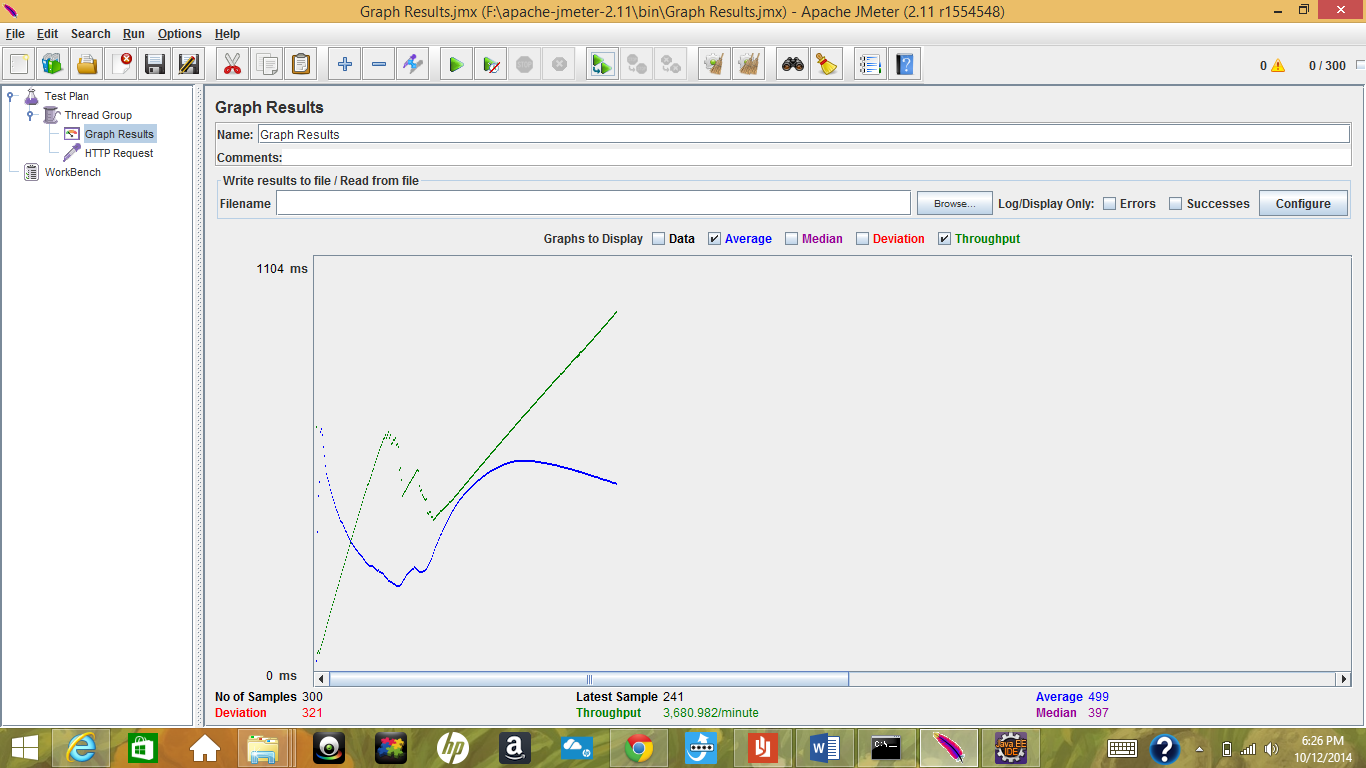
Users: 100



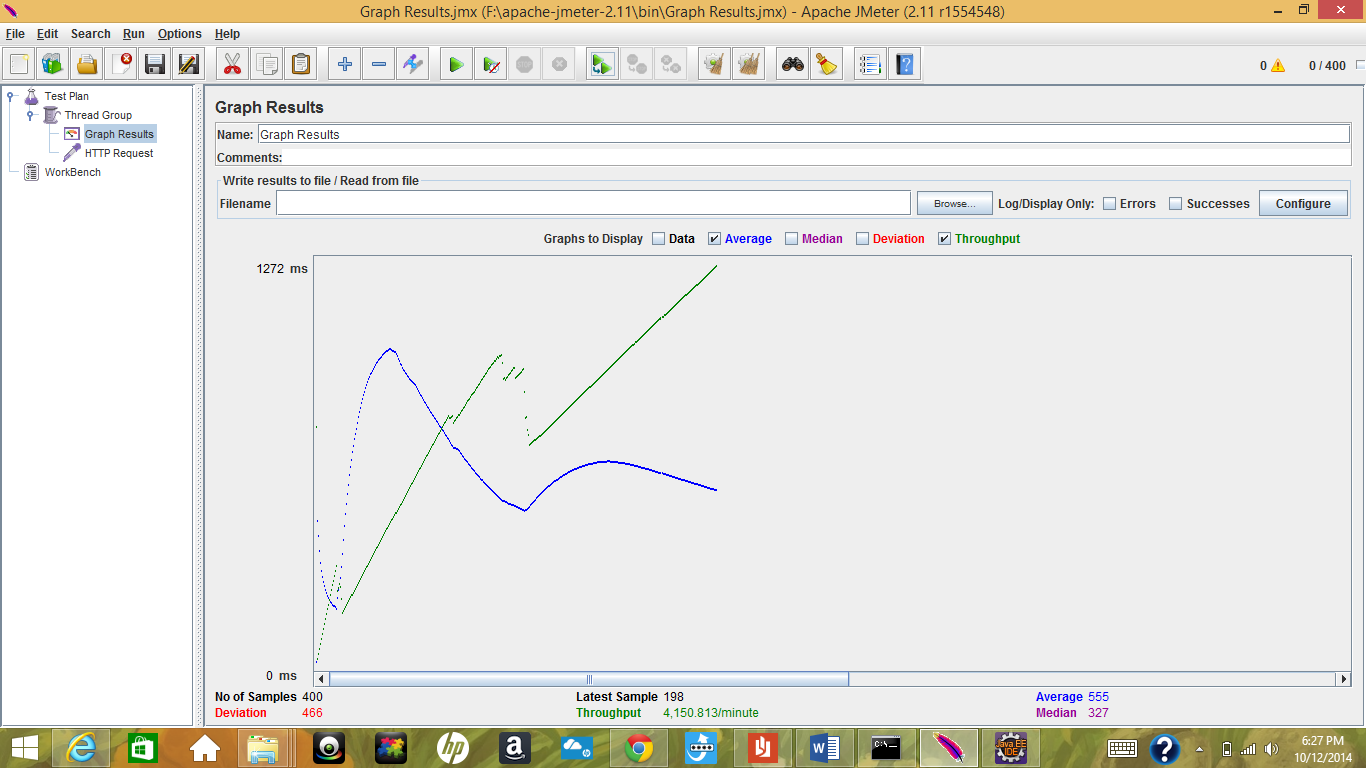
Users: 200



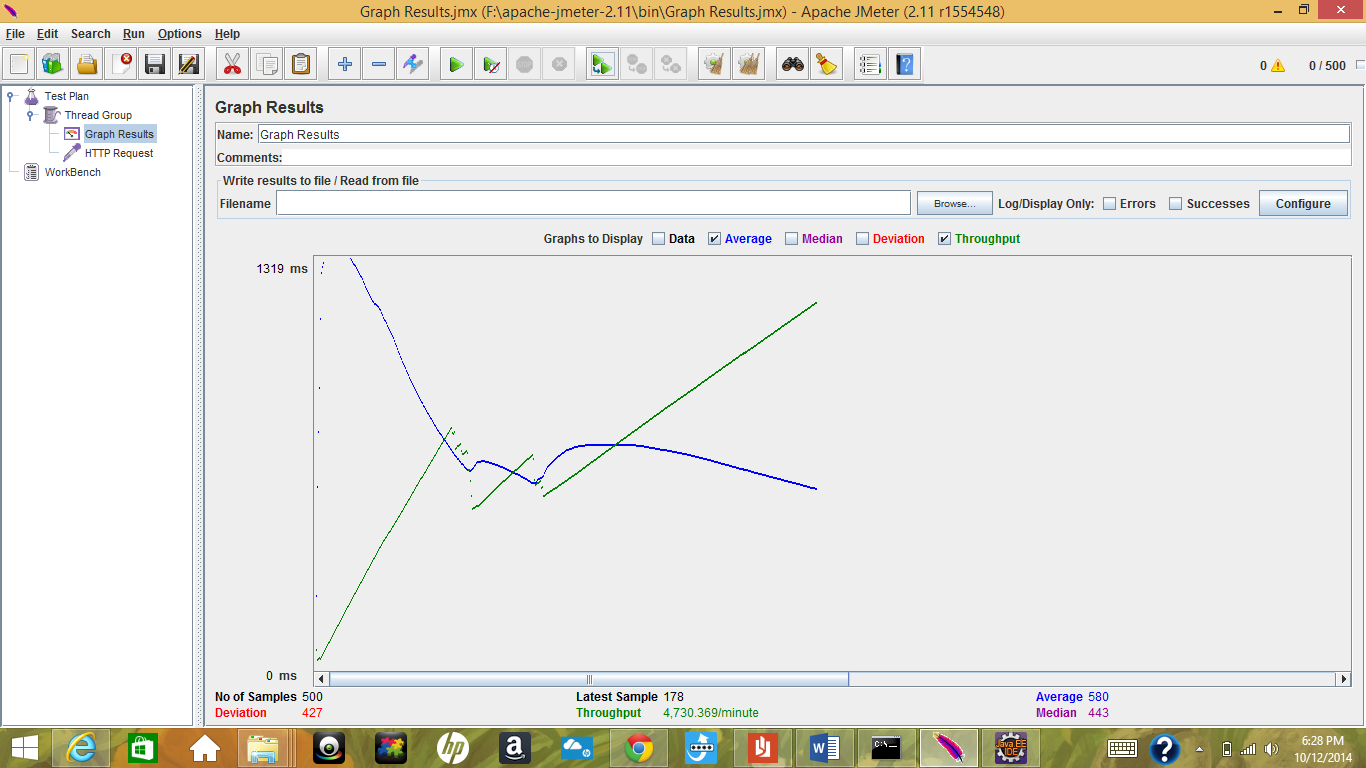
Users: 300



Users : 400



Users : 500



**Questions**

1. How to achieve horizontal scalability with single RabbitMQ Server? 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.

Using message queues to communicate between server and client gives the advantage of not burdening web servers. They allow you to create a separate machine pool for performing off-line processing. This allows you to target increases in resources to your current performance or throughput bottleneck rather than uniformly increasing resources across the bottleneck and non-bottleneck systems.

By being careful about distributing publishers, relative to the queues to which their messages are ultimately destined, we can to some extent avoid potential bottlenecks imposed by a single queue.

RabbitMQ can handle up to few thousands of messages per second, but when the system needed to be scaled up beyond this, simple load balancing could solve the problem of scaling up.

Throughput increases after using RabbitMQ messaging service.  When the queue is empty, when messages are ready to be received by consumers, tas soon as a message is received in the queue, it immediately goes to the consumer. In the case of a persistent it will also go to disk, but that's done in an asynchronous manner and is buffered heavily. The main point is that very little book-keeping needs to be done, very few data structures are modified, and very little additional memory needs allocating. This could be the reason for the increased throughput.

1. Explain the strategy used in implementing Sessions in this Lab. Compare your Sessions strategy with default express Sessions. Describe which Strategy is better.

It helps maintain session and the user data in the cookies and it is stored over every top level browsing context.

Session strategy used not handled as horizontally scalable. If more number of additional servers are added, then you need to store sessions across server .Before your application routes can use session data in a request, the data needs to be looked up. This is done by running each request through session middleware. Since sessions depend on cookies, the request first needs to run through cookie parsing middleware.

1. Describe the type of encryption algorithm used to store passwords in this lab and the reason to use it.

SHA-1 algorithm was used to encrypt passwords in the system. This was initially designed by NSA. SHA-1 produces a 160-[bit](https://en.wikipedia.org/wiki/Bit) hash value. A SHA-1 hash value is typically rendered as a [hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal) number, 40 digits long.

A hash function is simply an algorithm that takes a string of any length and reduces it to a unique fixed length string.

'test' => SHA-1 => 'a94a8fe6ccb194ba61c4c0875d391e987342fbbd3'   
only ‘test’ should produce the result a94a8fe6ccb194ba61c4c0875d391e987342fbbd3 being output.  
Also, every time that anyone anywhere runs the word 'test' through the SHA-1 function, they should always get: a94a8fe5ccb19ba61c4c0873d391e987982fbbd3.  
Finally, if the following pattern is given, 'a94a8fe5ccb19ba61c4c0873d391e987982fbbd3' and if SHA-1 was used, there should not be any way to figure out what was put into the function to create that.

SHA-1 is very simple to use. It is very difficult to produce collisions in SHA-1 compared to other algorithms.

It is also more robust and secure compared to other algorithms.