

**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 course. 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.

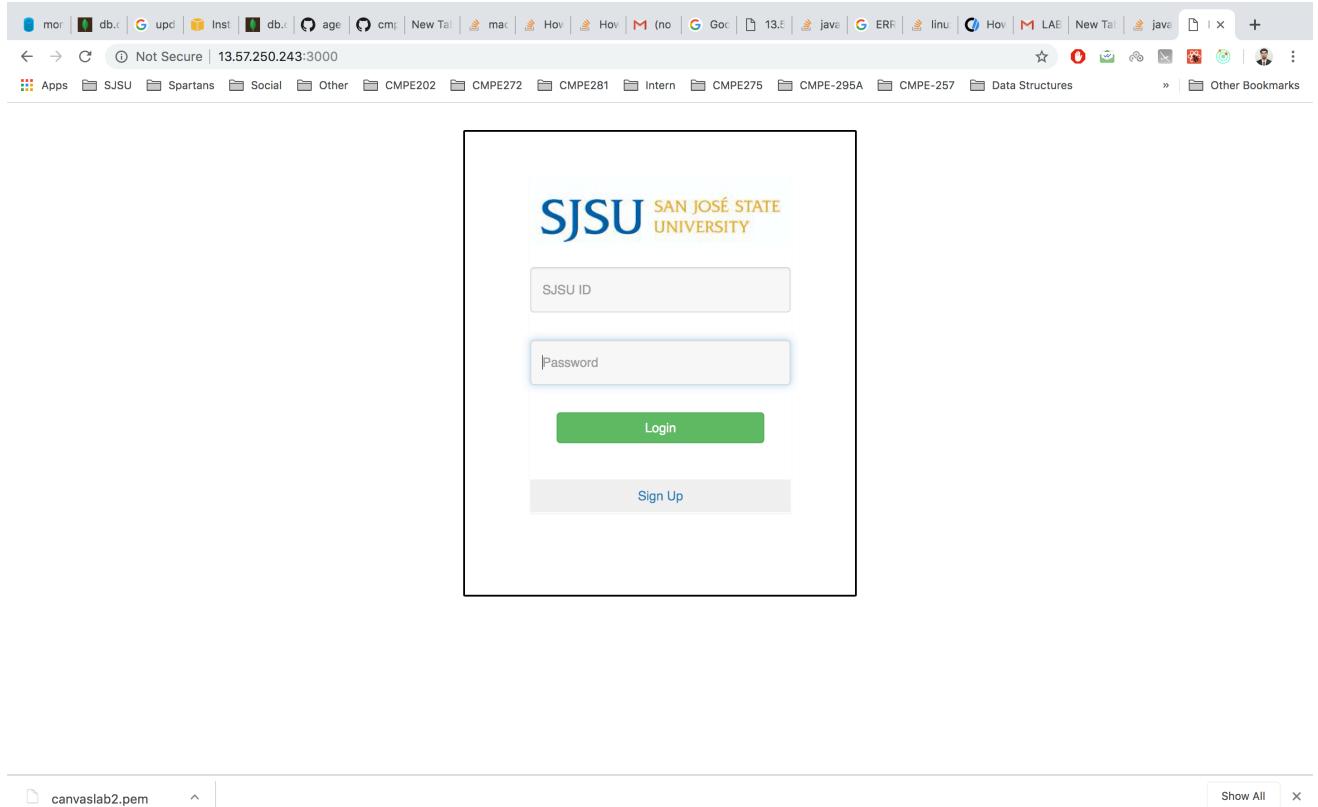
Where MongoDB provides availability as compared to MySQL. There will not be any issues with the availability if MongoDB but in some of the times there will be issues with the availability of MySQL during the execution of an application.

**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.eyJzaanNiaWQ10i150TiiLCJpYXQ10jE1NTUyMjA2MDgsImV4cCI6MTU1NTIzMDY40H0.EFB6jCcCiu0vnFVtJgb8mfgqpu1UKi1b_43bNcq-U4
{"logincheck":{"finalstatus":true,"facultyfnd":true,"pwdvalidity":true}},{"Token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJzaanNiaWQ10i150TiiLCJpYXQ10jE1NTUyMjA2MDgsImV4cCI6MTU1NTIzMDY40H0.EFB6jCcCiu0vnFVt
Jgb8mfgqpu1UKi1b_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": "5cb2a608995504fee2e2f9a",
  sjsuId: '993',
  password: '$2a$10$eBLUVquZgeHGcZk5GeNEoelVhM0omE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
In results Signup
Results { "_id": "5cb2a608995504fee2e2f9a",
  sjsuId: '993',
  password: '$2a$10$eBLUVquZgeHGcZk5GeNEoelVhM0omE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
Logged in successfully.
{ "_id": "5cb2a608995504fee2e2f9a",
  sjsuId: '993',
  password: '$2a$10$eBLUVquZgeHGcZk5GeNEoelVhM0omE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJzaanNiaWQ10i150TiiLCJpYXQ10jE1NTUyMjA2MDgsImV4cCI6MTU1NTIzMDcwNH0.zFAAA1o7xzt04-L3FASRR3GK_Ubc05yer2iM0f3VJ9
{"logincheck":[{"finalstatus":true,"facultyfnd":false,"pwdvalidity":true}],"Token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJzaanNiaWQ10i150TiiLCJpYXQ10jE1NTUyMjA2MDgsImV4cCI6MTU1NTIzMDcwNH0.zFAAA1o7xzt04-L
3FASRR3GK_Ubc05yer2iM0f3VJ9"
[ec2-user@ip-172-31-3-189 ~]$ 

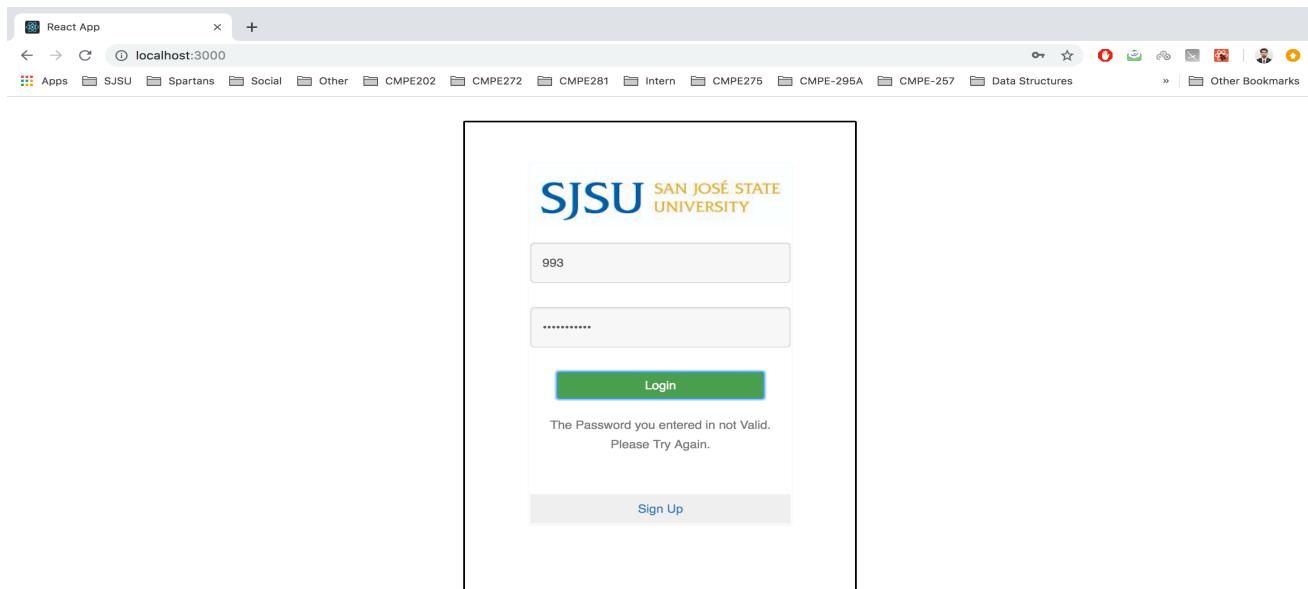
```

```

Downloads — ec2-user@ip-172-31-3-189:~ — ssh ec2-user@13.57.250.243 -i canvaslab2.pem — 193x45
[...]
User details { _id: 5cb2a654995504fee2e2f9c,
  sjsuId: '992',
  password: '$2a$10$QxJJBs97piELC7000fYE2u1HKIhCzEi/rqII19JWilM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  __v: 0 }
valid Credentials!
After request handling: { _id: 5cb2a654995504fee2e2f9c,
  sjsuId: '992',
  password: '$2a$10$QxJJBs97piELC7000fYE2u1HKIhCzEi/rqII19JWilM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  __v: 0 }
Data: { response_topic: { '0': 0 } }
message received for login [object Object]
"{"correlationId":"\\\"daci8de208e5f4594d6aae7577965916\\\",\\\"replyTo\\\":\\\"response_topic\\\",\\\"data\\\":{\\\"sjsuId\\\":\\\"993\\\",\\\"password\\\":\\\"993\\\"}}"
Inside Kafka Backend Signup
Message: { sjsuId: '993', password: '993' }
{ "_id": "5cb2a608995504fee2e2f9a",
  sjsuId: '993',
  password: '$2a$10$eBLUVquZgeHGcZk5GeNEoelVhM0omE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
User details { _id: 5cb2a608995504fee2e2f9a,
  sjsuId: '993',
  password: '$2a$10$eBLUVquZgeHGcZk5GeNEoelVhM0omE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
valid Credentials!
After request handling: { _id: 5cb2a608995504fee2e2f9a,
  sjsuId: '993',
  password: '$2a$10$eBLUVquZgeHGcZk5GeNEoelVhM0omE2SHJTXzA9HMCvKmp3MPavVy',
  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.



```
if finalstatus : false, facultyfnd : false, pwdvalidity : false}
Inside Login Post Request
Req Body : { sjsuid: '992', password: 'dsjk' }
992
val1sjsuid 992
val1pwd dsjk
encrypted 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.

The screenshot shows a web browser window with the URL <https://13.57.250.243:3000/signup>. The page title is "SJSU SAN JOSÉ STATE UNIVERSITY". The form fields are as follows:

- Year: 1999
- Name: Lucky
- Email: sd@dsf.fdf
- Password: \*\*\*\*
- Please click here if you are a Faculty.
- Profile Image: Choose File No file chosen
- Sign Up

The screenshot shows the MongoDB Atlas interface with the URL <https://cloud.mongodb.com/v2/5cb0fd62014b7660410329c4#metrics/replicaSet/5cb2a091cf09a225c287d9d0/>. The left sidebar includes:

- Project 0
- Clusters
- Alerts
- Backup
- Access
- Settings
- Stitch Apps
- Charts
- Docs
- Support

The main content area shows two document snippets:

```

courseid: "9129"
courseName: "dsf"
courseDept: "dsf"
courseDesc: "dsfdf"
courseRoom: "234"
courseCapacity: "10"
courseWaitList: "4"
courseTerm: "sfd"
__v: 0

_id: ObjectId("5cb2ce273f13d061477b33bb")
courseId: "9129"
courseName: "dsfjk"
courseDept: "wef"
courseDesc: "edfv"
courseRoom: "10"
courseCapacity: "10"
courseWaitList: "1"
courseTerm: "ds"
__v: 0

```

Below the documents are edit, delete, and copy icons. At the bottom, there is a footer with system status and support links.

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

The screenshot shows a Chrome browser window with the title bar "Chrome" and various menu options like File, Edit, View, History, Bookmarks, People, Window, and Help. Below the title bar, there's a toolbar with several icons. The main content area is titled "SJSU Dashboard". On the left, there's a sidebar with the following items:

- ACCOUNT (with a user profile icon)
- ANNOUNCEMENT (with a speech bubble icon)
- QUIZ (with a cube icon)
- GRADE (with a graduation cap icon)
- COURSES (with a document icon)
- ENROLL (with a graduation cap icon)
- HELP (with a question mark icon)
- LOGOUT (with a cross icon)

The main dashboard area is currently empty. At the bottom of the screen, the Mac OS X Dock is visible with various application icons.

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.

SJSU Dashboard

Please verify Below Details.

993

Lucky

lux@lucky.com

Change Details

ACCOUNT

DASHBOARD

COURSES

ENROLL

HELP

LOGOUT

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

SJSU Dashboard

Please verify Below Details.

993

Lucky

lux@lucky.com

Change Details

ACCOUNT

DASHBOARD

COURSES

ENROLL

HELP

LOGOUT

13.57.250.243:3000 says  
You are at SJSU Canvas DashBoard Page

OK

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

The screenshot shows a Chrome browser window with the URL [13.57.250.243:3000/Dashboard](http://13.57.250.243:3000/Dashboard). The page title is "SJSU Dashboard". The left sidebar contains the following menu items:

- ACCOUNT (User icon)
- DASHBOARD (Home icon)
- COURSES (Document icon, highlighted in teal)
- ENROLL (Boat icon)
- HELP (Question mark icon)
- LOGOUT (Logout icon)

The right side of the dashboard is currently empty. The browser's address bar shows "Not Secure" and the port number 3000. The bottom of the screen shows the Mac OS X Dock with various application icons.

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

The screenshot shows a web browser window with multiple tabs open at the top. The active tab is titled "localhost:3000/Dashboard/quizcreation". The main content area displays a series of questions and answer fields:

- Question 1: "is this first question?" with an "Answer" button.
- Question 2: "jkkkd" with an "Answer" button.
- Question 3: "sdiusd" with an "Answer" button.
- Question 4: "Are you sure ?" with an "Answer" button.
- Question 5: "Is this your last semester?" with an "Answer" button.

A green "Submit Answer" button is located at the bottom of the list of questions.

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

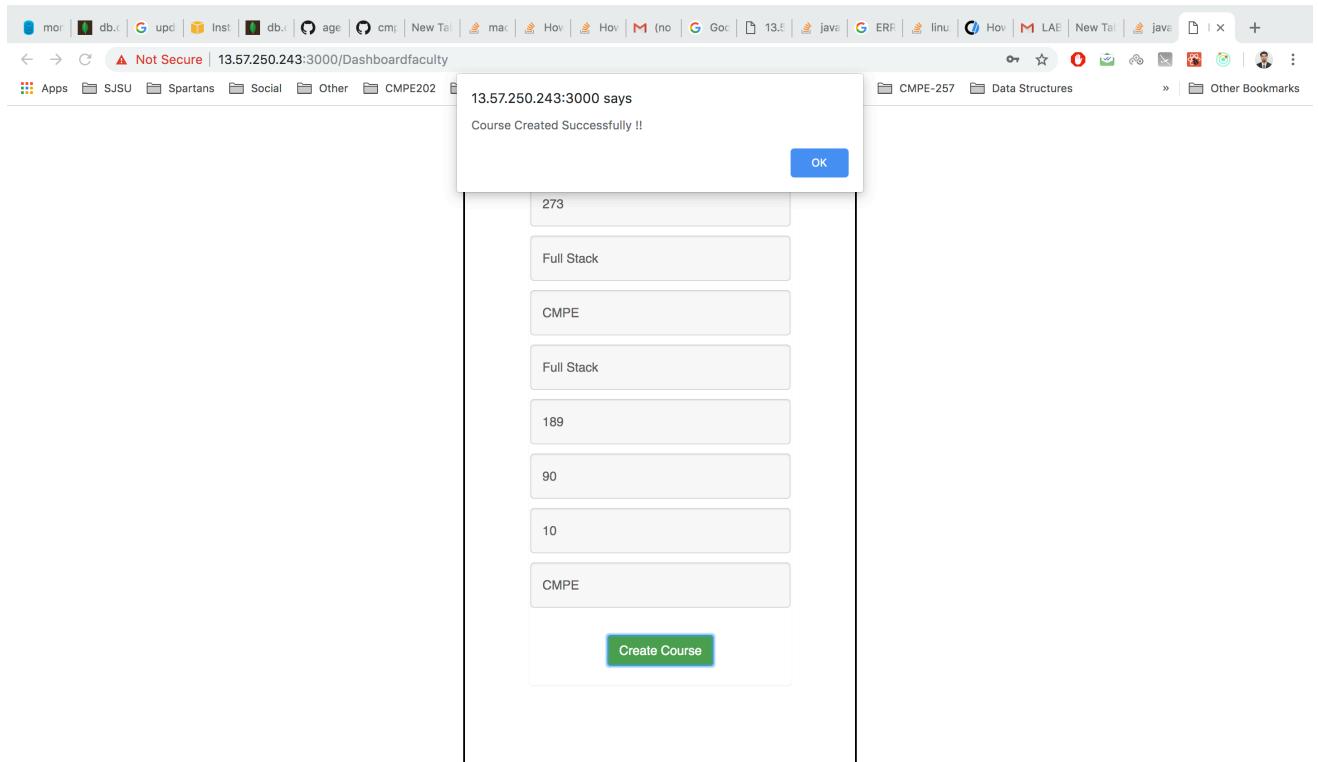
The screenshot shows a web browser window with the URL [13.57.250.243:3000/Dashboard](http://13.57.250.243:3000/Dashboard). A modal dialog box is centered on the screen with the text "13.57.250.243:3000 says" at the top and "Course Enrolled Successfully !!". Below the message is a blue "OK" button. The background shows a search bar with "Search Course" and three green buttons labeled "Term", "Course Id", and "Course Name". Below the search bar is a text input field with the placeholder "Enter Course details". The main content area displays a table titled "List of All Courses" with three rows of data. The table has columns for "CourseTerm", "Course Id", and "Course Name". Each row includes a green "Enroll" button. At the bottom of the page are two blue navigation buttons: "Prev" on the left and "Next" on the right.

CourseTerm	Course Id	Course Name	
CMPE	273	Environment Distributed System	<button>Enroll</button>
CMPE	272	Physics	<button>Enroll</button>
CMPE	202	Chemistry	<button>Enroll</button>

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

The screenshot shows a web browser window with the URL [13.57.250.243:3000/DashboardFaculty](http://13.57.250.243:3000/DashboardFaculty). On the left is a vertical sidebar menu with the following items: ACCOUNT (selected), DASHBOARD (selected), COURSES, HELP, and LOGOUT. The main content area is currently empty. The browser's address bar shows the URL and a "Not Secure" warning. The bookmarks bar at the top includes links for SJSU, Spartans, Social, Other, CMPE202, CMPE272, CMPE281, Intern, and CMPE275.

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



The screenshot shows the MongoDB Atlas interface for managing clusters. The left sidebar includes options like Project 0, Clusters, Alerts, Backup, Access, Settings, Stitch Apps, Charts, Docs, and Support. The main area displays three course documents:

```
_id: ObjectId("9129")
courseid: "9129"
courseterm: "ds1"
coursedept: "ds1"
coursedes: "sdfg"
courseroom: "234"
coursecapacity: "10"
coursewaitlist: "4"
coursestart: "sfd"
__v: 0
```

```
_id: ObjectId("5cb2ce273f13d061477b33bb")
courseid: "9129"
courseid: "9129"
coursedept: "dfdfk"
coursedes: "wef"
courseroom: "10"
coursecapacity: "10"
coursewaitlist: "1"
coursestart: "ds"
__v: 0
```

```
_id: ObjectId("5cb2eb2cf612e03867e08f58")
courseid: "273"
courseid: "273"
coursedept: "CMPE"
coursedes: "Full Stack"
courseroom: "189"
coursecapacity: "90"
coursewaitlist: "10"
coursestart: "CMPE"
__v: 0
```



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

The screenshot shows a web browser window with the following details:

- Address Bar:** Shows the URL <http://13.57.250.243:3000/DashboardFaculty/quizcreation> with a red warning icon indicating "Not Secure".
- Toolbar:** Includes standard browser icons for back, forward, search, and refresh.
- Menu Bar:** Shows various system and application icons.
- Content Area:**
  - A large text input field labeled "Question No 1".
  - Four smaller text input fields, each labeled "Answer" and preceded by a small checkbox.
  - A text input field labeled "Course Id".
  - Two green buttons at the bottom: "More Questions" and "Final Submit".
  - A small text input field at the bottom left containing the date "2019-04-104".

Chrome File Edit View History Bookmarks People Window Help

Not Secure | 13.52.177.109:3000/DashboardFaculty/announcement

Enter Course Id

Your Announcement Here

Announce

Chrome File Edit View History Bookmarks People Window Help

Not Secure | 13.52.177.109:3000/DashboardFaculty

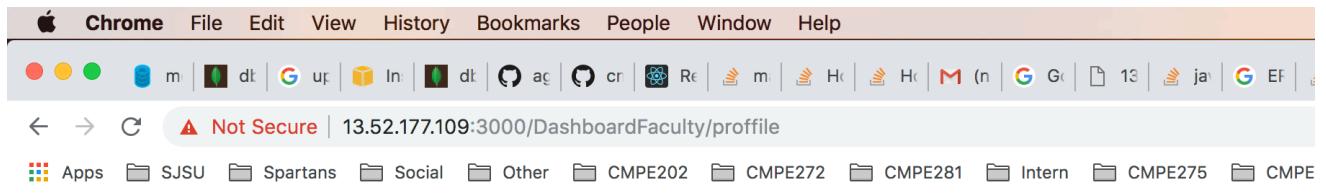
Announced Successfully !!

OK

1909

Check annoucemet!

Announce



```
[ec2-user@ip-172-31-3-189 ~]$ 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]$
```

A screenshot of a code editor with the file "Dashboard.js" open. The left sidebar shows an "EXPLORER" view with a tree structure of files and folders. The "uploads" folder under "CANVAS/test" is selected. The main pane displays the code for "Dashboard.js".

```
389     name="name"
390     placeholder="Name"
391     required
392     autoFocus
393     autoComplete
394     value={this.state.
395     /><br/><br/>
396   }
397
398
399   if(flagcheck1){
400     // this.state.flagche
401     textbox3 =
402     <div style={{width: '150
403     <h1> Please check below
404     <textarea
405     onChange = {this.announc
406     type="text"
```

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

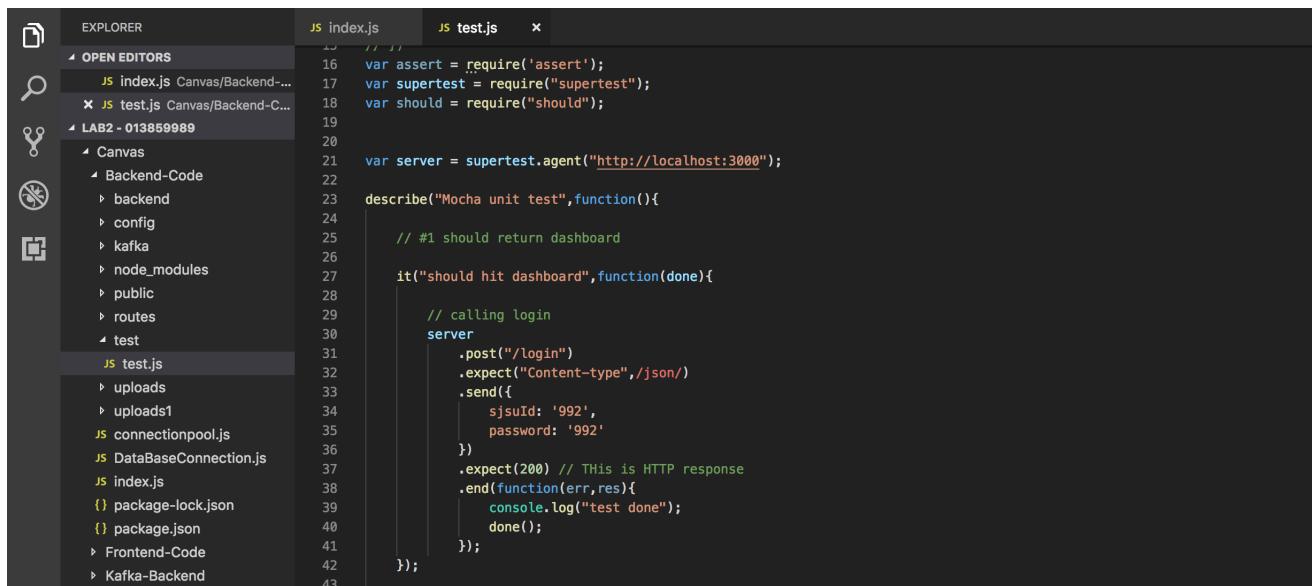
The screenshot shows a web browser window for Chrome on a Mac OS X desktop. The title bar says "localhost:3000/Dashboard". The main content area displays the "SJSU Dashboard" with a sidebar on the left containing icons for Account, Dashboard, Courses, Enroll, Help, and Logout. The Courses icon is highlighted. On the right, a message is displayed: "You have one Message in your Inbox." Below this, a box contains the text "Lab2 submission". The Mac OS X dock at the bottom shows various application icons.

Added draggable feature but failed to drop it.

The screenshot shows a web browser window for Chrome on a Mac OS X desktop. The title bar says "localhost:3000/Dashboard". The main content area displays the "SJSU Dashboard" with a sidebar on the left containing icons for Account, Dashboard, Courses, Enroll, Help, and Logout. The Courses icon is highlighted. At the top of the page, there is a horizontal banner with the text "CMPE" on a yellow background and "202 CMPE Software Systems" on a red background.

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

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



The screenshot shows the VS Code interface. On the left is the Explorer sidebar with a tree view of files and folders. In the center are two tabs: 'index.js' and 'test.js'. The 'test.js' tab contains the following Mocha test code:

```
15 // ...
16 var assert = require('assert');
17 var supertest = require('supertest');
18 var should = require('should');
19
20
21 var server = supertest.agent("http://localhost:3000");
22
23 describe("Mocha unit test",function(){
24
25     // #1 should return dashboard
26
27     it("should hit dashboard",function(done){
28
29         // calling login
30         server
31             .post("/login")
32             .expect("Content-type",/json/)
33             .send({
34                 sjsuid: '992',
35                 password: '992'
36             })
37             .expect(200) // This is HTTP response
38             .end(function(err,res){
39                 console.log("test done");
40                 done();
41             });
42     });
43});
```

```
[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)

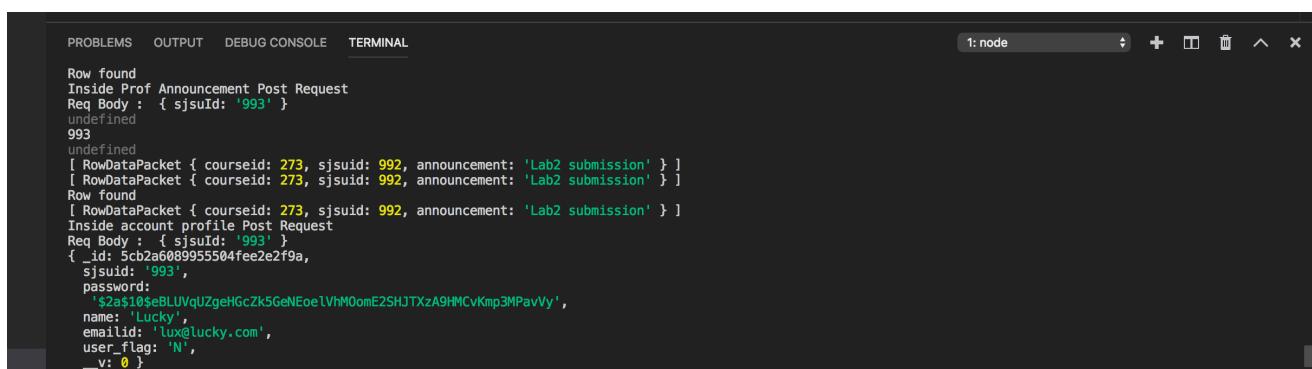
Mocha unit test
test done
  ✓ should display Signup (693ms)

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$
```



The screenshot shows the VS Code terminal window. The output shows the results of the Mocha test run, indicating 5 passing tests in 1 second. Below the test results, there is some additional log output related to database queries and user profiles.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
TERMINAL: 1: node

Row found
Inside Prof Announcement Post Request
Req Body : { sjsuid: '993' }
undefined
993
undefined
[ RowDataPacket { courseid: 273, sjsuid: 992, announcement: 'Lab2 submission' } ]
[ RowDataPacket { courseid: 273, sjsuid: 992, announcement: 'Lab2 submission' } ]
Row found
[ RowDataPacket { courseid: 273, sjsuid: 992, announcement: 'Lab2 submission' } ]
Inside account profile Post Request
Req Body : { sjsuid: '993' }
{ _id: 5cb2a608995504fee2e2f9a,
  sjsuid: '993',
  password:
    '$2a$10$eBLUVqUZgeHgCZk5GeNEoeIVhM0omE2SHJTXzA9HMcvKnp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  __v: 0 }
```

I have tested AWS deployed application with MOCHA.

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of project files. Under 'OPEN EDITORS', there is a file named 'test.js'. The main area shows the content of 'test.js' which contains Mocha test code. The code includes imports for assert, supertest, and should, and defines a describe block for 'Mocha unit test' with a nested it block for 'should hit dashboard'. The code is syntax-highlighted in green, blue, and yellow.

```
var assert = require('assert');
var supertest = require("supertest");
var should = require("should");

//var server = supertest.agent("http://localhost:3000");
var server = supertest.agent("http://13.52.177.109:3000/");

describe("Mocha unit test",function(){
  // #1 should return dashboard
  it("should hit dashboard",function(done){
```

```
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:~ sachinwaghmode$ mocha
>Error: No test files found: "test"
[Sachins-MacBook-Pro:~ sachinwaghmode$ cd /Users/sachinwaghmode/Desktop/Lab2\ -\ 013859989/Canvas/Backend-Code/test
[Sachins-MacBook-Pro:~ 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:~ sachinwaghmode$ ]
```

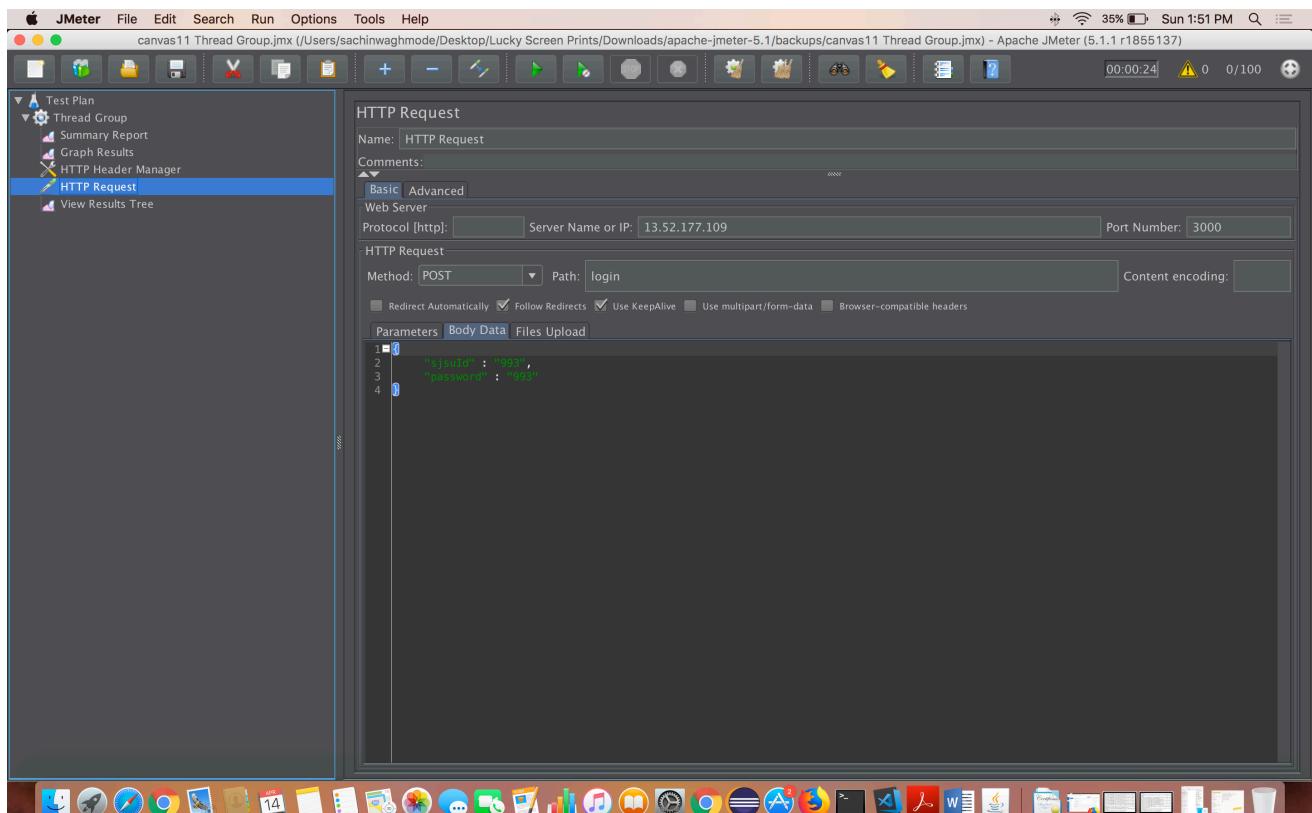
Below message is from Server which is running on AWS.

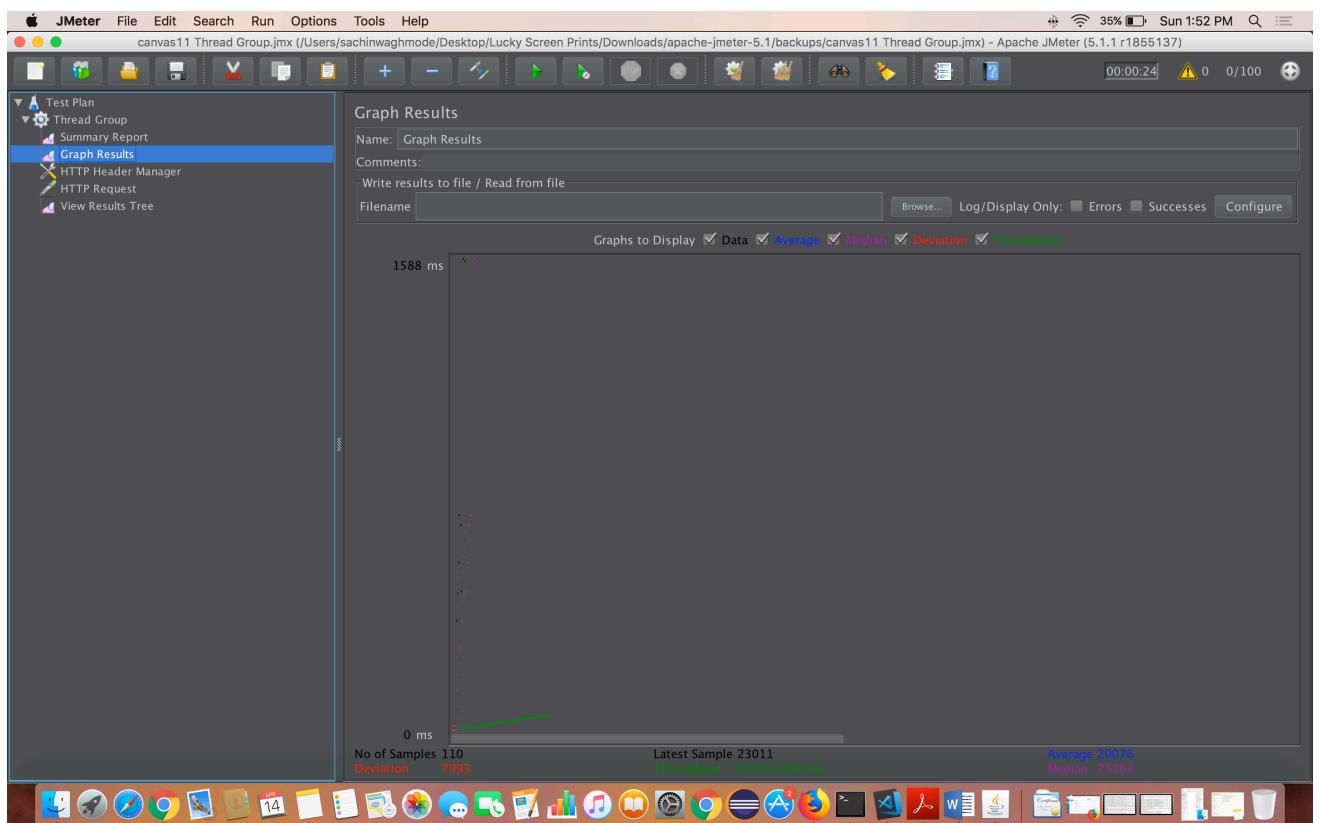
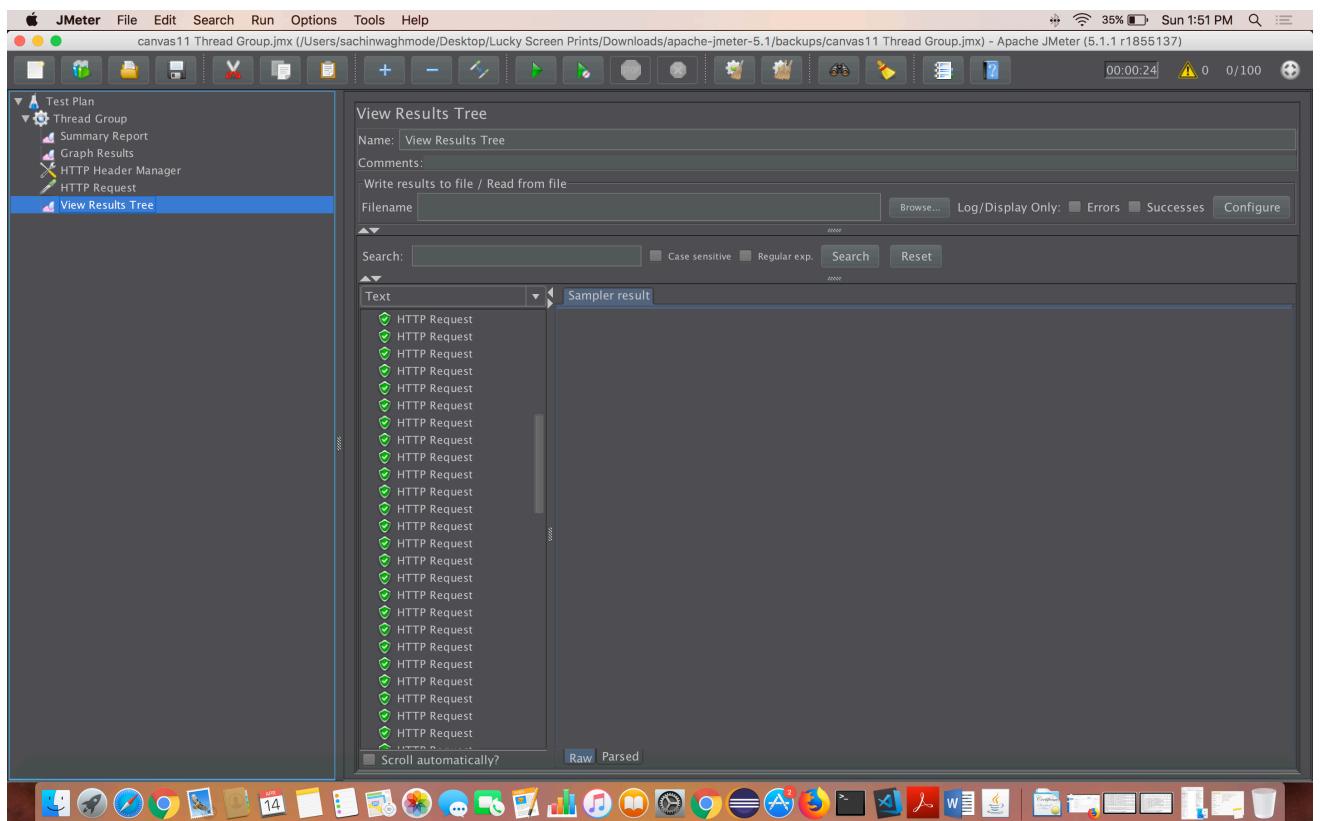
```
msg received
response { _id: '5cb2a608995504fee2e2f9a',
  sjsuid: '993',
  password: 'S2as10seBLUVqUZgeHGcZk5GeNEoe1VhM0omE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  v: 0 }
In results Signup
Results: { _id: '5cb2a608995504fee2e2f9a',
  sjsuid: '993',
  password: 'S2as10seBLUVqUZgeHGcZk5GeNEoe1VhM0omE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  v: 0 }
Logged in successfully.
{ _id: '5cb2a608995504fee2e2f9a',
  sjsuid: '993',
  password: 'S2as10seBLUVqUZgeHGcZk5GeNEoe1VhM0omE2SHJTXzA9HMCvKmp3MPavVy',
  name: 'Lucky',
  emailid: 'lux@lucky.com',
  user_flag: 'N',
  v: 0 }
eyJhbGciOiJIUzI1NiI6InRzCjG1kpXVCj9.eyJzanN1aWQiOii50TMilCJpYXQiOjE1NTUyNzM0MzgsImV4cCI6MTU1NTI4MzUxO0.6nAvez7cMcu866HNuvbMe5p0B5atrnQUaggnlUEVR8
{"logincheck": {"finalStatus": true, "facultyfn": false, "pwdValidity": true}}, {"Token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCj9.eyJzanN1aWQiOii50TMilCJpYXQiOjE1NTUyNzM0MzgsImV4cCI6MTU1NTI4MzUxO0.6nAvez7cMcu866HNuvbMe5p0B5atrnQUaggnlUEVR8
HNuvbMe5p0B5atrnQUaggnlUEVR8"}
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)

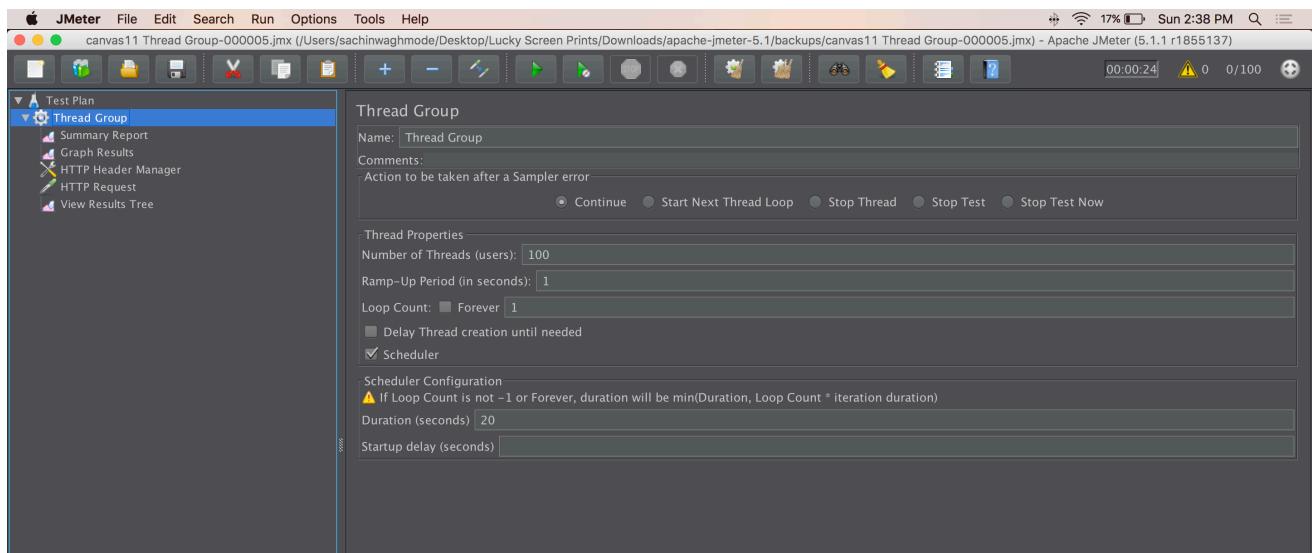




Below screen print is from the AWS where Server.js file is running

```
YL4xww2xiRAZKvmdvtTnVzDSJpE")
msg received
response { _id: '5cb2a6549955504fee2e2f9c',
  sjsuid: '992',
  password: 'S2as10$0xJJ8s97piELC7000fYE2u1HkIhCzEi/rqII19JwIM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  _v: 0 }
In results Signup
Results: { _id: '5cb2a6549955504fee2e2f9c',
  sjsuid: '992',
  password: 'S2as10$0xJJ8s97piELC7000fYE2u1HkIhCzEi/rqII19JwIM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  _v: 0 }
Logged in successfully.
{ _id: '5cb2a6549955504fee2e2f9c',
  sjsuid: '992',
  password: 'S2as10$0xJJ8s97piELC7000fYE2u1HkIhCzEi/rqII19JwIM5tNDNBob.q',
  name: 'Professor',
  emailid: 'prof@prof.com',
  user_flag: 'Y',
  _v: 0 }
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJzaanN1aWQ1Oii5OTIiLCJpYXQiOjE1NTUyNzU2D0EsImV4cCI6MTU1NTI4NTc2MX0.PCIU1lIrW0QDMmNsYL4xww2xiRAZKvmdvtTnVzDSJpE
{"logincheck": {"finalstatus": true, "facultyfn": true, "pwdValidity": true}}, {"Token": "eyJhbGciOiJIUzI1NiisInR5cCI6IkpxVCJ9.eyJzaanN1aWQ1Oii5OTIiLCJpYXQiOjE1NTUyNzU2D0EsImV4cCI6MTU1NTI4NTc2MX0.PCIU1lIrW0QDMmNsYL4xww2xiRAZKvmdvtTnVzDSJpE"}
[ec2-user@ip-172-31-3-189 ~]$
```

## 100 Concurrent Users: (Local run)



**EXPLORER**

- OPEN EDITORS
  - JS settings.js
- LAB2 - 013859989
  - Canvas
  - Backend-Code
  - Frontend-Code
  - node\_modules
  - public
  - src
  - AWS
  - JS settings.js
  - components
  - AccountProfile
  - acctdetails
  - actions
  - JS postActions.js
  - JS types.js
  - Announcement
  - CourseCreation
  - Dashboard
  - Enroll
  - File
  - LandingPage
  - \_mocks\_
  - app
  - JS Login.js
  - SJSUI.png
  - Quiz
  - reducers
  - SignUp
  - JS Main.js
  - config
  - logo
  - # App.css
  - JS App.js
  - JS App.test.js
  - # index.css
- OUTLINE

**settings.js**

```

1 //export const rooturl = "13.57.250.243";
2 export const rooturl = "localhost";
```

**PROBLEMS**

```

sjsuid: '993',
password: '$2a$10$eBLUVqUZgeHgCzK5GeNEoeVhM0mE2SHJTxzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
user_flag: 'N',
v: 0 }
```

**valid Credentials!**

```

After request handling: { _id: 5cb2a608995504fee2e2f9a,
sjsuid: '993',
password: '$2a$10$eBLUVqUZgeHgCzK5GeNEoeVhM0mE2SHJTxzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
user_flag: 'N',
v: 0 }
{ _id: 5cb2a608995504fee2e2f9a,
sjsuid: '993',
password: '$2a$10$eBLUVqUZgeHgCzK5GeNEoeVhM0mE2SHJTxzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
user_flag: 'N',
v: 0 }
```

**User details**

```

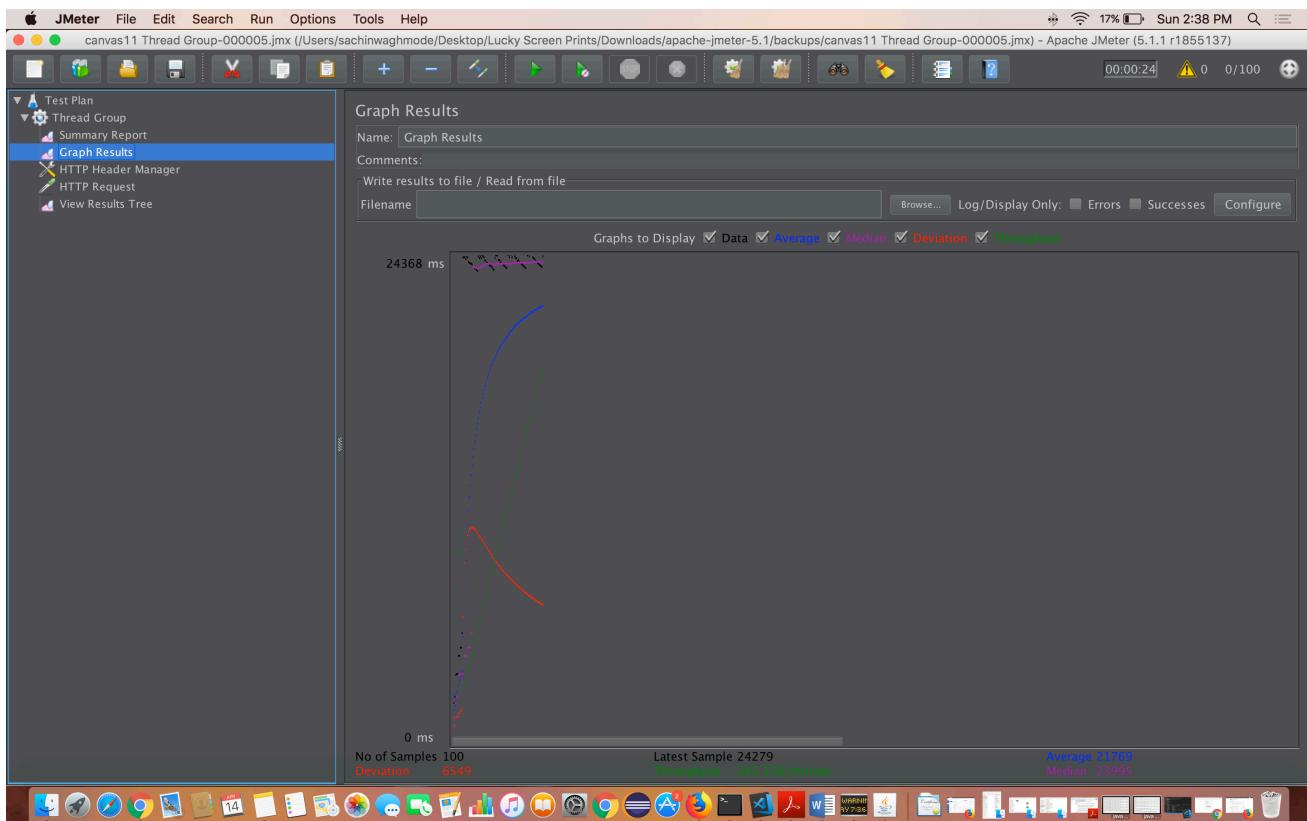
{ _id: 5cb2a608995504fee2e2f9a,
sjsuid: '993',
password: '$2a$10$eBLUVqUZgeHgCzK5GeNEoeVhM0mE2SHJTxzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
user_flag: 'N',
v: 0 }
```

**valid Credentials!**

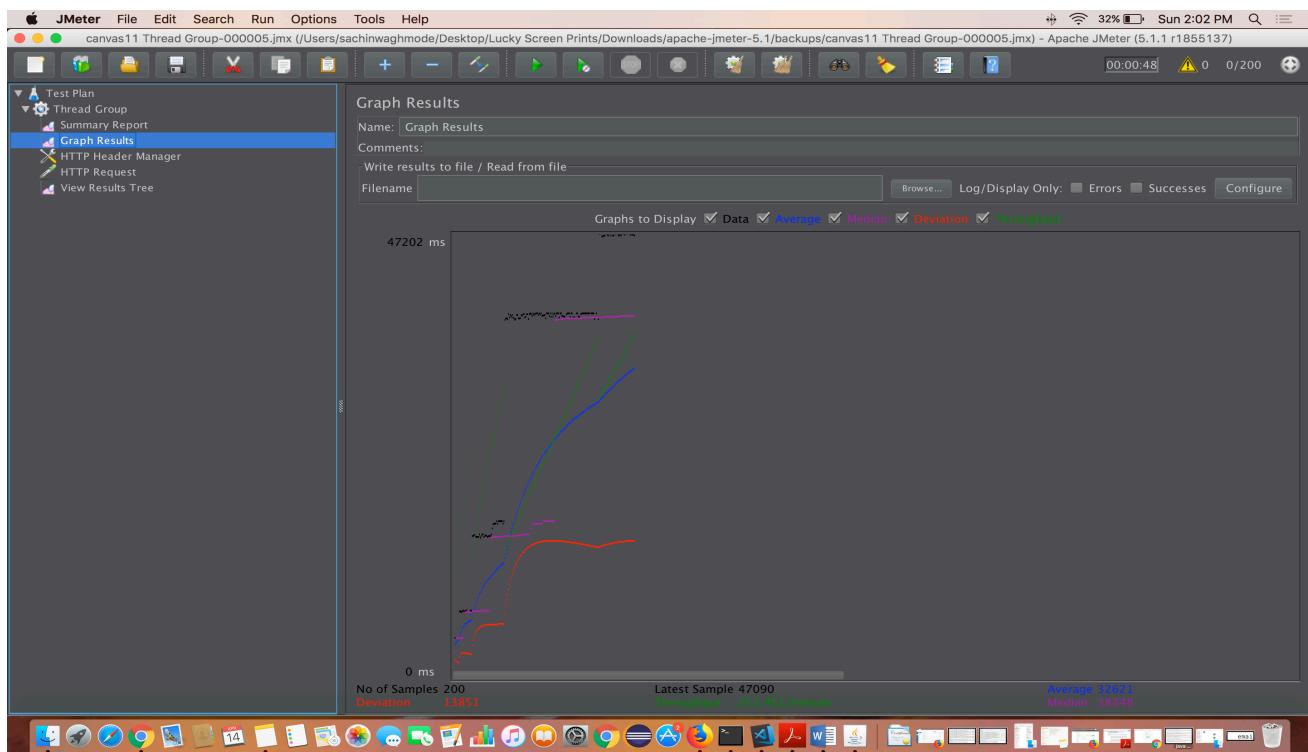
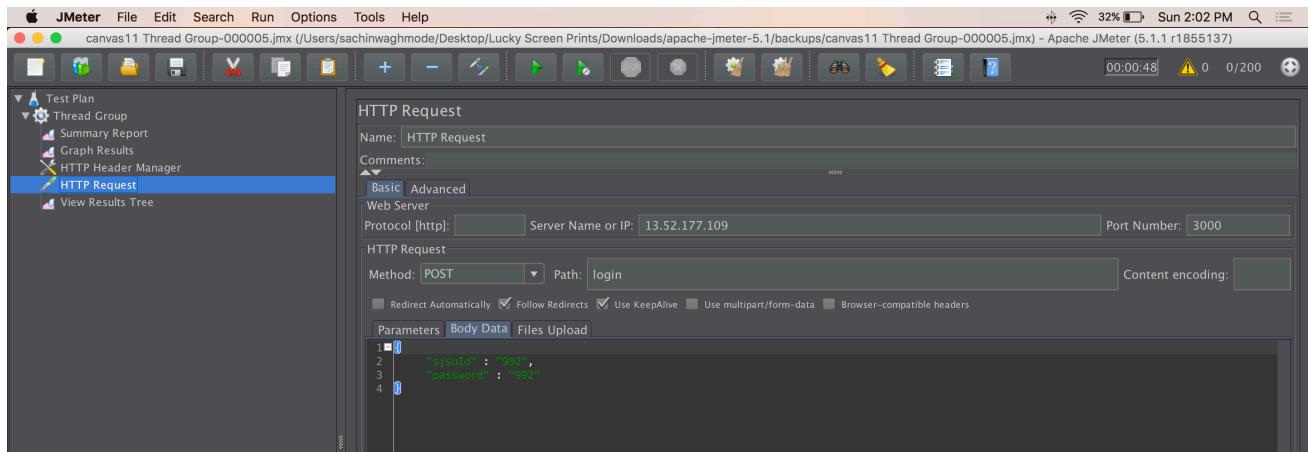
```

After request handling: { _id: 5cb2a608995504fee2e2f9a,
sjsuid: '993',
password: '$2a$10$eBLUVqUZgeHgCzK5GeNEoeVhM0mE2SHJTxzA9HMCvKmp3MPavVy',
name: 'Lucky',
emailid: 'lux@lucky.com',
```

**Ln 2, Col 36 Spaces: 4 UTF-8 LF Javascript (Babel) ☺**

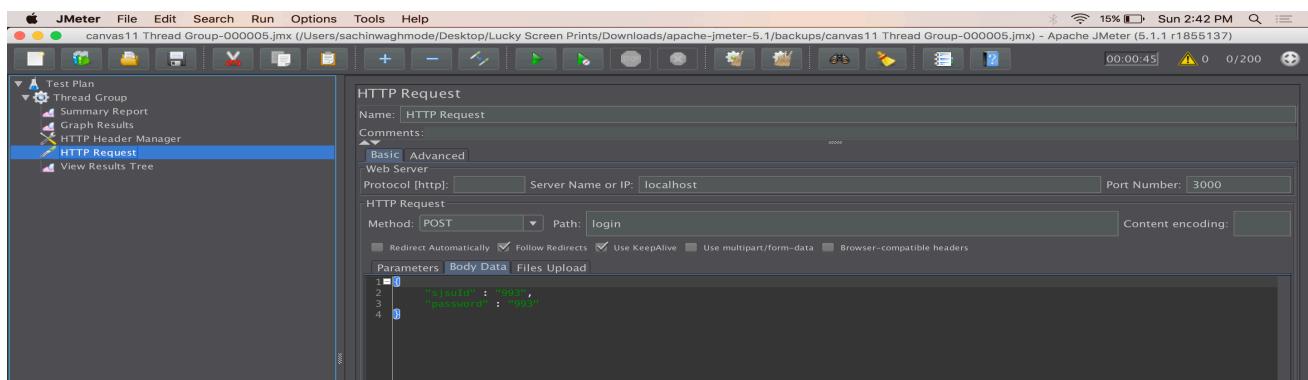
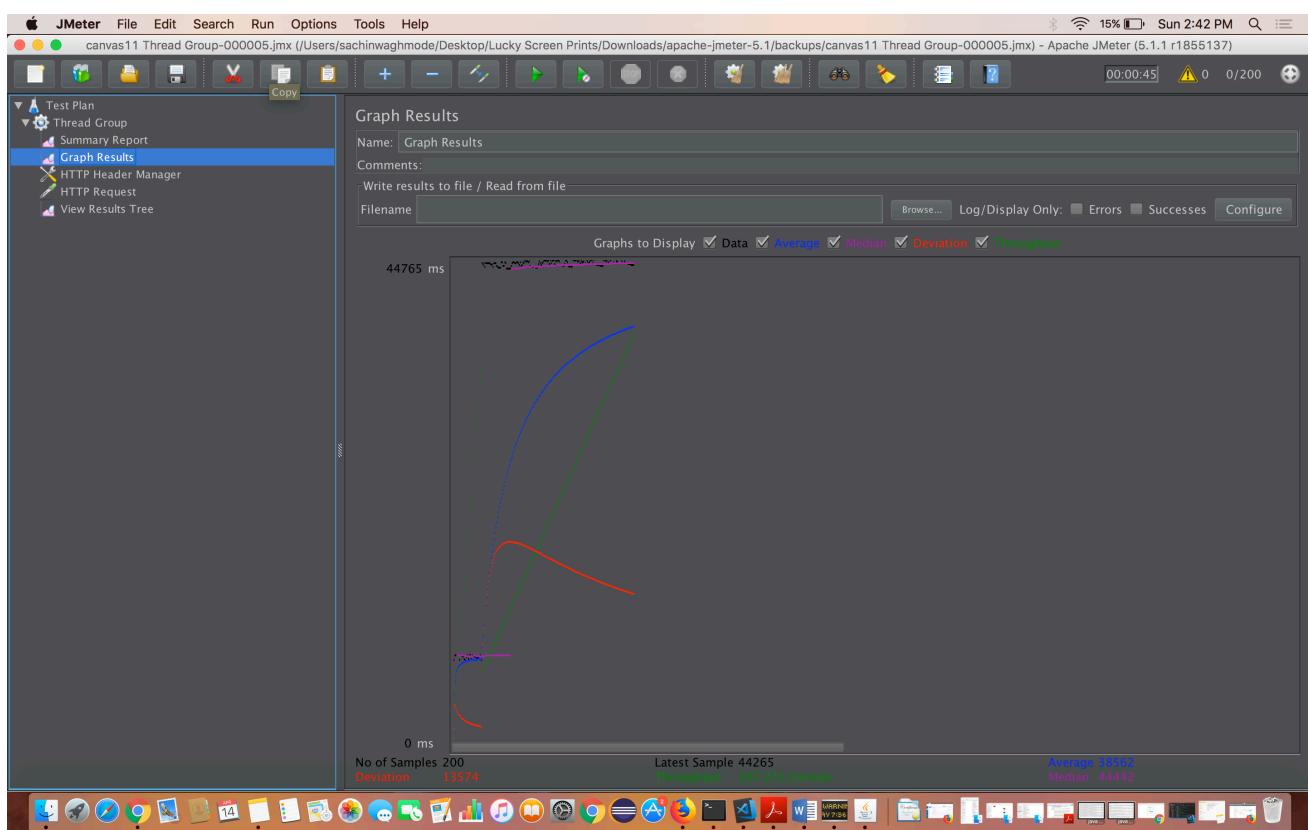
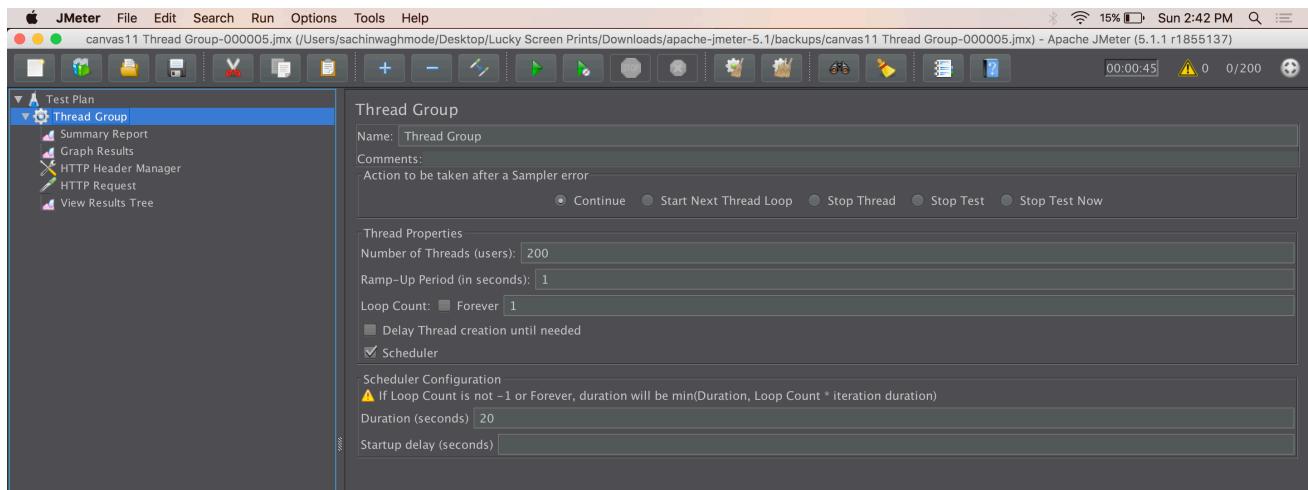


## 200 Concurrent Users: (Deployed on AWS)



```
Last login: Sun Apr 14 00:25:33 on ttys004
{
  "_id": "5cb2a6089955504fee2e2f9a",
  "ssuid": "993",
  "password": "52a$10$eBLUVqUzGeHGcZk5GeNEoelVhM0omE2SHJTxzA9HMCvKmp3MPavVy",
  "name": "Lucky",
  "emailid": "luxelucky.com",
  "user_flag": "N",
  "__v": 0
}
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQ1OjI50TMlCJpYXQ1OjE1NTUyNzUxODMsImV4cCI6MTU1NTI4NTI2M30.jry5JjGig70npD0F040bV0kcRS0Eku7jUNFsRQ5yRM
{"loginCheck":[{"finalStatus":true,"facultyId":false,"pwdValidity":true}],"Token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQ1OjI50TMlCJpYXQ1OjE1NTUyNzUxODMsImV4cCI6MTU1NTI4NTI2M30.jry5JjGig70npD0
uF040bV0kcRS0Eku7jUNFsRQ5yRM"}
msg received
msg {
  "_id": "5cb2a6089955504fee2e2f9a",
  "ssuid": "993",
  "password": "52a$10$eBLUVqUzGeHGcZk5GeNEoelVhM0omE2SHJTxzA9HMCvKmp3MPavVy",
  "name": "Lucky",
  "emailid": "luxelucky.com",
  "user_flag": "N",
  "__v": 0
}
In results Sign up
Results: {
  "_id": "5cb2a6089955504fee2e2f9a",
  "ssuid": "993",
  "password": "52a$10$eBLUVqUzGeHGcZk5GeNEoelVhM0omE2SHJTxzA9HMCvKmp3MPavVy",
  "name": "Lucky",
  "emailid": "luxelucky.com",
  "user_flag": "N",
  "__v": 0
}
Logged in successfully.
{
  "_id": "5cb2a6089955504fee2e2f9a",
  "ssuid": "993",
```

## 200 Concurrent Users: (Deployed on Local)



## 300 Concurrent Users: (Deployed on AWS)

JMeter File Edit Search Run Options Tools Help

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)

Test Plan

Thread Group

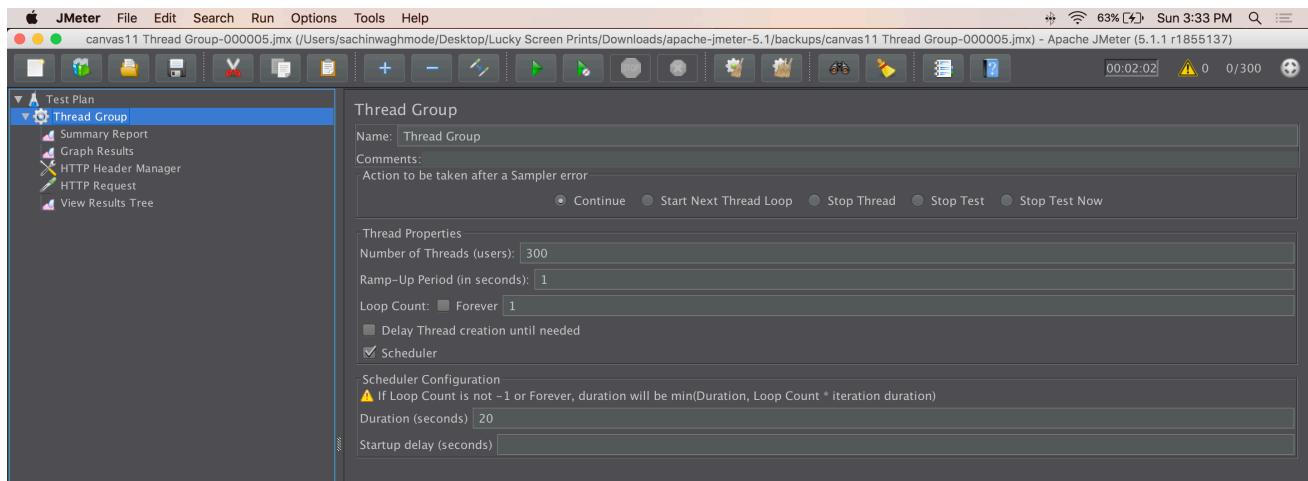
Name: Thread Group  
Comments:  
Action to be taken after a Sampler error:

Thread Properties

Number of Threads (users): 300  
Ramp-Up Period (in seconds): 1  
Loop Count:  Forever | 1  
 Delay Thread creation until needed  
 Scheduler

Scheduler Configuration

If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count \* iteration duration)  
Duration (seconds): 20  
Startup delay (seconds):



JMeter File Edit Search Run Options Tools Help

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)

Test Plan

Thread Group

HTTP Request

Name: HTTP Request  
Comments:  
Basic Advanced

Web Server

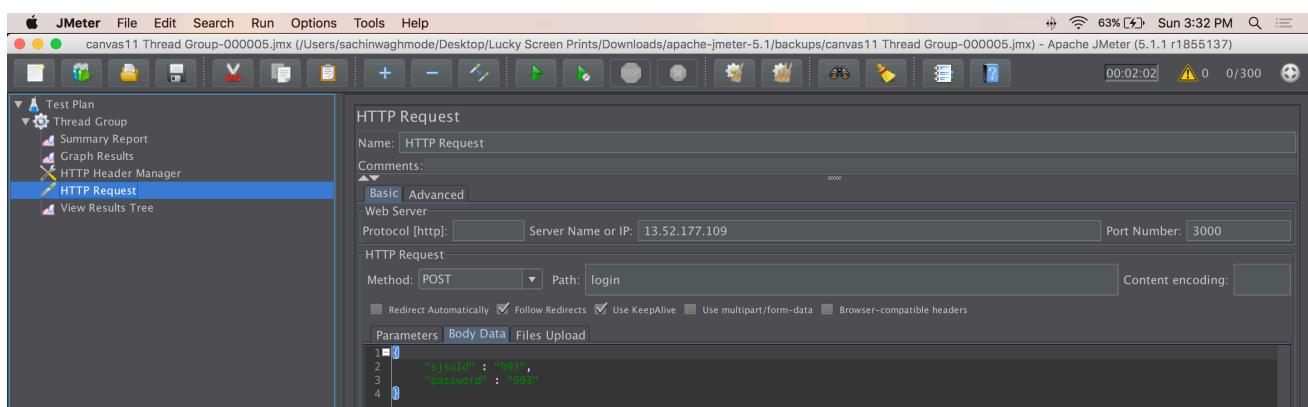
Protocol [http]: Server Name or IP: 13.52.177.109 Port Number: 3000

HTTP Request

Method: POST Path: login  
 Redirect Automatically  Follow Redirects  Use KeepAlive  Use multipart/form-data  Browser-compatible headers

Parameters Body Data Files Upload

1  
2     "userId" : "999",  
3     "password" : "654"  
4



JMeter File Edit Search Run Options Tools Help

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)

Test Plan

Thread Group

Graph Results

Name: Graph Results  
Comments:  
Write results to file / Read from file  
Filename:

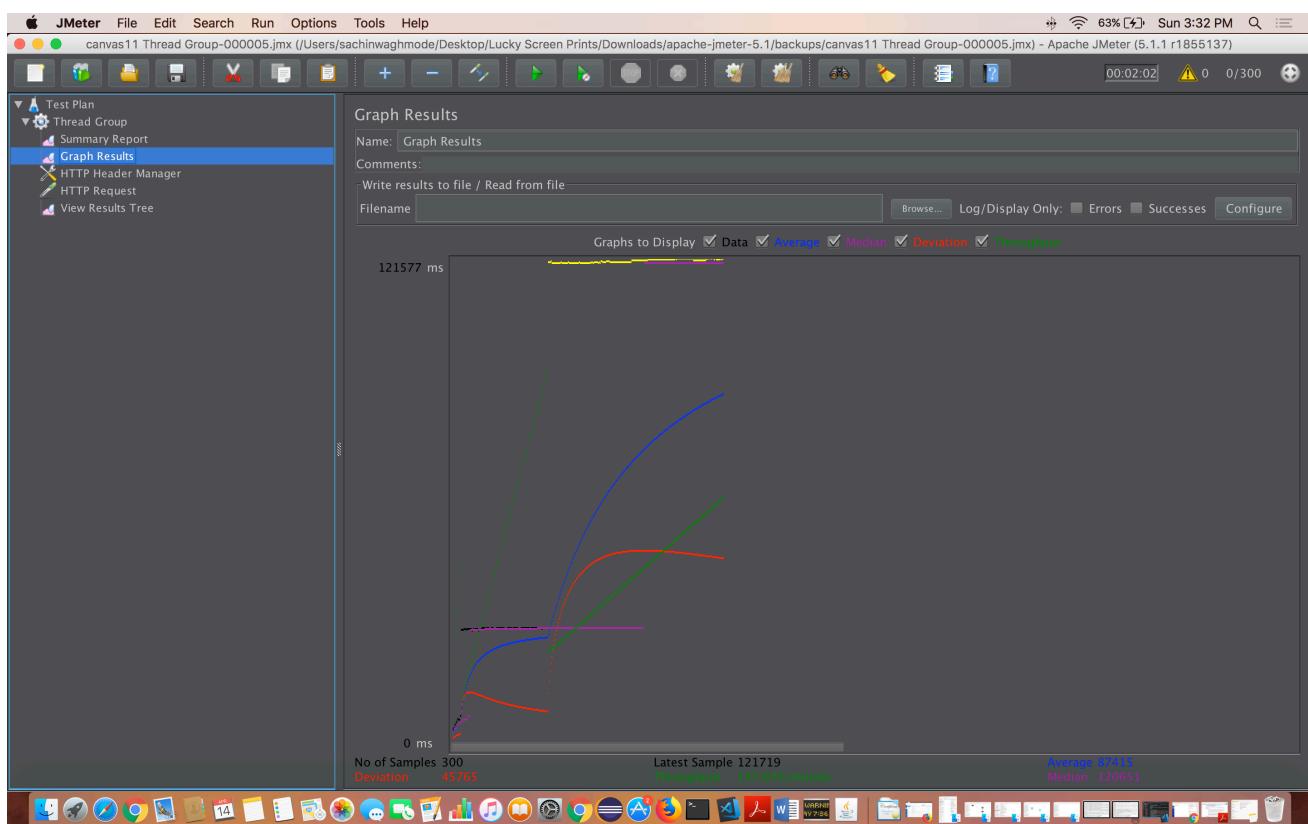
Graphs to Display  Data  Average  Median  Deviation  Throughput

121577 ms

0 ms

No of Samples 300 Deviation 45765 Latest Sample 121719 Throughput 147.645/minute

Average 87415 Median 120651



## 300 Concurrent Users: (Deployed on local)

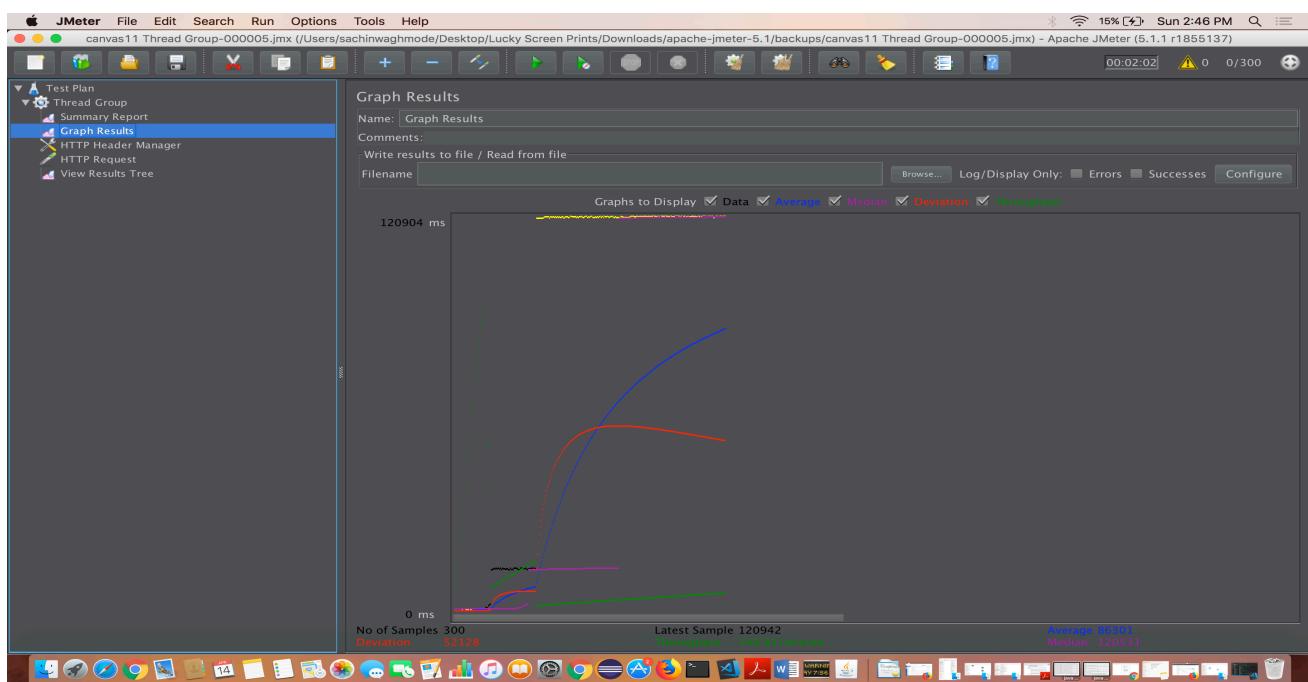
The screenshot shows the Apache JMeter interface with the following details:

- Test Plan Structure:** The left sidebar shows a tree view with "Test Plan" expanded, containing "Thread Group", "Summary Report", "Graph Results", "HTTP Header Manager", "HTTP Request", and "View Results Tree".
- HTTP Request Configuration:** The main panel displays the "HTTP Request" configuration. It includes:
  - Basic:** Protocol [http], Server Name or IP: localhost, Port Number: 3000.
  - Advanced:** Method: POST, Path: login.
  - Parameters:** A table with four rows:

1	username	:	admin
2	password	:	password
3			
4			
  - Content encoding:** A dropdown menu.

The screenshot shows the Apache JMeter interface with the following details:

- Test Plan Structure:** The left sidebar shows a tree view with "Test Plan" expanded, containing "Thread Group", "Summary Report", "Graph Results", "HTTP Header Manager", "HTTP Request", and "View Results Tree".
- Thread Group Configuration:** The main panel displays the "Thread Group" configuration. It includes:
  - Name:** Thread Group.
  - Action to be taken after a Sampler error:** Continue (radio button selected).
  - Thread Properties:** Number of Threads (users): 300, Ramp-Up Period (in seconds): 1, Loop Count: Forever (checkbox checked), Delay Thread creation until needed (checkbox unchecked), Scheduler (checkbox checked).
  - Scheduler Configuration:** If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count \* iteration duration). Duration (seconds): 20.
  - Startup delay (seconds):** A dropdown menu.



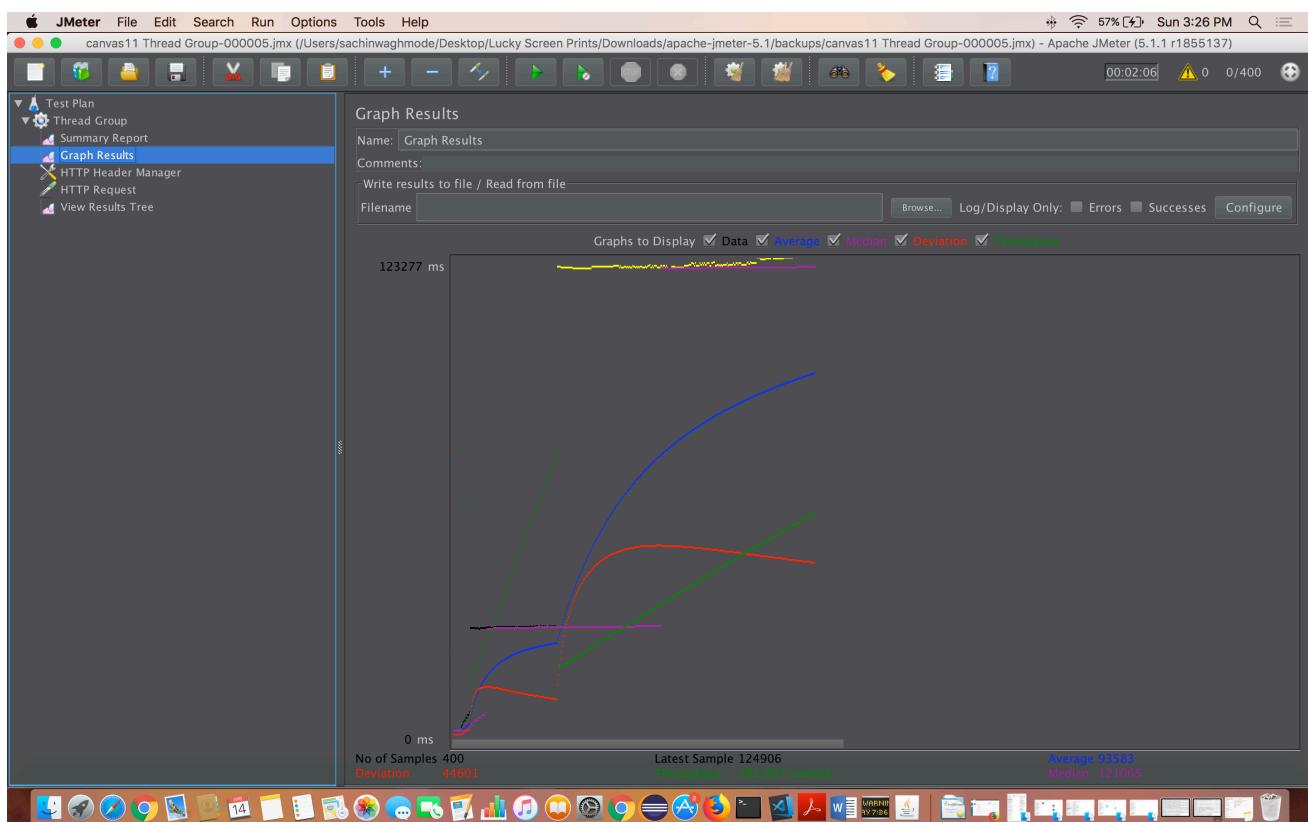
## 400 Concurrent Users: (Deployed on AWS)

The screenshot shows the Apache JMeter interface with the following details:

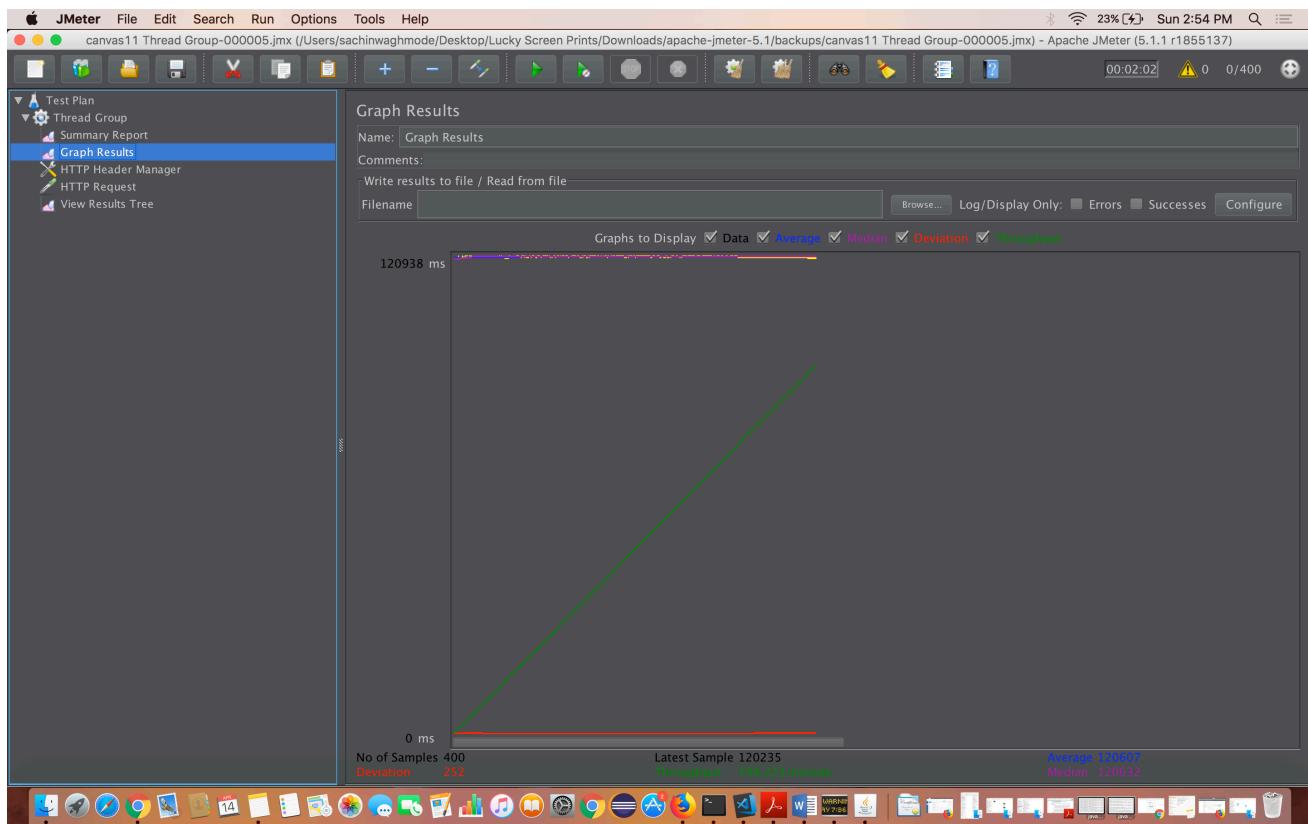
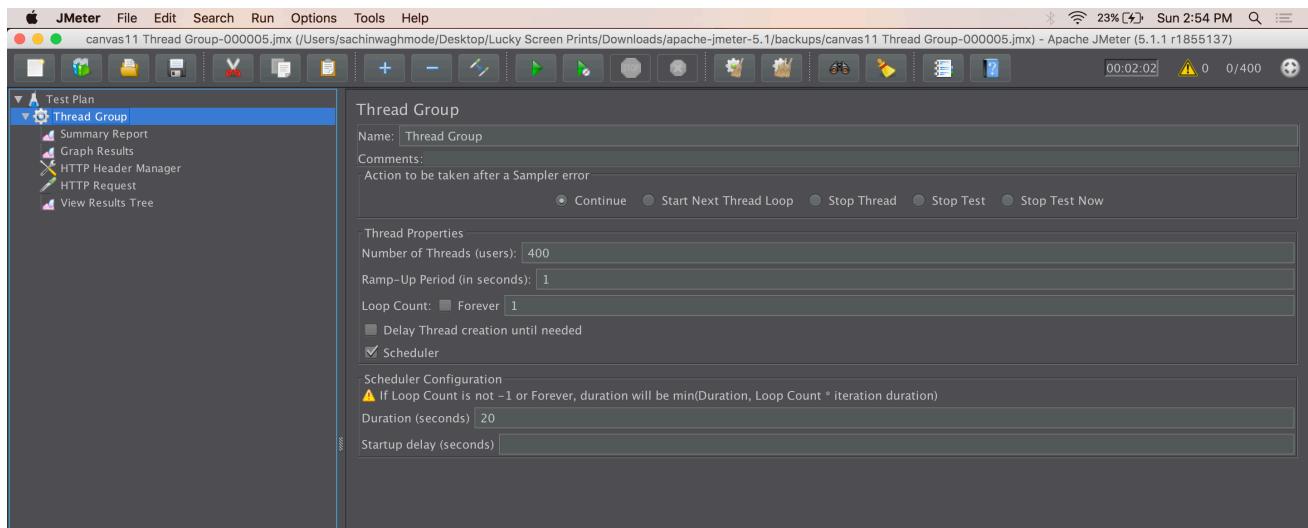
- Test Plan:** canvas11 Thread Group-000005.jmx
- Thread Group:** 1 thread group
- HTTP Request:** Configuration includes:
  - Protocol: http, Server Name or IP: 13.52.177.109, Port Number: 3000
  - Method: POST, Path: login
  - Content encoding: (empty)
  - Advanced settings: Redirect Automatically (unchecked), Follow Redirects (checked), Use KeepAlive (checked), Use multipart/form-data (unchecked), Browser-compatible headers (unchecked)
  - Parameters tab: Contains parameters 2, 3, and 4 with their values set to "1".
- Summary Report:** Viewed in the tree.

The screenshot shows the Apache JMeter interface with the following details:

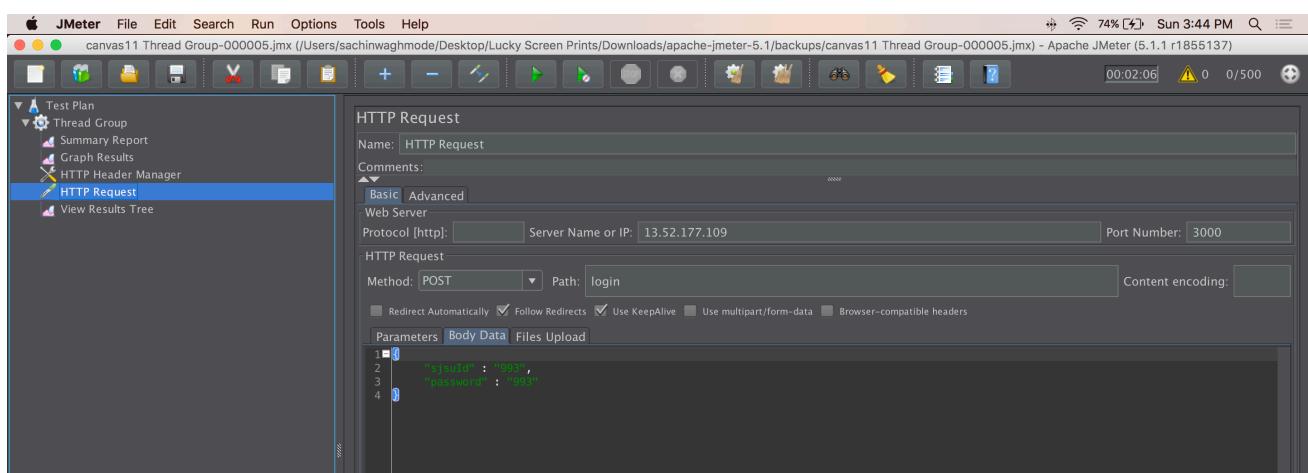
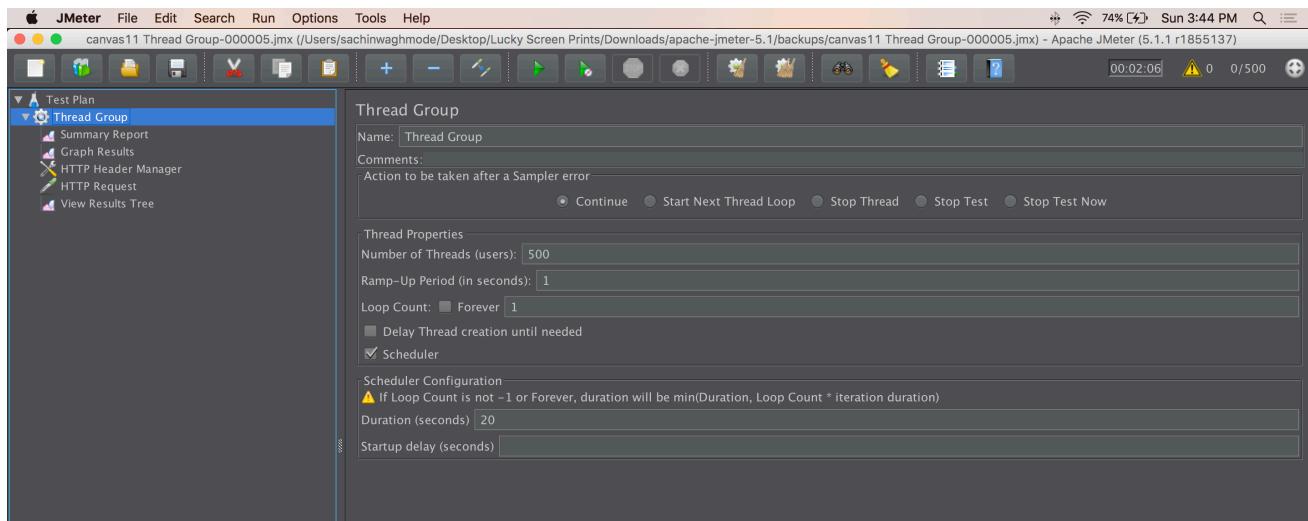
- Test Plan:** canvas11 Thread Group-000005.jmx
- Thread Group:** 1 thread group
- Thread Properties:** Number of Threads (users): 400, Ramp-Up Period (in seconds): 1, Loop Count: Forever 1, Delay Thread creation until needed (unchecked), Scheduler (checked).
- Scheduler Configuration:** If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count \* iteration duration). Duration (seconds): 20.
- Summary Report:** Viewed in the tree.



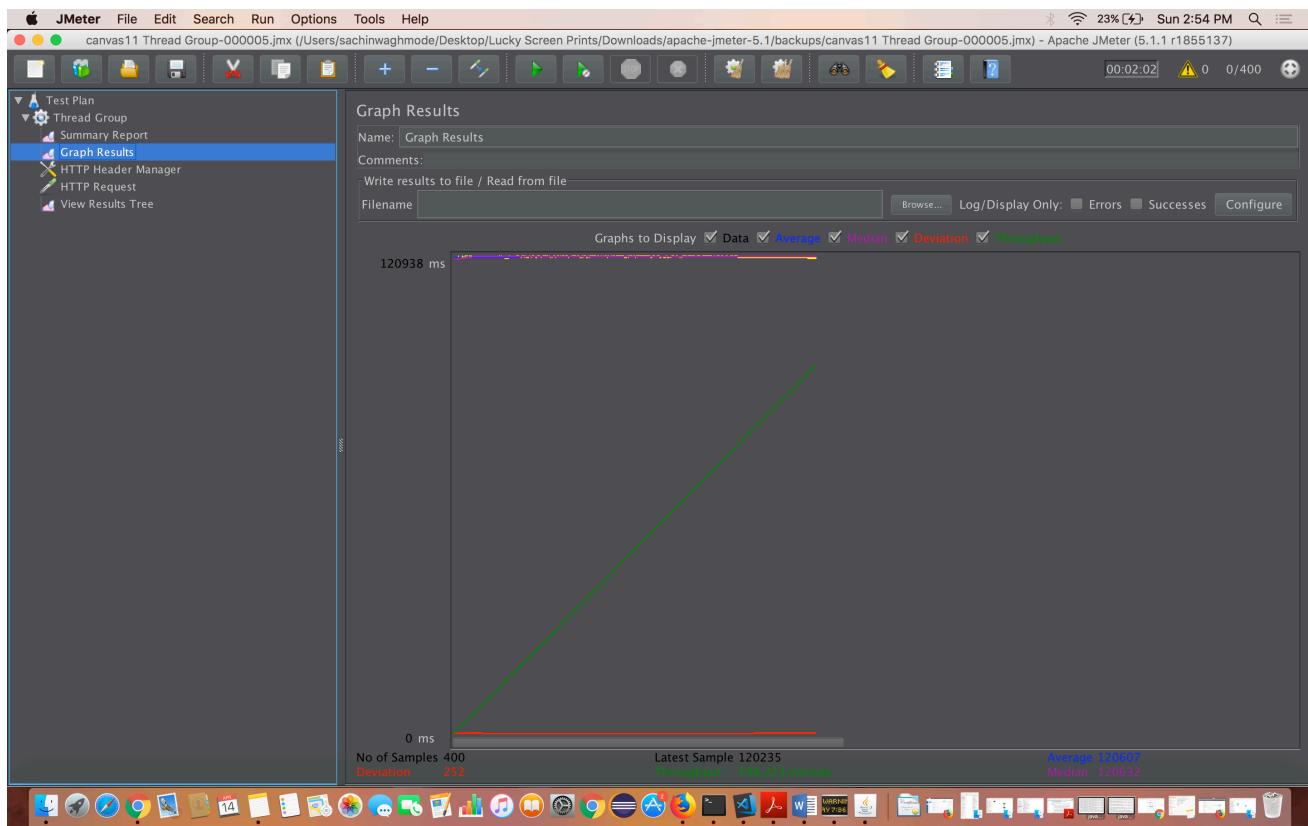
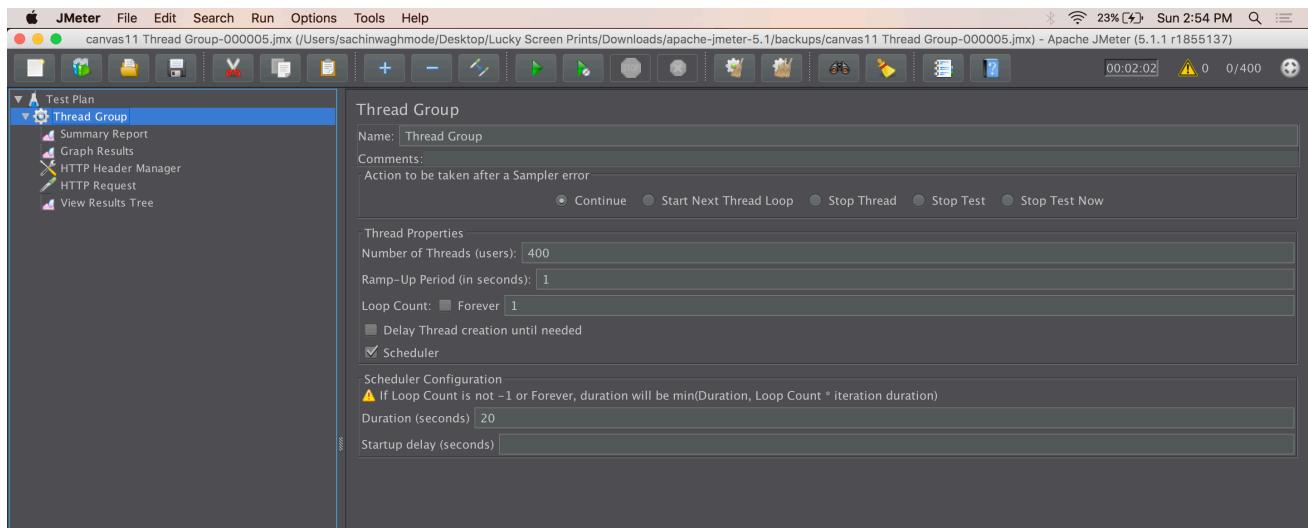
## 400 Concurrent Users: (Deployed on local)



## 400 Concurrent Users: (Deployed on AWS)



## 400 Concurrent Users: (Deployed on Local)



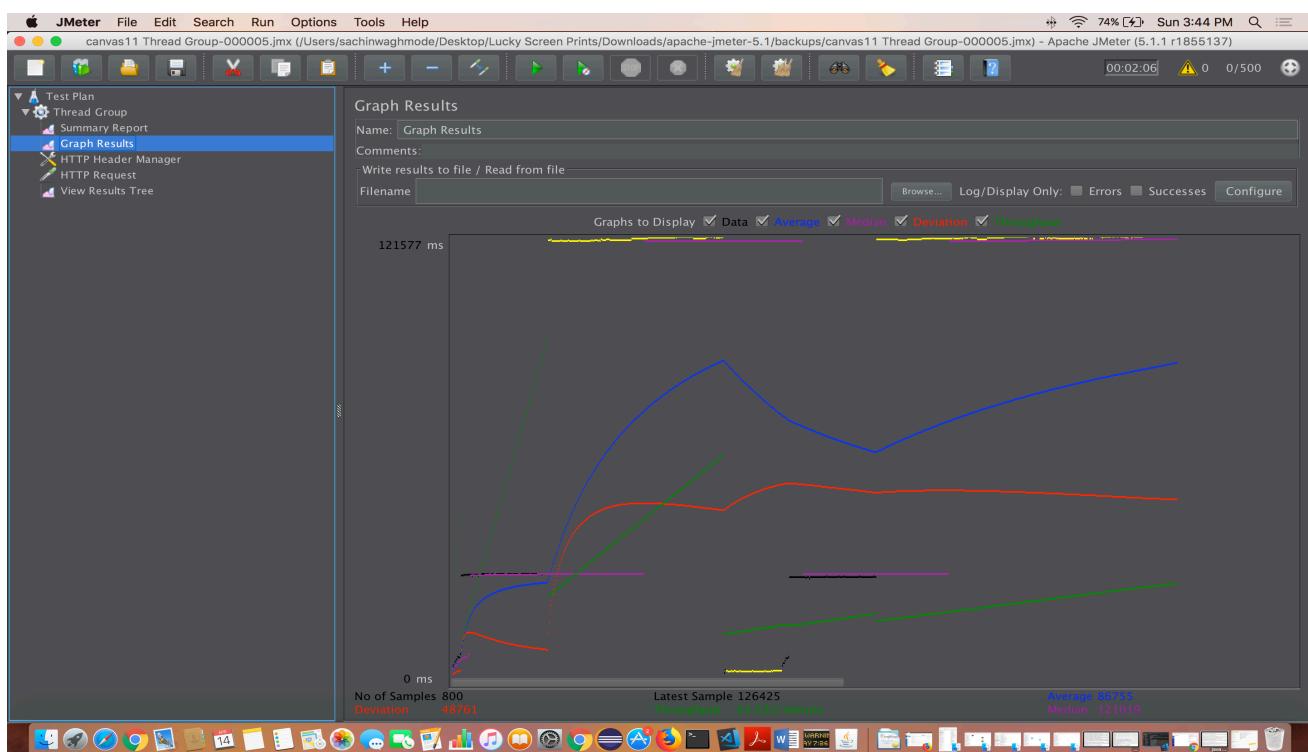
## 500 Concurrent Users: (Deployed on AWS)

The screenshot shows the Apache JMeter interface with a test plan containing an 'HTTP Request' sampler. The 'HTTP Request' configuration panel is open, showing the following details:

- Name: HTTP Request
- Protocol: http
- Server Name or IP: 13.52.177.109
- Port Number: 3000
- Method: POST
- Path: login
- Content encoding: (unchecked)
- Parameters tab (with 4 parameters):
  - 1: "username": "sachinwaghmode"
  - 2: "password": "1234567890"
  - 3: "rememberMe": "true"
  - 4: "j\_username": "sachinwaghmode"
- Body Data tab (disabled)
- Files Upload tab (disabled)

The screenshot shows the Apache JMeter interface with a test plan containing a 'Thread Group'. The 'Thread Group' configuration panel is open, showing the following details:

- Name: Thread Group
- Action to be taken after a Sampler error:
  - Continue (radio button selected)
  - Start Next Thread Loop
  - Stop Thread
  - Stop Test
  - Stop Test Now
- Thread Properties:
  - Number of Threads (users): 500
  - Ramp-Up Period (in seconds): 1
  - Loop Count: Forever (checkbox checked)
  - Delay Thread creation until needed (checkbox unchecked)
  - Scheduler (checkbox checked)
- Scheduler Configuration:
  - If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count \* iteration duration)
  - Duration (seconds): 20
  - Startup delay (seconds): (empty)



## 500 Concurrent Users: (Deployed on local)

JMeter File Edit Search Run Options Tools Help

26% Sun 2:57 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)

Test Plan

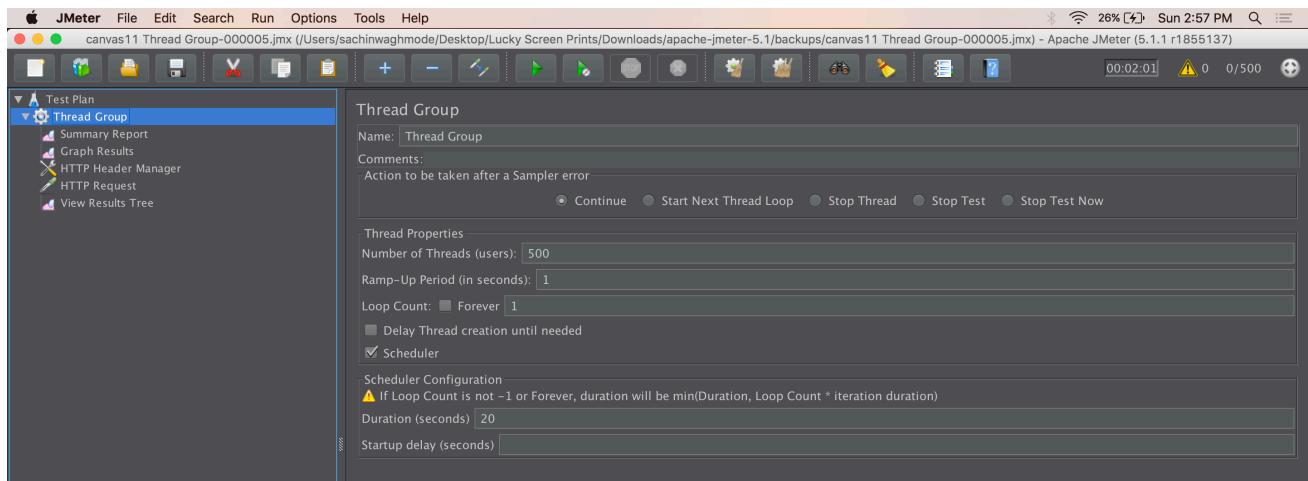
Thread Group

Name: Thread Group  
Comments:  
Action to be taken after a Sampler error: Continue

Thread Properties  
Number of Threads (users): 500  
Ramp-Up Period (in seconds): 1  
Loop Count: Forever 1  
Delay Thread creation until needed  
Scheduler

Scheduler Configuration  
If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count \* iteration duration)  
Duration (seconds): 20  
Startup delay (seconds):

00:02:01 0 0/500



JMeter File Edit Search Run Options Tools Help

26% Sun 2:57 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)

Test Plan

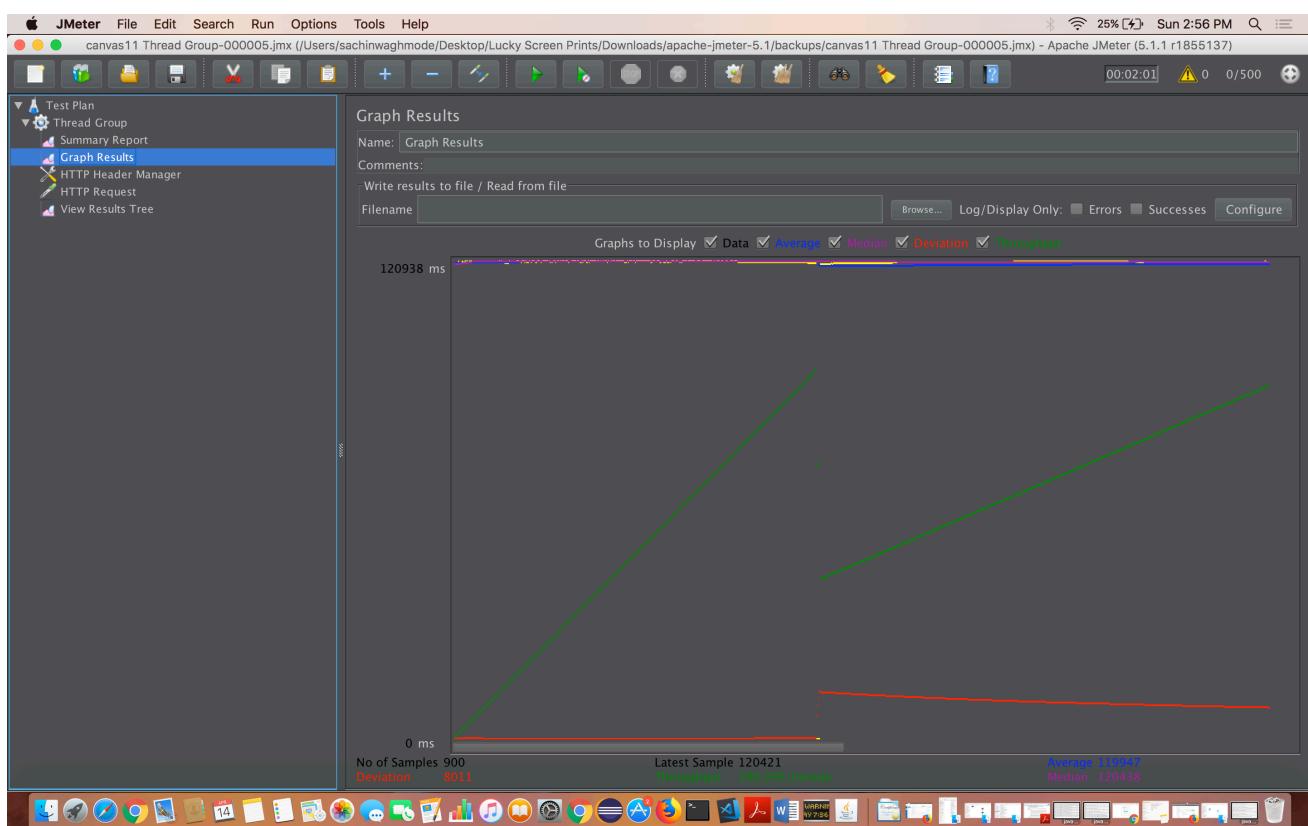
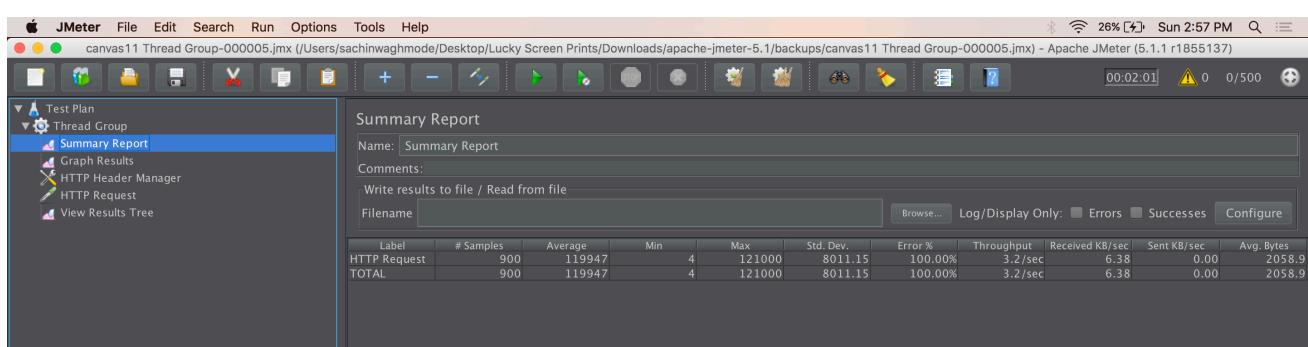
Thread Group

Summary Report

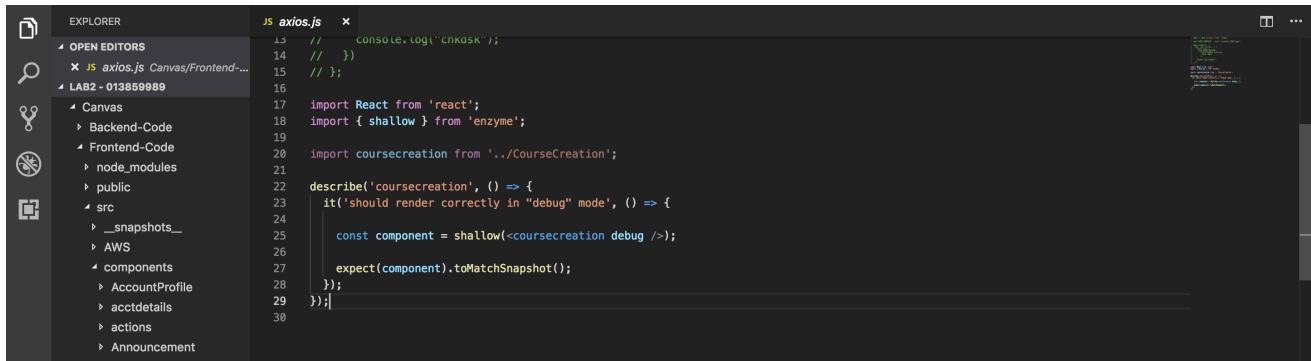
Name: Summary Report  
Comments:  
Write results to file / Read from file  
Filename:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	900	119947	4	121000	8011.15	100.00%	3.2/sec	6.38	0.00	2058.9
TOTAL	900	119947	4	121000	8011.15	100.00%	3.2/sec	6.38	0.00	2058.9

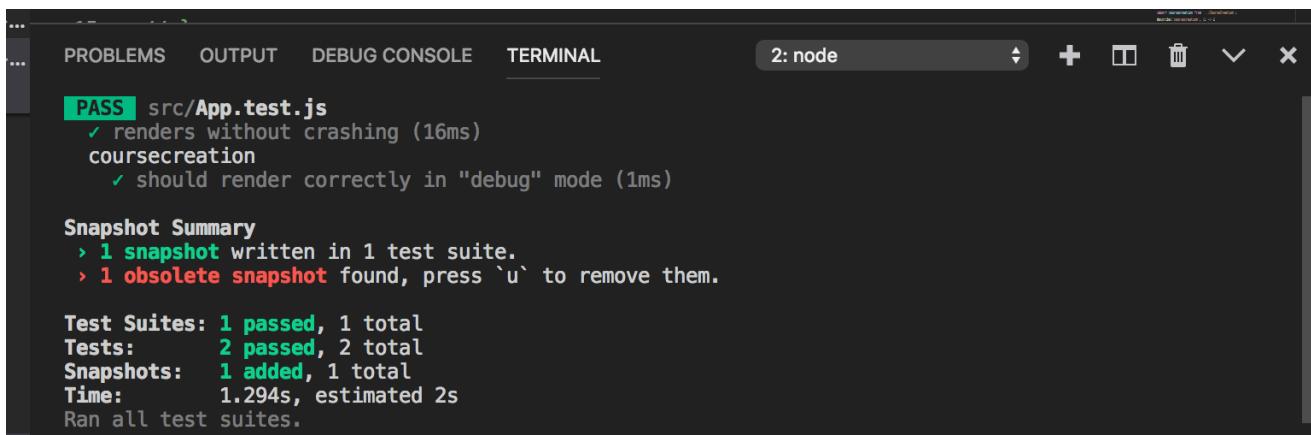
00:02:01 0 0/500



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



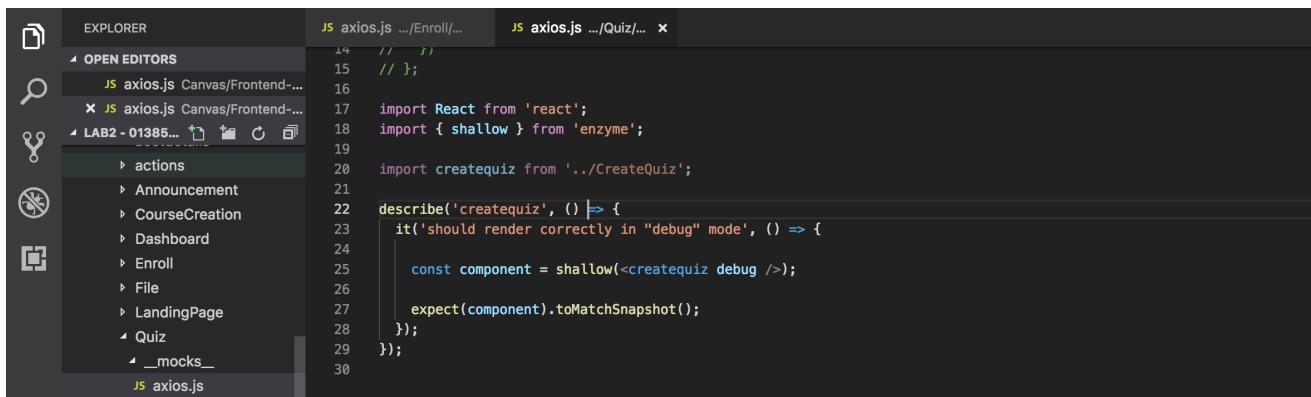
```
JS axios.js x
1 // ...
2 // ...
3 // ...
4 import React from 'react';
5 import { shallow } from 'enzyme';
6
7 import coursecreation from '../CourseCreation';
8
9 describe('coursecreation', () => {
10   it('should render correctly in "debug" mode', () => {
11     const component = shallow(<coursecreation debug />);
12
13     expect(component).toMatchSnapshot();
14   });
15 });
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```



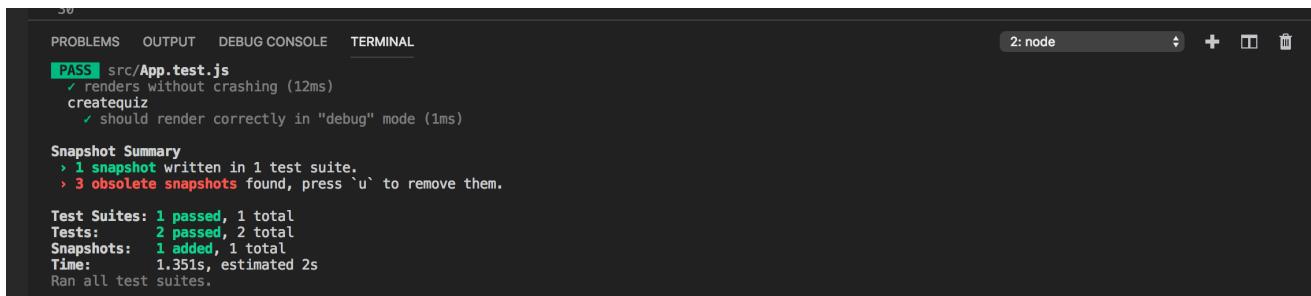
```
PASS src/App.test.js
  ✓ renders without crashing (16ms)
    coursecreation
      ✓ should render correctly in "debug" mode (1ms)

Snapshot Summary
  > 1 snapshot written in 1 test suite.
  > 1 obsolete snapshot found, press `u` to remove them.

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:  1 added, 1 total
Time:        1.294s, estimated 2s
Ran all test suites.
```



```
JS axios.js .../Enroll/... x
1 // ...
2 // ...
3 // ...
4 import React from 'react';
5 import { shallow } from 'enzyme';
6
7 import createquiz from '../CreateQuiz';
8
9 describe('createquiz', () => {
10   it('should render correctly in "debug" mode', () => {
11     const component = shallow(<createquiz debug />);
12
13     expect(component).toMatchSnapshot();
14   });
15 });
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```



```
PASS src/App.test.js
  ✓ renders without crashing (12ms)
    createquiz
      ✓ should render correctly in "debug" mode (1ms)

Snapshot Summary
  > 1 snapshot written in 1 test suite.
  > 3 obsolete snapshots found, press `u` to remove them.

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:  1 added, 1 total
Time:        1.351s, estimated 2s
Ran all test suites.
```

```
15 // };
16
17 import React from 'react';
18 import { shallow } from 'enzyme';
19
20 import enroll from '../Enroll';
21
22 describe('enroll', () => {
23   it('should render correctly in "debug" mode', () => {
24     const component = shallow(<enroll debug />);
25
26     expect(component).toMatchSnapshot();
27   });
28 });
29
30 );
```

```
PASS src/App.test.js
✓ renders without crashing (13ms)
  enroll
    ✓ should render correctly in "debug" mode (1ms)

Snapshot Summary
> 1 snapshot Written in 1 test suite.
> 2 obsolete snapshots found, press 'u' to remove them.

Test Suites: 1 passed, 1 total
Tests: 2 passed, 2 total
Snapshots: 1 added, 1 total
Time: 1.287s, estimated 2s
Ran all test suites.
```

## GITHUB commits and steps added in README.md →

- ↳ Commits on Apr 13, 2019
  - added changes for lab2  
Laxmikant Bhaskar Pandhare authored and Laxmikant Bhaskar Pandhare committed an hour ago
- ↳ Commits on Apr 11, 2019
  - added mongodb Atlas connectivity changes  
Laxmikant Bhaskar Pandhare authored and Laxmikant Bhaskar Pandhare committed 19 hours ago
  - added mongoDB in every call  
Laxmikant Bhaskar Pandhare authored and Laxmikant Bhaskar Pandhare committed 2 days ago
- ↳ Commits on Apr 11, 2019
  - added few more functionality  
Laxmikant Bhaskar Pandhare authored and Laxmikant Bhaskar Pandhare committed 3 days ago
  - added changes for account profile with mongo and kafka  
Laxmikant Bhaskar Pandhare authored and Laxmikant Bhaskar Pandhare committed 4 days ago
  - added changes for passport JS  
Laxmikant Bhaskar Pandhare authored and Laxmikant Bhaskar Pandhare committed 4 days ago
- ↳ Commits on Apr 8, 2019
  - added changes for kafka  
Laxmikant Bhaskar Pandhare authored and Laxmikant Bhaskar Pandhare committed 6 days ago
- ↳ Commits on Apr 2, 2019
  - added mongodb changes for login page as well and its working now  
Laxmikant Bhaskar Pandhare authored and Laxmikant Bhaskar Pandhare committed 13 days ago
- ↳ Commits on Mar 31, 2019
  - removed calculator  
Laxmikant Bhaskar Pandhare authored and Laxmikant Bhaskar Pandhare committed 14 days ago

The screenshot shows a GitHub repository page for the CMPE273-SP19-17 project. The repository is private. The README.md file contains the following content:

```
Create you ec2 instance on AWS and run KAFKA folder on it.

• a. First start Zookeeper and then Kafka Server.
• b. Once your zookeeper and kafka server is up create required topics.
• c. Clone your repo on the AWS instance.
• d. install all dependencies (npm install,passportjs etc).
• e. First run with the folder Backend-Code
    ○ node index.js
• f. Then start your Kafka server in Kafka-Backend folder by command
    ○ node server.js
• g At the end, start your React by below command.
    ○ npm start
```

The commit history for README.md shows the following changes:

- laxmikantbandhare Update README.md (Latest commit c857e38 just now)
- ..
- Canvas added changes for lab2 (an hour ago)
- test added mongo changes in signup and login (14 days ago)
- README.md Update README.md (just now)

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/>