**San Jose State University**

**Fall 2017**

****

# Project Abstract

LAB 1 –Calculator/Dropbox

CMPE 273

SUBMITTED BY – SUBMITTED TO-

Siddharth Suthar(011439636) Prof. Simon Shim

Enterprise Distributed System Lab 1 Report

**Introduction**

**Goal :**

The goal of this assignment is to understand and implement the basic concepts of stateless connection , sql connection pooling and understanding the concepts of node js framework , jmeter , react js , redux ,password encryption and managing the security aspects of a fully fledged cyber attacks like sql injejction and cross site scripting.

To understand and implement this we have developed two systems.

**Purpose :**

To develop a working calculator app to learn the basic communication between React Js and node Js . And to understand the working concept of state change in Redux.

**System Design :**

Calculator , the architecture that I have made for this system is a simple single tier system with one front end react server that renders the Display components that we see in the browser and In the backend I have node server that serves my request from the react server and serves them by calculating the results.

I have created separate API’s for separate calculation calls , for example if one user is using the calculator for addition operation then the api generated would be “localhost:300/calculate/add” and simultaneously if another user is using the calculator for subtraction then the api call would be made to a different api “localhost:3000/calculate/sub” same way different api calls would be made for different operations , Basically this architecture helps in handling multiple concurrent users at a time , that reduces the average time to handle the request.

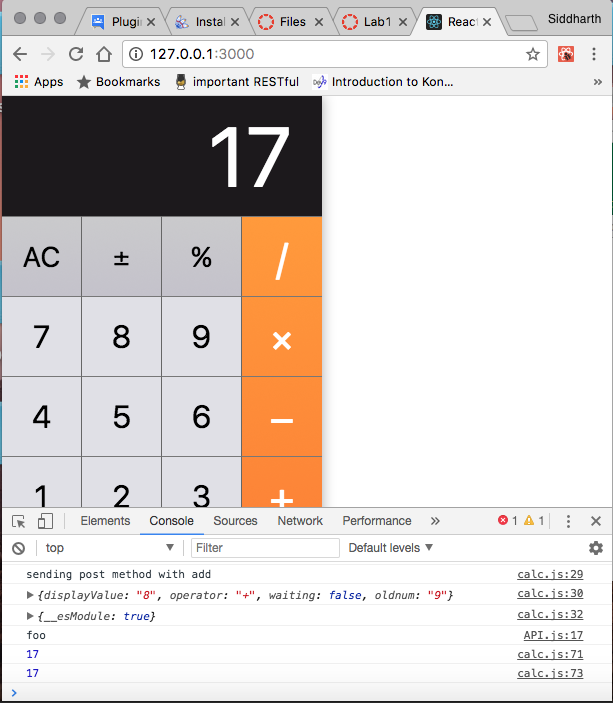
And on the front end part I have used react , that helps in minimizing the loading time , since it only re-renders the component that has been changed and does not re-render the entire application , for example when the answer comes after computation from the server only the display component changes , and not the entire calculator.

That is done by react internally , ie whenever any state changes of the application the component responsible re-renders. And to pass the state amongst all the components instead of passing it individually in the props I have used redux store, which creates a store for me to store all the states on it and I use the states stored in the store through reducers,

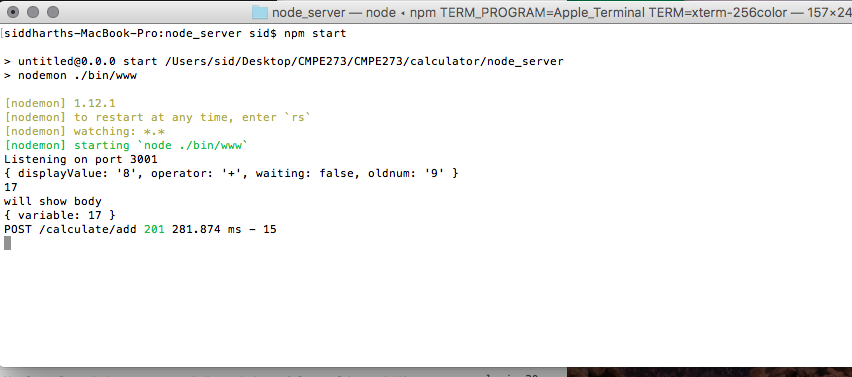
Front end Part (I have used the already created css file of a very famous React developer – micheal jackson) .

The calculator that I made handles the cases where even if the user does not press an equal to (‘=’) sign and presses 2+2+ , then also the request will be sent with two operators and an operand when the server computes and sends a response it is then displayed on the screen and the next operand is taken as + if the user doesnot enter a new operator.

The Frontend part of the calculator looks like as shown in the below image and , I have also included the console where the state of the operation is displayed.



When the request is sent to the backend server , it is received as a json string . and then the result is sent back as a json to the react server.

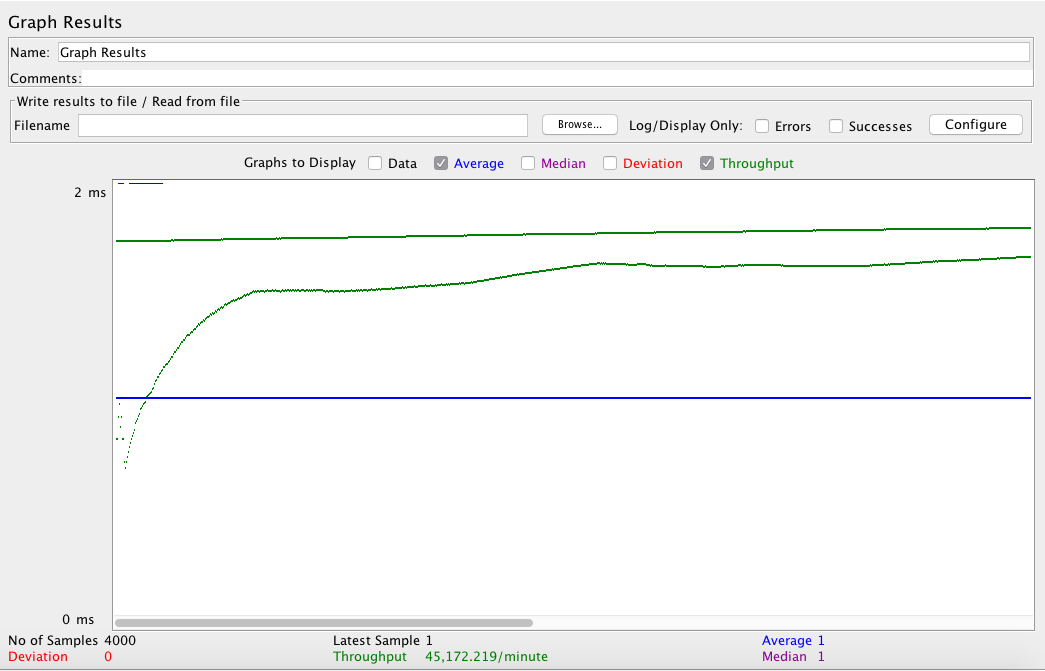


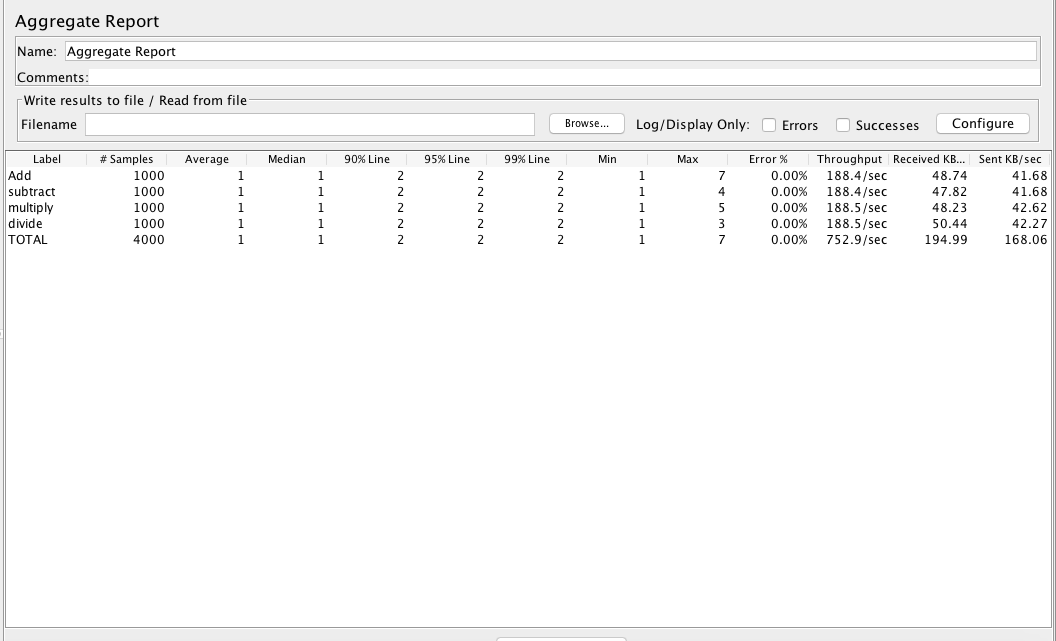
**Perfomance :**

For measuring the performance of the application I have used Apache Jmeter , I have created different thread groups to check for different number of users and different number of requests.

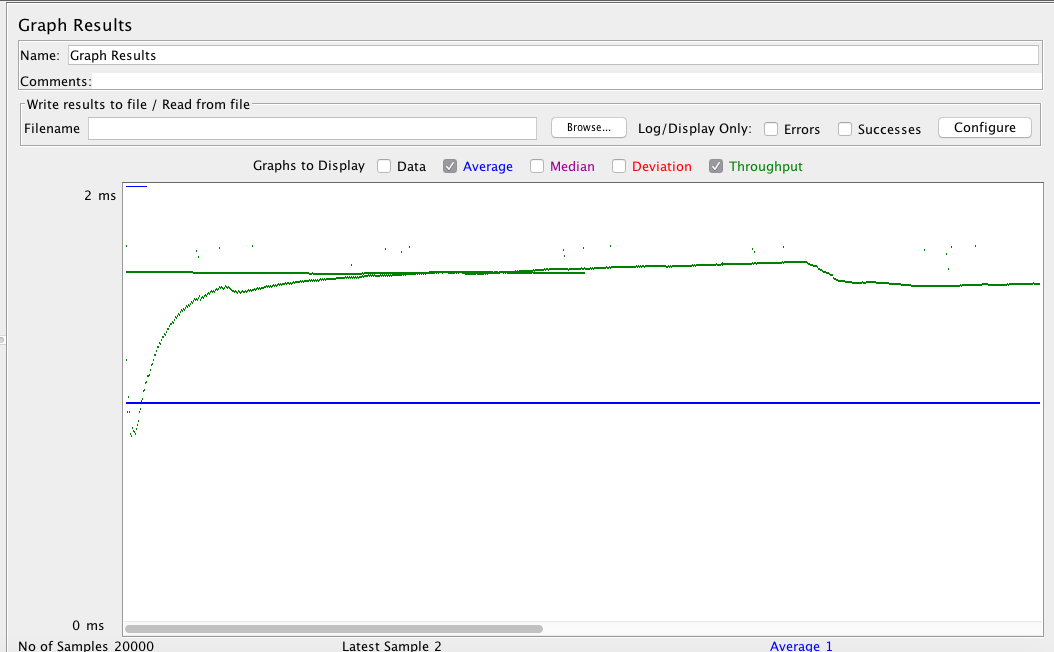
1. For 1000 requests the Jmeter output graph for random numbers generated is :
2. For 5000 requests the Jmeter output graph for random number generated is as shown in the graphs.

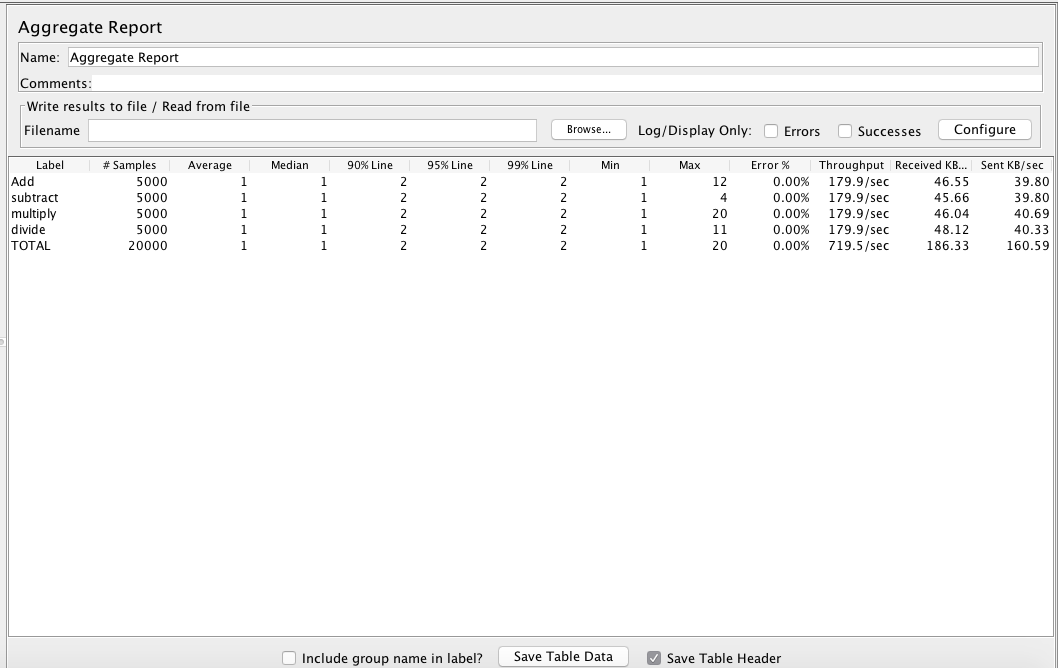
The Average time for both 1000 and 5000 consecutive requests is pretty decent with 1ms. As shown in the below graphs

****



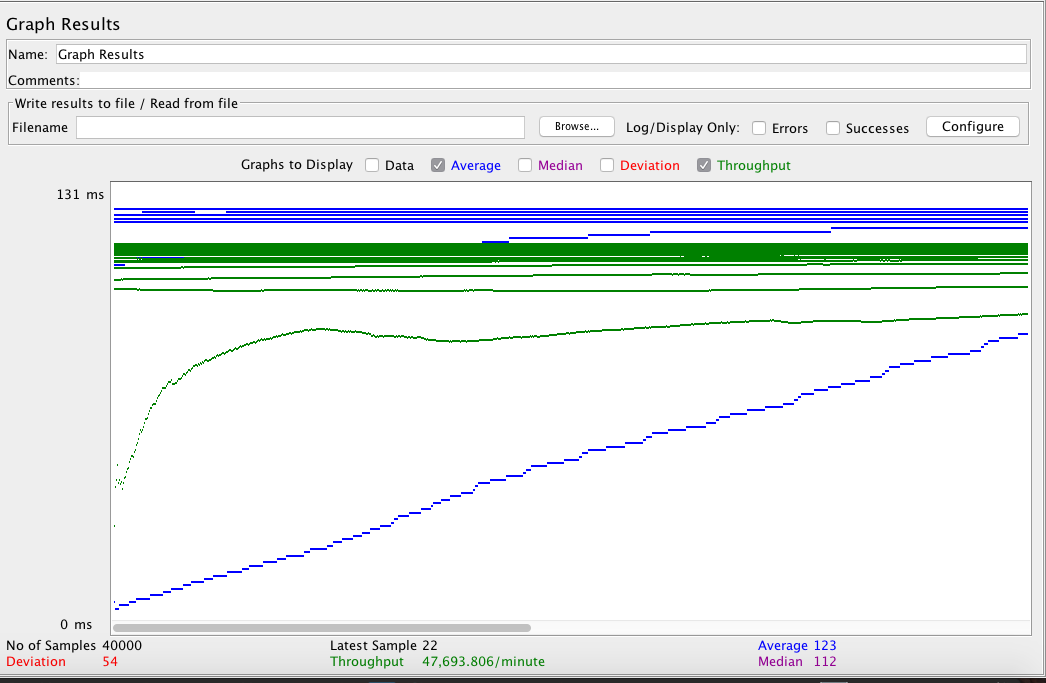
the above graph shows the average time for 1 user sending 1000 requests for random number’s ranging from 0-500 , and for random operations, and the aggregate reports.

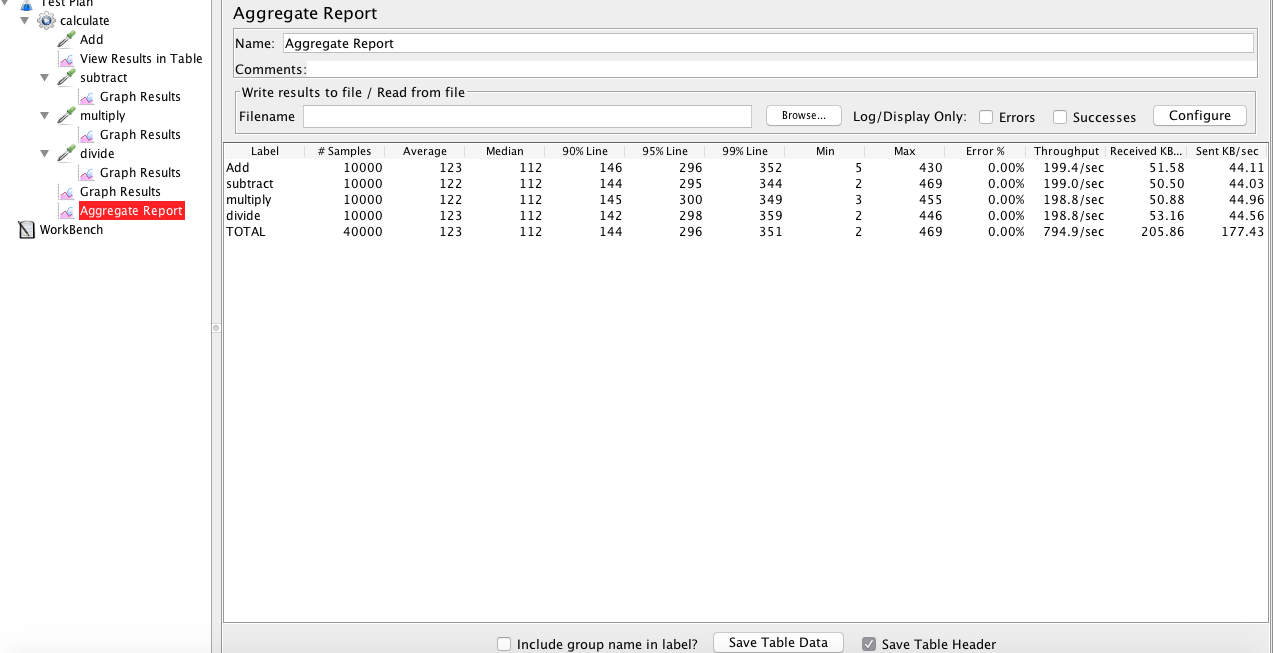




The result of jmeter testing for 5000 requests.

1. For 100 users using the application concurrently and sending 1000 requests for the operations , for that case the average time increases to 123 ms





**System 2**

**Purpose :**

To develop a dropbox like application , that does the basic functionality of the DROPBOX application , like creating users , uploading a file , downloading a file , creating folders and sharing the folders. Displaying user Info , user logs , maintaining the state , and encryption of Password using salt and hash.

**System Design :**

Dropbox , For this system I have one client server , one backend server and a Database . For serving the client side I have used a combination of react and redux , I have made several components in react for the displaying . and I have created a redux store that stores my states , I used redux to uncomplicate things by not passing the state and methods inside every component as a prop to avoid confusion and complications.

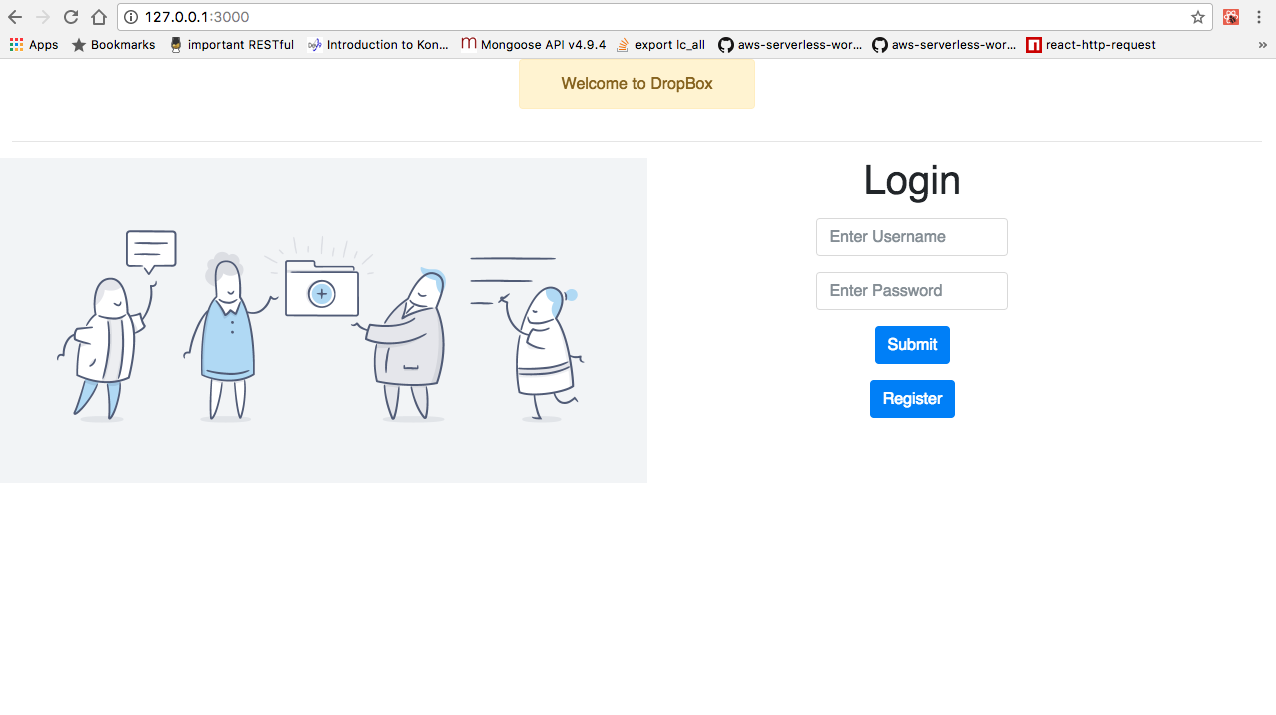
The system has separate API’s for handling different requests. The list of api’s are

1. Localhost:3001/users/login
2. Localhost:3001/users/logout
3. Localhost:3001/users/getAll
4. Localhost:3001/files/upload
5. Localhost:3001/files/share
6. Localhost:3001/files/delete

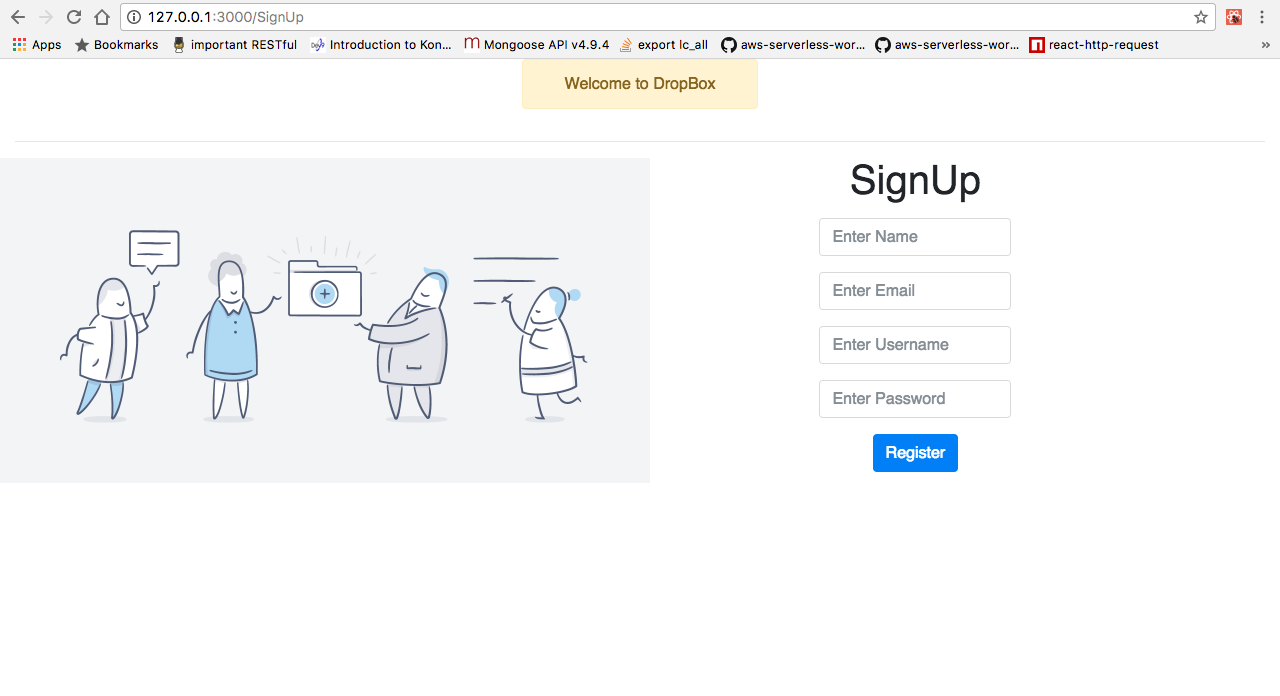
The system also handles the session management , for that express-session has been installed as the middleware.

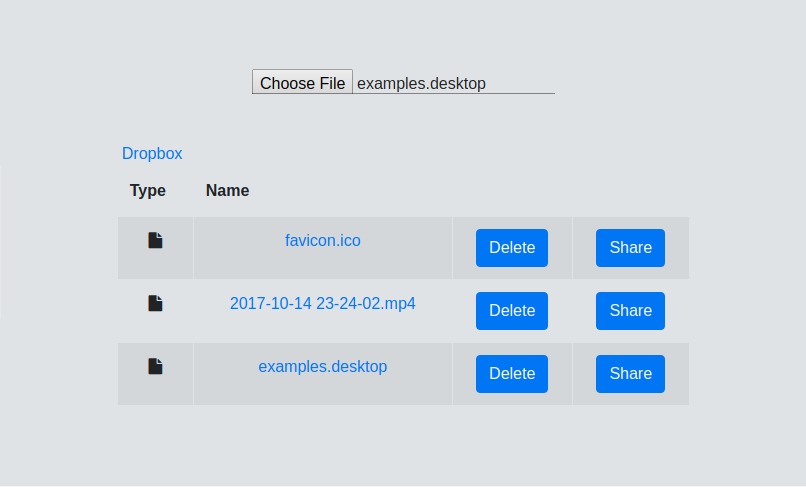
Attatched are the screen shots of the working application.

1. Sign In page –



when you click on Register a SignUp page appears.





by clicking on delete the selected file would be deleted , Deleting the folder functionality has not been implemented , and by clicking the share button , you can choose the person you want to share the file with.

**Performance :**

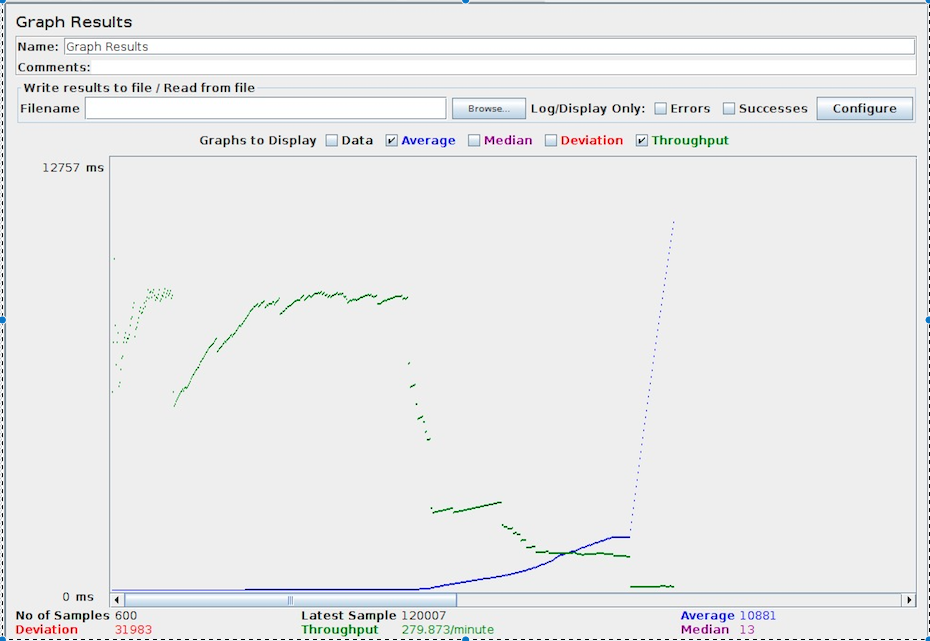
The Performance of the system was pretty decent considering the amount of concurrent requests it could handle , mysql connection pooling played a major role in decreasing the average time of the serving the request , I used systems default pool that created 150 threads ready to be used.

Below are the graphs that were created by testing the api’s with Apache Jmeter

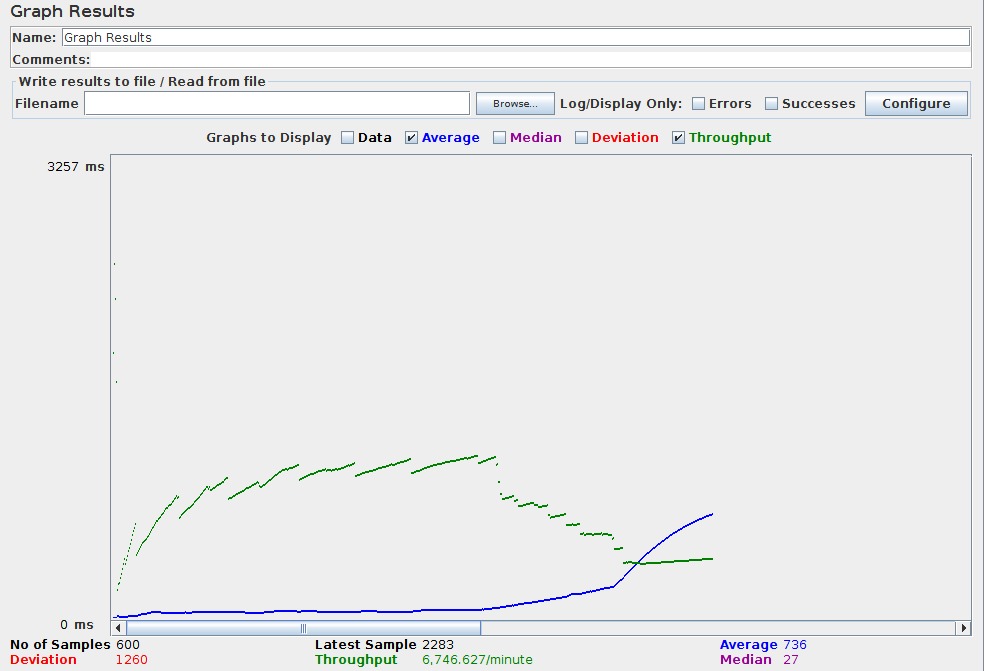
1. 100 concurrent users – connection pooling



non connection pooling –



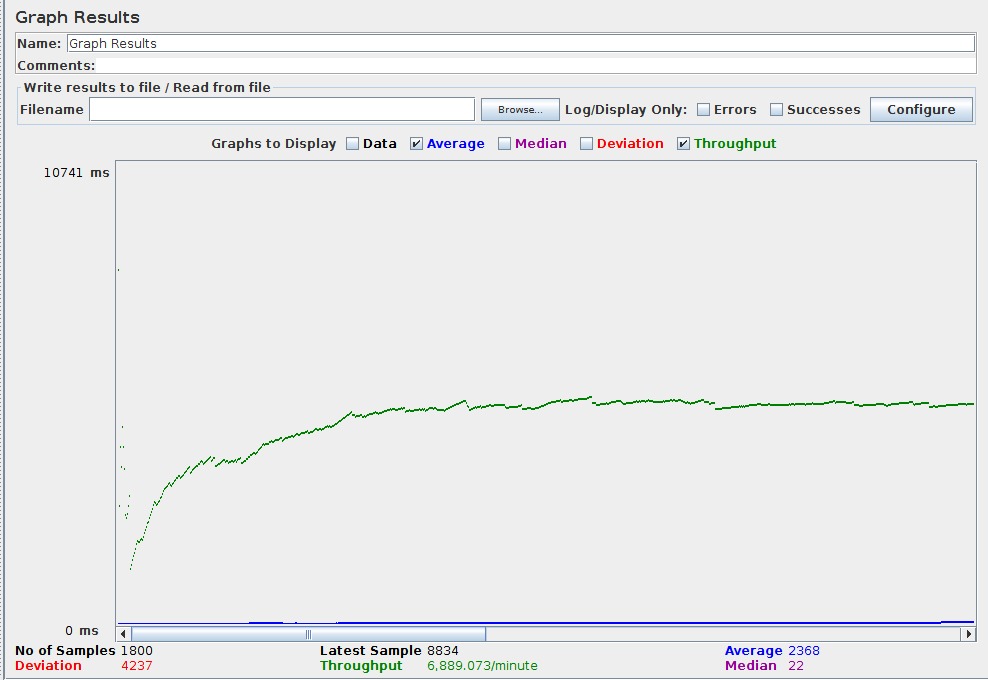
1. 200 concurrent users – connection pooling



non connection pooling :



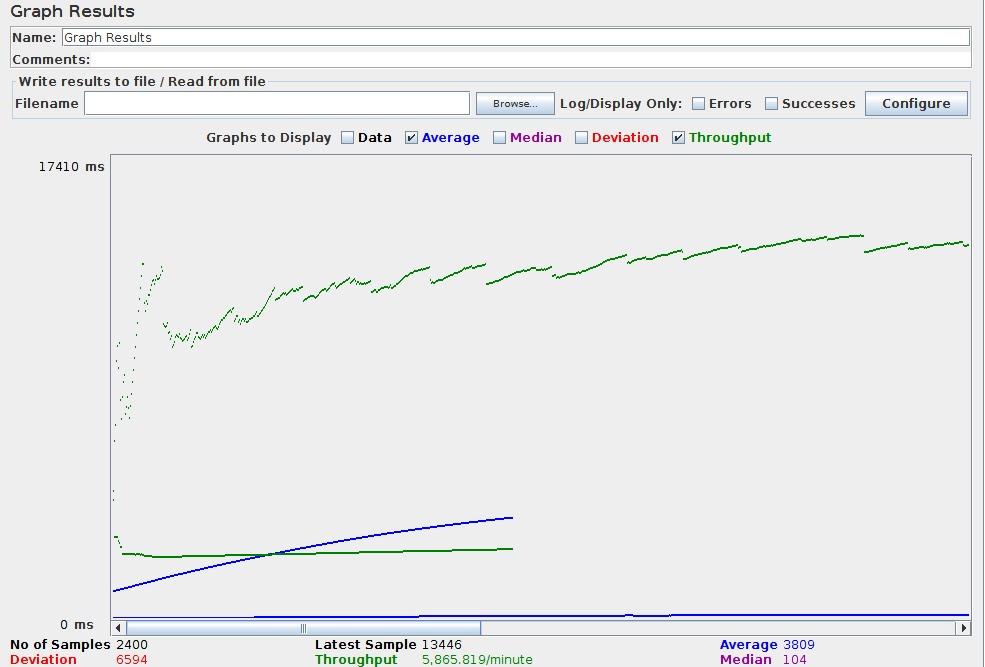
1. 300 concurrent users – connection pooling



non connection pooling –

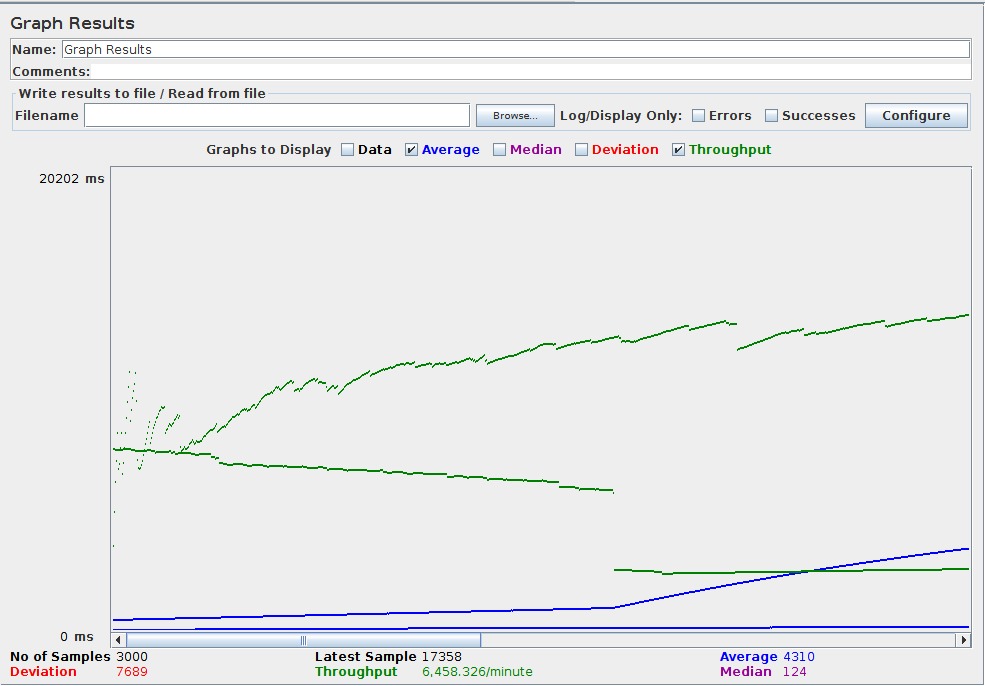


4) 400 concurrent users – connection pooling :

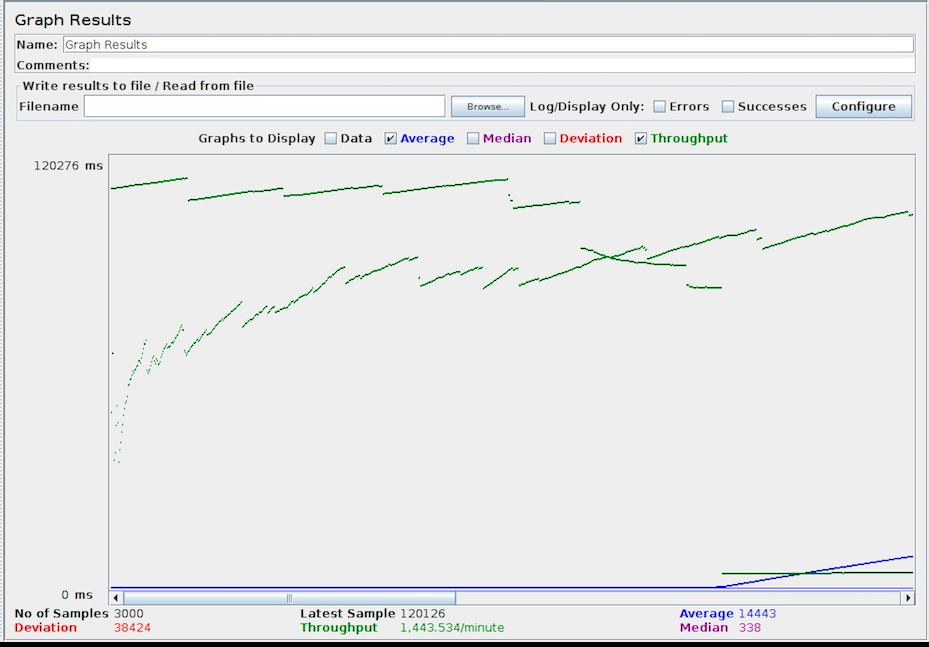


non connection pooling :



5)500 concurrent users – connection pooling

non connection pooling



As you can see from the above graphs and that the average time required when the database connection is connection pooling is signigicantly less than when the connection pooling is disabled. This gives us quite an overview of the fact that how much amount of time is being wasted waiting for the connection and waiting for I/O

Questions and Answers :

Question 1 : Explain the encryption algorithm used in your application. Mention different encryption algorithms available and the reason for your selection of the algorithm used.

Answer : I have used Salting and hashing for my Password encryption for this project. What I have done basically is when the user enters the password the password comes in the body of the request from the client side and then after receiving the password in my request package and then after unpacking it ,the first thing that I have done is I have added “salt” to the password to avoid if from the dictionary attacks , adding salt is basically adding a random string at the front and back of the password to help prevent dictionary attacks.

After that I have then hashed the new generated string to prevent it from the brute force attack and then I have stored it in the data base. To achieve so I have used the predefined functions in mysql 4.1 and later(line Encode() , compress() ,AESdecrypt()) . these functions did all the heavy lifting for me. All I had to do was use them in a proper way.

Also another security measure that I have taken in this project is I have used certain ways to prevent SQL injection that is when the SQL query is generated I did not directly added the parameters , I did some basic validations on the username and password . ie I tried avoiding any special characters in the username and no empty spaces .

And for cross site scripting . to prevent cross site scripting I am not parsing the query string to get the data instead I have passed my data in the body of the request.

Also I have not kept any untrusted jquery links on my script tags in html .

Other different types of excryption algorithms are

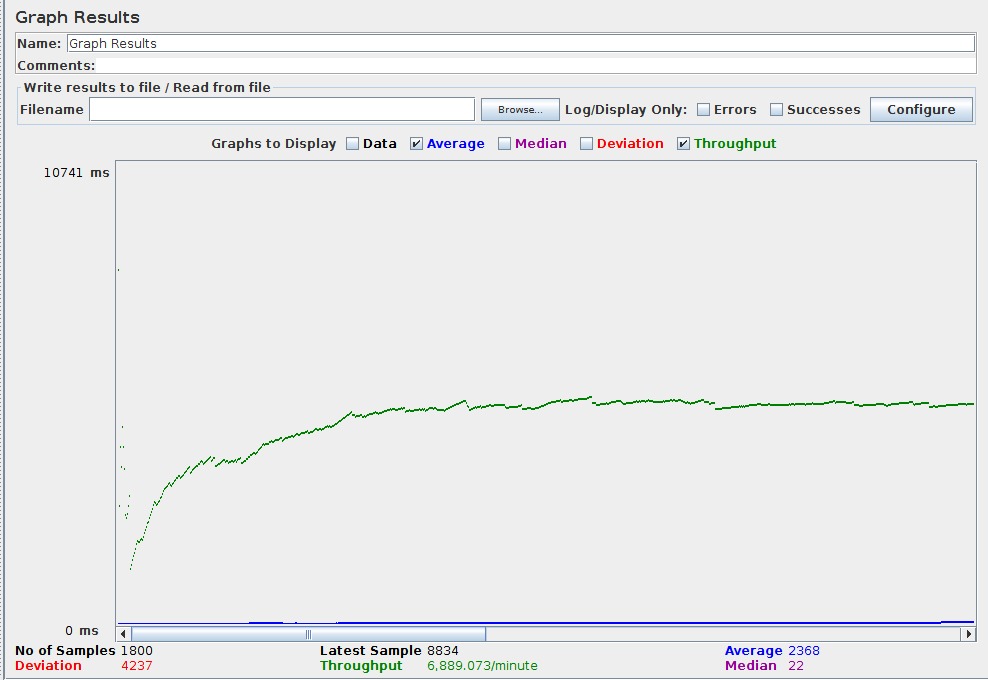
* AES
* Twofish
* IDEA
* MD5
* SHA 1
* HMAC.

Question 2 : Compare the results of graphs with and without connection pooling of database. Explain the result in detail and describe the connection pooling algorithm used in your code

Answer : I have used the predefined function of mysql , ie pool instead of creating my own algorithm I have used the predefined pool available from SQL which creates 150 connections for me , I have just used that . so instead of creating a new connection , the request would just use the already created threads. And using connection pooling significantly reduces the amount of time it takes to serve a request , as you might have seen from the screen shots above that without the connection pooling the amount of time required is almost more that 10 times , so using connection pooling can save a lot of time and can handle more number of requests per second and the throughput increases.

For example consider the case of handling 300 concurrent users . with connection pooling the average time is 2368ms

This further proves my point that connection pooling saves significant amount of time



but for the same case the average time required without connection pooling is



Question 3 : What is SQL caching? What all types of SQL caching is available and which suits your code the most. You don’t need to implement the caching, write pseudo code or explain in detail.

Answer : SQL Server has a pool of memory that is used to store both execution plans and data buffers. The part of the memory pool that is used to store execution plans is referred to as the procedure cache.

So basically we store outputs of some of the very common sql queries inside the caching pool so we wont have to wait for the database to give us the results to a lot of time is saved for fetching the data from the databse .

There are various different types of caching

1. Adhoc query caching
2. Autoparameterization
3. Prepared queries
4. Stored procedures.

The caching that suits my need the most is Autoparameterization , that is because the query that gets used the most for my project is getting the list of files , so the query is same just except for the where part where username is there .

I would create procedures .

EXEC sp\_executesql N'SELECT Filepath

FROM File\_Table

WHERE username = @p', N'@p tinyvar', “username”

EXEC sp\_executesql N'SELECT Filepath

FROM File\_Table

WHERE username = @p', N'@p tinyvar', “username2”

EXEC sp\_executesql N'SELECT Filepath

FROM File\_Table

WHERE username = @p', N'@p tinyvar', “username3”

Question 4 : Is your session strategy horizontally scalable? If **YES**, explain your session handling strategy. If **NO**, then explain how can you achieve it.

Answer : No the application is not horizontally scalable right now but to make it horizontally scalable , I can add my session data to a common database or a redis database that is shared amongst the different servers and every time a request comes the session can be checked from the database , and to further help the cause I can enable “stickiness” on my loadbalancer which balances my load , so that the consecutive requests can go to the same server.   
since my application is already a stateless application scaling and caching becomes easier than the statefull application.

Further more the scalability can be achieved using a proper load balancer that would be smart enough to route the requests to the appropriate server.