CMPE 273 – Lab2

Canvas Prototype with MongoDB and Kafka.

Student ID: **013859989** **Name:** Laxmikant Bhaskar Pandhare

Goals of the above Applications:

The goal of this lab is to develop a Canvas Application wherein it provides the students and faculty with the access to colleges processes. The students can enroll to the students and able to view the coursework. The coursework contains Announcement, Quiz and course materials provided by the faculty of that subject. It has provision to upload images for profile and files of the subject. The entire development is based on React, Node, MongoDB and Kafka. Canvas prototype application deployed to AWS server with MongoDB Atlas Database connected to it. Also, this application contains implementation of Passport.js for authentication and security provided to it.

Purpose of Developing System:

The main purpose behind making of Canvas Application is to give students the facility to enroll for course, upload assignment, give quiz and all other stuff related to coursework. Also, the faculty is having access to create a course, upload files, create quiz for the subject. Initially, the system validates the user. If user is valid then he can proceed with further functionalities on the dashboard else appropriate error will be thrown at the front end. This project can be utilized as an enrollment and college process during the semester. It was deployed to AWS so that it will be available for everyone and anyone can use it. It also has pagination provided for the course enrollment part.

System Design and the implemented system description:

Canvas →

- The design process used to build Canvas system is MERN Stack (Backend for this is MongoDB).
- This system is based on 3 tier architecture where backend processes will be handled and processed according to requirement by Node JS and Kafka. Node JS process is a single thread driven process and all the data fetch and stored in MongoDB. The react calls Node JS via axios and pass the input data from the user. Further, Node will make a call to messaging queue Kafka. After processing the data Kafka Server will send the response data to the again with response code 200 as a success to node and node will further pass it to React.
- To connect with MongoDB database, the path of the Cluster was provided. The cluster is created on MongoDB Atlas. Also, the sessions on the served side used to handle client processes. Same, I have used for logout process as well. This will check whether user is

already logged into the system or not. In the logout process, all these fields reset to the original value.

- The system is designed using React and some part of react-native as a frontend technology and node JS as backend and MongoDB as a database.
- To connect node JS and React and for the data passing from front end to backend and vice-versa, I have utilized axios call method. It will handle all the data passing processes. This is the easy way I found for passing the data from client to server and vice-versa.
- This project contains Redux implementation, the redux will be used to store and manage the data.
- The best use of this process is that, the entire application is single page application which is the requirement of this project. In react, I have utilized Redirect method to divert from one page to another depending upon the user actions. For example, once the faculty clicked on Course Creation process it get divert to the Couse Create page. This all has been handled by redirect method in React JS.
- This process is based on user validation as well, if user is part of the system and if it is not faculty then diversion takes place for Student and if it is faculty then it will be diverted towards Faculty Dashboard. This process is performed by axios and redirect in React. This flow in Canvas application helped us to develop enterprise application.
- Also, Bcrypt algorithm implemented in this lab as it has more powerful structure as compare to AES. The encrypted password got stored in the table and it will be verified against user when user tries with the SJSUID which was provided by him at the time of signup process. This algorithm provides the security which will provide the cyber-attacks.
- The Login Page, this page will take and input from the user and check into the database for the user entry. If it is a valid user with valid password then he is allowing access to the services. If not, then appropriate message will be thrown at the front end. There is signup link added in the process.
- The signup process, this will get the data from user. It also gets user profile and store it in the backend node. This will be further get downloaded during Dashboard process.
- Dashboard, in this the student or faculty able to do the process according to his/her needs.
- Quiz, in this tab, the professor can create the quiz and the same quiz get availed by the respective student in the same section at the student side process.
- Announcement, in this tab the professor can make announcement with the coursed. The course id added as the professor may has multiple subjects with him. The student enrolled for that subject gets announcement in announcement tab.

- Inbox, in this tab the professor/student can send message with the SJSUID. The student with SJSUID gets message in the message tab.
- Files, this will help to have proper communication between professor and student for submission of assignments or sharing of lecture notes from professor end.
- Account tab, this tab will show the details of the student or professor.
- Profile, this tab shows the small profile photo of the student or faculty.
- Course, this tab will allow professor to create the new course.
- Thus, this design provides the most suitable real time application and fast processes.
-

Performance of the System:

The above developed application has below mentioned benefits.

- The system mains the proper data handling from client side to the server and server side to the client.
- These applications are very fast and provides the services in very minimal time.
- The complete flow of the application is as follow:

  React JS (Frontend or Client side) → API → Node JS → Kafka → MongoDB → Kafka → Node JS → API → React JS (display response.)

- The React JS is main part of the Frontend Side, the system processes only those parts which was requested by the student or faculty. It will not change the entire page.
- It has implementation of **Bcrypt** mode algorithm for password encryption facility. It will help to improve the security of the application.
- Added Kafka as a messaging queue which solved the issue of running concurrent multiple users at a time. In previous case, it is failing as multiple users will not be handled properly.

  The combination of React as client and Node JS as server increases the performance by using functionality provided by both the JS.

Questions:

1. Compare performance with Kafka services, MongoDB deployed to cloud and local. Explain in detail the reason for difference in performance.

→

Deployment and implementation of KAFKA and MongoDB in Canvas prototype application, due to inclusion of this, the performance of Canvas system increased by large amount as compared to what we did in Lab1. We implemented the KAFKA because it has ability to manage multiple concurrent users. In previous system(Lab1), the multiple users can't operate at a same time. The Kafka solves this issue by using Consumer and Producer.

KAFKA is reliable, scalable and robust and handles a load using its system architecture. We created topics and it has improved parallel consumption. Also, we have included a MongoDB which increased a high availability of a system as compared to the Lab1.

Further, in this Lab, we deployed our application on AWS and it again improved the performance of the system by high margin. I have created an EC2 instance which has Java already installed. It provided space and high performance system to the application.

The connectivity between EC2 AWS machine instance and MongoDB Atlas also works perfectly. And the performance of the application got increased by some margin. Even Zookeeper and KAFKA servers ran fast on the AWS machine as compared to the local machine.

In sum, the performance of the system increased on AWS as compared to the local machine.

2. 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

→

The inclusion of MySQL and MongoDB depends on the application and the requirement. In some of the cases MongoDB works better than MySQL and sometimes MySQL. The use of MySQL is good as compare to MongoDB when there is need of very high performance, flexible and more reliable with ease of management.

But, MongoDB will be better whenever the developed application if very complex and it is not structured. In these cases, MongoDB is better option as compared to the MySQL.

Further, MongoDB will be suitable where applications contains more complexity, multiple attributes and when schema is unstable.

Due to inclusion of MongoDB in Canvas prototype, the speed of the application and the scalability got increased efficiently. The scalability is very high in MongoDB such that, it can be scaled across distributed systems. We create documents in the MongoDB which is very flexible. Also, MySQL does not have process to validate schema. The application developer needs to add it manually where MongoDB have that provision already. In this way, the MongoDB helps to improve productivity of application as well as the developer.

MySQL will be much helpful when consistency is required. It means child will not be referred unless and until the parent has data in it. MySQL is more structured data and consistent too. It required the data in more structured way as compared to the MongoDB.

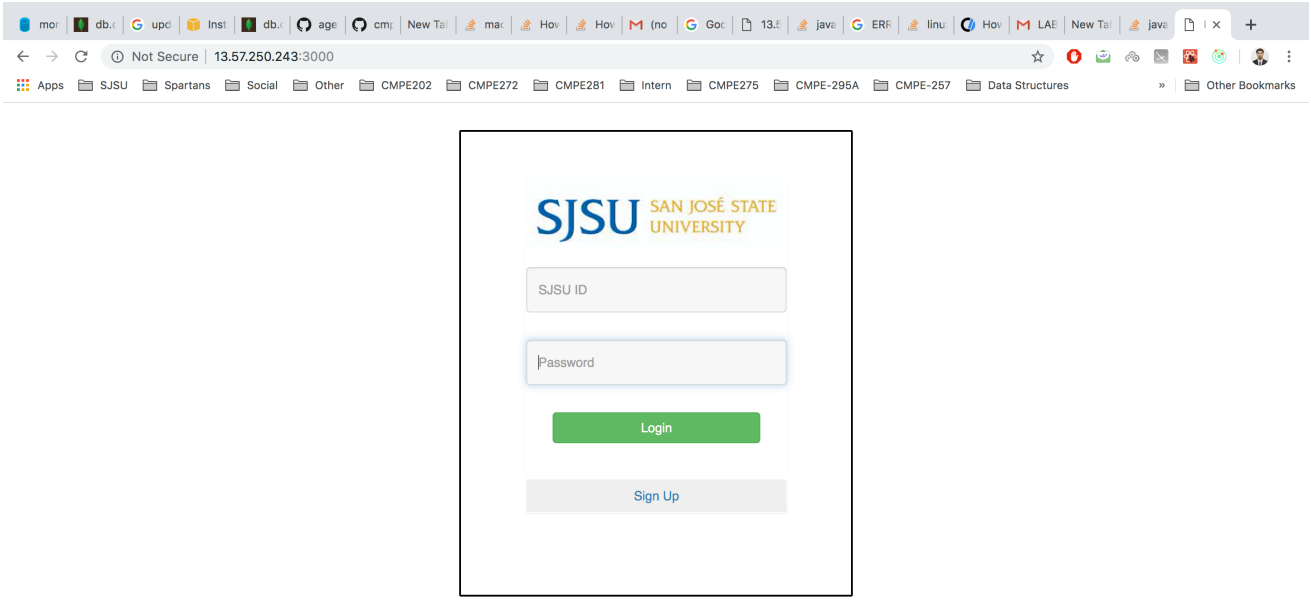
Where MongoDB provides availability as compared to MySQL. There will not be any issues with the availability if MongoDB but in some of the times there will be issues with the availability of MySQL during the execution of an application.

Canvas →

Login Page →

Initially, if user student the wrong username and password then appropriate message will be and if student enters valid password then it got diverted to the Dashboard page.

The page when React app ran for the first time.

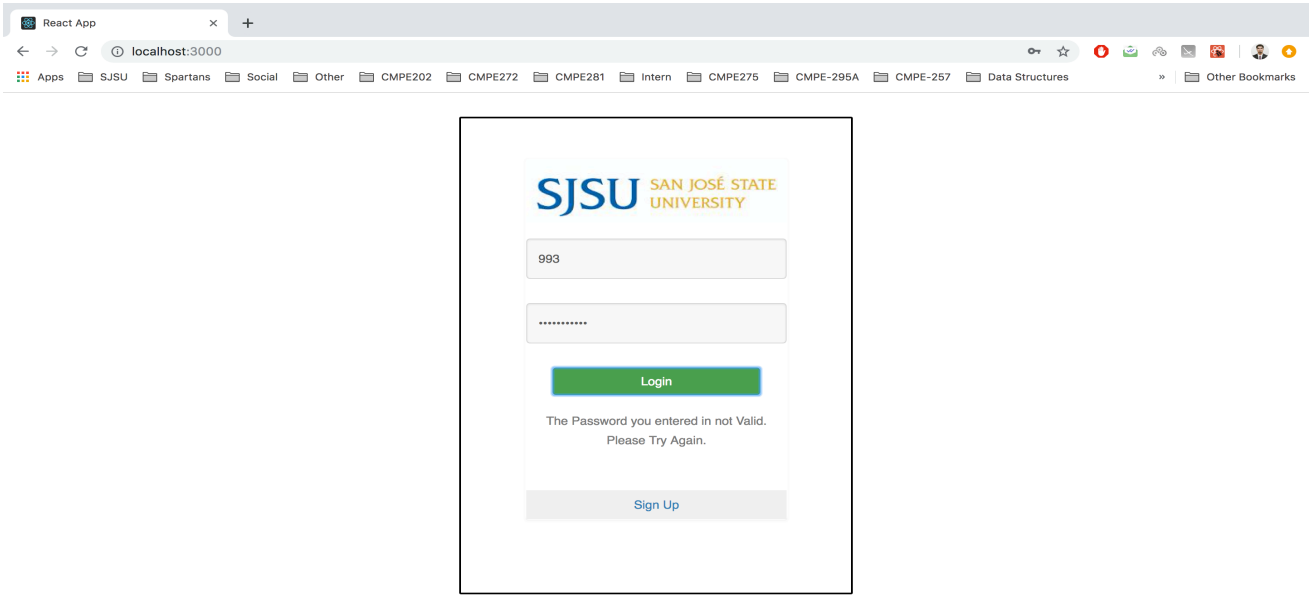


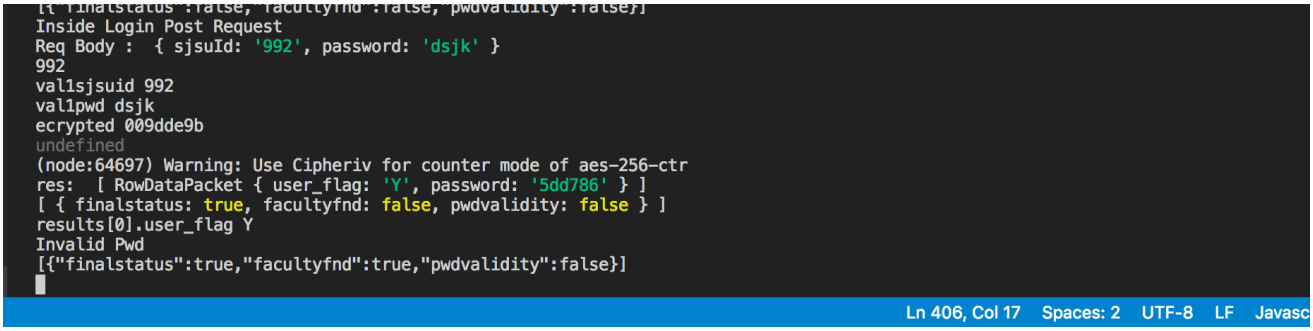
a. If user enters invalid user name.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTIiLCJpYXQiOjE1NTUyMjA2MDgsImV4cCI6MTU1NTIzMDY4OH0.EFB6jCcCiu0vnFVtJgb8mqfgqpuiUki1b_43bNcq-U4
{"logincheck":[{"finalstatus":true,"facultyfnd":true,"pwdvalidity":true}],"Token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTIiLCJpYXQiOjE1NTUyMjA2MDgsImV4cCI6MTU1NTIzMDY4OH0.EFB6jCcCiu0vnFVtJgb8mqfgqpuiUki1b_43bNcq-U4"}
Inside Login Post Request
Req Body :  { sjsuId: '993', password: '993' }
in make request
{ sjsuId: '993', password: '993' }
in response
in response1
true
in response2
{ login: { '0': 1 } }
msg received
response { _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
In results Signup
Results:  { _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
Logged in successfully.
{ _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTMiLCJpYXQiOjE1NTUyMjA2MjQsImV4cCI6MTU1NTIzMDcwNH0.zfAAA1o7xzt04-L3FASRR3GK_Ubc05yer2iMQf3VJjg
{"logincheck":[{"finalstatus":true,"facultyfnd":false,"pwdvalidity":true}],"Token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTMiLCJpYXQiOjE1NTUyMjA2MjQsImV4cCI6MTU1NTIzMDcwNH0.zfAAA1o7xzt04-L3FASRR3GK_Ubc05yer2iMQf3VJjg"}
[ec2-user@ip-172-31-3-189 ~]$
```

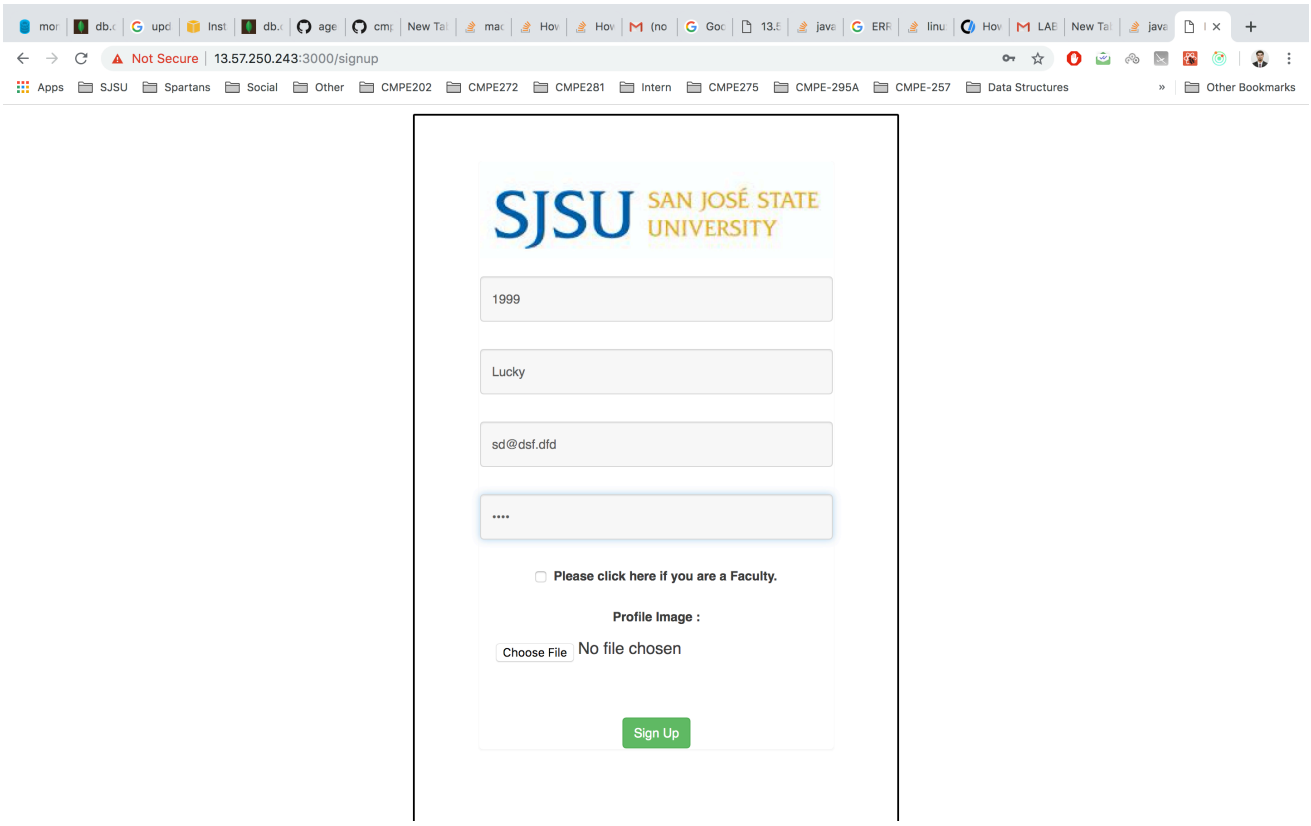
```
Downloads — ec2-user@ip-172-31-3-189:~ — ssh ec2-user@13.57.250.243 -i canvaslab2.pem — 193×45
  __v: 0 }
User details  { _id: 5cb2a6549955504fee2e2f9c,
  sjsuid: '992',
  password: '$2a$10$QxJJBs97piELC7OO0fYE2u1HkIhCzEi/rqII19JWilM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  __v: 0 }
valid Credentials!
After request handling:  { _id: 5cb2a6549955504fee2e2f9c,
  sjsuid: '992',
  password: '$2a$10$QxJJBs97piELC7OO0fYE2u1HkIhCzEi/rqII19JWilM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  __v: 0 }
Data:  { response_topic: { '0': 0 } }
message recieved for login [object Object]
"{\"correlationId\":\"dac18de208e5f4594d6aae7577965916\",\"replyTo\":\"response_topic\",\"data\":{\"sjsuId\":\"993\",\"password\":\"993\"}}"
Inside Kafka Backend Signup
Message:  { sjsuId: '993', password: '993' }
{ _id: 5cb2a6089955504fee2e2f9a,
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
User details  { _id: 5cb2a6089955504fee2e2f9a,
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
valid Credentials!
After request handling:  { _id: 5cb2a6089955504fee2e2f9a,
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
Data:  { response_topic: { '0': 1 } }
[ec2-user@ip-172-31-3-189 ~]$
```

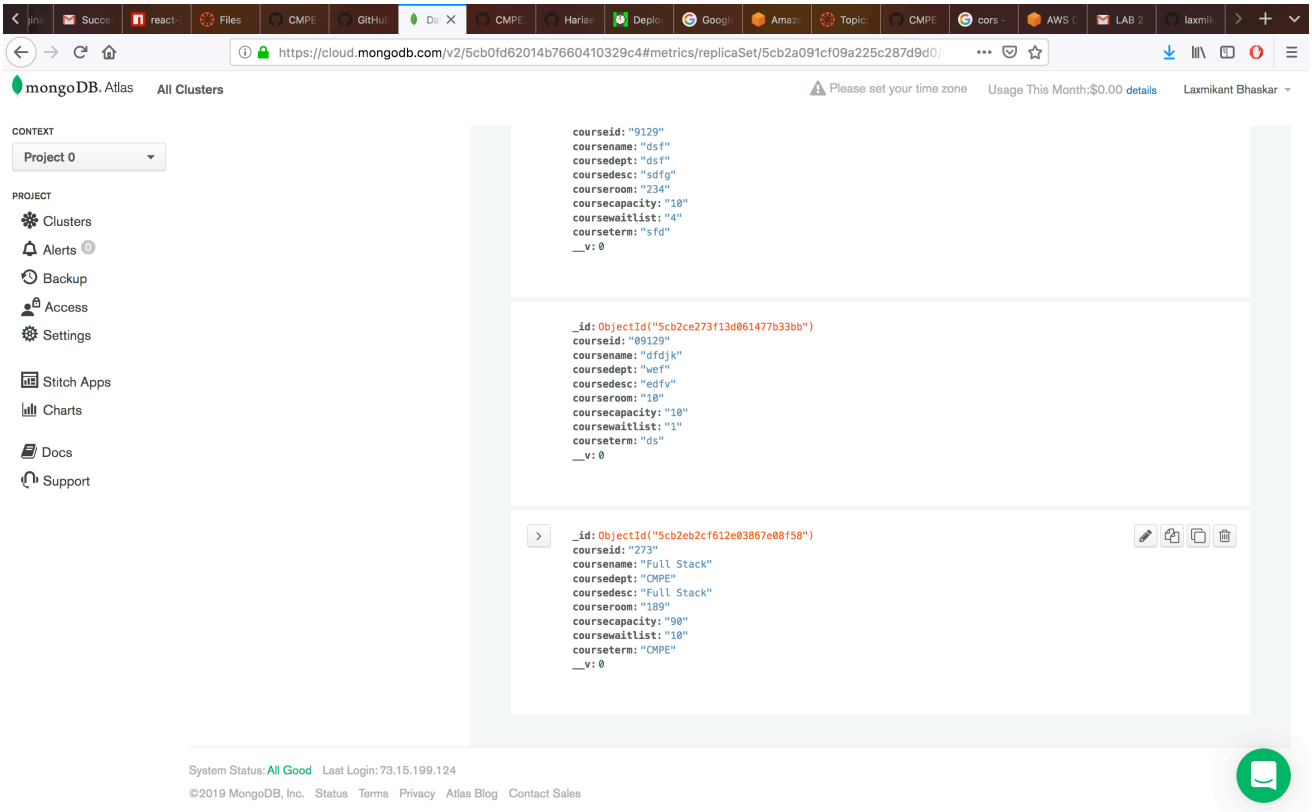
b. If the student enters invalid password with correct SJSU ID.


React App
localhost:3000
Apps
SJSU
Spartans
Social
Other
CMPE202
CMPE272
CMPE281
Intern
CMPE275
CMPE-295A
CMPE-257
Data Structures
Other Bookmarks
SJSU SAN JOSÉ STATE UNIVERSITY
993
Login
The Password you entered in not Valid.
Please Try Again.
Sign Up

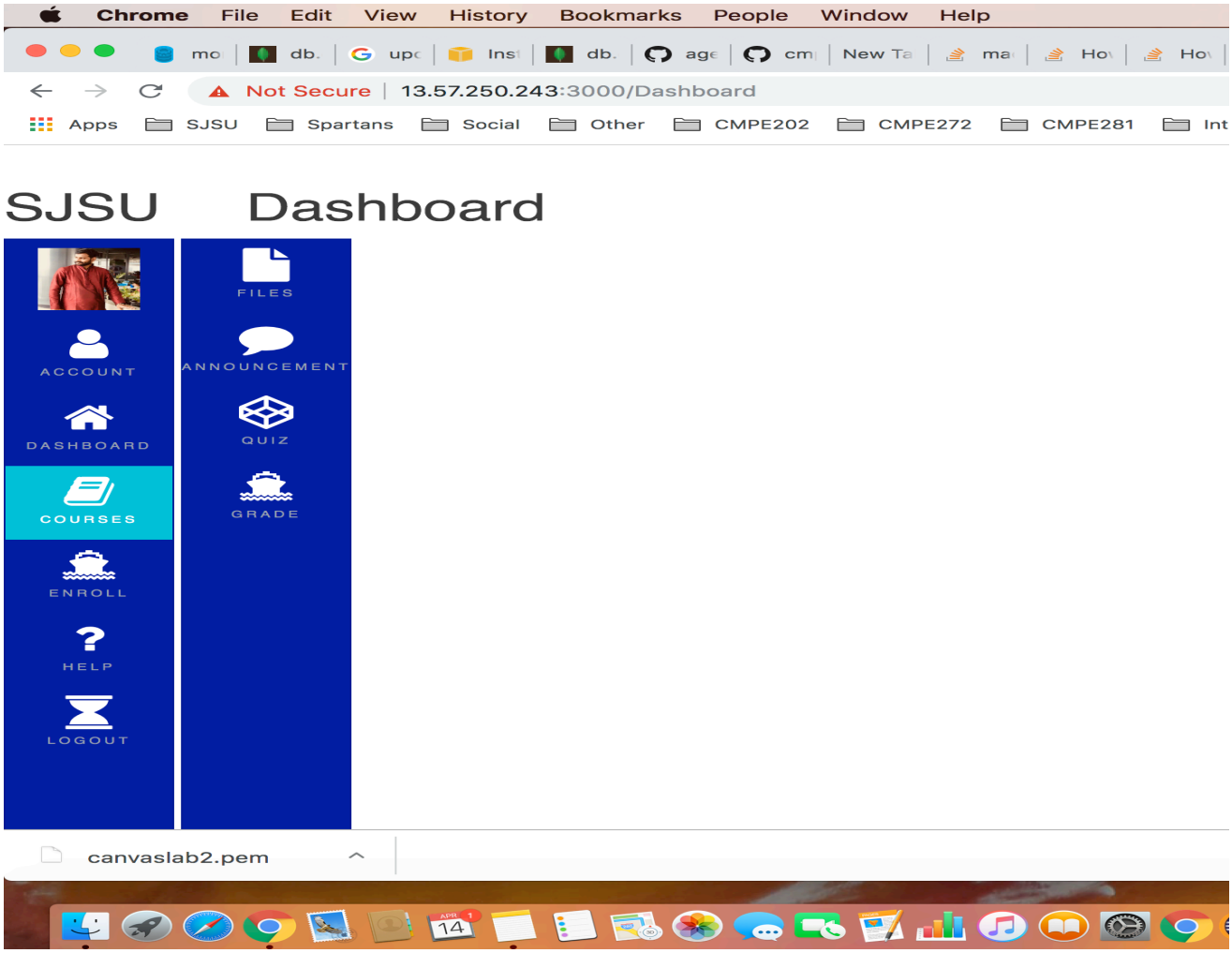

Inside Login Post Request
Req Body : { sjsuId: '992', password: 'dsjk' }
992
val1sjsuid 992
val1pwd dsjk
ecrypted 009dde9b
undefined
(node:64697) Warning: Use Cipheriv for counter mode of aes-256-ctr
res: [RowDataPacket { user_flag: 'Y', password: '5dd786' }]
[{ finalstatus: true, facultyfnd: false, pwdvalidity: false }]
results[0].user_flag Y
Invalid Pwd
[{"finalstatus":true,"facultyfnd":true,"pwdvalidity":false}]
Ln 406, Col 17
Spaces: 2
UTF-8
LF
Javasc

c. If the User clicks on Signup Page.



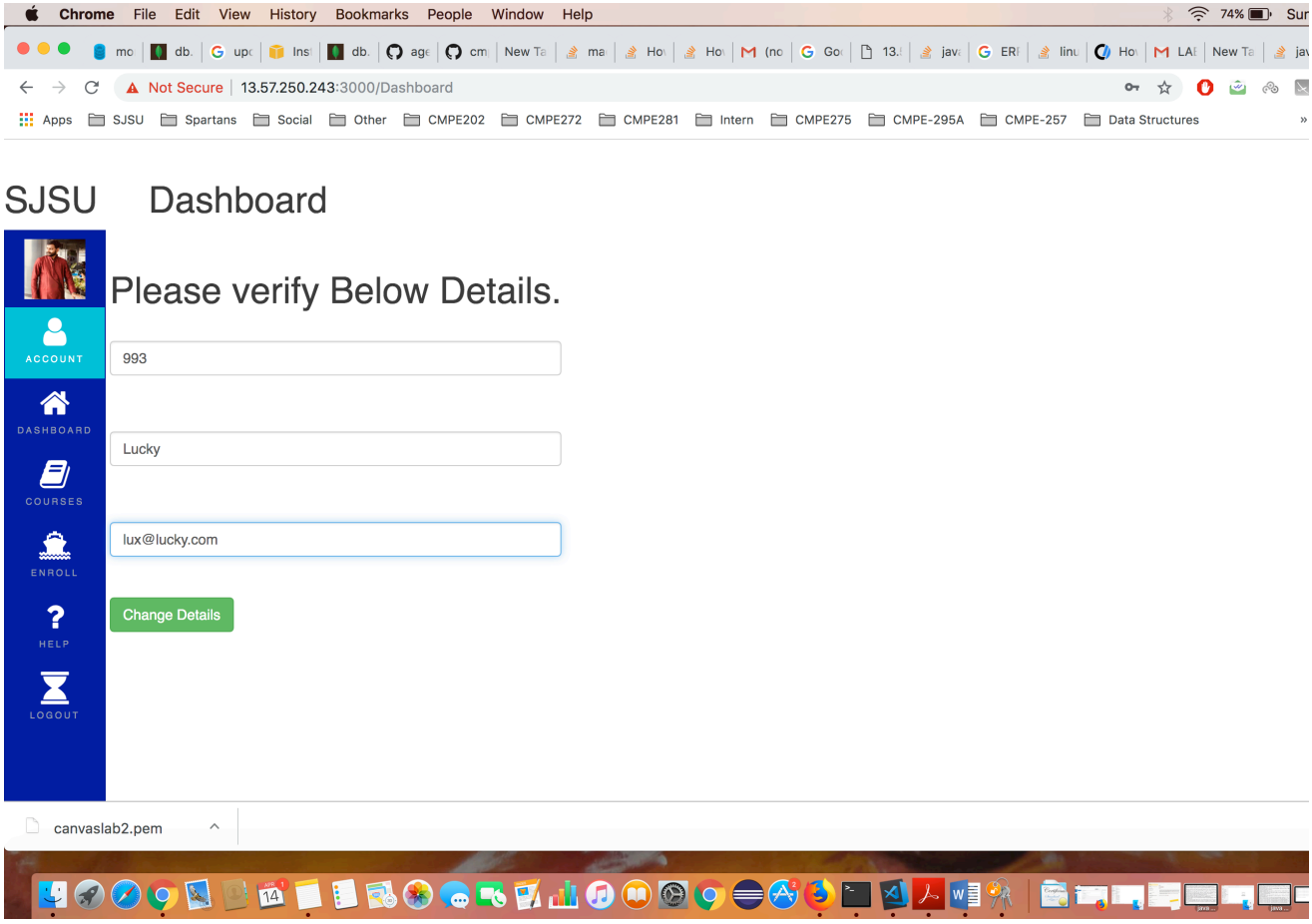


d. If Student enters valid details then it got diverted to the Dashboard page.

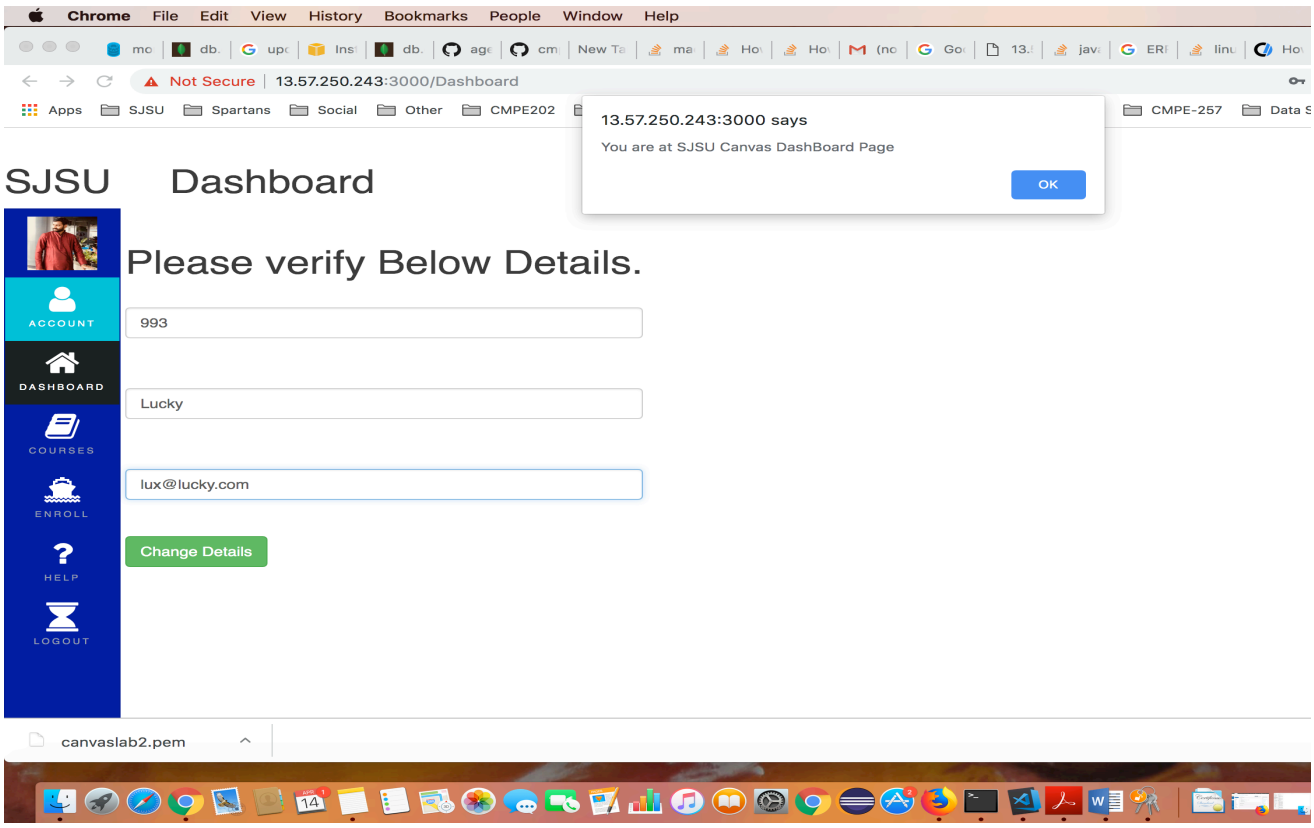


On the Dashboard Page, the Logout page will destroy the session and Student/faculty will be back to login page. At the top, the image of the student got downloaded and added as a profile of a Student.

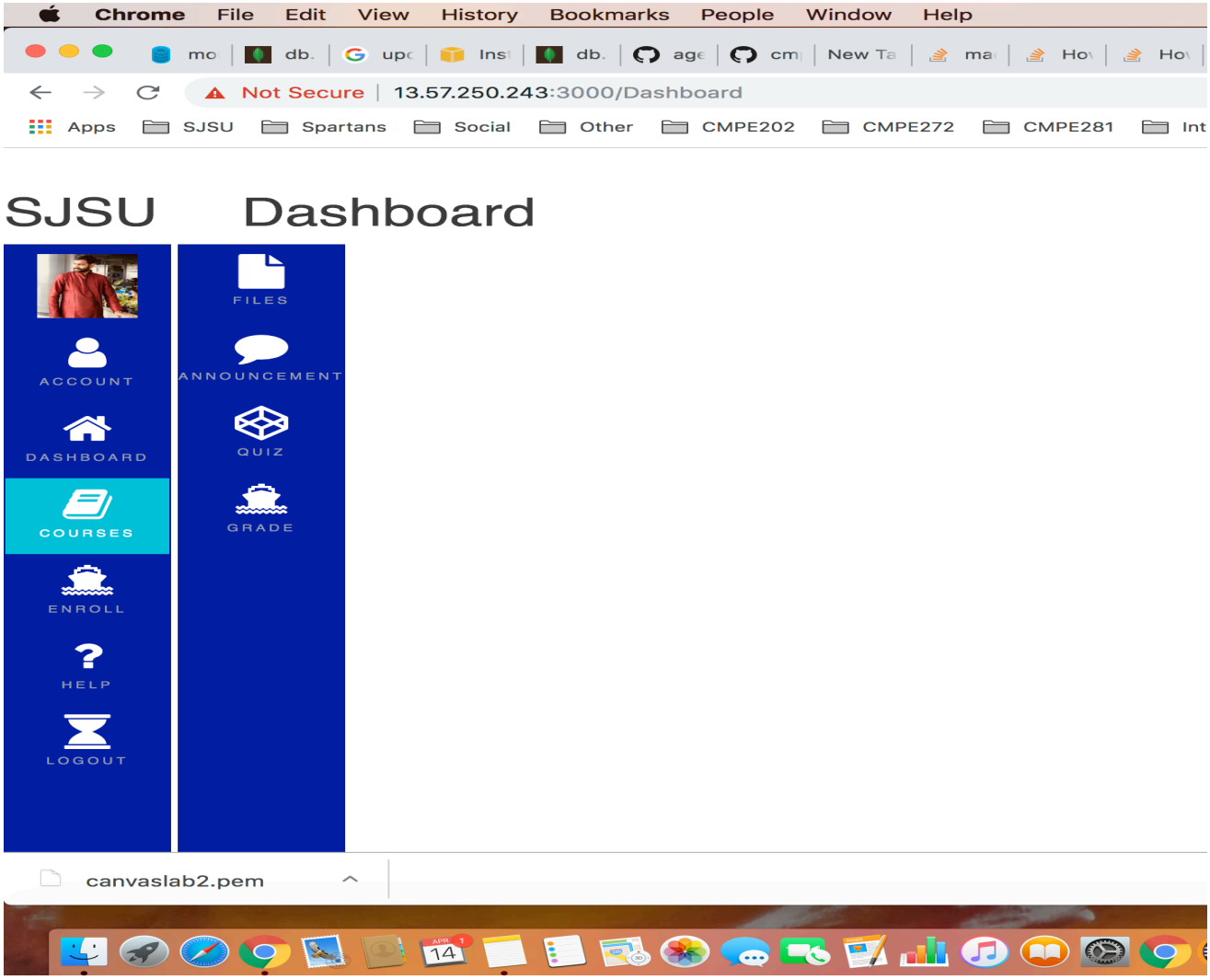
2. Once Student Click on Account, it will display the details of the user.



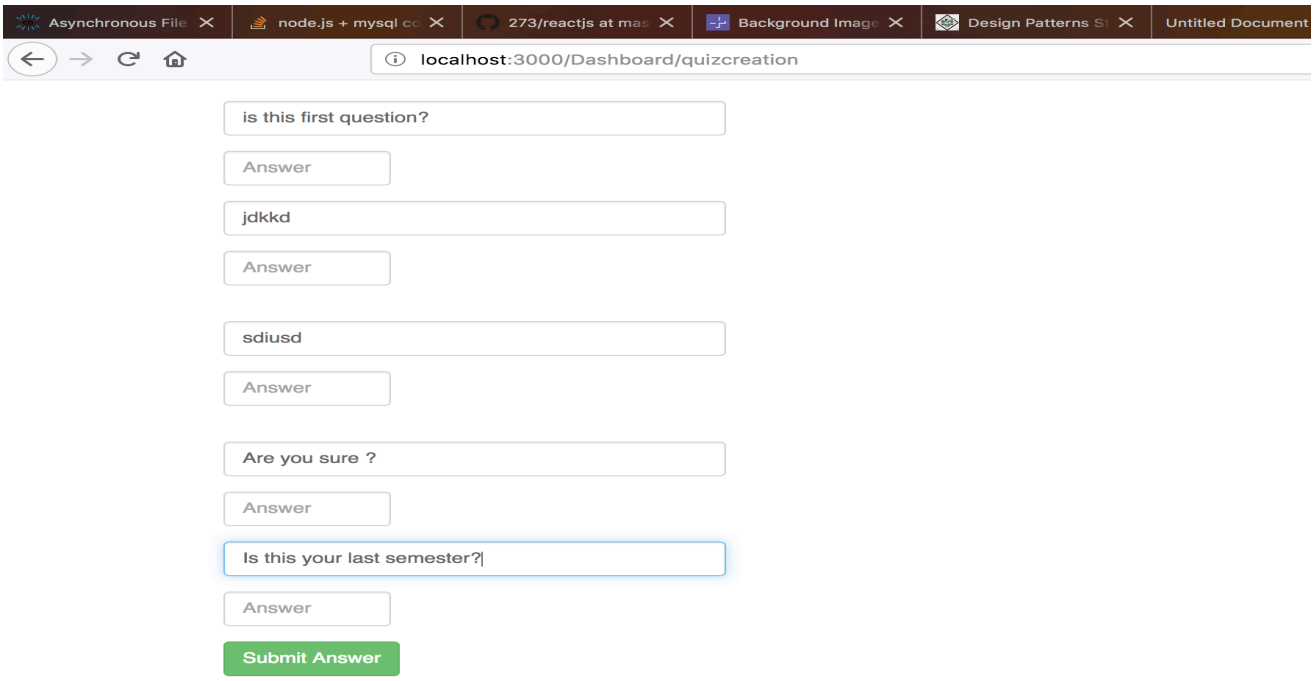
If User Clicks on Dashboard then it will be moved to same page



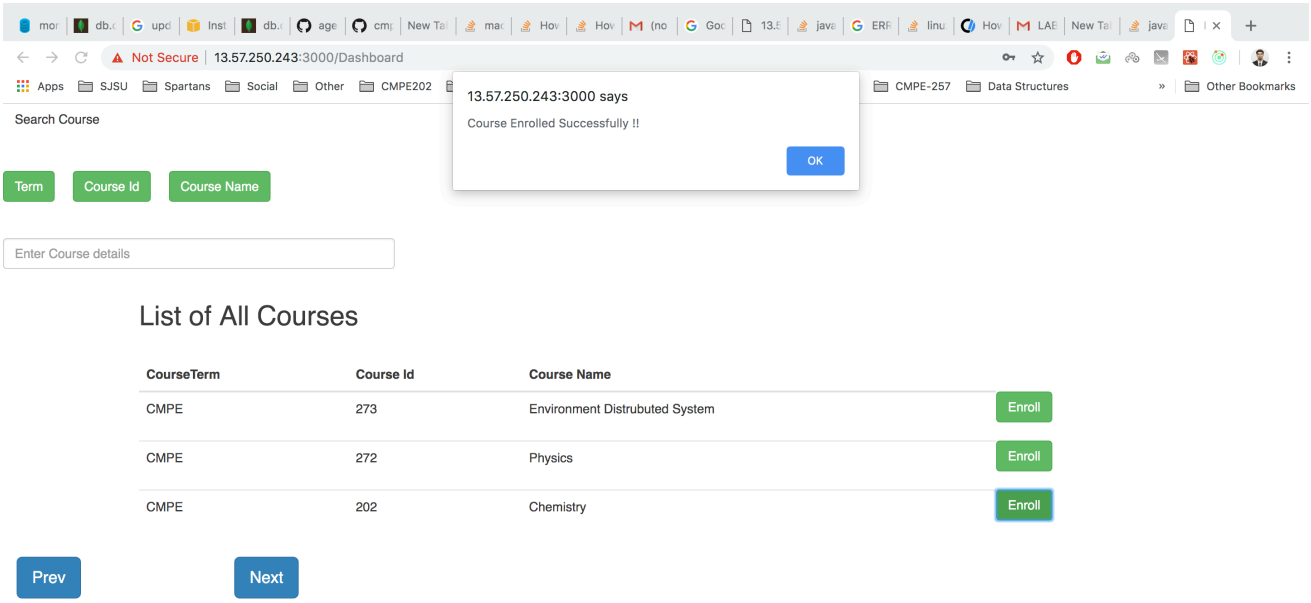
Once Student Clicks on Dashboard it will display the options Student wants to perform.



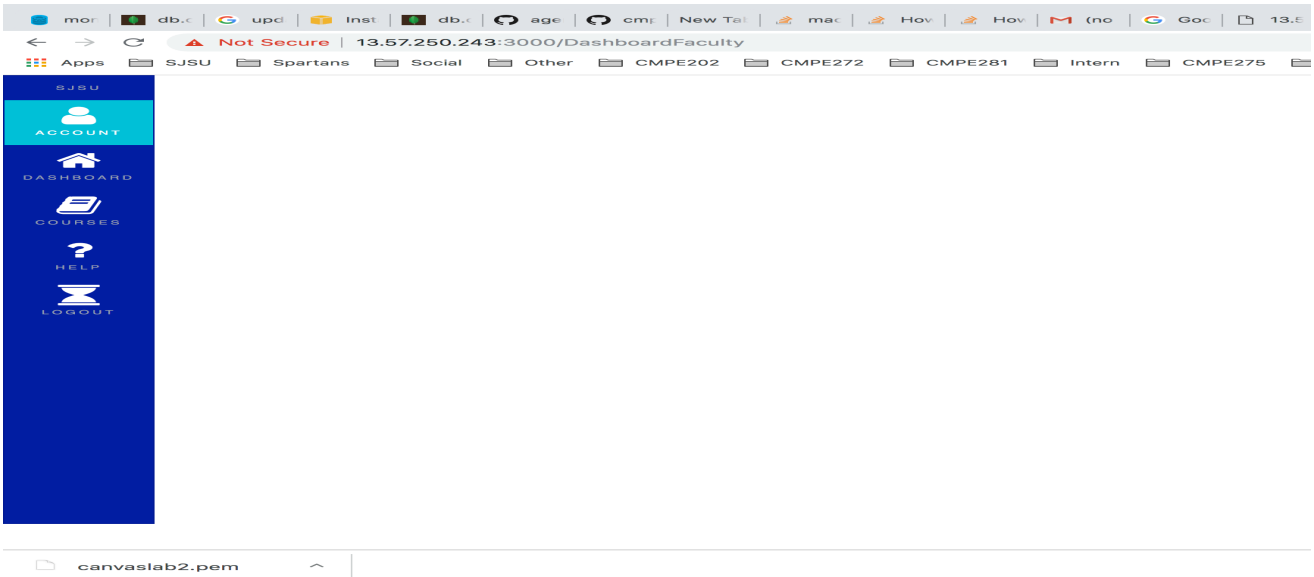
Once Student Clicks on Quiz then it will be diverted to new page and Student will be able to give the Quiz.



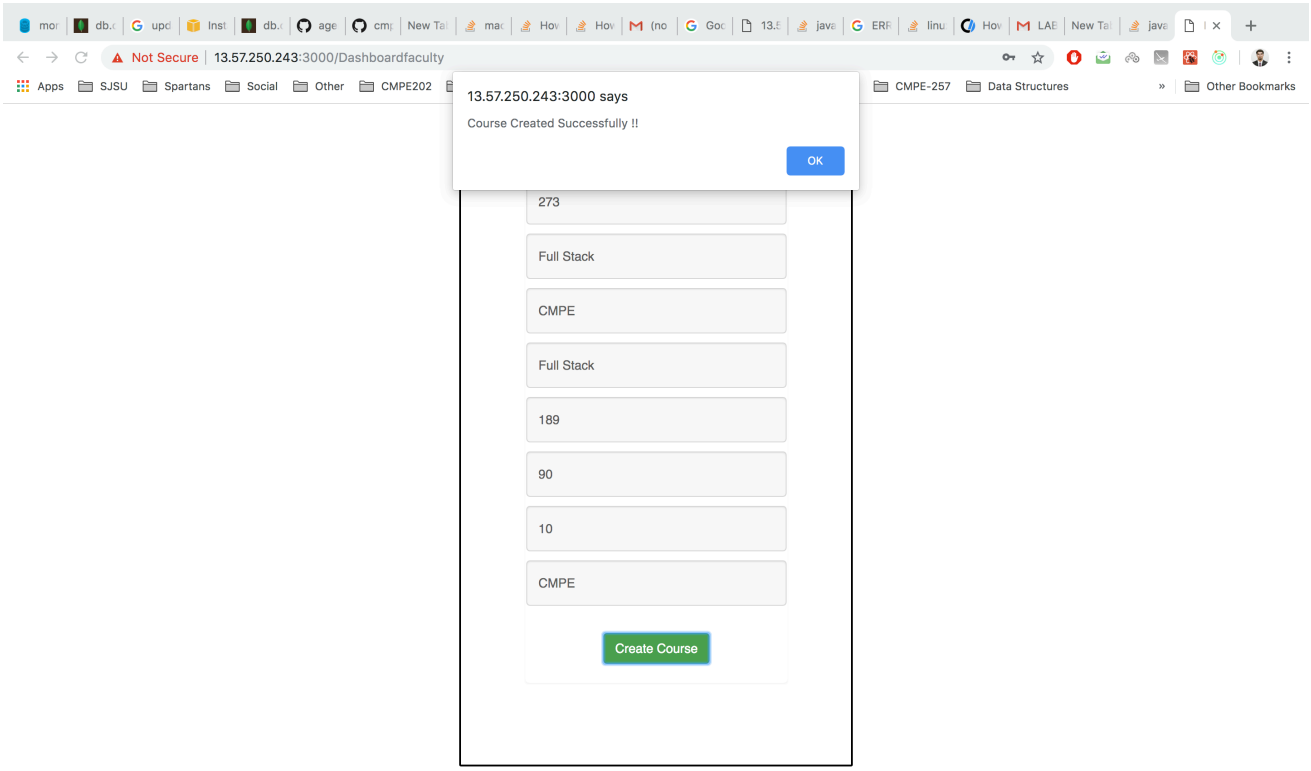
Student Enrollment process. Once Student clicks on Enroll then the respective subject gets added with SJSU Id of Student

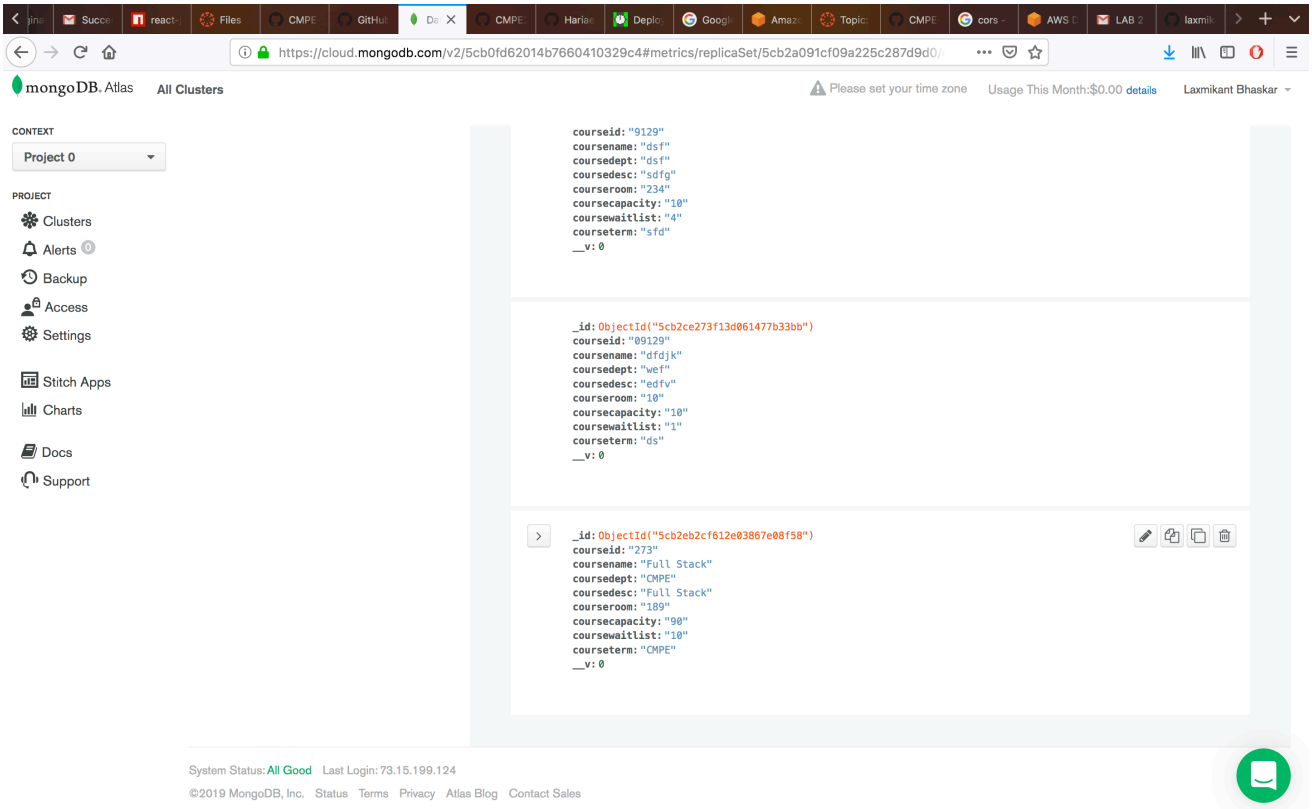


Now, the pages and actions which was added for Faculty.

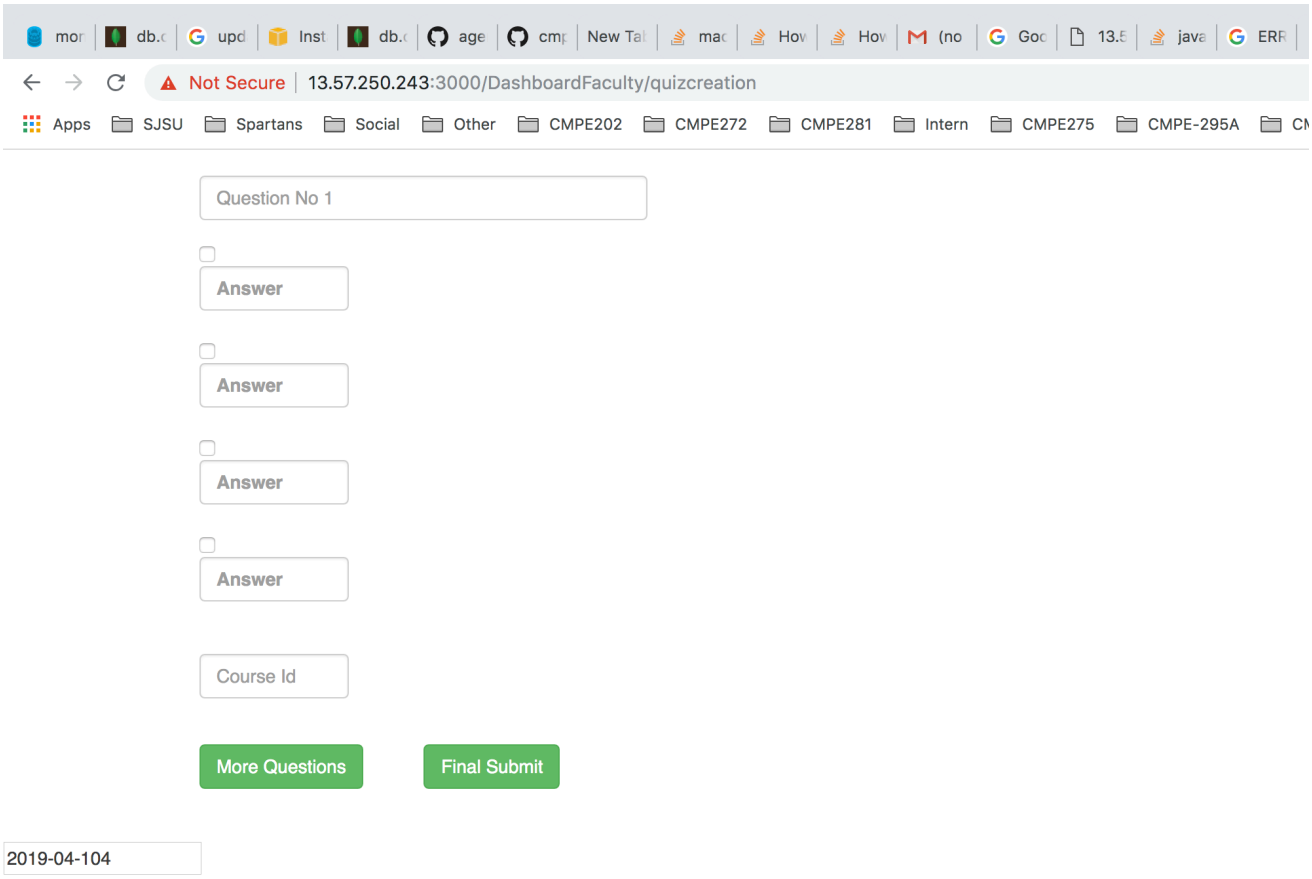


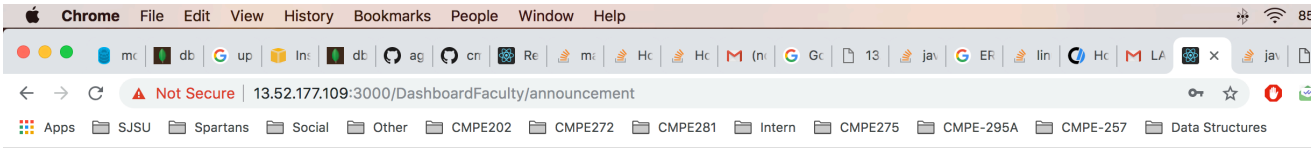
Creation of Course Page. Faculty can create a Course with below details.





Below are the more options performed by the Faculty. (Announcement, Quiz, file upload)



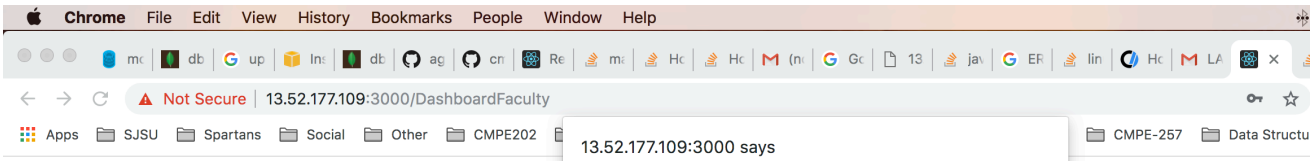


Add Announcement in Below Area :

Enter Course Id

Your Announcement Here

Announce



13.52.177.109:3000 says

Announced Successfully !!

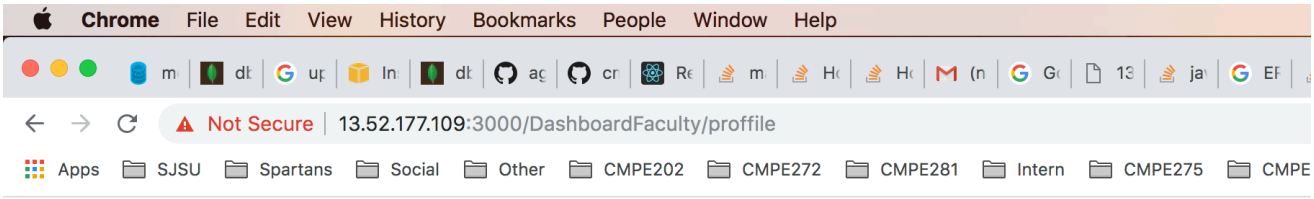
OK

Add Announcement in Below A

1909

Check annoucnemet!

Announce

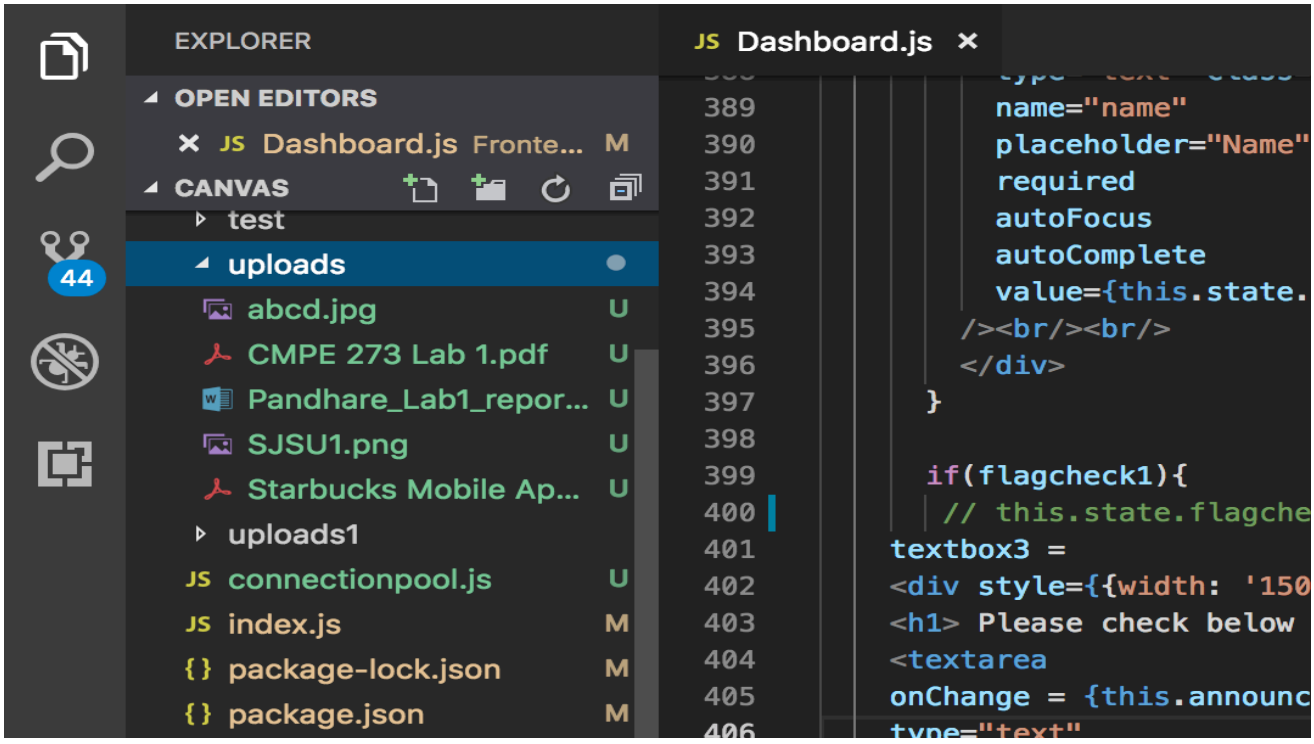


Course Files:

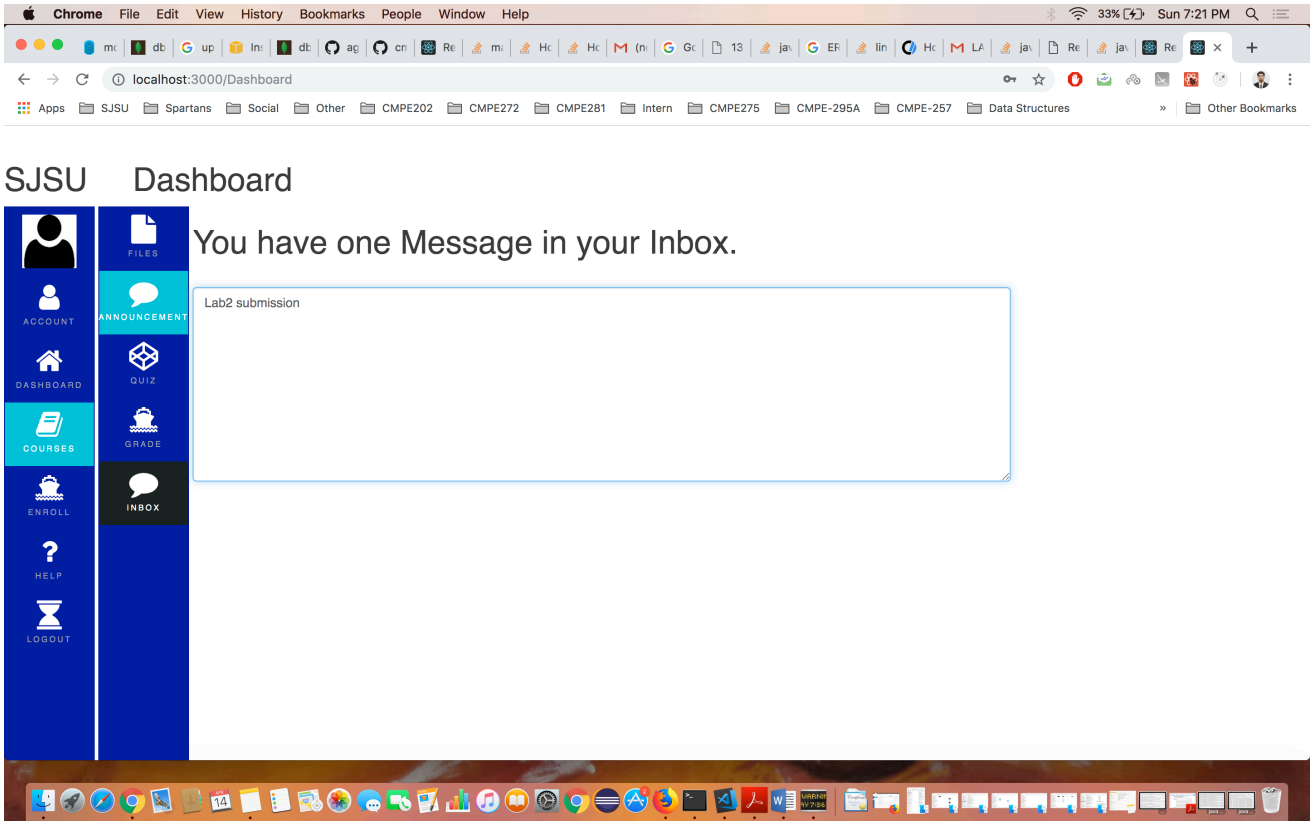
Choose File jmeter.log

Sign Up

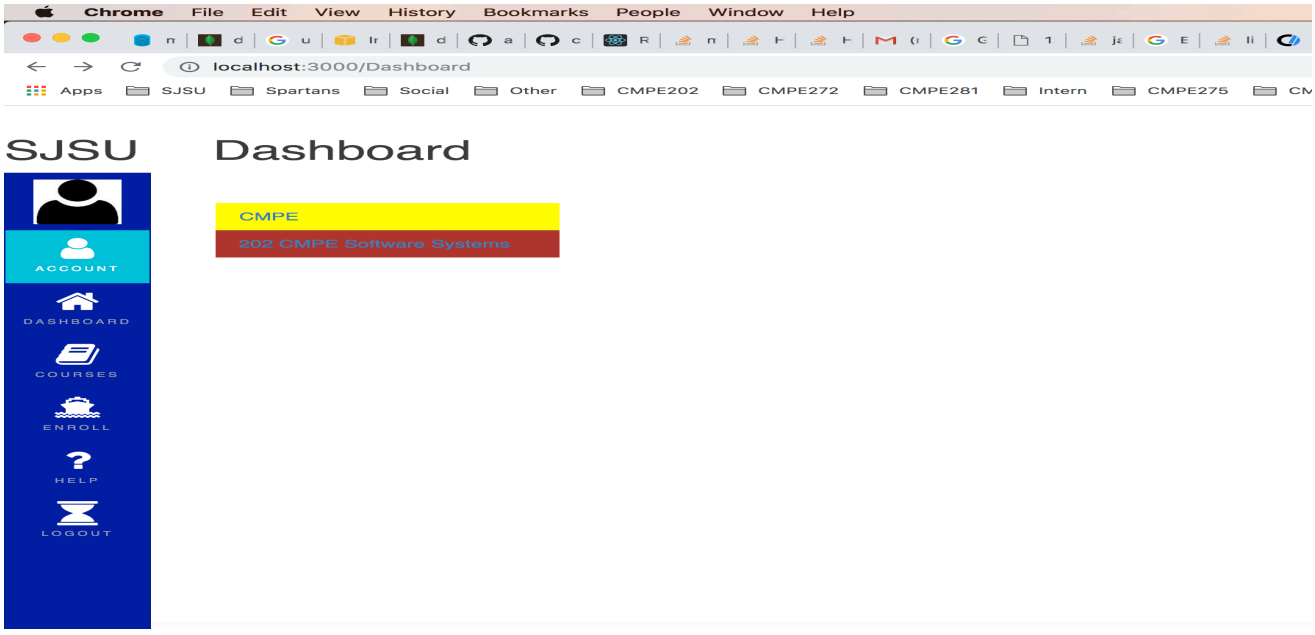
```
backend  config  connectionpool.js  DataBaseConnection.js  index.js  kafka  node_modules  package.json  package-lock.json  routes  test  uploads
[ec2-user@ip-172-31-3-189 Backend-Code]$ cd uploads
[ec2-user@ip-172-31-3-189 uploads]$ ls
abcd.jpg  CMPE 273 Lab 1.pdf  Pandhare_Lab1_report.doc  SJSU1.png  SJSU.jpg  Starbucks Mobile App - Project Requirements 2018.V2-1.pdf
[ec2-user@ip-172-31-3-189 uploads]$ ls
abcd.jpg  CMPE 273 Lab 1.pdf  Pandhare_Lab1_report.doc  SJSU1.png  SJSU.jpg  Starbucks Mobile App - Project Requirements 2018.V2-1.pdf
[ec2-user@ip-172-31-3-189 uploads]$
```



Added New Functionality for sending messages from one User to Another.



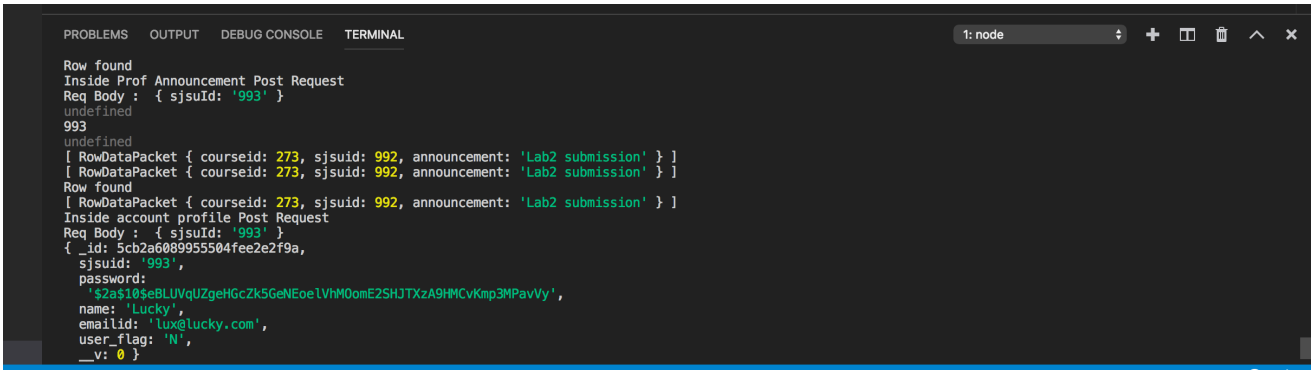
Added draggable feature but failed to drop it.



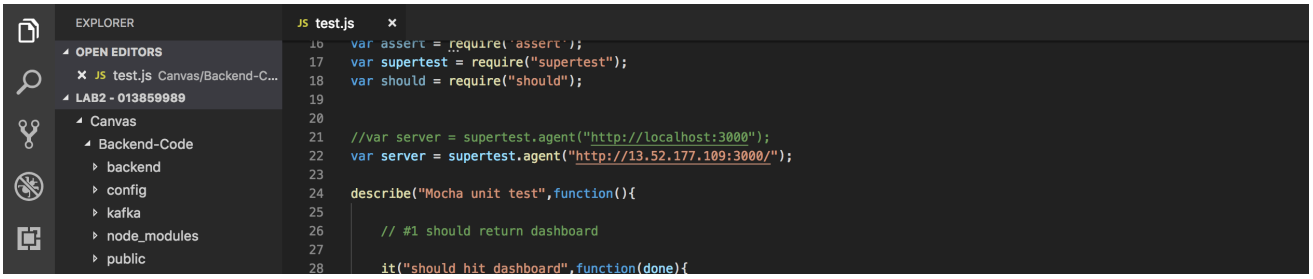
Mocha Testing for Canvas: (Test Cases added on the GITHUB Repo)

Below Coding snippet added to show that, it is running on localhost.


```
Sachins-MacBook-Pro:Backend-Code sachinwaghmode$ ls
DataBaseConnection.js   config                  index.js            node_modules        package.json        routes              uploads
backend                 connectionpool.js       kafka               package-lock.json   public              test                uploads1
Sachins-MacBook-Pro:Backend-Code sachinwaghmode$ mocha


  Mocha unit test
test done
    ✓ should hit dashboard (368ms)

  Mocha unit test
test done
    ✓ should display Signup (693ms)

  Mocha unit test
test done
    ✓ should display quizcreation (52ms)

  Mocha unit test
test done
    ✓ should display announcement

  Mocha unit test
test done
    ✓ should display profiledetails (166ms)


  5 passing (1s)

Sachins-MacBook-Pro:Backend-Code sachinwaghmode$
```



I have tested AWS deployed application with MOCHA.




```
Last login: Sun Apr 14 11:12:41 on ttys006
Sachins-MacBook-Pro:~ sachinwaghmode$ mocha
Error: No test files found: "test"
Sachins-MacBook-Pro:~ sachinwaghmode$ cd /Users/sachinwaghmode/Desktop/Lab2\ -\ 013859989/test
Sachins-MacBook-Pro:test sachinwaghmode$ mocha
Error: No test files found: "test"
Sachins-MacBook-Pro:test sachinwaghmode$ cd /Users/sachinwaghmode/Desktop/Lab2\ -\ 013859989/Canvas/Backend-Code/test
Sachins-MacBook-Pro:test sachinwaghmode$ mocha


  Mocha unit test
test done
    ✓ should hit dashboard (417ms)

  Mocha unit test
test done
    ✓ should display Signup (113ms)

  Mocha unit test
test done
    ✓ should display quizcreation (52ms)

  Mocha unit test
test done
    ✓ should display announcement

  Mocha unit test
test done
    ✓ should display profiledetails (62ms)


  5 passing (688ms)

Sachins-MacBook-Pro:test sachinwaghmode$
```

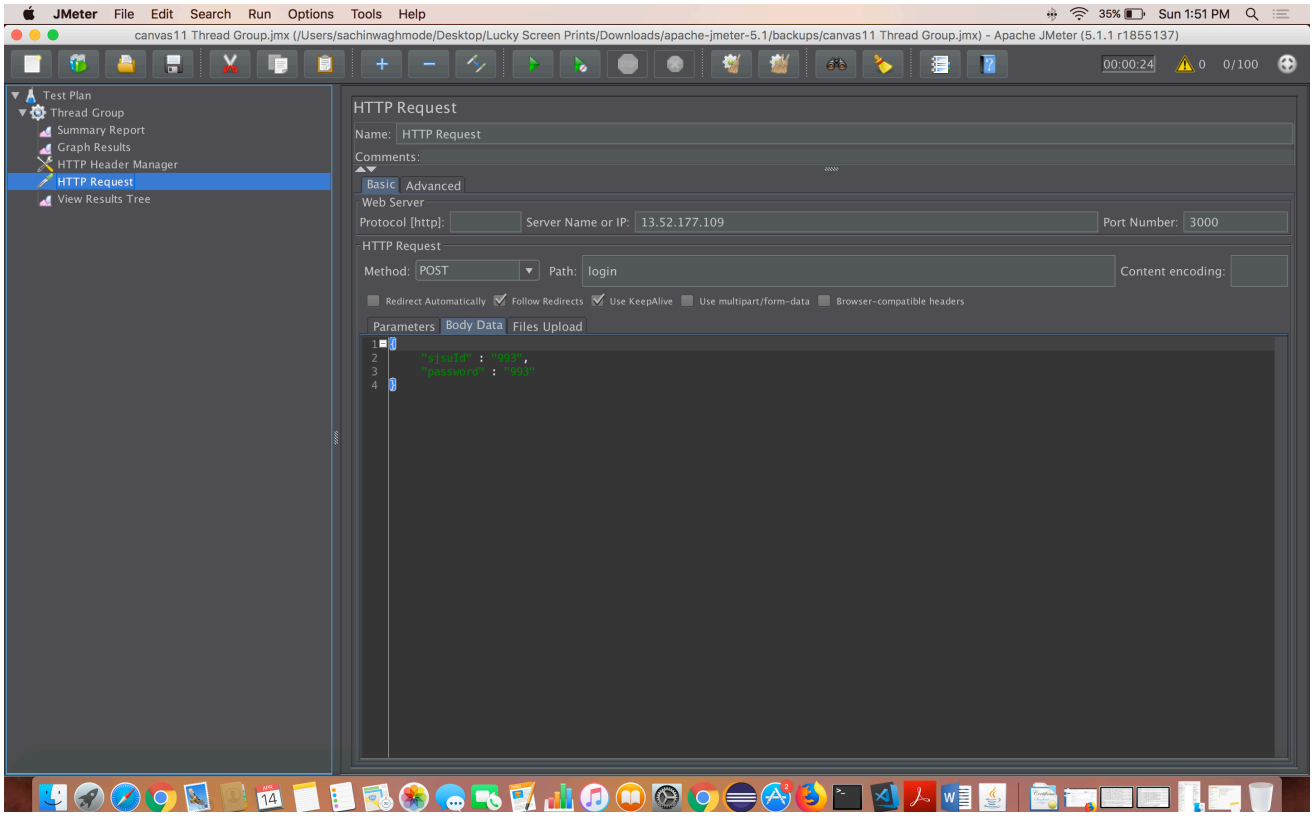
Below message is from Server which is running on AWS.

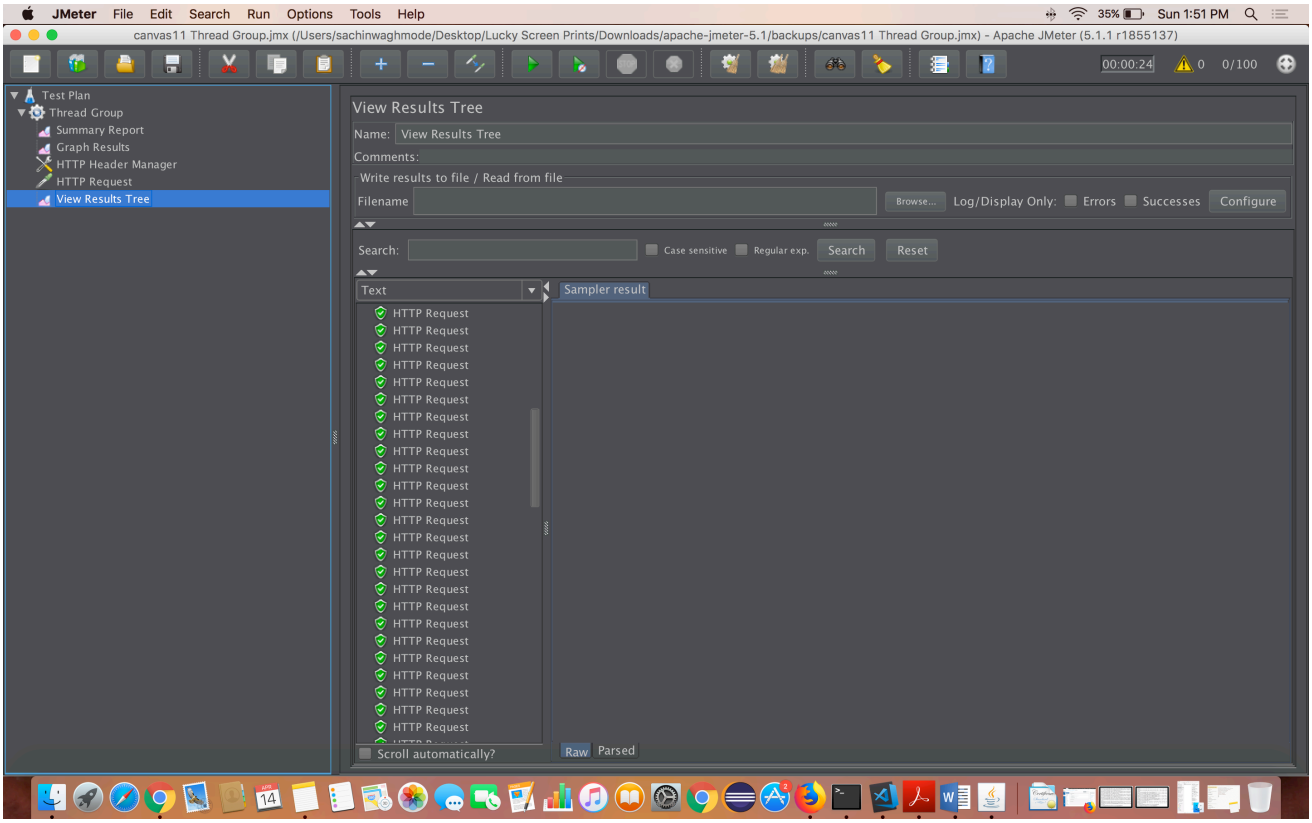
```
msg received
response { _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
In results Signup
Results:  { _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
Logged in successfully.
{ _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTMiLCJpYXQiOjE1NTUyNzM0MzgsImV4cCI6MTU1NTI4MzUxOH0.h6nAvez7cMcu866HNUvbMe5p0R5atrnQUaggnlUEVR8
{"logincheck":[{"finalstatus":true,"facultyfnd":false,"pwdvalidity":true}],"Token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTMiLCJpYXQiOjE1NTUyNzM0MzgsImV4cCI6MTU1NTI4MzUxOH0.h6nAvez7cMcu866
HNUvbMe5p0R5atrnQUaggnlUEVR8"}
Inside DOwnload File
Inside Login Post Request
```

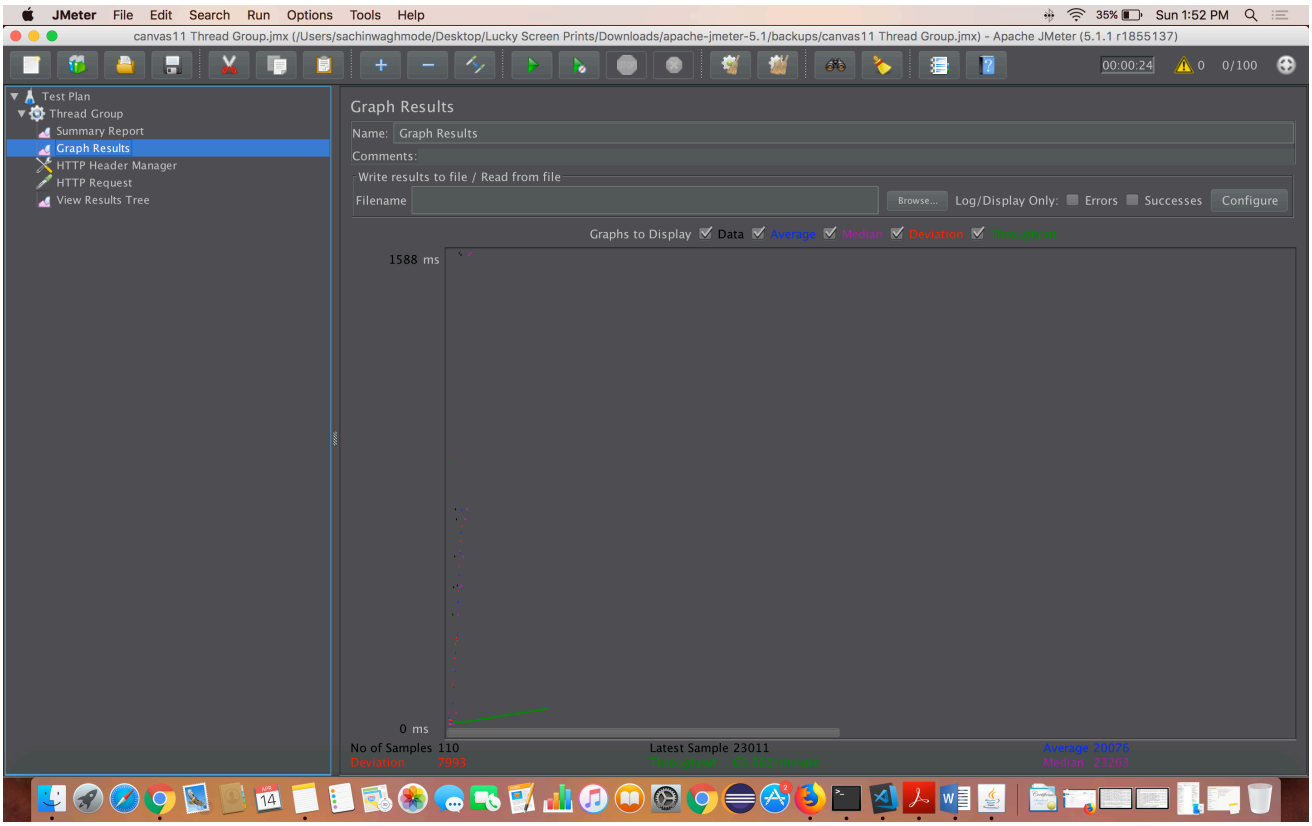
Jmeter Testing for Canvas →

I have tried with multiple concurrent user with and without connection pooling. I have noticed that, when I triggered multiple users with connection pooling is faster than without connection pooling.

100 Concurrent Users: (Deployed on AWS)



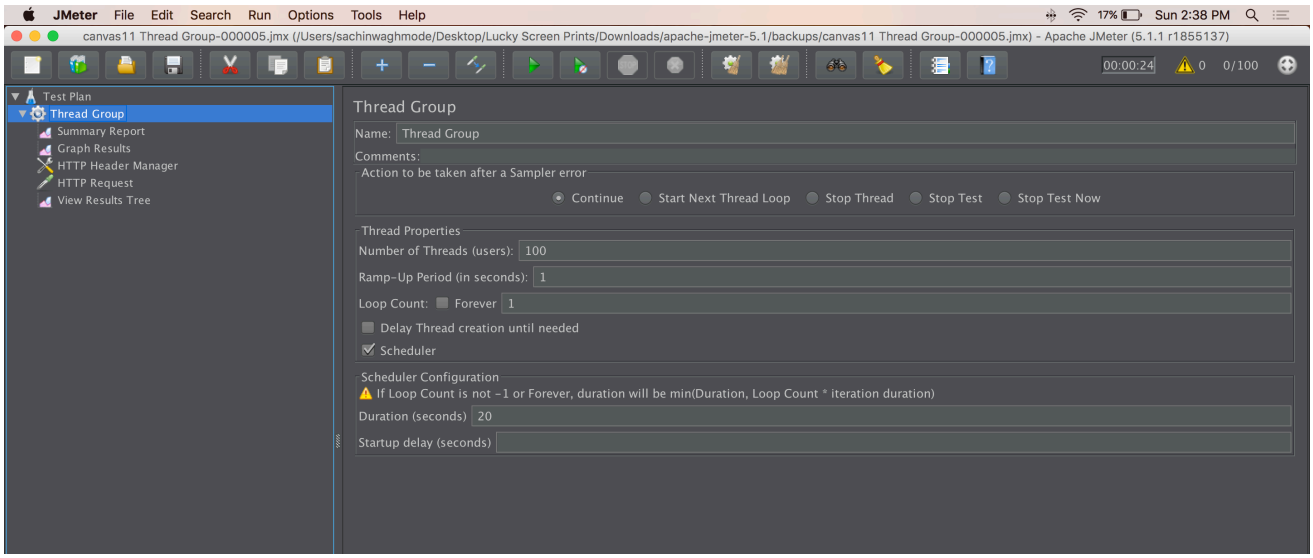

JMeter File Edit Search Run Options Tools Help
35% Sun 1:51 PM
canvas11 Thread Group.jmx (/Users/sachinwaghmode/Desktop/Lucky Screen Prints/Downloads/apache-jmeter-5.1/backups/canvas11 Thread Group.jmx) - Apache JMeter (5.1.1 r1855137)
00:00:24
0
0/100
Test Plan
Thread Group
Summary Report
Graph Results
HTTP Header Manager
HTTP Request
View Results Tree
View Results Tree
Name:
View Results Tree
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Successes
Configure
Search:
Case sensitive
Regular exp.
Search
Reset
Text
Sampler result
HTTP Request
Scroll automatically?
Raw
Parsed

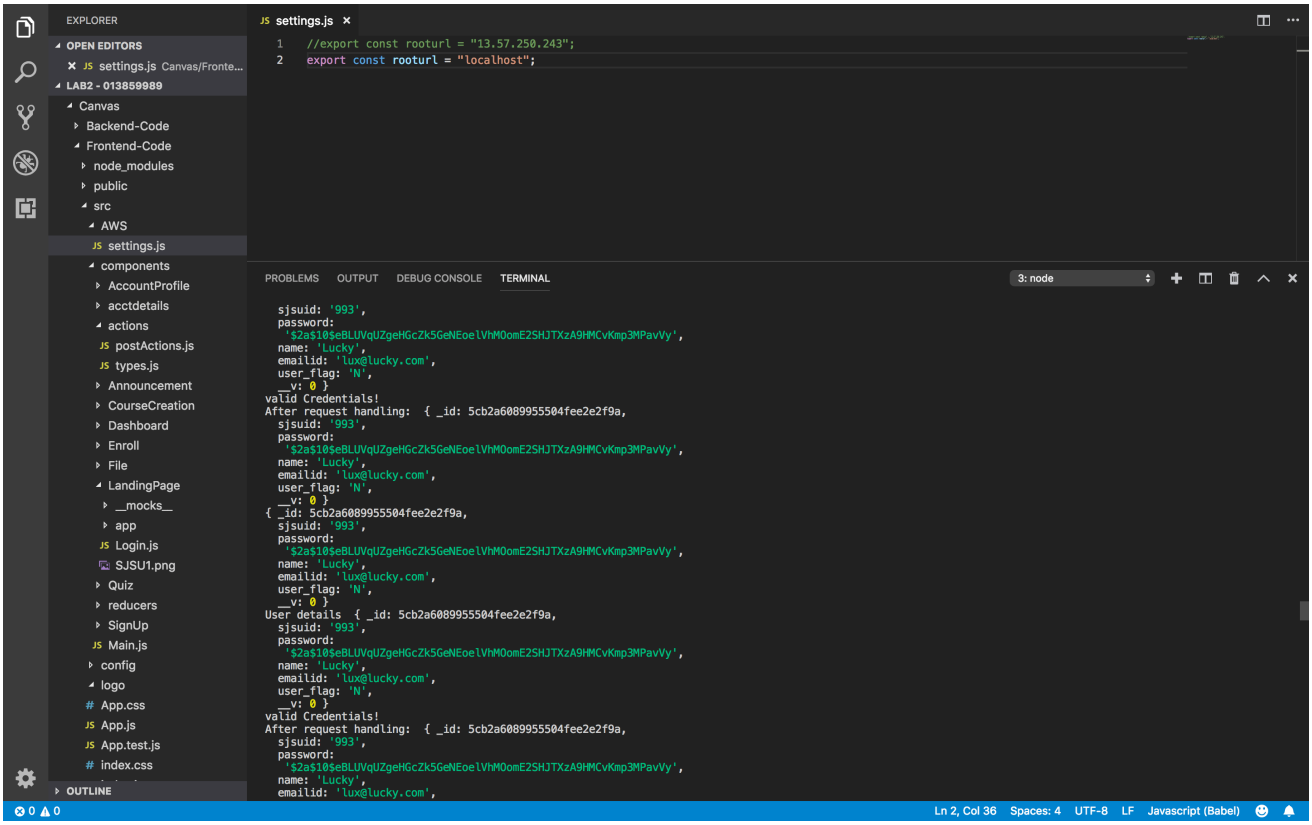

JMeter File Edit Search Run Options Tools Help
35% Sun 1:52 PM
canvas11 Thread Group.jmx (/Users/sachinwaghmode/Desktop/Lucky Screen Prints/Downloads/apache-jmeter-5.1/backups/canvas11 Thread Group.jmx) - Apache JMeter (5.1.1 r1855137)
00:00:24
0
0/100
Test Plan
Thread Group
Summary Report
Graph Results
HTTP Header Manager
HTTP Request
View Results Tree
Graph Results
Name:
Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Successes
Configure
Graphs to Display
Data
Average
Median
Deviation
Throughput
1588 ms
0 ms
No of Samples 110
Latest Sample 23011
Deviation 7993
Throughput 63.562/minute
Average 20078
Median 23263

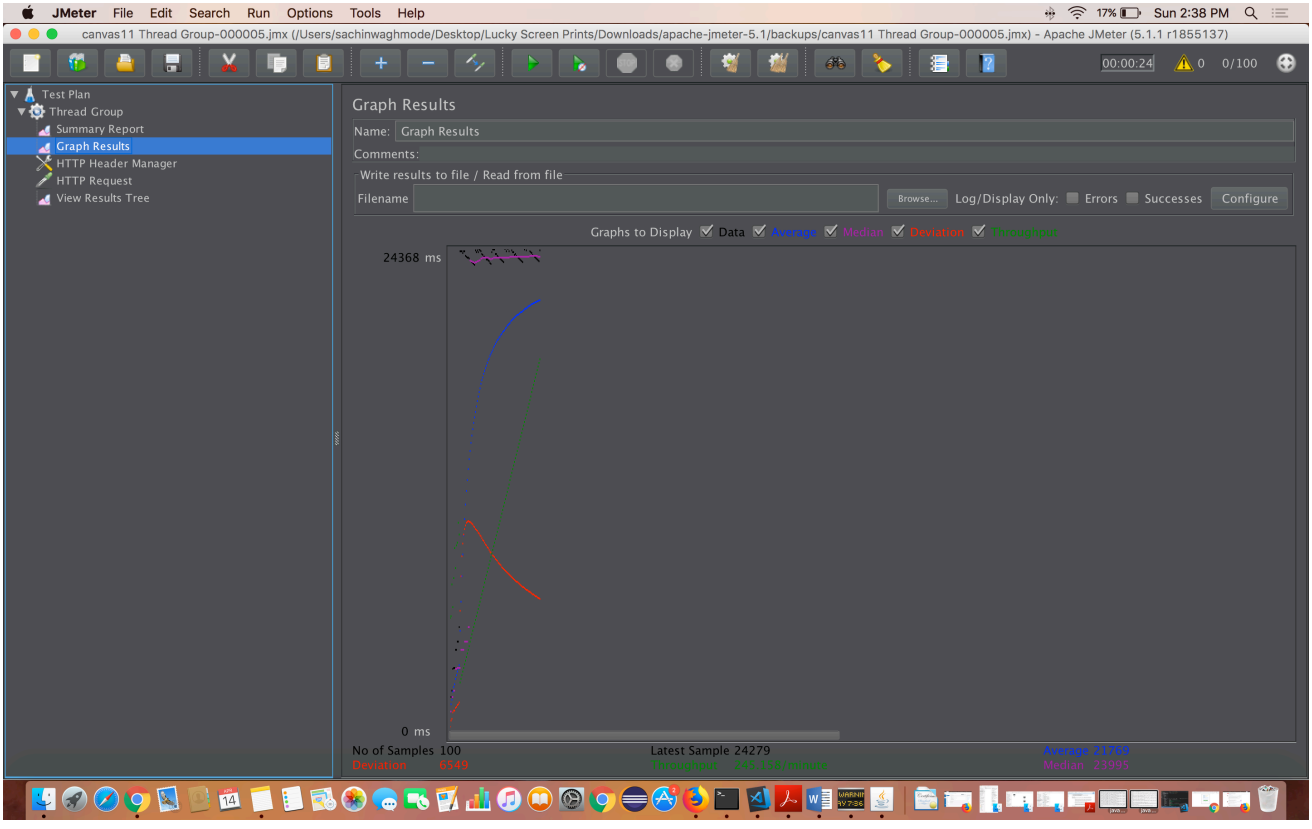
Below screen print is from the AWS where Server.js file is running

```
YL4xww2xiRAZKvmdvtTnVzDSJpE"}
msg received
response { _id: '5cb2a6549955504fee2e2f9c',
  sjsuid: '992',
  password: '$2a$10$QxJJBs97piELC7OO0fYE2u1HkIhCzEi/rqII19JWilM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  __v: 0 }
In results Signup
Results:  { _id: '5cb2a6549955504fee2e2f9c',
  sjsuid: '992',
  password: '$2a$10$QxJJBs97piELC7OO0fYE2u1HkIhCzEi/rqII19JWilM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  __v: 0 }
Logged in successfully.
{ _id: '5cb2a6549955504fee2e2f9c',
  sjsuid: '992',
  password: '$2a$10$QxJJBs97piELC7OO0fYE2u1HkIhCzEi/rqII19JWilM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  __v: 0 }
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTIiLCJpYXQiOjE1NTUyNzU2ODEsImV4cCI6MTU1NTI4NTc2MX0.PCIU1lIrWOQDMmNsYL4xww2xiRAZKvmdvtTnVzDSJpE
{"logincheck":[{"finalstatus":true,"facultyfnd":true,"pwdvalidity":true}],"Token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTIiLCJpYXQiOjE1NTUyNzU2ODEsImV4cCI6MTU1NTI4NTc2MX0.PCIU1lIrWOQDMmNs
YL4xww2xiRAZKvmdvtTnVzDSJpE"}
[ec2-user@ip-172-31-3-189 ~]$
```

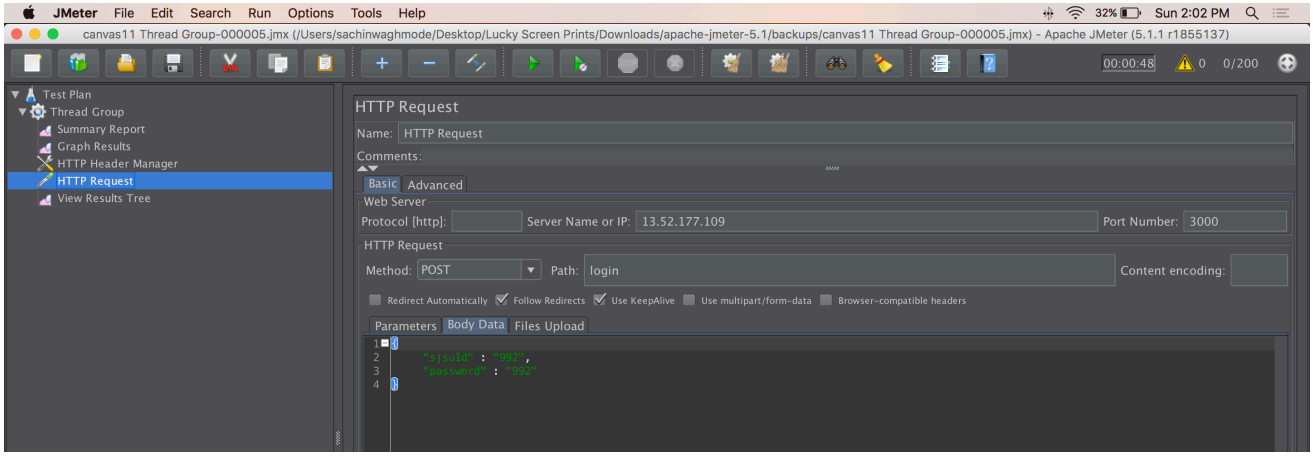
100 Concurrent Users: (Local run)

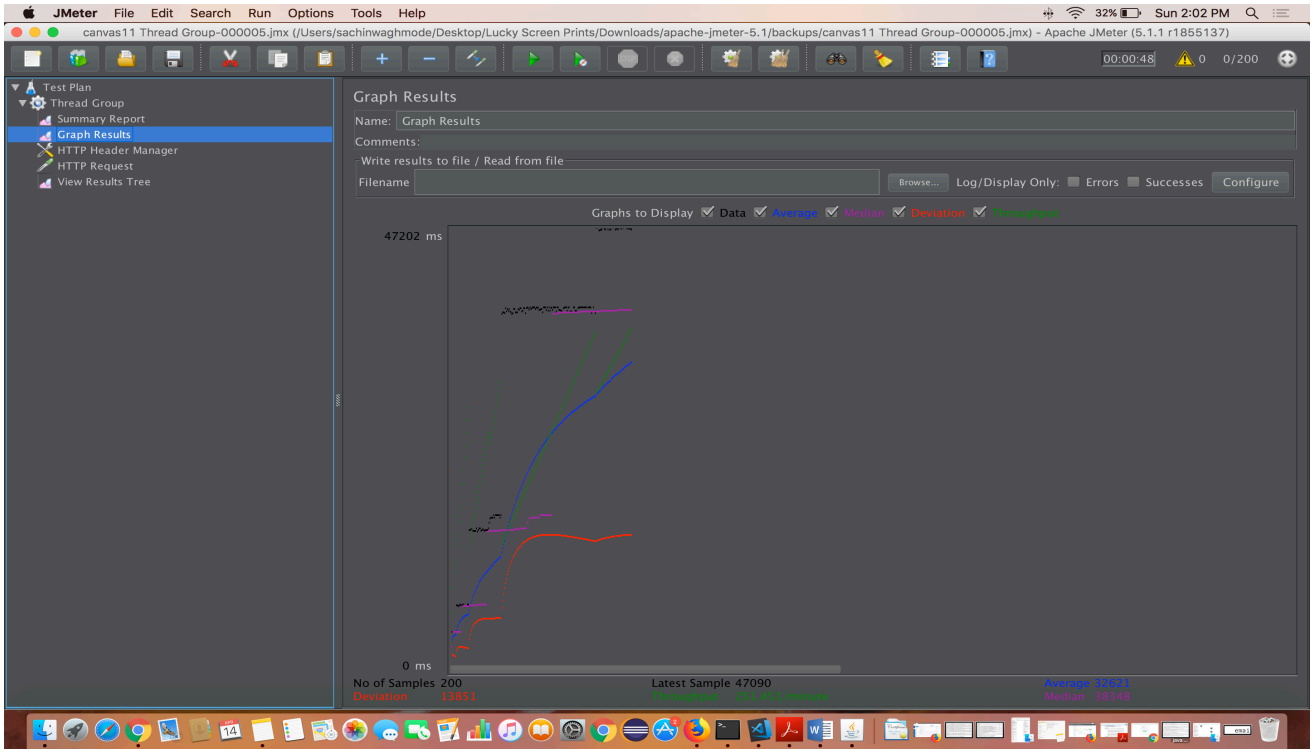



EXPLORER
OPEN EDITORS
settings.js Canvas/Fronte...
LAB2 - 013859989
Canvas
Backend-Code
Frontend-Code
node_modules
public
src
AWS
settings.js
components
AccountProfile
acctdetails
actions
postActions.js
types.js
Announcement
CourseCreation
Dashboard
Enroll
File
LandingPage
__mocks__
app
Login.js
SJSU1.png
Quiz
reducers
SignUp
Main.js
config
logo
App.css
App.js
App.test.js
index.css
OUTLINE
settings.js
1 //export const rooturl = "13.57.250.243";
2 export const rooturl = "localhost";
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
3: node
sjsuid: '993',
password:
'$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
user_flag: 'N',
__v: 0 }
valid Credentials!
After request handling: { _id: 5cb2a6089955504fee2e2f9a,
sjsuid: '993',
password:
'$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
user_flag: 'N',
__v: 0 }
{ _id: 5cb2a6089955504fee2e2f9a,
sjsuid: '993',
password:
'$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
user_flag: 'N',
__v: 0 }
User details { _id: 5cb2a6089955504fee2e2f9a,
sjsuid: '993',
password:
'$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
user_flag: 'N',
__v: 0 }
valid Credentials!
After request handling: { _id: 5cb2a6089955504fee2e2f9a,
sjsuid: '993',
password:
'$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
Ln 2, Col 36
Spaces: 4
UTF-8
LF
Javascript (Babel)


JMeter File Edit Search Run Options Tools Help
Sun 2:38 PM
canvas11 Thread Group-000005.jmx (/Users/sachinwaghmode/Desktop/Lucky Screen Prints/Downloads/apache-jmeter-5.1/backups/canvas11 Thread Group-000005.jmx) - Apache JMeter (5.1.1 r1855137)
00:00:24
0/100
Test Plan
Thread Group
Summary Report
Graph Results
HTTP Header Manager
HTTP Request
View Results Tree
Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
24368 ms
0 ms
No of Samples 100
Latest Sample 24279
Deviation 6549
Median 23995

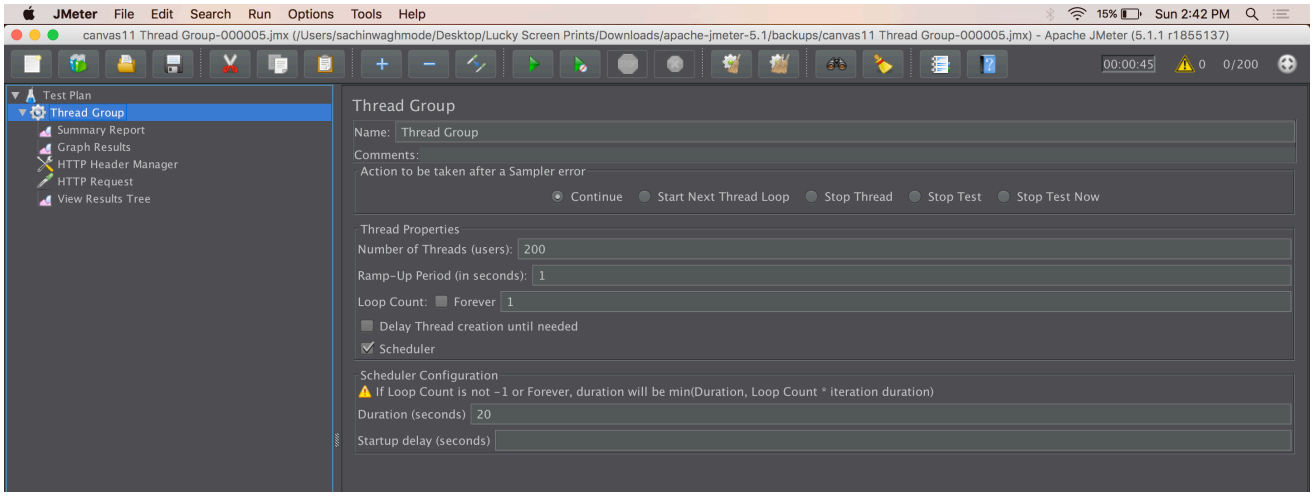
200 Concurrent Users: (Deployed on AWS)

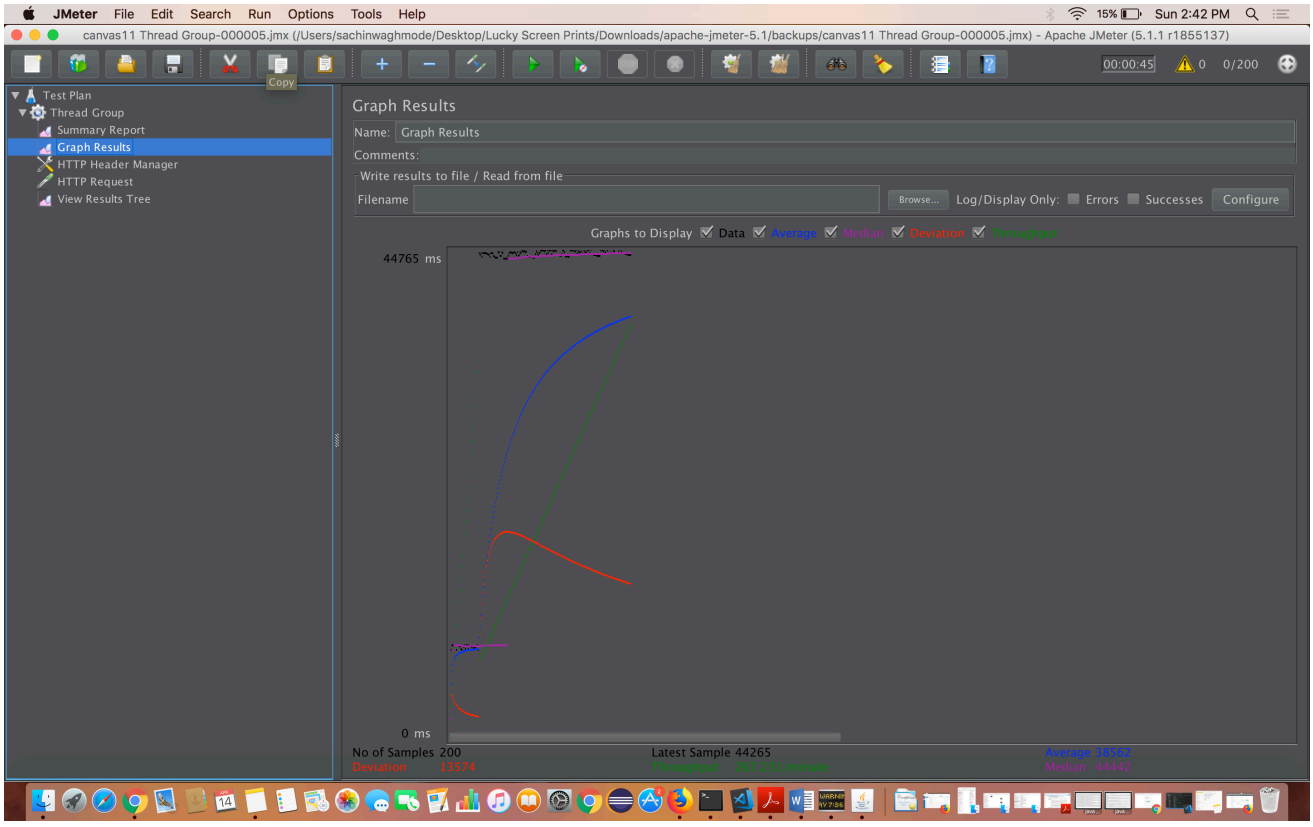


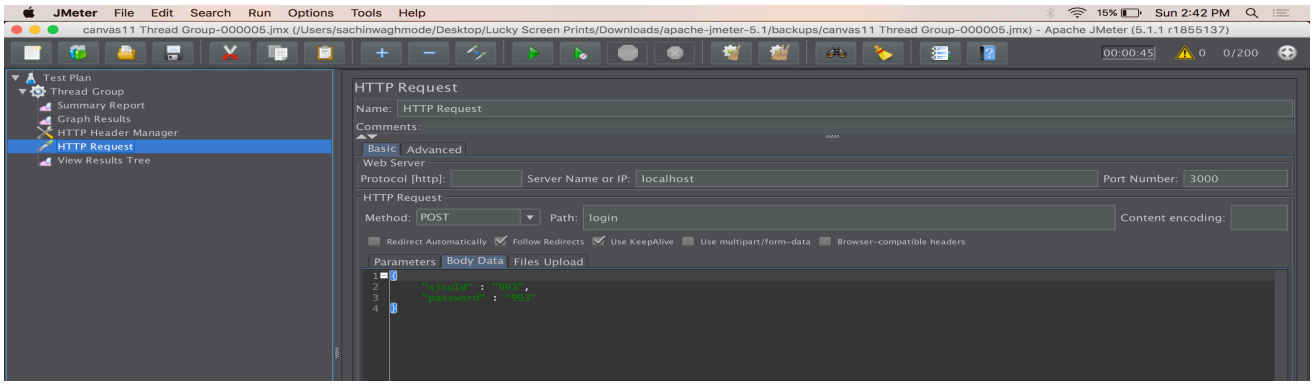



```
Last login: Sun Apr 14 00:25:33 on ttys004

{ _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTMiLCJpYXQiOjE1NTUyNzUxODMsImV4cCI6MTU1NTI4NTI2M30.jry5JjGig7OnpDouF040bVOkcRSOEku7jUNFsRQSyRM
{"logincheck":[{"finalstatus":true,"facultyfnd":false,"pwdvalidity":true}],"Token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzanN1aWQiOiI5OTMiLCJpYXQiOjE1NTUyNzUxODMsImV4cCI6MTU1NTI4NTI2M30.jry5JjGig7OnpDouF040bVOkcRSOEku7jUNFsRQSyRM"}
msg received
response { _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
In results Signup
Results:  { _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
  password: '$2a$10$eBLUVqUZgeHGcZk5GeNEoelVhMOomE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
Logged in successfully.
{ _id: '5cb2a6089955504fee2e2f9a',
  sjsuid: '993',
```

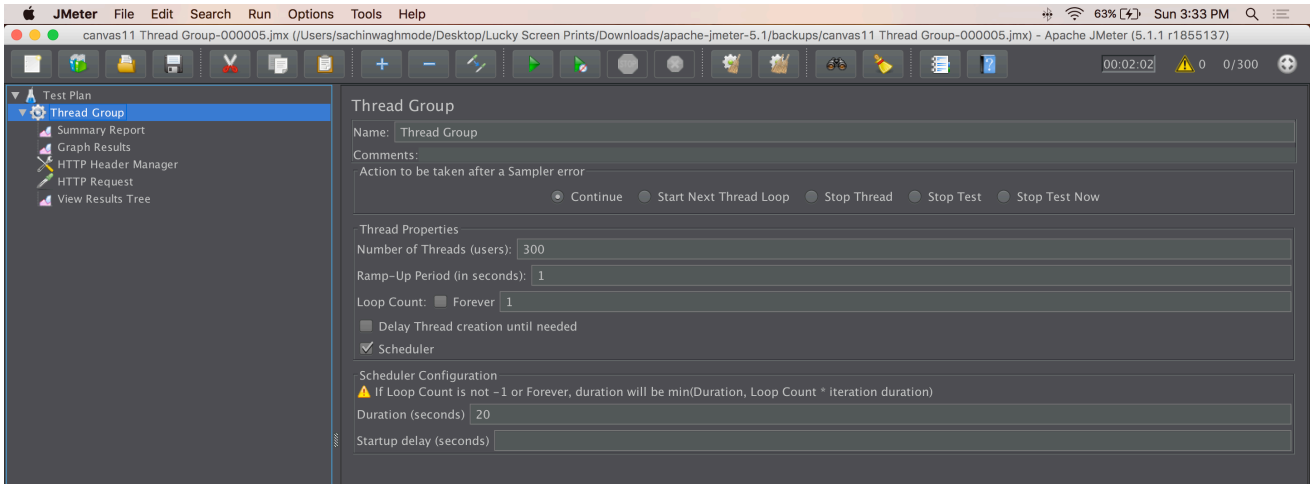
200 Concurrent Users: (Deployed on Local)

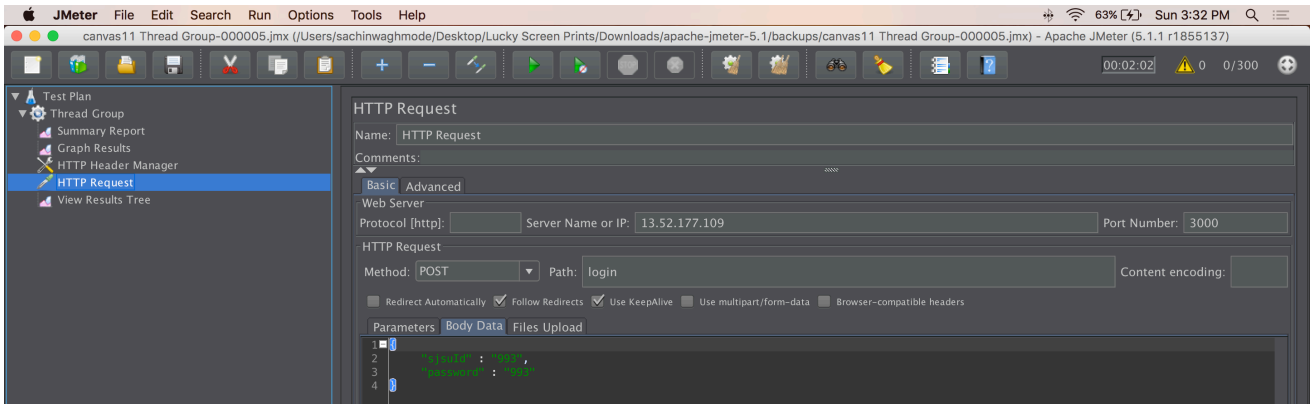


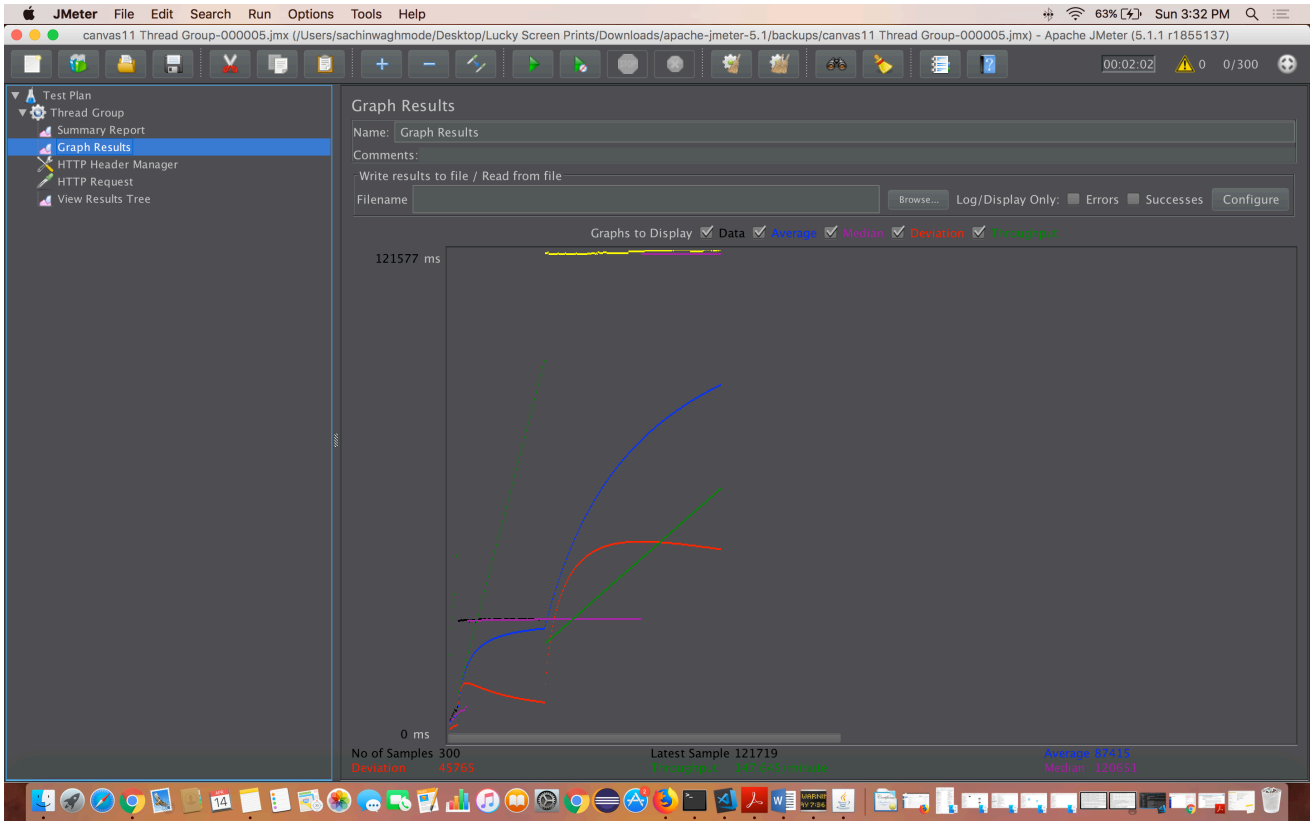




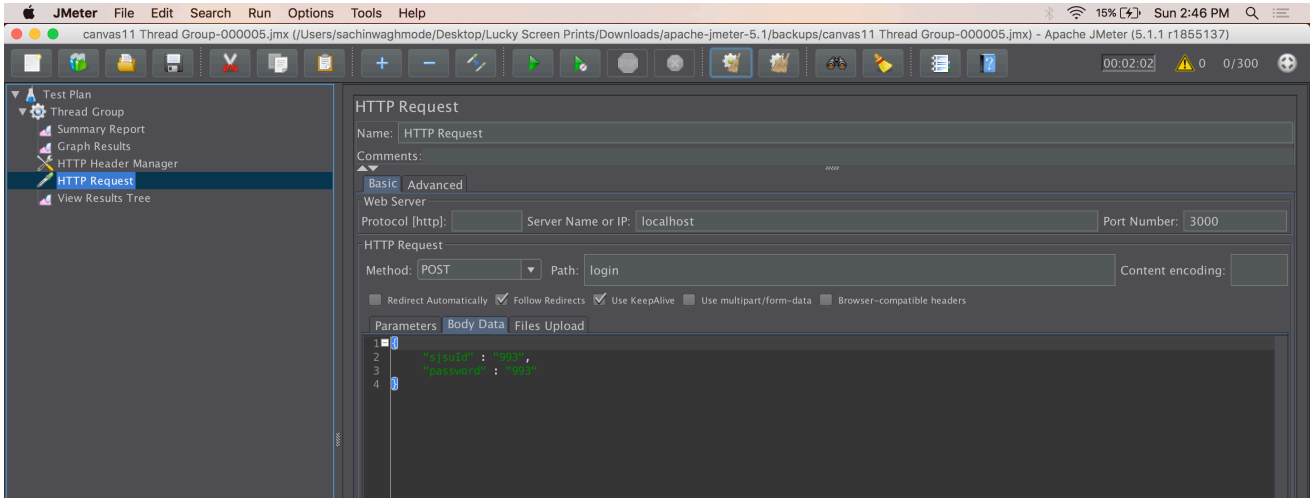
300 Concurrent Users: (Deployed on AWS)

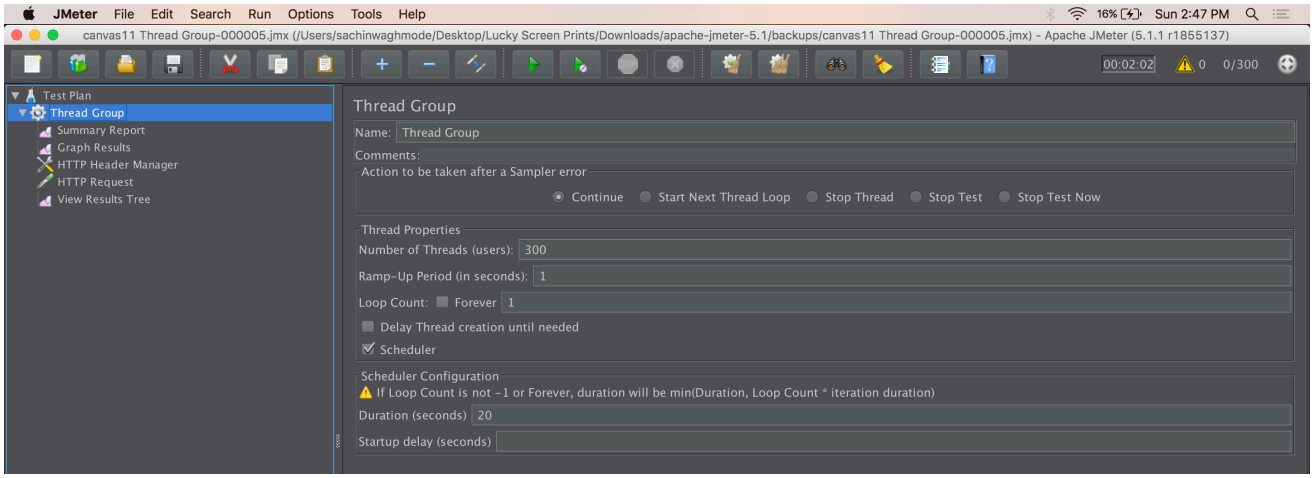


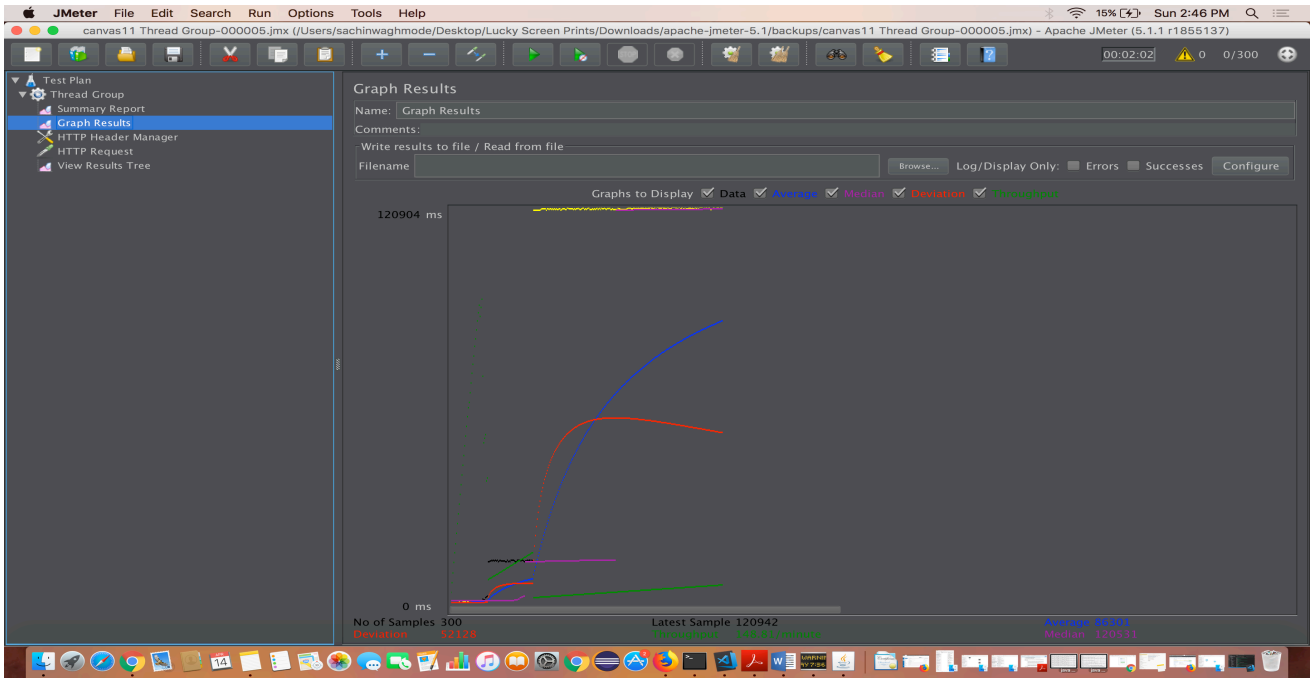




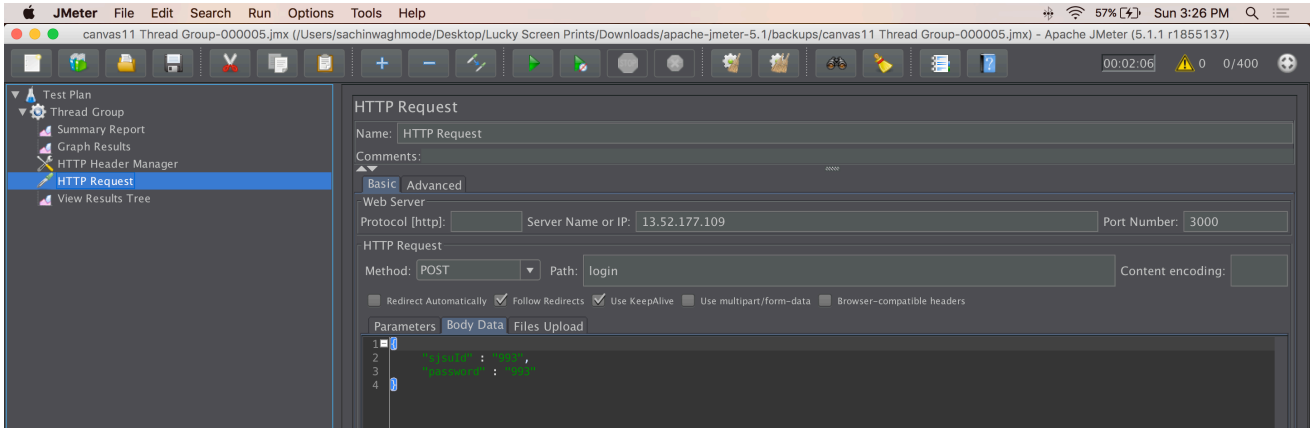
300 Concurrent Users: (Deployed on local)

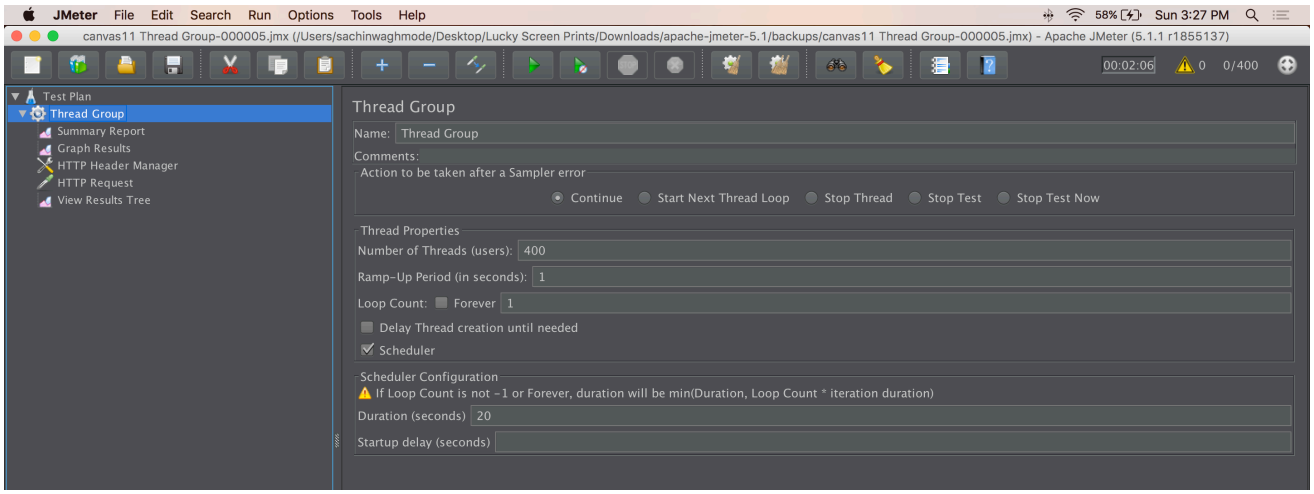


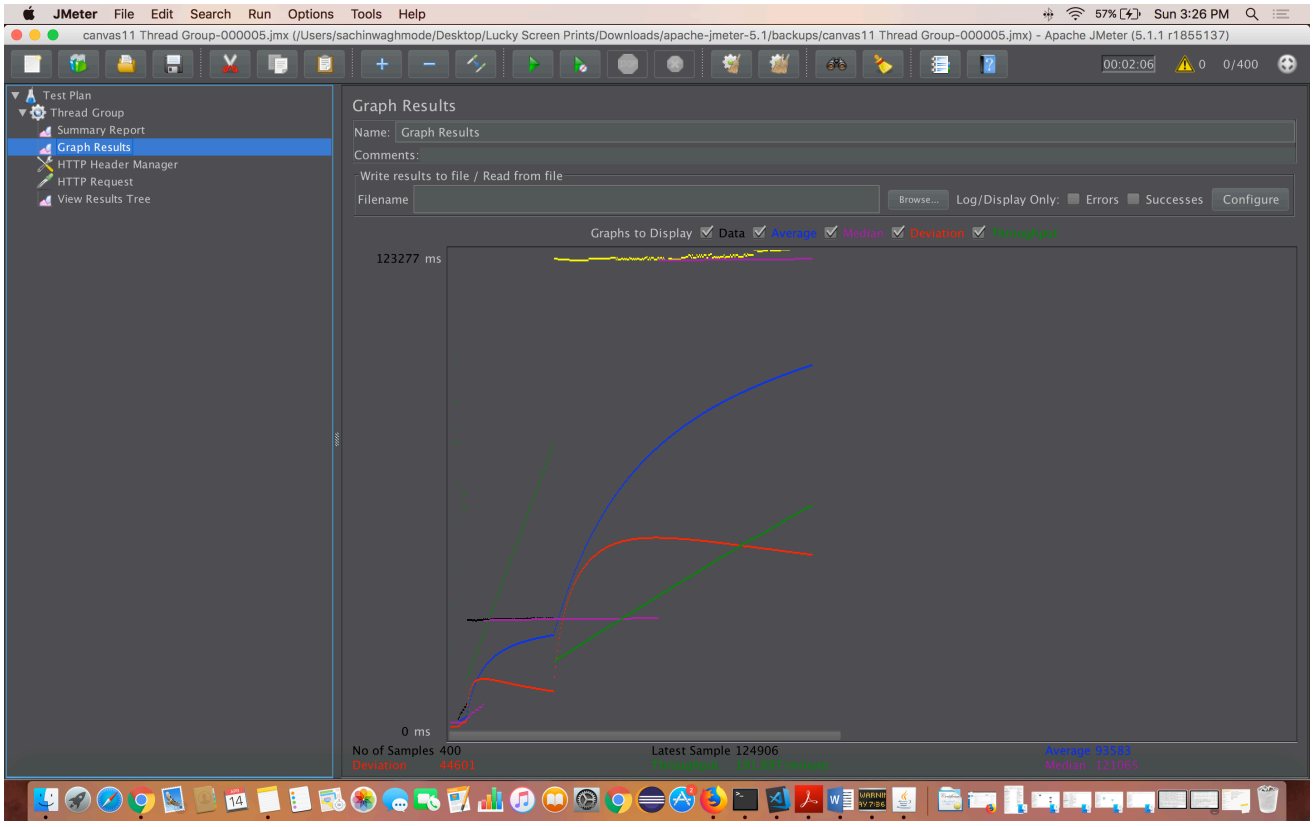




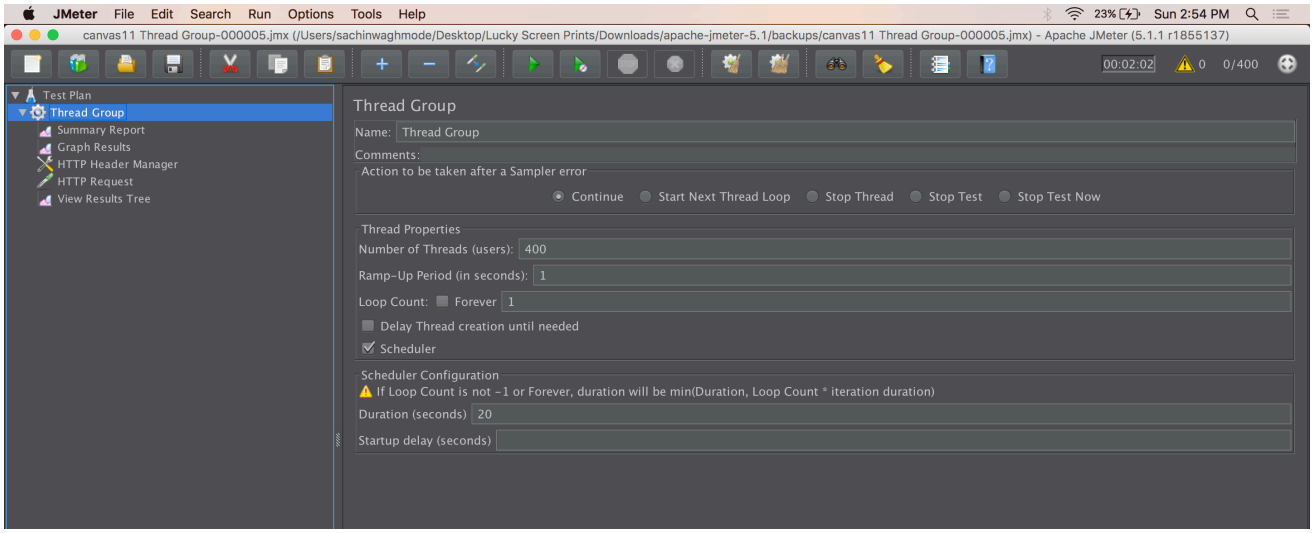
400 Concurrent Users: (Deployed on AWS)

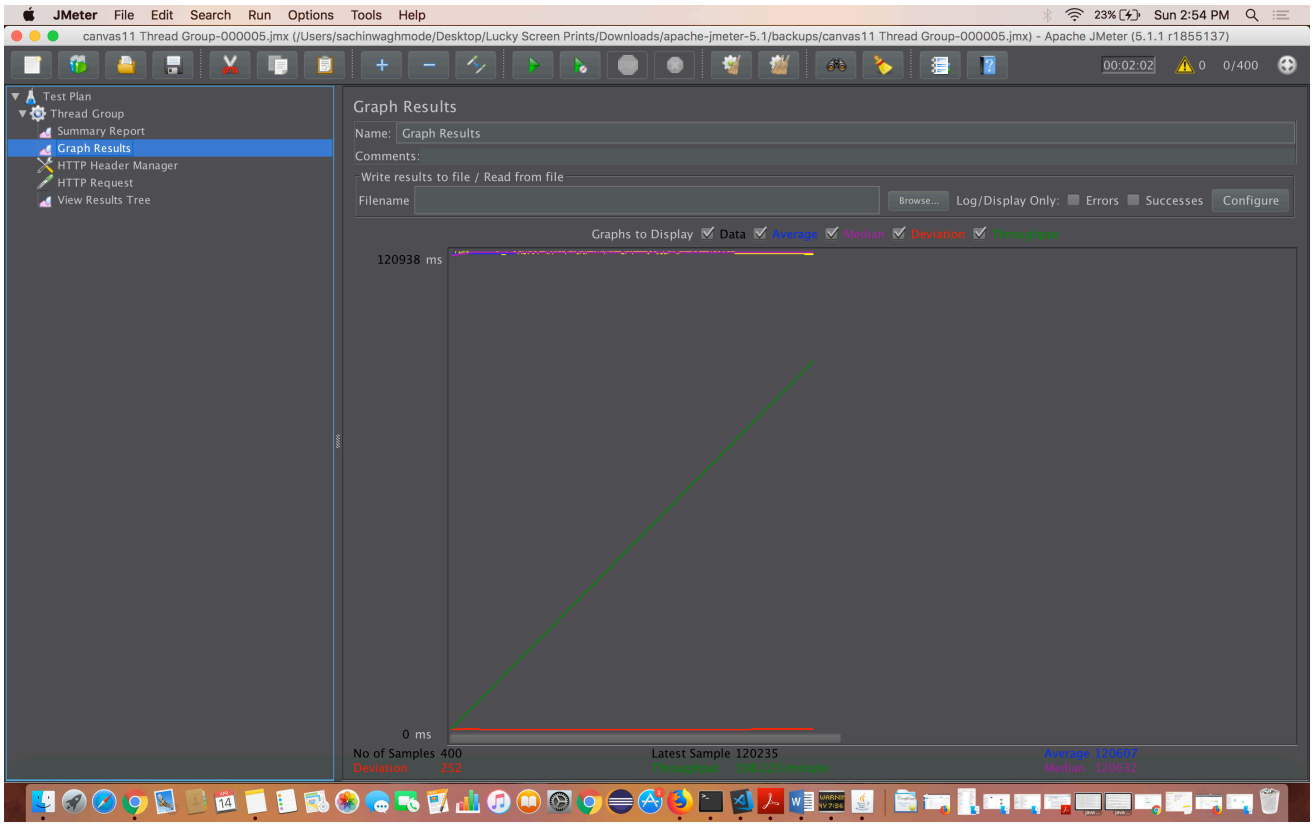




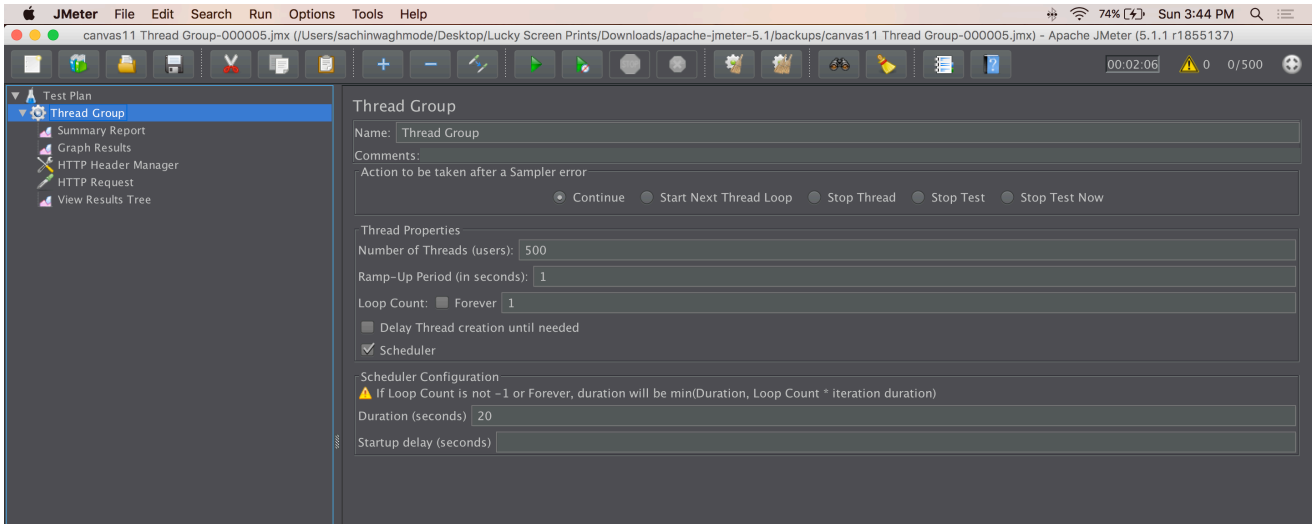


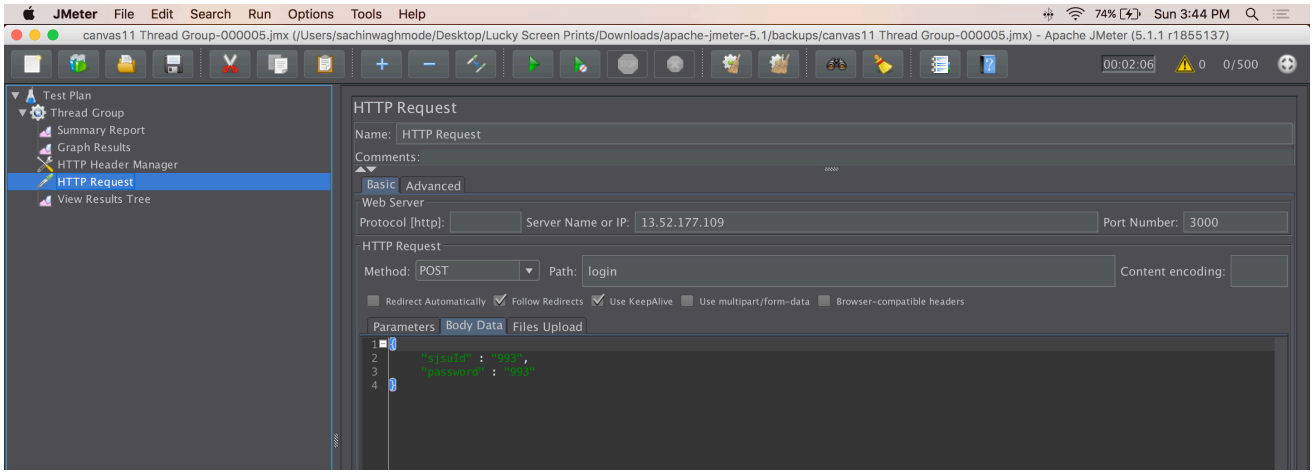
400 Concurrent Users: (Deployed on local)

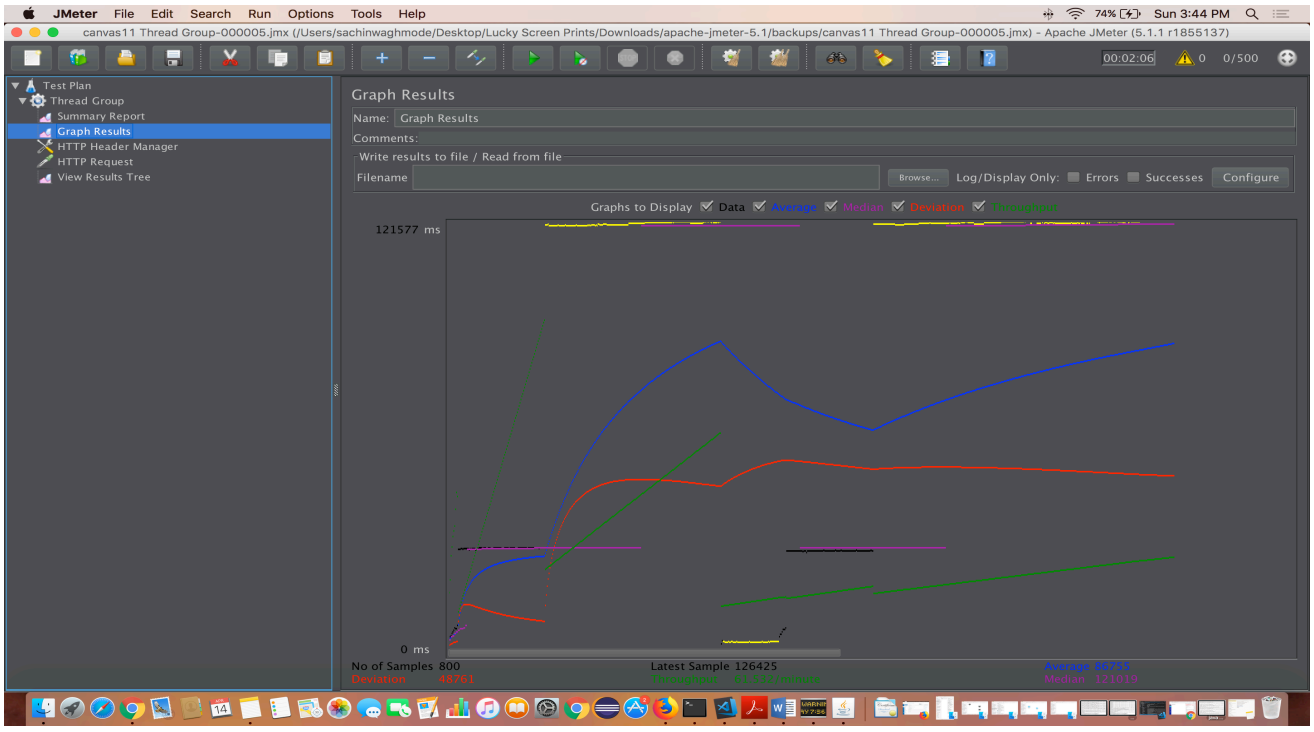




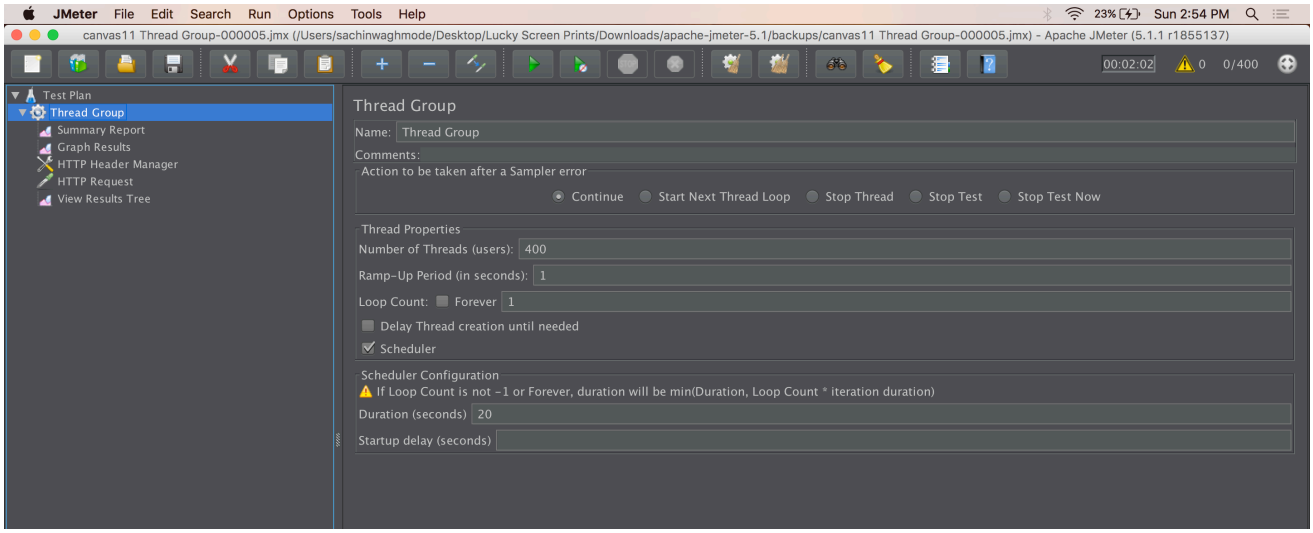
400 Concurrent Users: (Deployed on AWS)

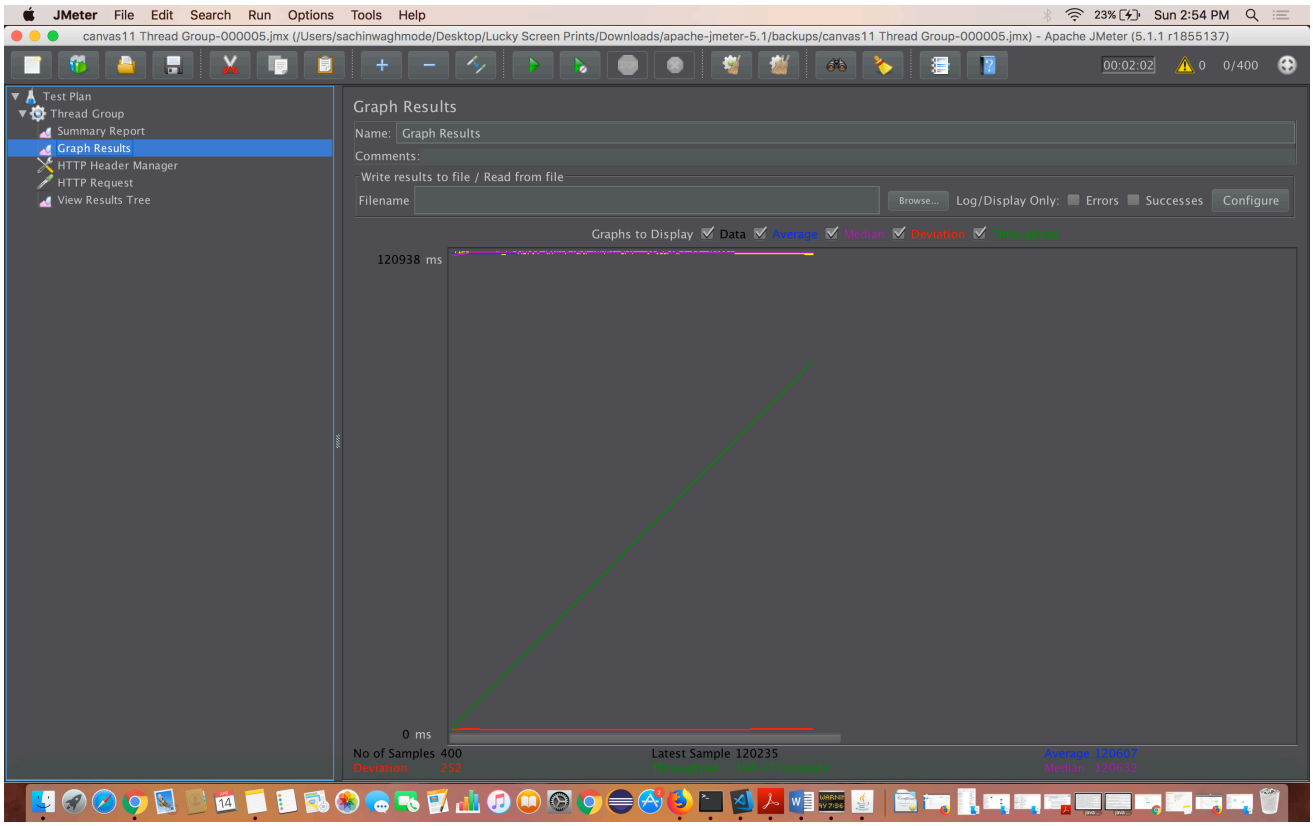




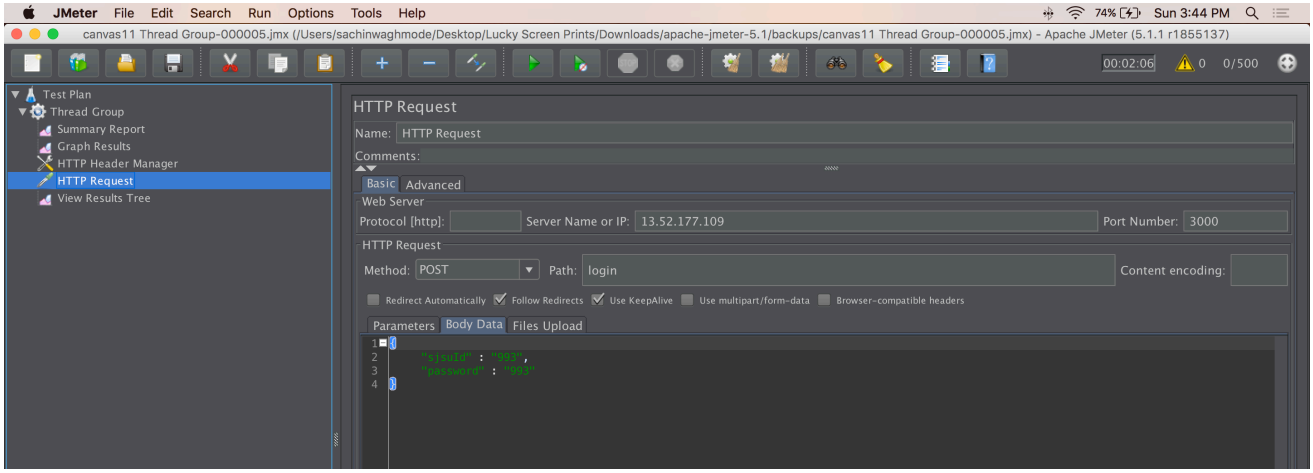


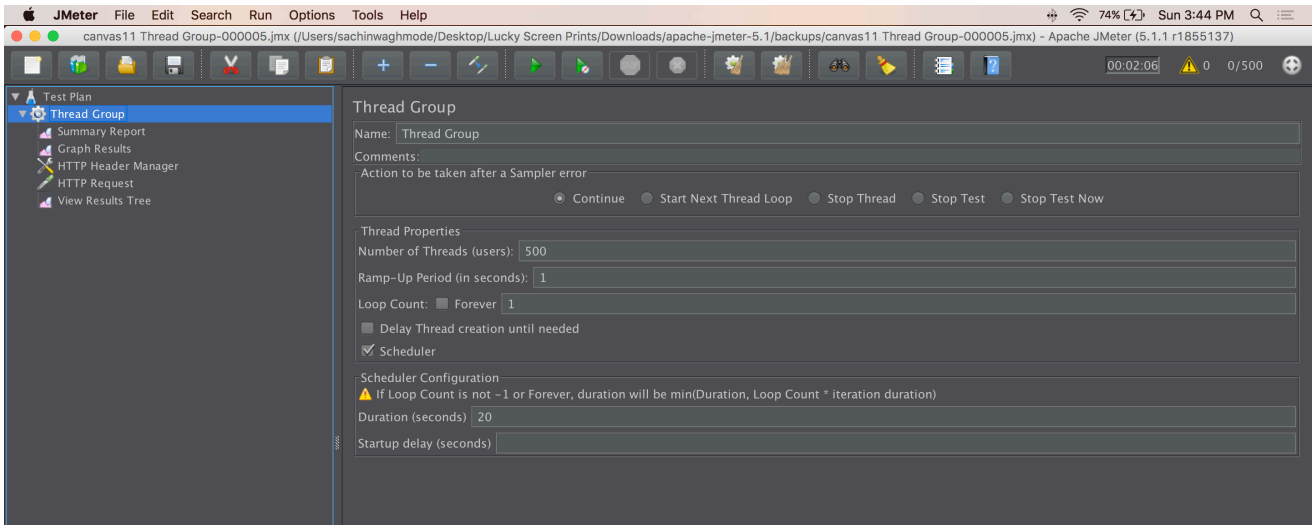
400 Concurrent Users: (Deployed on Local)

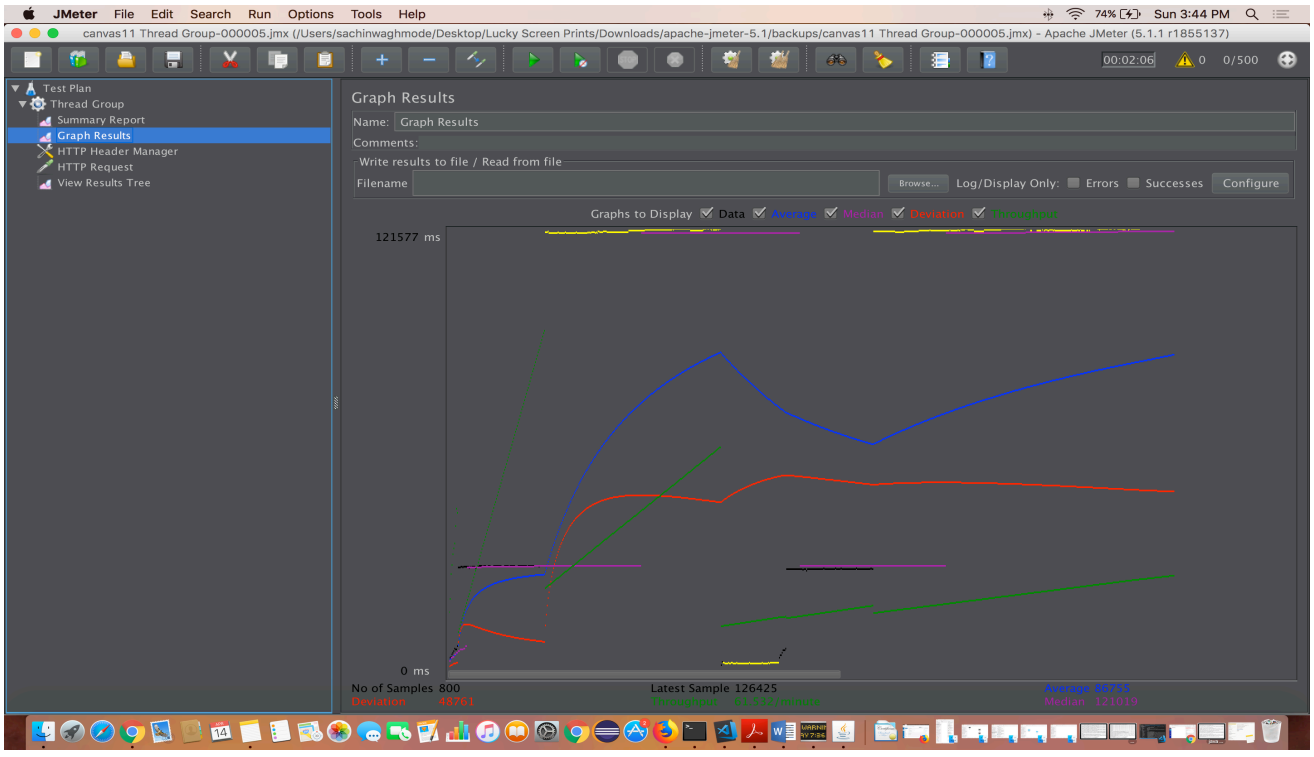




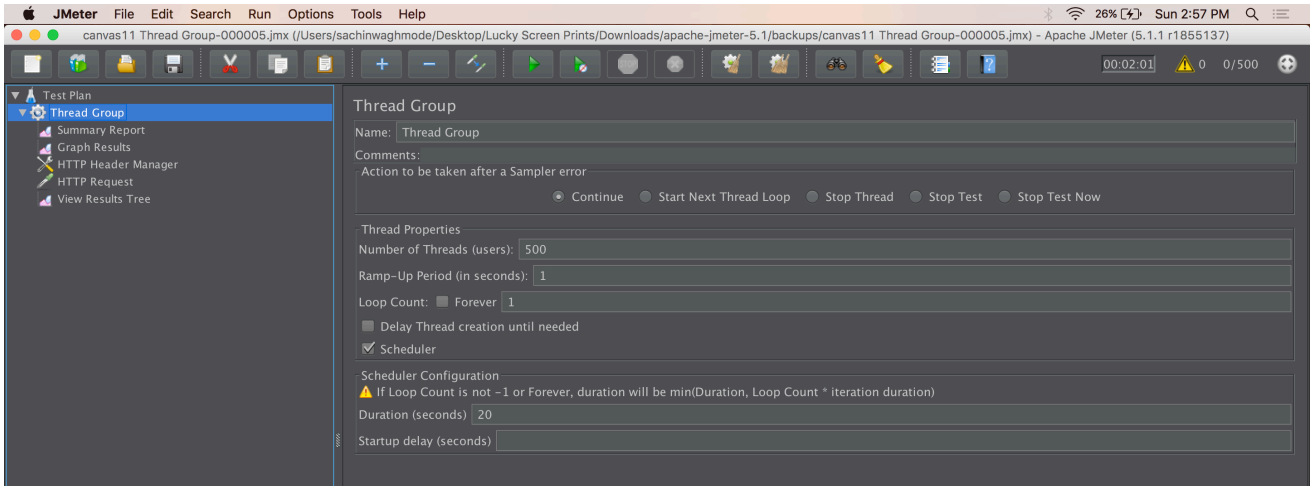
500 Concurrent Users: (Deployed on AWS)

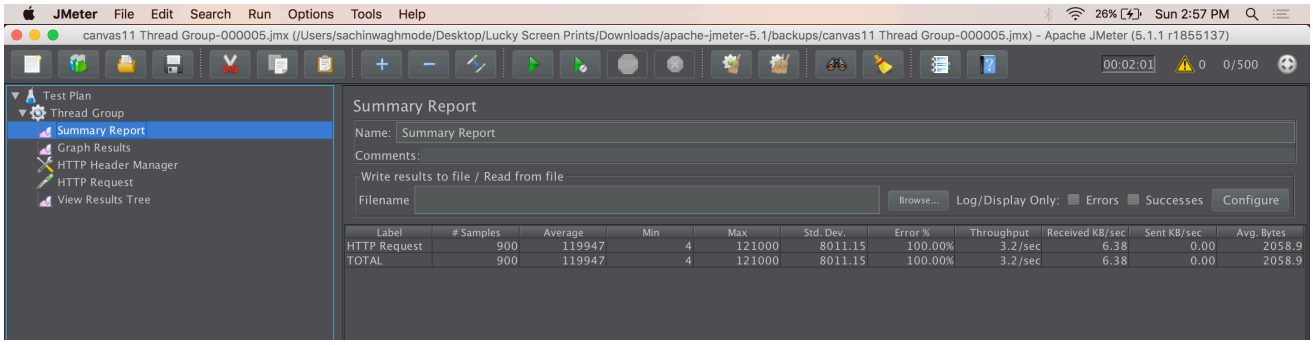


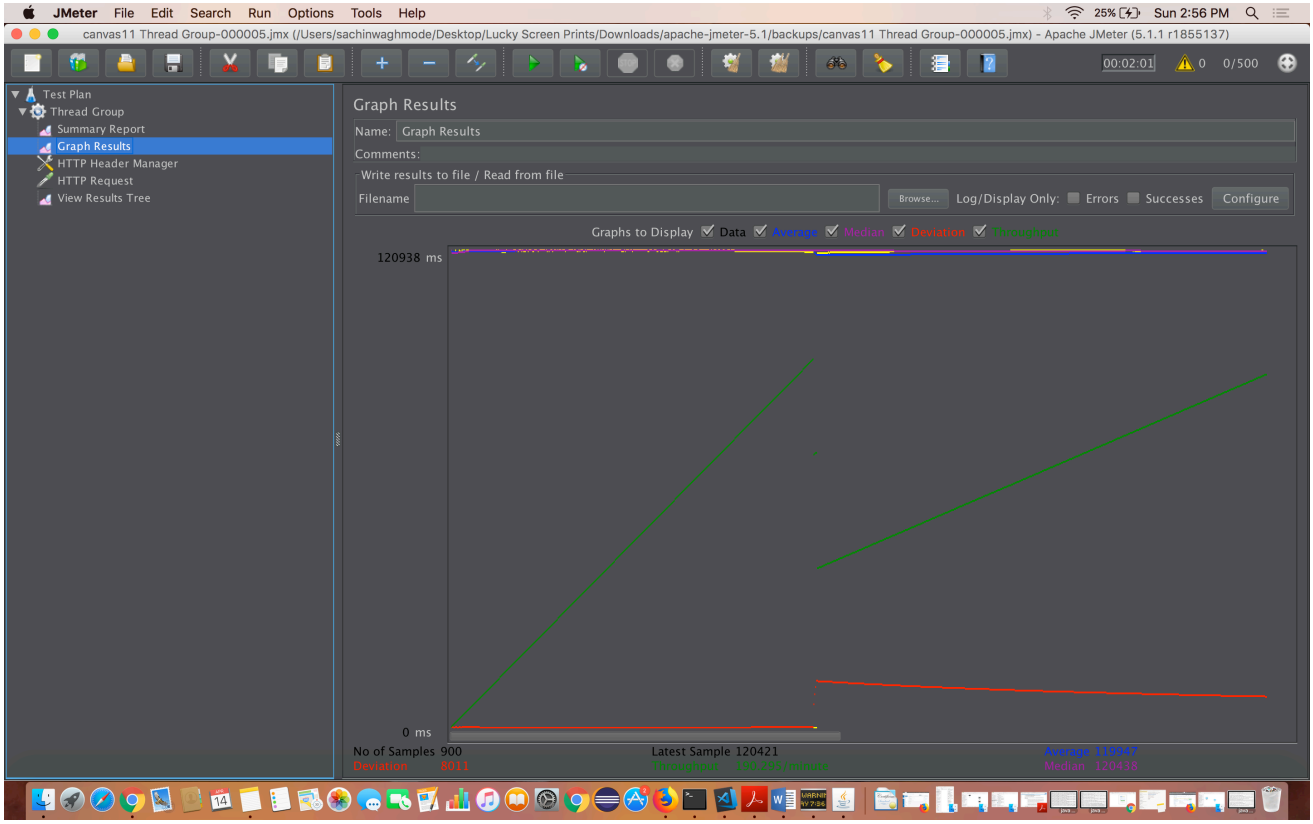




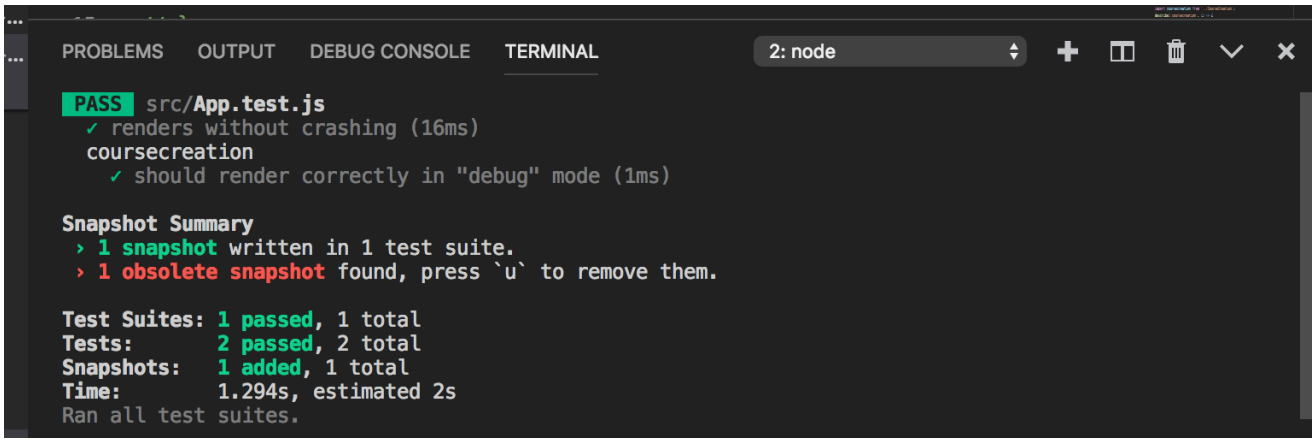
500 Concurrent Users: (Deployed on local)

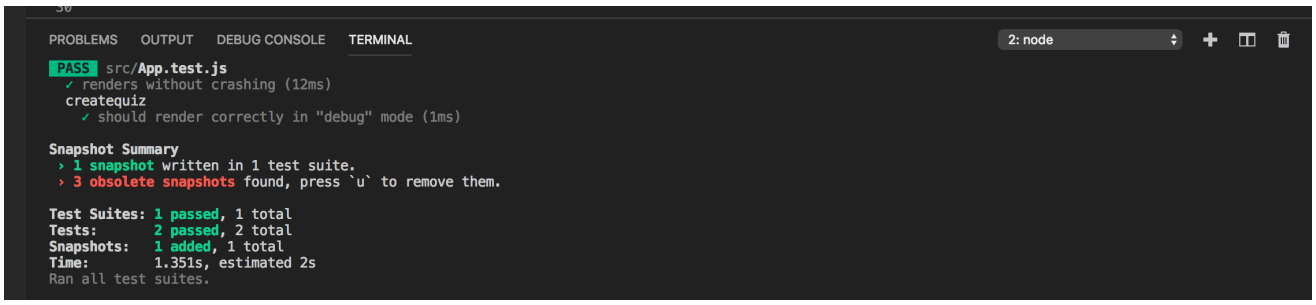


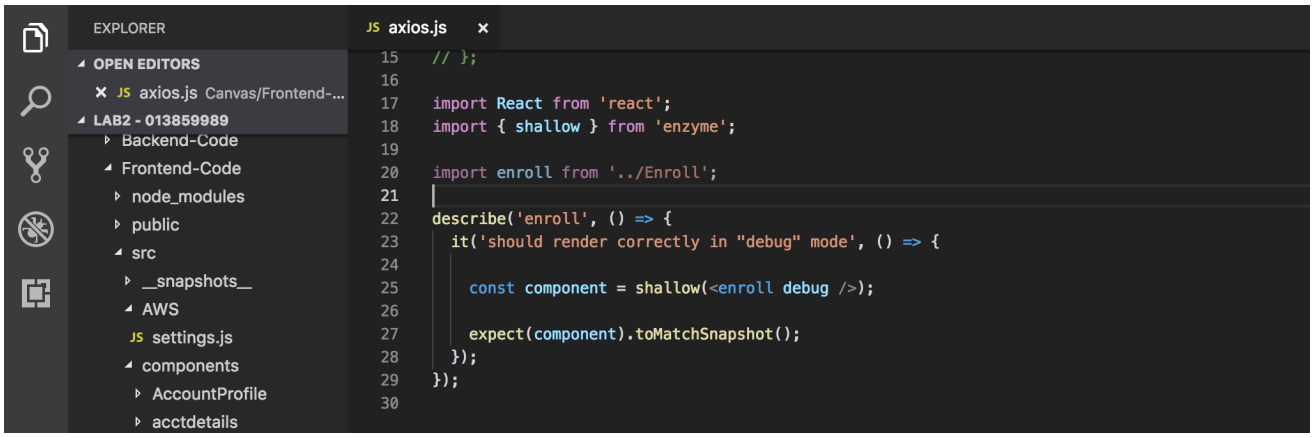




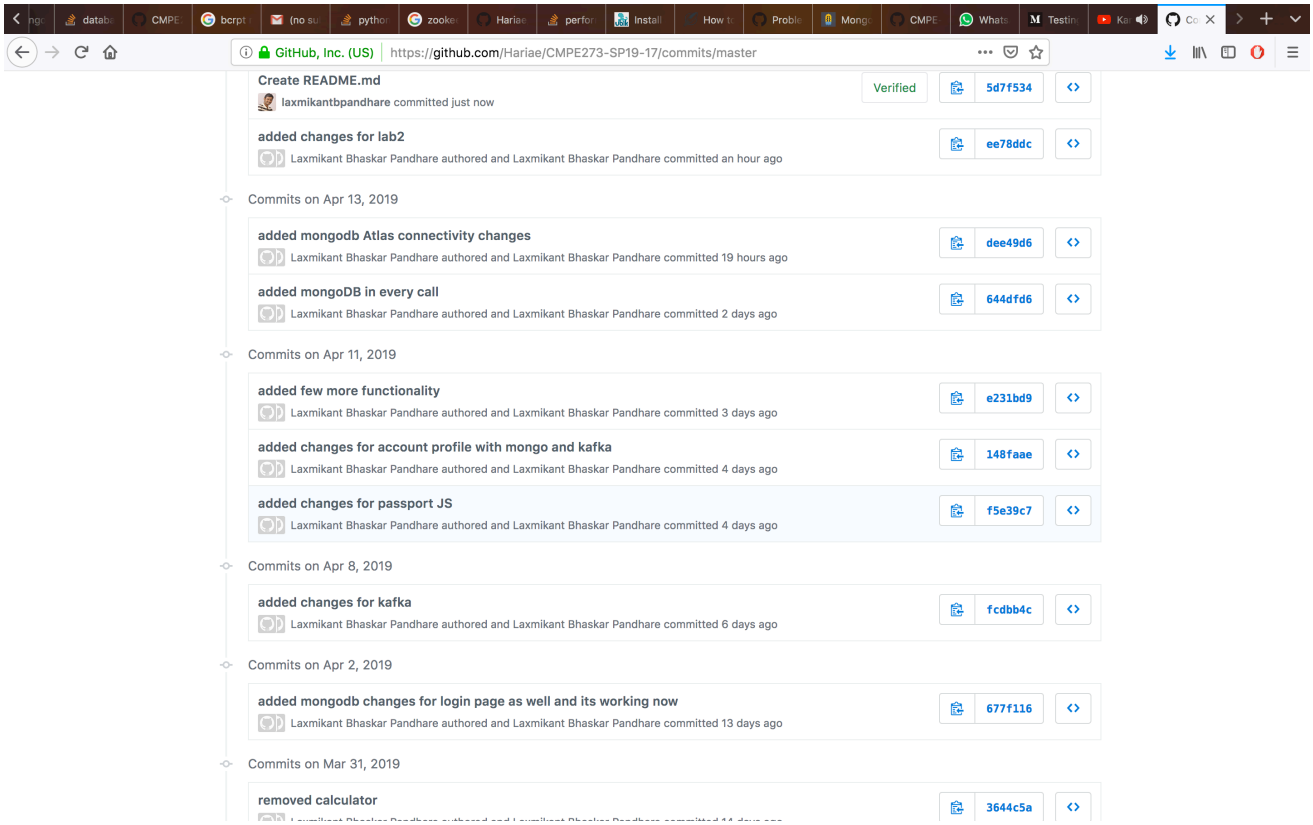
Completed **Jest Enzyme** Test for Course Creation, Quiz Creation and for Course Enrollment.

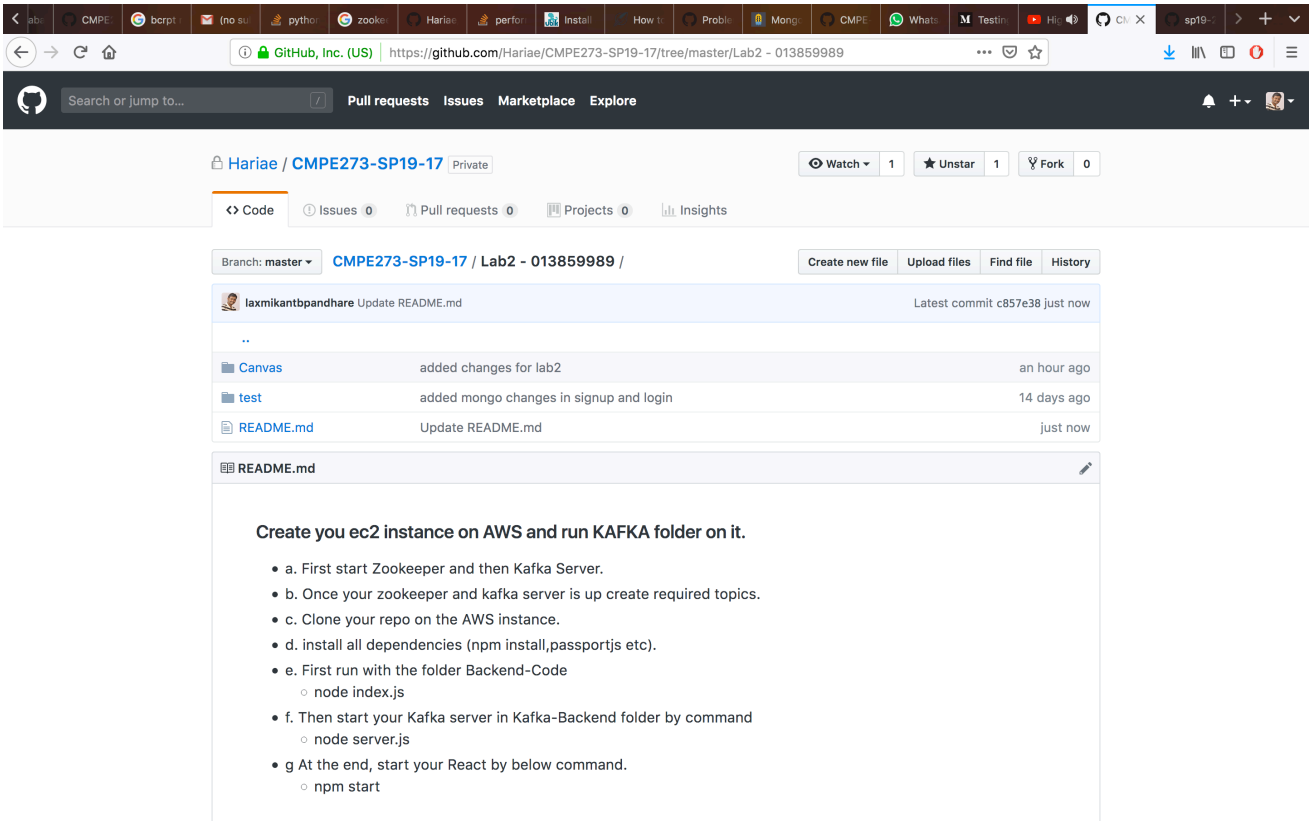






GITHUB commits and steps added in README.md →





GitHub link → https://github.com/Hariae/CMPE273-SP19-17/tree/master/Lab2%20-%20013859989

EC2 Instance Created → ec2-13-52-177-109.us-west-1.compute.amazonaws.com

Port : http://13.52.177.109:3000/