

LAB – 2 Report -- Koushik Kumar Kamala

AWS Frontend:	ec2-3-213-4-33.compute-1.amazonaws.com
AWS Backend:	ec2-34-201-199-224.compute-1.amazonaws.com

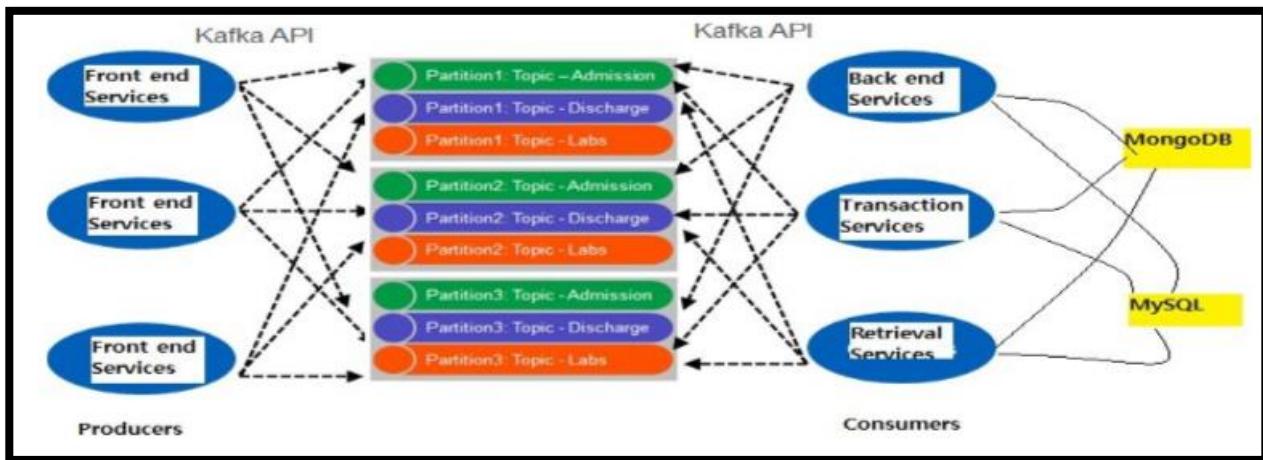
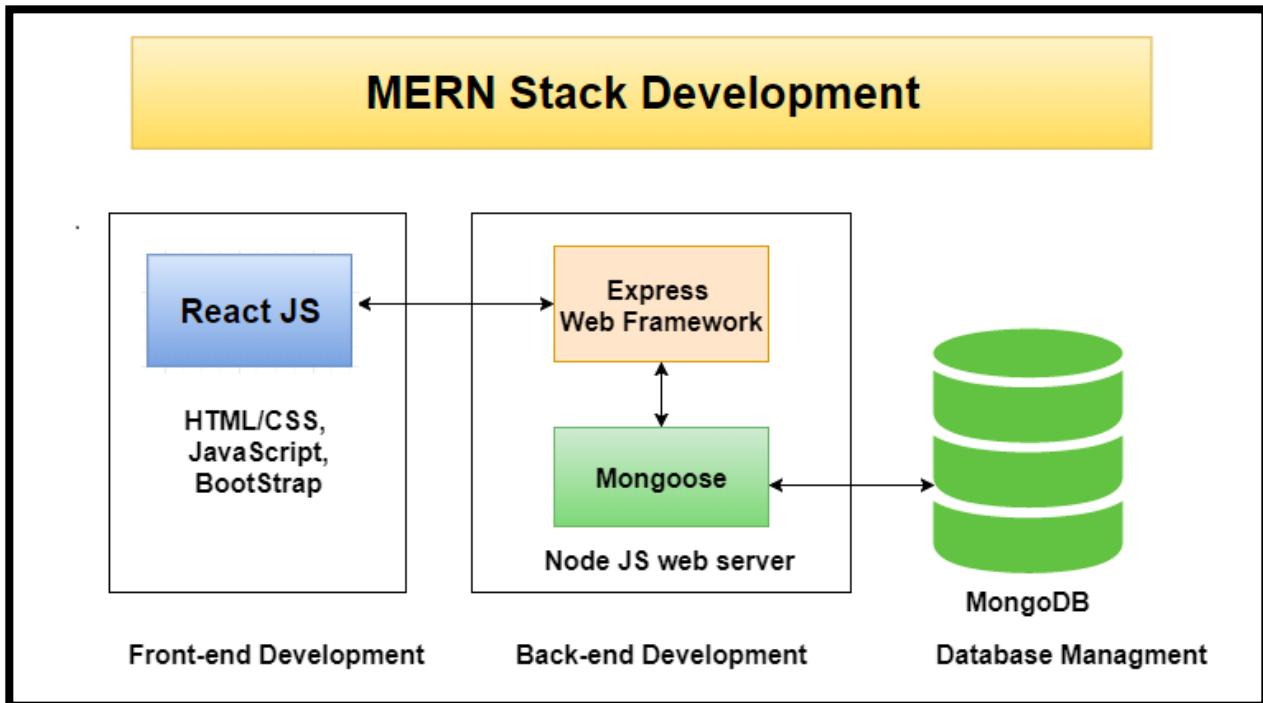
Canvas Application

Introduction:

Goals: Design and implement the prototype of Canvas Application, using MERN stack i.e., MongoDB, Express.js, React and Node.js and Passport.js and JWT for authentication, Redux for state management, Kafka for messaging queues, provided this application is a student-faculty based app with features like registering for courses, dropping, requesting for permission number etc.

System UseCases: This application has two kinds of users mainly, i.e., Student and Professor. A student can search for the courses, register, drop, request for permission number, submit assignments, view their grades, contact professors, view class member, have a course dashboard, showing the courses, a side navigation bar to view calendar, files, courses and profile editing.

System Design:



Following is the operation flow,

1. Starting with a login page, build using JWT and passport, state being managed with ReactJS and redux. Once the user logs in, based on the type of the user, related dashboard is rendered. All the front end code being built by react using npm modules.
2. As soon as a request is made, the same is sent to the backend, being built using node.js, where it uses the route module of the npm manager and routes the request based on the type of the request being made.
3. Kafka is the messaging queue, which communicates with the database and send the corresponding response back to the node which further send the response to the front end. And then using the axios calls' response, according to the response is being reflected on to the browser
4. For this, we have used a MongoDB on cloud using mongodb atlas for storing the data where connection pooling is implemented automatically, and we have also used mongoose npm module for object data modeling. Once the application is built, the frontend is tested using enzyme and unit testing using Mocha

Functionality Flow:

Basically, two types of users, Professor and Student, where the professor has functionalities like create a course, drop students, grant permission, conduct online quizzes, grade them, communicate with the students.

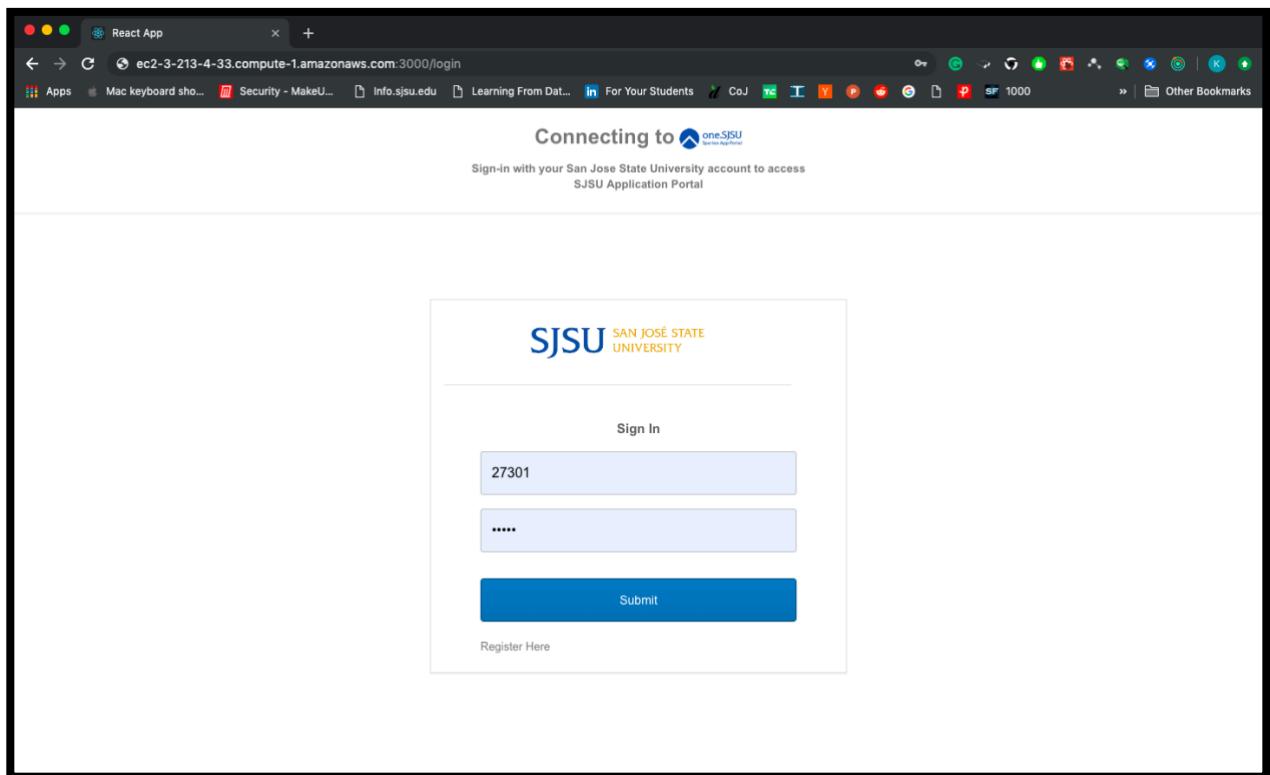
Both the users have following functionalities,

1. Signup new user
2. Signing in for existing users
3. Signout
4. Profile (Name, Contact, Email, Contact, About Me, City, Country, Company, School, Home town, Languages, Gender)
5. Able to edit user profile data anytime
6. Faculty
 - a. The Faculty able to create a course, with course specific details, required by the student and the course

- b. Grant Permission number on request
 - c. Able to view all the courses created by the professor
 - d. Remove the student from a course.
 - e. Make an announcement
7. Student
- a. Student should be able to see all the courses in his dashboard
 - b. Student able to submit the assignments, download the files.
8. Able to organize course cards on the dashboard
9. Students and faculty should be able to message with one another and display the messages under Inbox tab of sidebar.
10. Support the pagination for people search and course search.

Screenshots:

- Sign In



- Sign Up and validation

The screenshot shows a registration form titled "Register!" displayed in a web browser window. The browser's address bar indicates the URL is `ec2-3-213-4-33.compute-1.amazonaws.com:3000/login#`. The page has a dark grey background with a white registration form in the center.

The form fields include:

- Gender selection: Radio buttons for "Student" (selected) and "Professor".
- User ID input field.
- Email ID input field.
- First Name input field.
- Last Name input field.
- Gender selection: Radio buttons for "Male" (selected) and "Female".
- Password input field.
- File upload input field labeled "Choose file" with the message "No file chosen".
- Contact input field.
- About Me text area.
- City input field.

- Sign Up and validation

The screenshot shows a registration form titled "Register!" displayed in a web browser window. The URL in the address bar is "ec2-3-213-4-33.compute-1.amazonaws.com:3000/login#". The form includes fields for User ID, Email ID, FirstName, LastName, gender selection (Male/Female), Password, file upload (Choose file), and Contact information. Error messages are visible for the User ID, Email ID, FirstName, and LastName fields.

Register!

Student Professor

User ID
Invalid User ID

Email ID
Invalid Email ID

FirstName
Please enter first name

LastName
Please enter last name

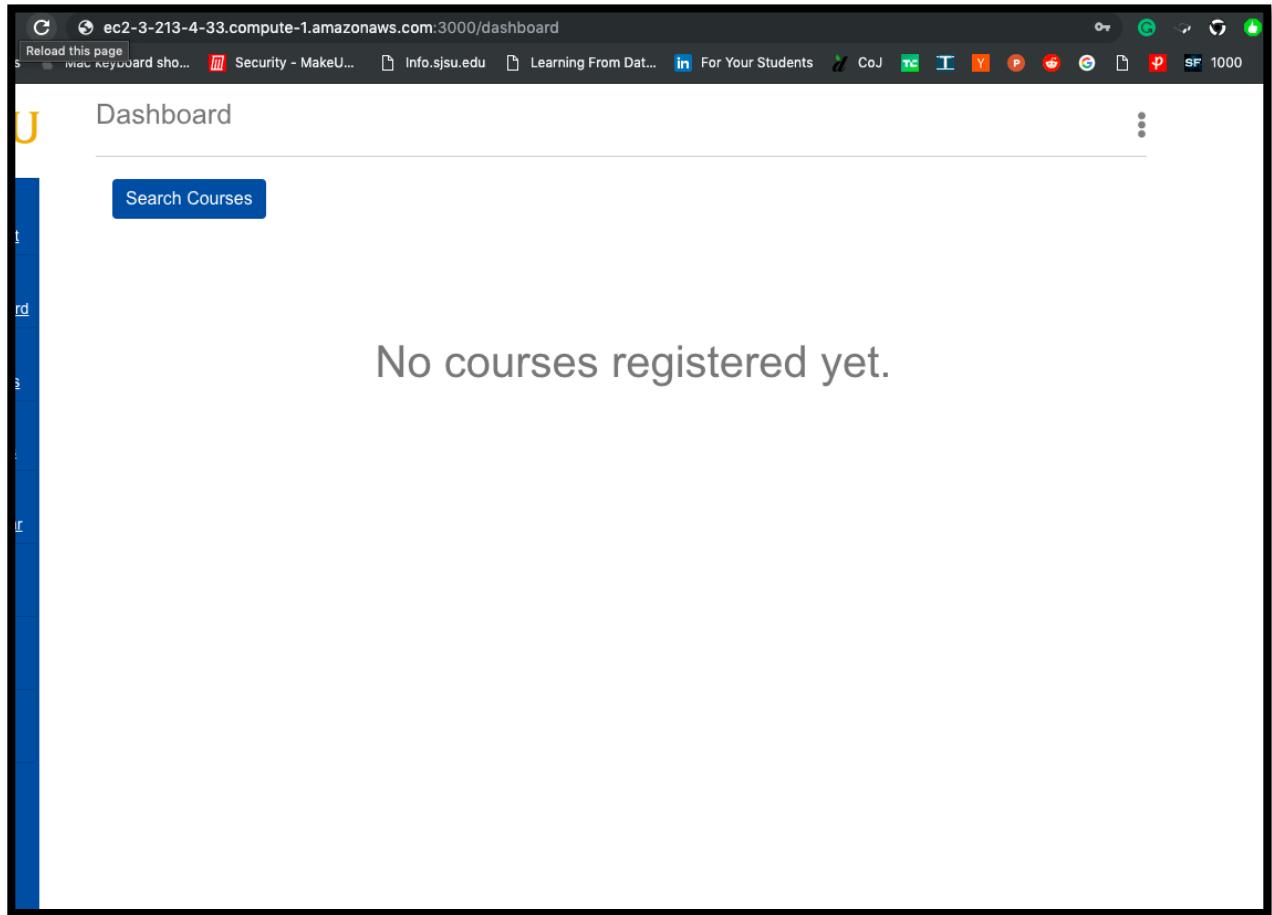
Male Female

Password
Please enter password

Choose file No file chosen

Contact

- No Courses message – and search courses button



- Dashboard – Professor/ student viewing courses

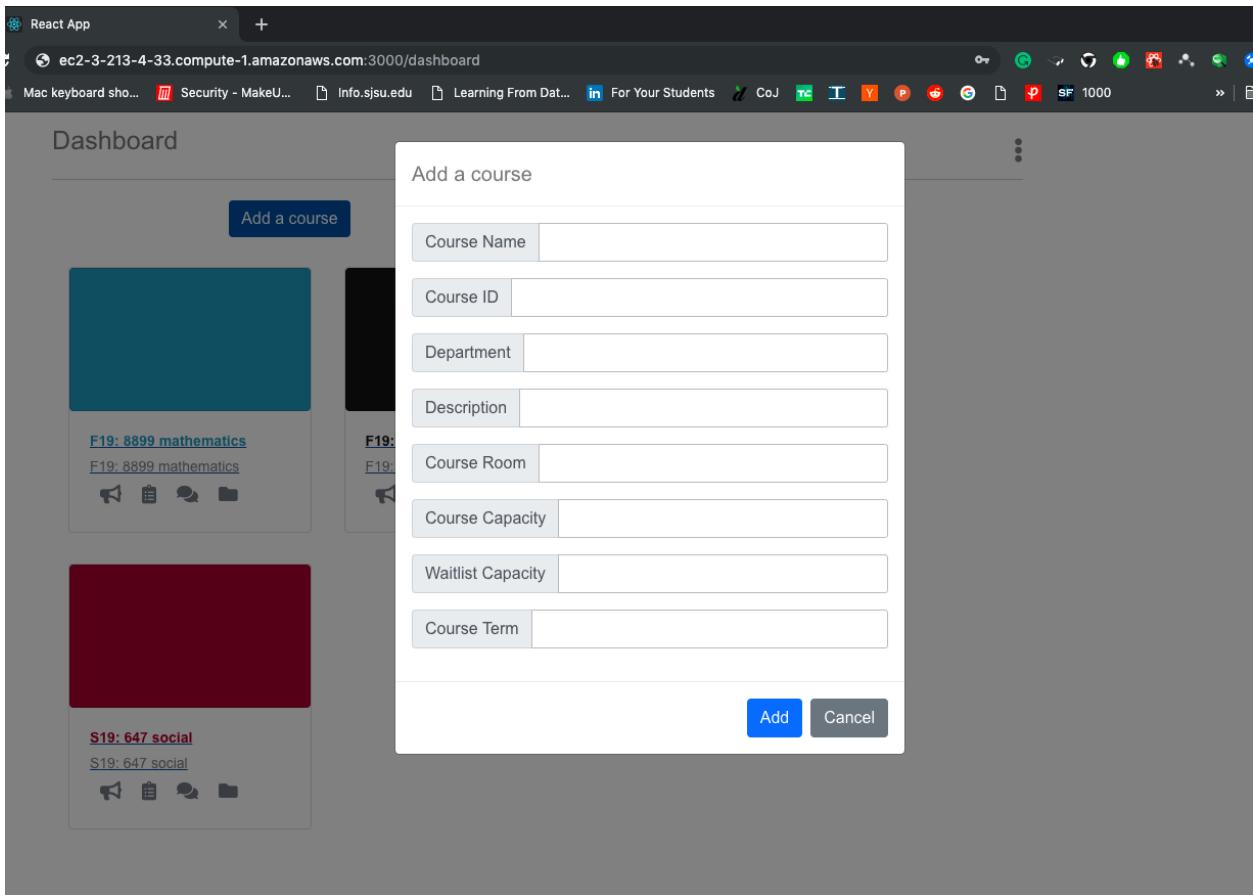
The screenshot shows a web browser window titled "React App" with the URL "ec2-3-213-4-33.compute-1.amazonaws.com:3000/dashboard". The page is titled "Dashboard" and features a sidebar on the left with the SJSU logo. The sidebar contains links for Account, Dashboard, Courses, Groups, Calendar, Inbox, Help, and Library. The main content area displays four course cards arranged in a grid:

- F19: 8899 mathematics** (Blue card)
F19: 8899 mathematics
View Details
- F19: 177 zoology** (Black card)
F19: 177 zoology
View Details
- S19: 647 social** (Red card)
S19: 647 social
View Details
- 619: 76 Physics** (Red card)
619: 76 Physics
View Details

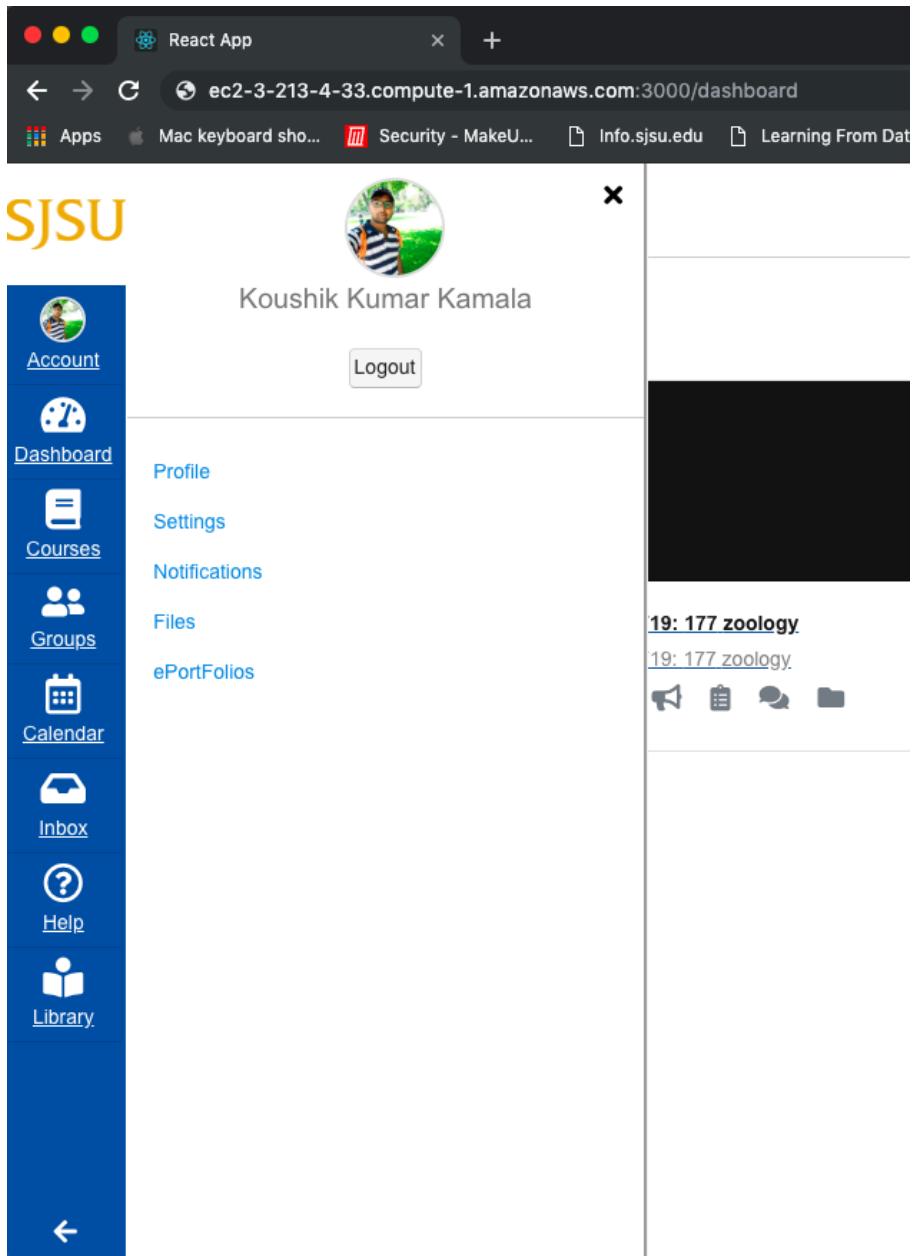
Each card includes icons for a megaphone, a document, a video camera, and a folder.

Add course button visible only to the Professor

- Add a new course



- Sidebar Navigation and Profile Tray



- Sidebar Navigation and Courses Tray

The screenshot shows a web browser window titled "React App" with the URL "ec2-3-213-4-33.compute-1.amazonaws.com:3000/dashboard". The browser's address bar also lists "Apps", "Mac keyboard sho...", "Security - MakeU...", "Info.sjsu.edu", and "Learning Fr...".

The main content area displays the SJSU Learning Management System. At the top left is the SJSU logo. To its right, the word "Courses" is displayed above a list of course links. A large black rectangular redaction box covers the right side of the main content area.

The sidebar on the left contains a vertical navigation menu with the following items:

- Account
- Dashboard
- Courses** (selected)
- Groups
- Calendar
- Inbox
- Help
- Library

The main content area includes the following course links:

- [27319: 273 Enterprise Distributed Systems](#)
- [Fall19: 8899 mathematics](#)
- [27319: 273 Enterprise Distributed Systems](#)
- [Fall19: 177 zoology](#)
- [619: 76 Physics](#)
- [SPRING19: 647 social](#)

Below the course links, a welcome message reads:

Welcome to your courses! To customize the list of courses, click on the "All Courses" link and star the courses to display.

At the bottom right of the main content area, there are four small icons: a megaphone, a clipboard, a speech bubble, and a folder.

- Generate Permission Numbers by Professor

The screenshot shows a web browser window titled "React App" with the URL "ec2-3-213-4-33.compute-1.amazonaws.com:3000/courses/8899". The page is for a course titled "8899 : mathematics". On the left is a vertical sidebar with icons and labels: Account, Dashboard, Courses (highlighted), Groups, Calendar, Inbox, Help, and Library. The main content area has a title "Course Enrollment Details" and displays the following information in boxes:

- Course Capacity: 40
- Waitlist Capacity: 10
- Total Students: 50

Below these boxes is a button labeled "Generate Permission Numbers". A large text box contains a list of generated permission numbers:

```
6062781, 4815516, 7550587, 9397837, 7421879, 6980670, 4374188, 6120774, 7699755, 5773458, 7891552, 3980522,  
2147329, 1867073, 9220857, 5772934, 5418626, 7916493, 2107480, 2159592,
```

- Assignment View for a course

The screenshot shows a web browser window with a URL starting with "33.compute-1.amazonaws.com:3000/courses/8899?asgnmtname=Assignment3&asgnmtdue=2019-04-30&asgnmtmark...". The page displays a list of assignments for a course. The assignments are:

Assignment	Due Date
Assignment1	Due 2019-04-18T00:00:00.000Z
Assignment2	Due 2019-04-30T00:00:00.000Z
Assignment3	Due 2019-04-30T00:00:00.000Z

There is a blue button labeled "+Assignments" in the top right corner of the assignment list.

- Files Upload

The screenshot shows a web browser window with the URL <http://ec2-3-213-4-33.compute-1.amazonaws.com:3000/courses/8899#>. The page title is "8899 : mathematics". On the left, there is a sidebar with various course links: Home, PermissionCodes, Announcements, Assignments, Grades, People, Files, Quizzes, Conferences, Collaborations, Chat, Criterion, Portfolium, and SOTE/SOLATE. The main content area is titled "Files". It contains instructions for creating a folder ("To create a folder: Folder Name Create Folder") and uploading a file ("Upload a file: Choose file No file chosen into / Folder Name Upload File"). Below these, there is a file list with a single item: "200-canvaDB (1).pdf". The file path is shown above the list as "demos" and "notes".

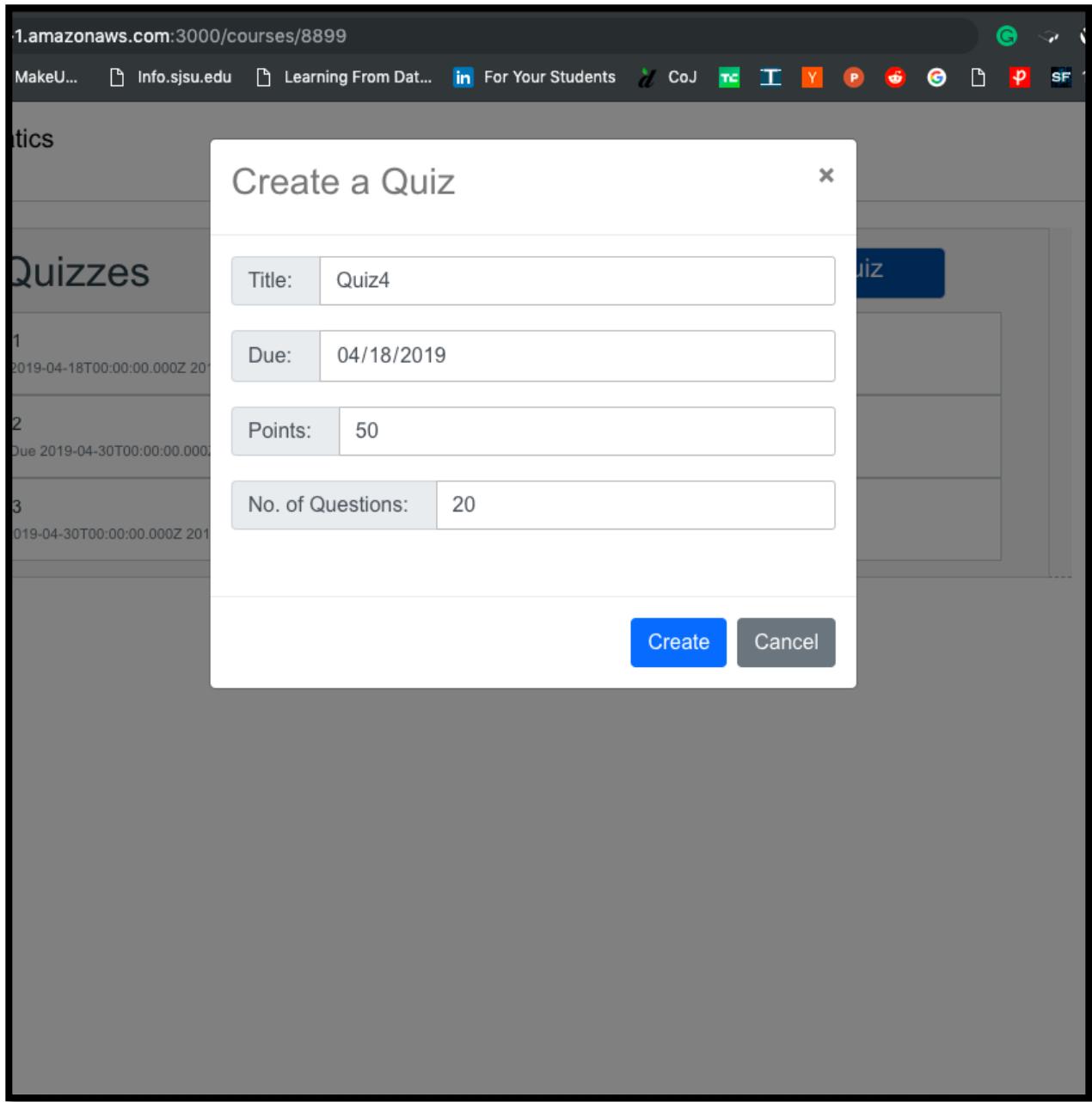
- Quizzes - list

The screenshot shows a web browser window with the URL ec2-3-213-4-33.compute-1.amazonaws.com:3000/courses/8899. The page title is "8899 : mathematics". On the left, there is a sidebar with various navigation links: Home, PermissionCodes, Announcements, Assignments, Grades, People, Files, Quizzes, Conferences, Collaborations, Chat, Criterion, Portfolium, and SOTE/SOLATE. The main content area is titled "Quizzes" and contains three entries:

- Quiz1**
Due: 2019-04-18T00:00:00.000Z | points 50 | Questions 10
- Quiz2**
Due: 2019-04-30T00:00:00.000Z | points 40 | Questions 20
- Quiz3**
Due 2019-04-30T00:00:00.000Z | points 30 | Questions 15

A blue button labeled "+Quiz" is located in the top right corner of the main content area.

- Quizzes – Add a quiz



- Announcements – Add a announcement

The screenshot shows a web browser window with a modal dialog box titled "Make an announcement". The modal contains fields for "Title:" and "Announcement:", both currently empty. At the bottom right of the modal are two buttons: "Post" (highlighted in blue) and "Cancel". In the background, there is a list of announcements. The first announcement is titled "Announcement" and contains the text "Hi everyone, this homework consists of basic coding snippets on javascript.". This announcement was posted on 29th Oct, 2018. The second announcement is also titled "Announcement" and contains the same text. It was also posted on 29th Oct, 2018. The third announcement is partially visible. The browser's address bar shows "mpute-1.amazonaws.com:3000/courses/8899". The title bar of the browser window says "Announcements".

ec2-3-213-4-33.compute-1.amazonaws.com:3000/courses/8899

Mac keyboard sho... Security - MakeU... Info.sjsu.edu Learning From Dat... For Your Students CoJ Y P G D SF 1000

8899 : mathematics

Home **Grades for Koushik Kumar Kamala**

PermissionCodes

Announcements Course Arrange By

Assignments zoology Assignment Group

Grades

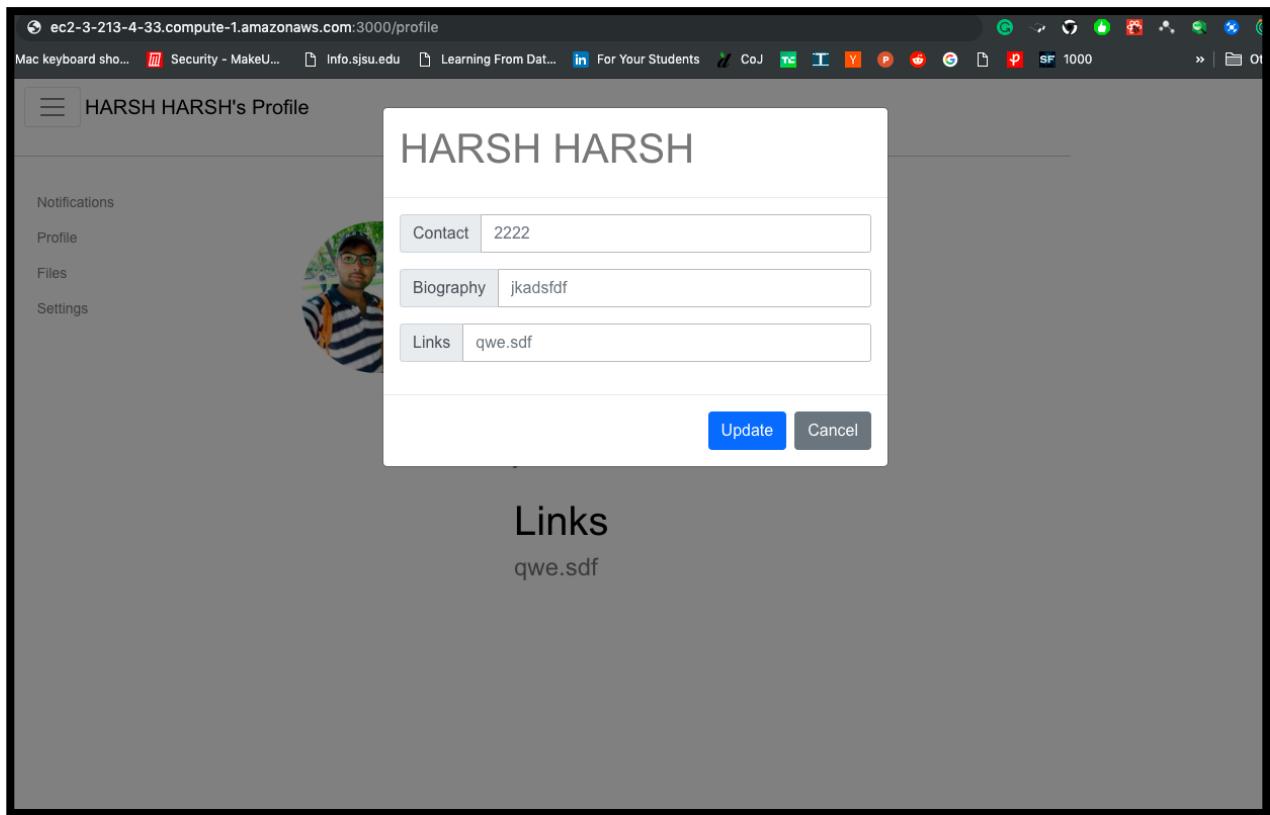
People

	Name	Due	Status	Score	Out of
Files	Homework1	Feb 23 by 11:59pm	Late	10	10
Quizzes	Midterm	Feb 23 by 11:59pm		5	10
Conferences					
Collaborations	Homework1	Feb 23 by 11:59pm		10	10
Chat					
Criterion					
Portfolium					
SOTE/SOLATE					

- Profile View

The screenshot shows a web browser window with a profile page titled "HARSH HARSH's Profile". The browser's address bar displays the URL "ec2-3-213-4-33.compute-1.amazonaws.com:3000/profile". The sidebar on the left contains several icons with labels: "S", "Board", "ses", "Jobs", "Calendar", "ox", "Op.", and "ary.". The main content area features a circular profile picture of a man with glasses and a striped shirt. The name "HARSH HARSH" is prominently displayed above a "Contact" section which includes the number "2222". Below this is a "Biography" section with the text "jkadsfdf" and a "Links" section with the text "qwe.sdf". An "Edit Profile" button is located in the top right corner of the main content area.

- Profile Edit



- Login – Backend console

```
Mongo DB is connected!
BACKEND: ----- INSDIE LOGIN POST
Request Body: { UserID: '27301', password: '27301' }
in make request
{ UserID: '27301', password: '27301' }
1
returning next
in response
in response1
true
=====Inside Backend - Get Courses =====
Request Body: { UserID: '27301', password: '27301' }
in make request
{ UserID: '27301', password: '27301' }
in response
in response1
true
client ready!
in response2
{ login: { '0': 70 } }
in response2
{ getCourses: { '0': 40 } }
msg received
response { coursesEnrolled:
[ { CourseID: '190',
  TeacherID: '27301',
  EnrollmentStatus: 'Course Teacher' },
  { CourseID: '87',
  TeacherID: '27301',
  EnrollmentStatus: 'Course Teacher' },
  { CourseID: '78',
  TeacherID: '27301',
  EnrollmentStatus: 'Course Teacher' },
  { CourseID: '8789',
  TeacherID: '27301',
  EnrollmentStatus: 'Course Teacher' },
  { CourseID: '13234',
```

- Login – Backend console

```
client ready!
(node:11372) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
message received for login { handle_request: [Function: handle_request] }
"{"correlationId":"\\"7e582366b38bf476830f54da3d64cb9\\","\\replyTo\\":\\"response_topic\\","\\data\\":{\\\"UserID\\\":\"27301\\\",\\\"password\\\":\"27301\\\"}}"
=====Inside Kafka Backend - Login =====
Message { UserID: '27301', password: '27301' }
message received for getCourses { handle_request: [Function: handle_request] }
"{"correlationId":"\\"92ecf1bb89db732f12638ccc72595ad9\\","\\replyTo\\":\\"response_topic\\","\\data\\":{\\\"UserID\\\":\"27301\\\",\\\"password\\\":\"27301\\\"}}"
=====Inside Kafka Backend - Get courses =====
Message { UserID: '27301', password: '27301' }
User Details : { coursesEnrolled:
  [ { CourseID: '190',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '87',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '78',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '8789',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '13234',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '89',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '123',
      TeacherID: '27301',
```

- Add a course – Backend console

```
Mongo DB is connected!
Backend: ----- Inside addCourse route
req { CUID: 20,
  CourseName: 'biology',
  CourseID: '202',
  department: 'CMPE',
  description: 'biology',
  room: '34',
  capacity: '23',
  waitlist: '4',
  term: 'Fall',
  color: '#FF3A15',
  UserID: '27301' }
Updating User data
Successfully Added a course
POST /createCourse 201 234.377 ms - 57
(node:11421) DeprecationWarning: collection.findAndModify is deprecated. Use
{ coursesEnrolled:
  [ { CourseID: '190',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '87',
      TeacherID: '27301' } ] }
```

- Get Profile Data – Kafka console

```
client ready!
(node:11415) DeprecationWarning: collection.ensureIndex is deprecated
message received for getProfileData { handle_request: [Function: han
"{"correlationId": "9cd88426b342a0b4f83bb741bcfc2c91"}, {"replyTo": ""
=====Inside Kafka Backend - Get Profile Data ===
Message { UserID: '27301' }
Redering user details....
after handle{ coursesEnrolled:
  [ { CourseID: '190',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '87',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '78',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '8789',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '13234',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' }.
```

- Update Profile Data – Kafka console

```
}"
=====Inside Kafka Backend - UpdateProfile =====
Message { UserID: '27301',
  contact: '273458',
  biography: 'PROFESSOR',
  links: 'ABC.ABC' }
(node:11415) DeprecationWarning: collection.findAndModify is deprecated. Use findOneAndUpdate, find0
Profile updated
after handle{ coursesEnrolled:
  [ { CourseID: '190',
    TeacherID: '27301',
    EnrollmentStatus: 'Course Teacher' },
  { CourseID: '87',
    TeacherID: '27301',
    EnrollmentStatus: 'Course Teacher' },
  { CourseID: '78',
    TeacherID: '27301',
    EnrollmentStatus: 'Course Teacher' },
  { CourseID: '8789',
    TeacherID: '27301' }
```

- Add a course – Kafka console

```
(node:11421) DeprecationWarning: collection.ensureIndex is
Mongo DB is connected!
Backend: ----- Inside addCourse route
req { CUID: 20,
  CourseName: 'biology',
  CourseID: '202',
  department: 'CMPE',
  description: 'biology',
  room: '34',
  capacity: '23',
  waitlist: '4',
  term: 'Fall',
  color: '#FF2A2E' }
```

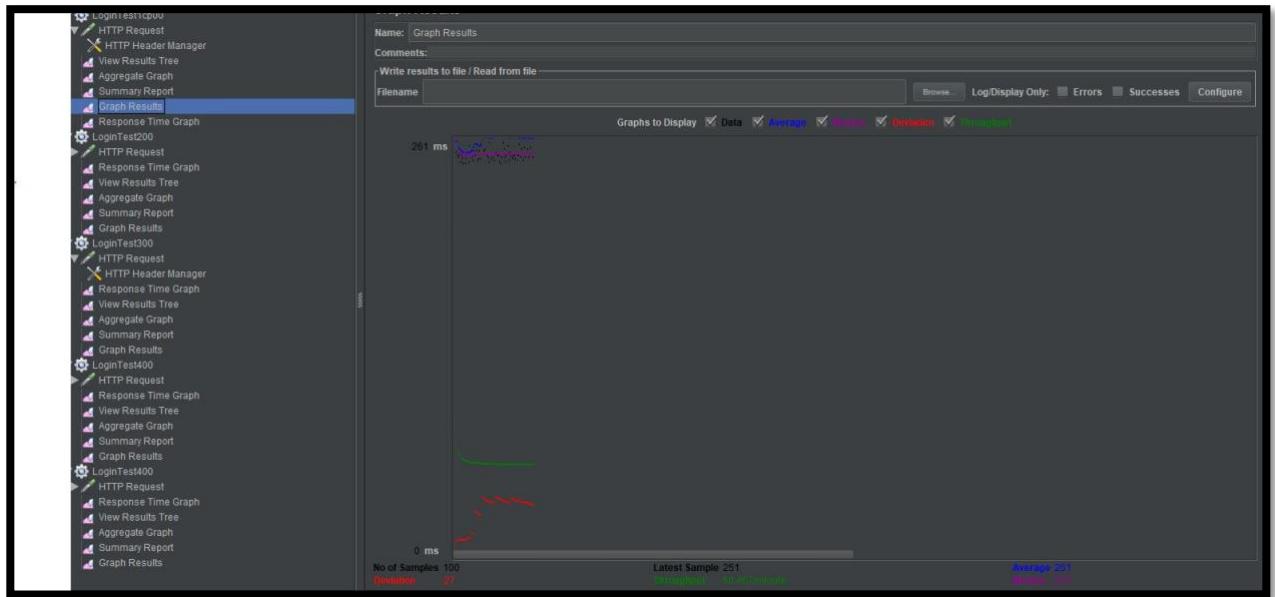
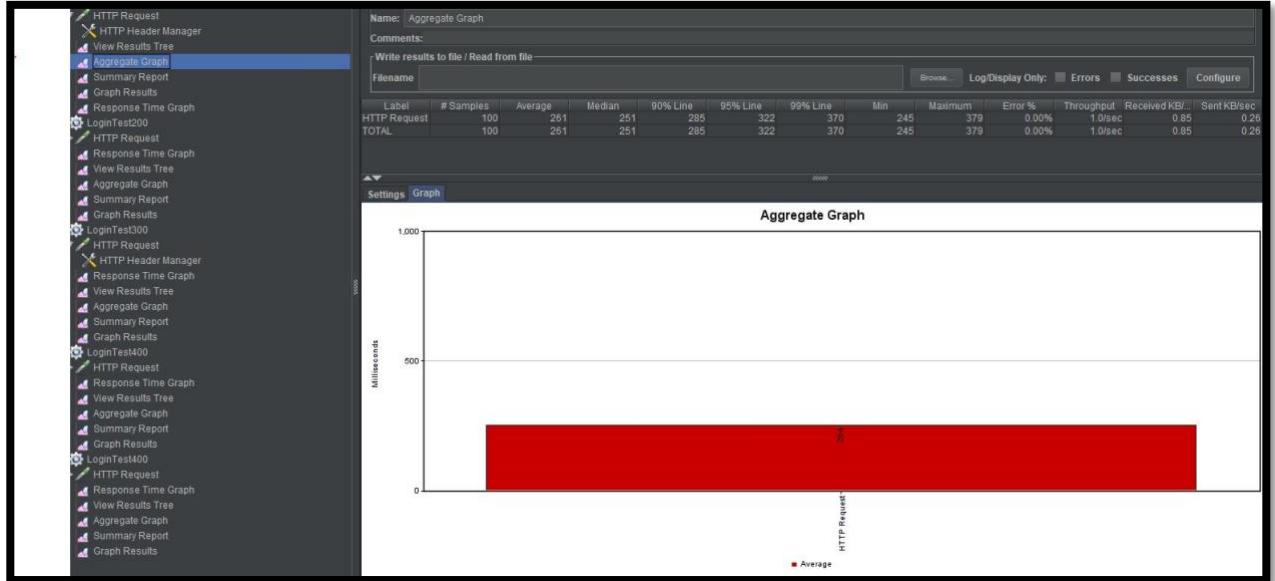
```
Successfully Added a course
POST /createCourse 201 234.377 ms - 57
(node:11421) DeprecationWarning: collection.findAndModify
{ coursesEnrolled:
  [ { CourseID: '190',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '87',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '78',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' },
    { CourseID: '8789',
      TeacherID: '27301',
      EnrollmentStatus: 'Course Teacher' } ] }
```

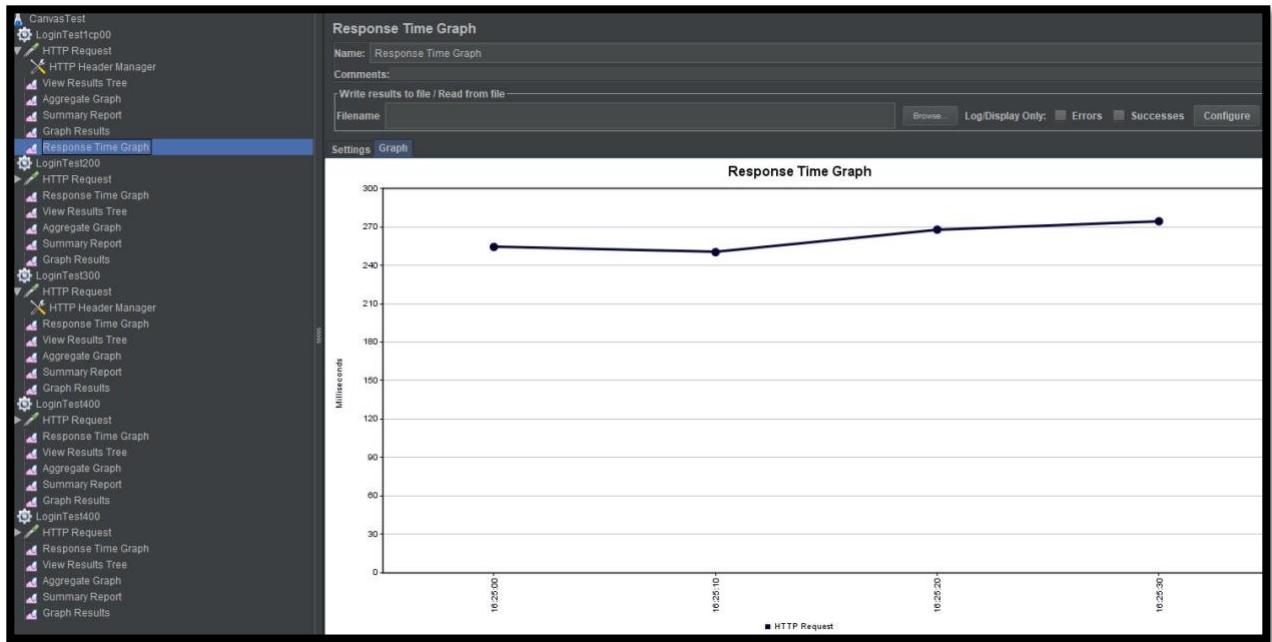
- **Pagination**

People	
Student Name	Student Id
ranjith	443
vinay	444
Previous	Next

People	
Student Name	Student Id
99	99
Previous	Next

- JMeter
For 100 calls,



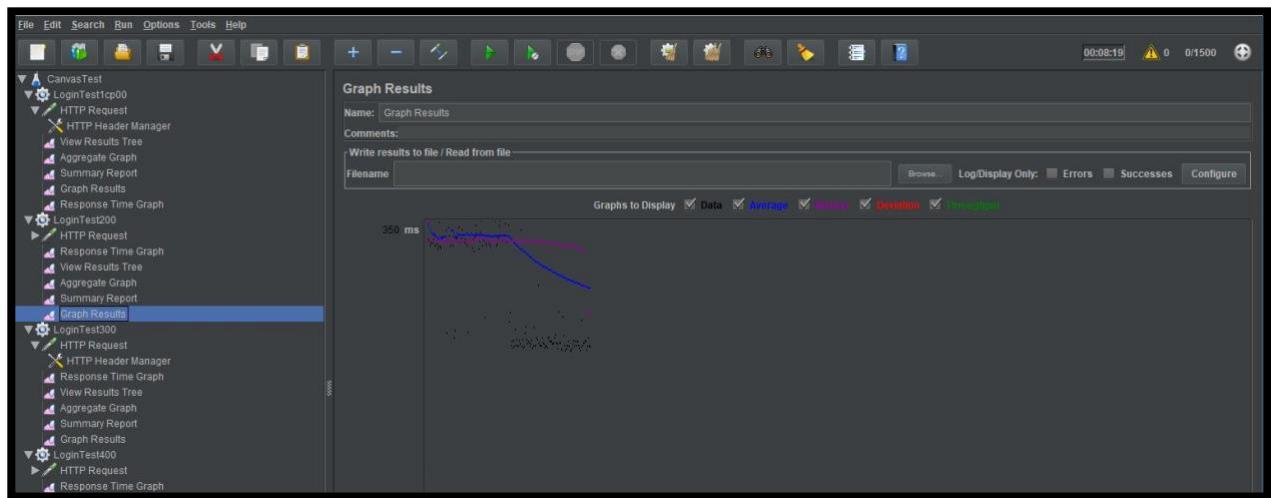
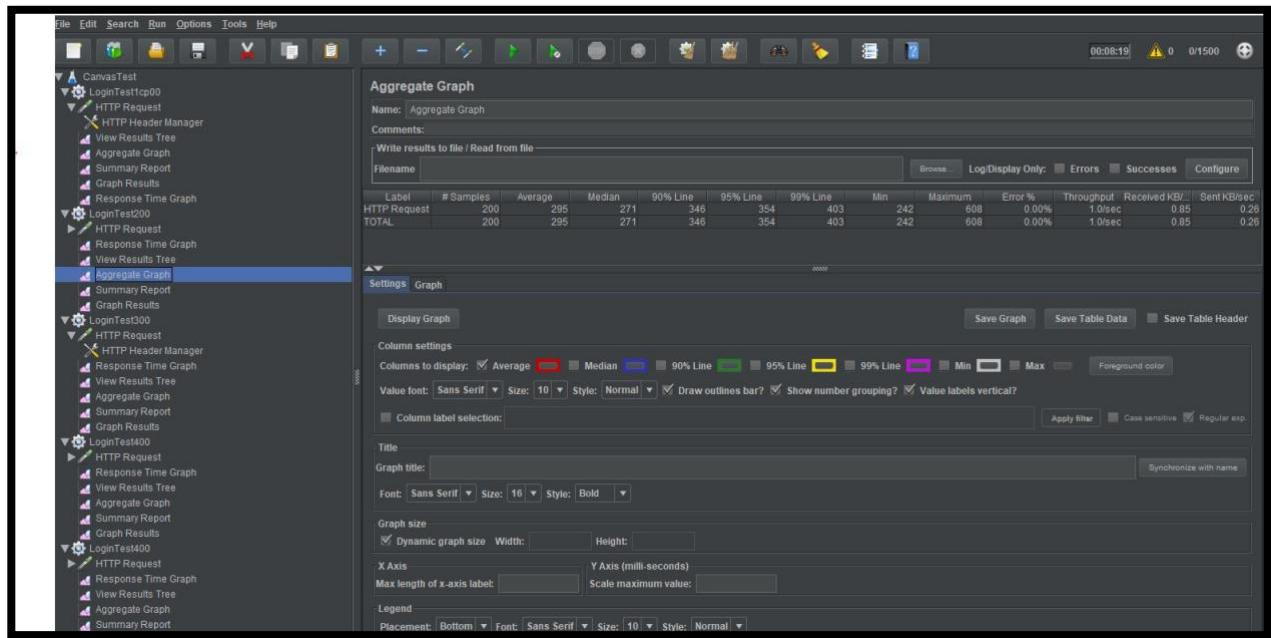


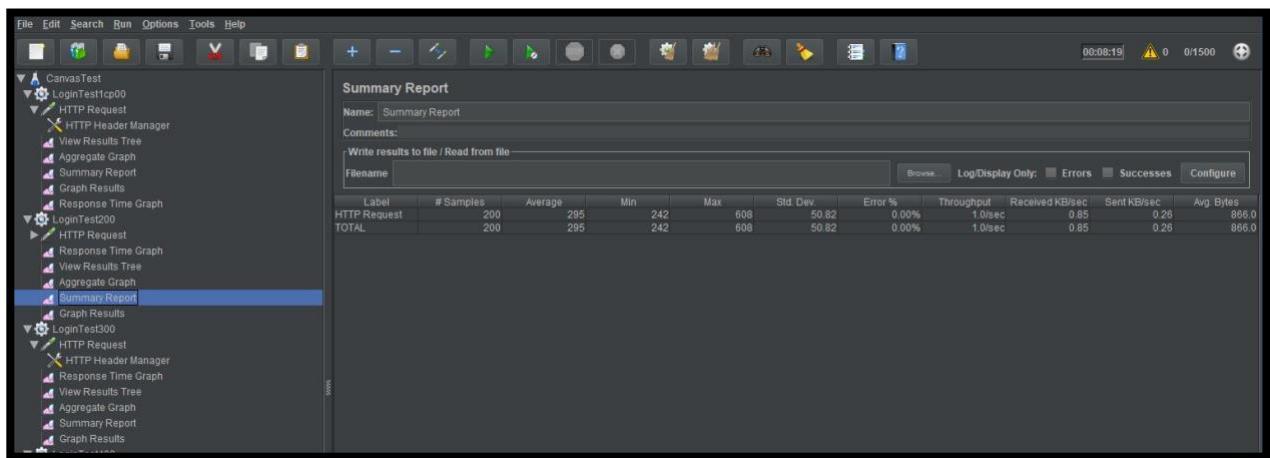
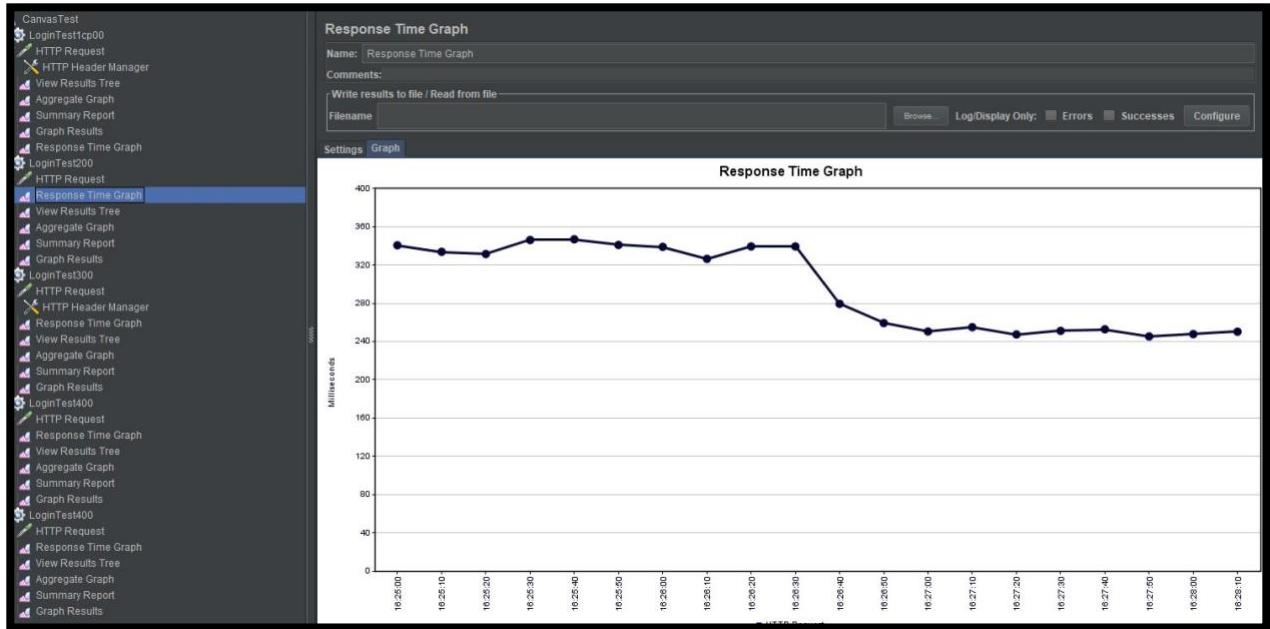
Summary Report

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	100	261	245	379	27.99	0.00%	1.0/sec	0.85	0.26	866.0
TOTAL	100	261	245	379	27.99	0.00%	1.0/sec	0.85	0.26	866.0

For 200 calls,





For 300 calls,

Screenshot of the JMeter Aggregate Graph results window.

Aggregate Graph

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

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	300	375	335	507	598	766	241	3346	0.00%	1.0/sec	0.85	0.24
TOTAL	300	375	335	507	598	766	241	3346	0.00%	1.0/sec	0.85	0.24

Settings **Graph**

Display Graph **Save Graph** **Save Table Data** **Save Table Header**

Column settings
Columns to display: Average Median 90% Line 95% Line 99% Line Min Max Foreground color

Value font: Sans Serif **Size**: 10 **Style**: Normal Draw outlines bar? Show number grouping? Value labels vertical?

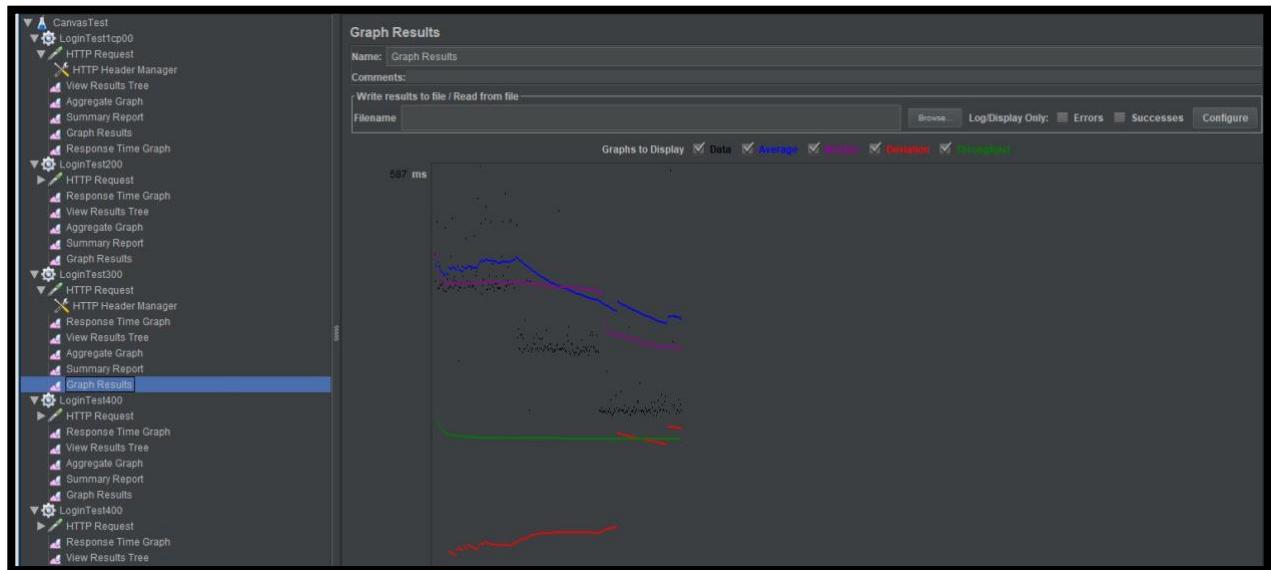
Column label selection: Apply filter Case sensitive Regular exp **Synchronize with name**

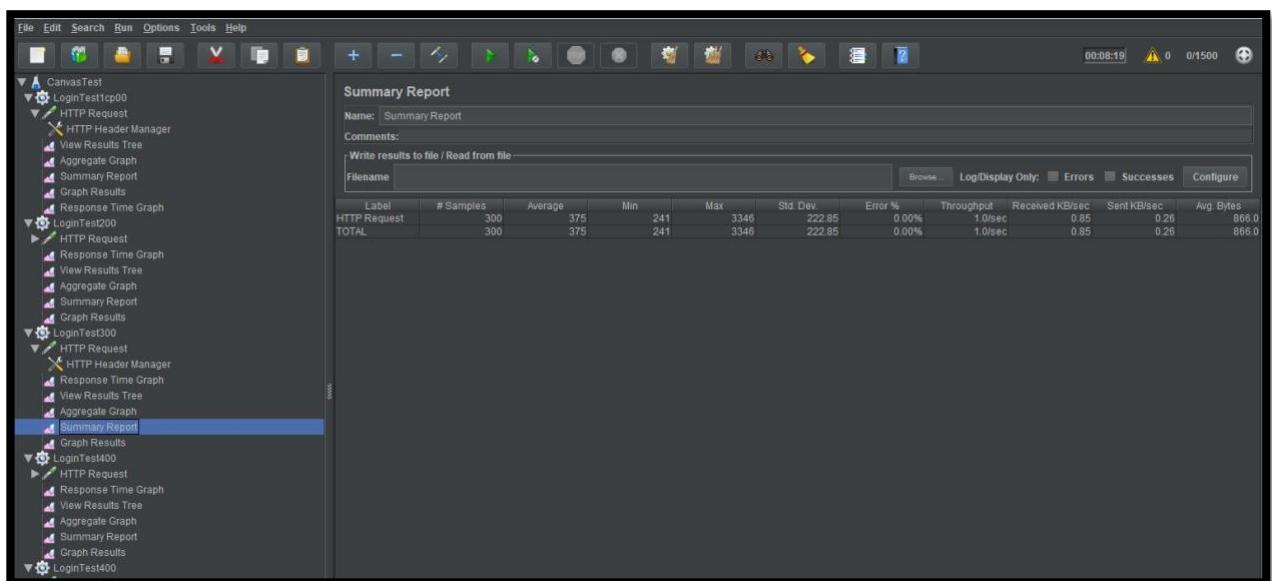
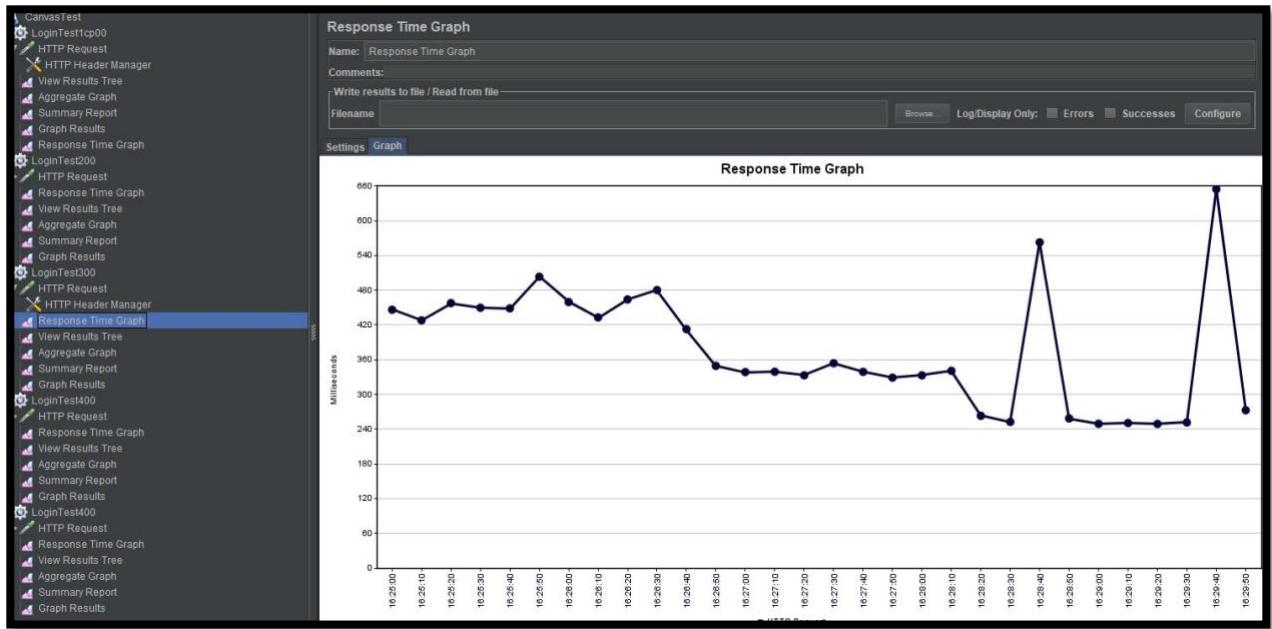
Title
Graph title:
Font: Sans Serif **Size**: 16 **Style**: Bold

Graph size
 Dynamic graph size **Width**: **Height**:

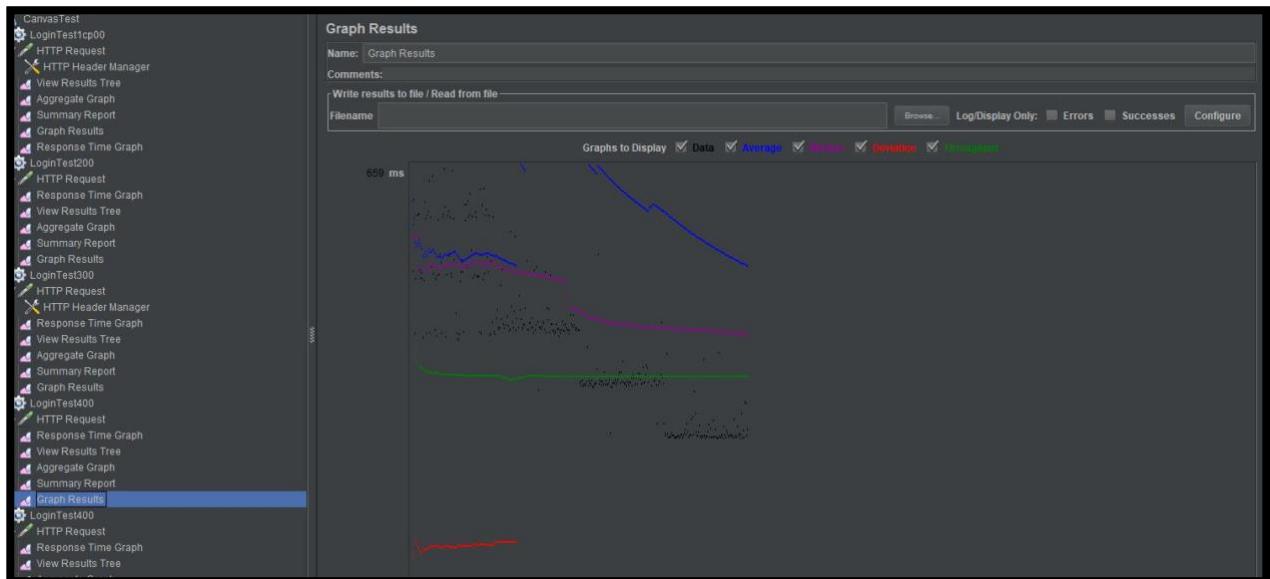
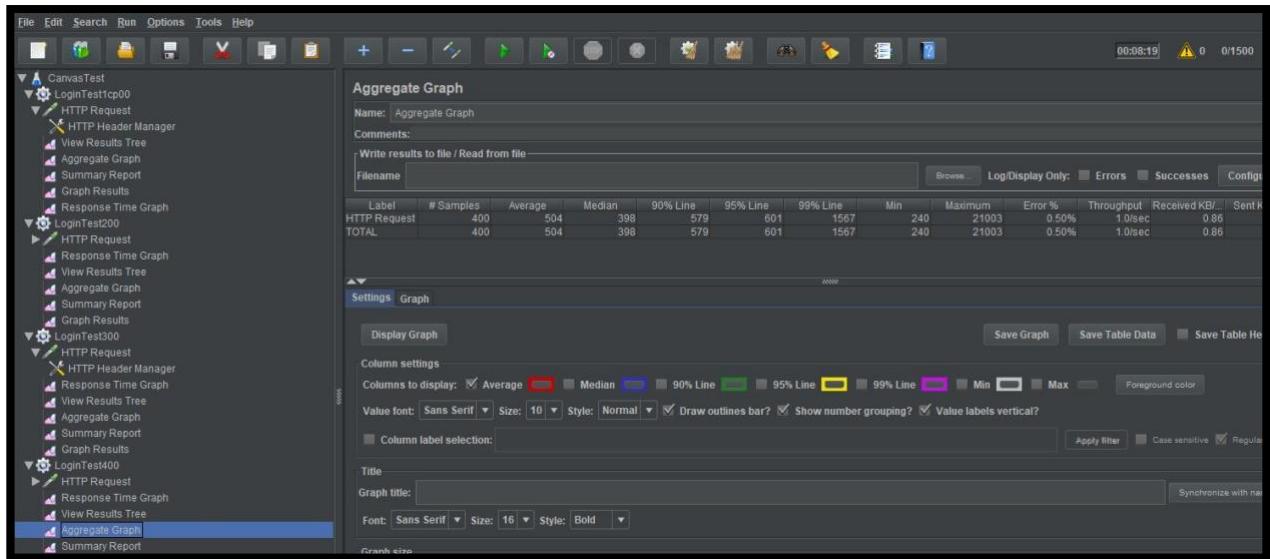
X Axis **Y Axis (milli-seconds)**
Max length of x-axis label: **Scale maximum value**:

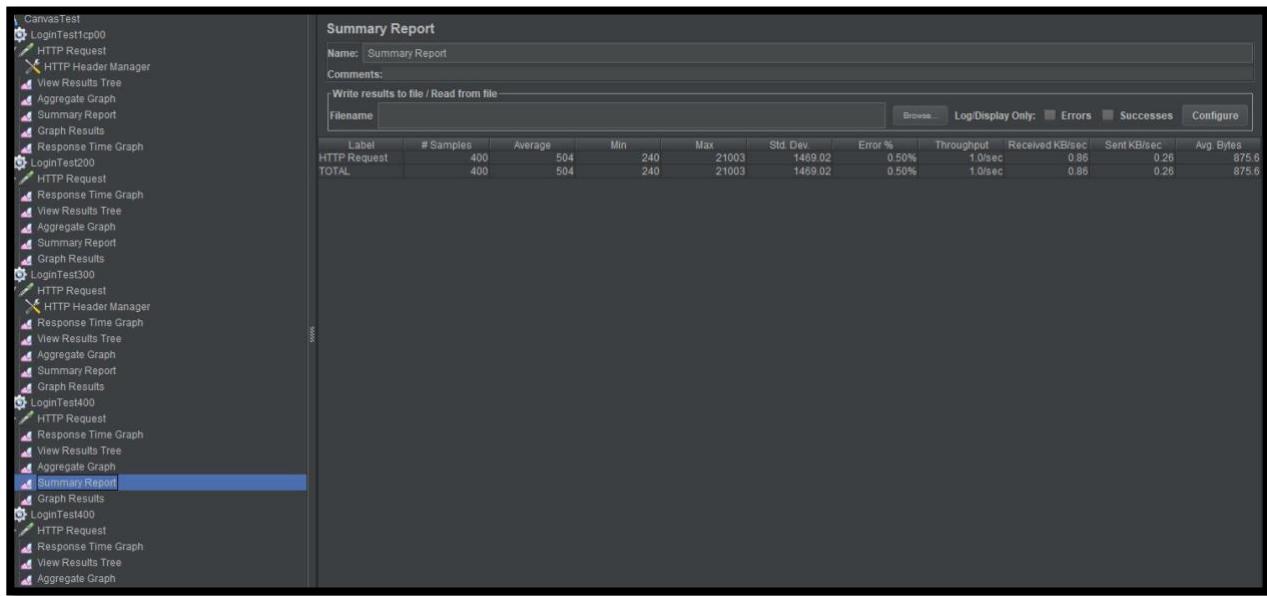
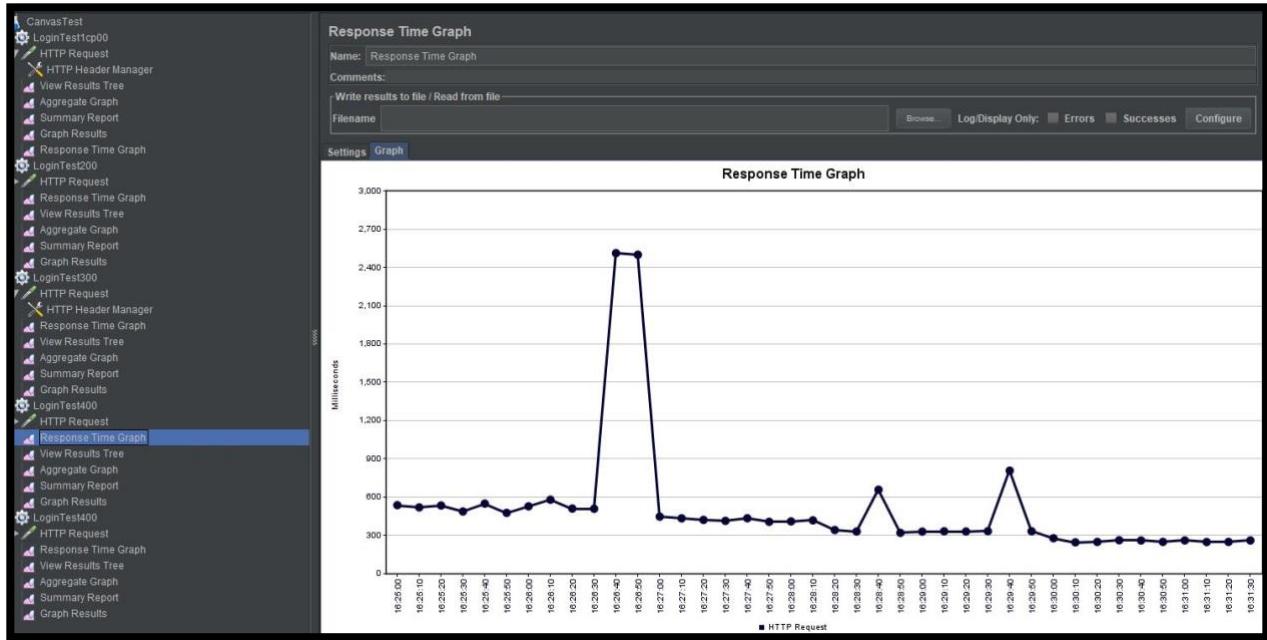
Legend
Placement: Bottom **Font**: Sans Serif **Size**: 10 **Style**: Normal





For 400 calls,





For 500 calls,

Aggregate Graph

Name: Aggregate Graph
Comments:
Write results to file / Read from file
Filename: Browse Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB	Sent KB/sec
HTTP Request	500	404	390	547	560	902	237	3702	0.00%	1.0/sec	0.85	0.26
TOTAL	500	404	390	547	560	902	237	3702	0.00%	1.0/sec	0.85	0.26

Settings Graph

Display Graph Save Graph Save Table Data Save Table Header

Column settings: Average Median 90% Line 95% Line 99% Line Min Max Foreground color

Value font: Sans Serif Size: 10 Style: Normal Draw outlines bar? Show number grouping? Value labels vertical?

Column label selection: Apply filter Case sensitive Regular exp

Title: Graph title: Synchronize with name

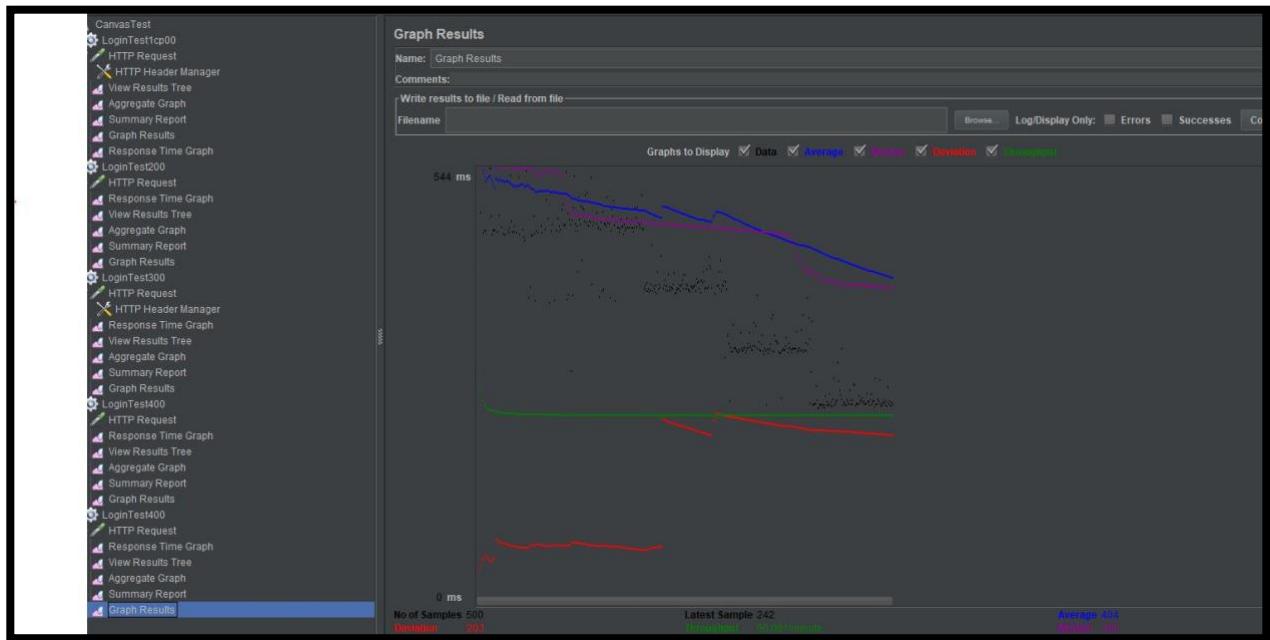
Font: Sans Serif Size: 16 Style: Bold

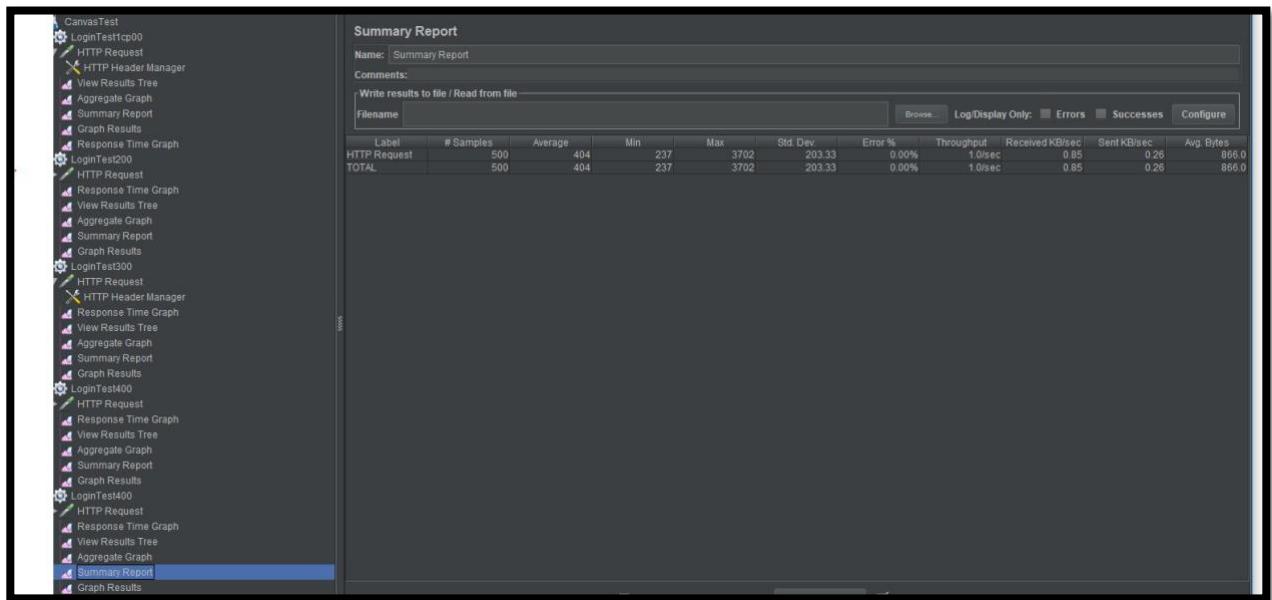
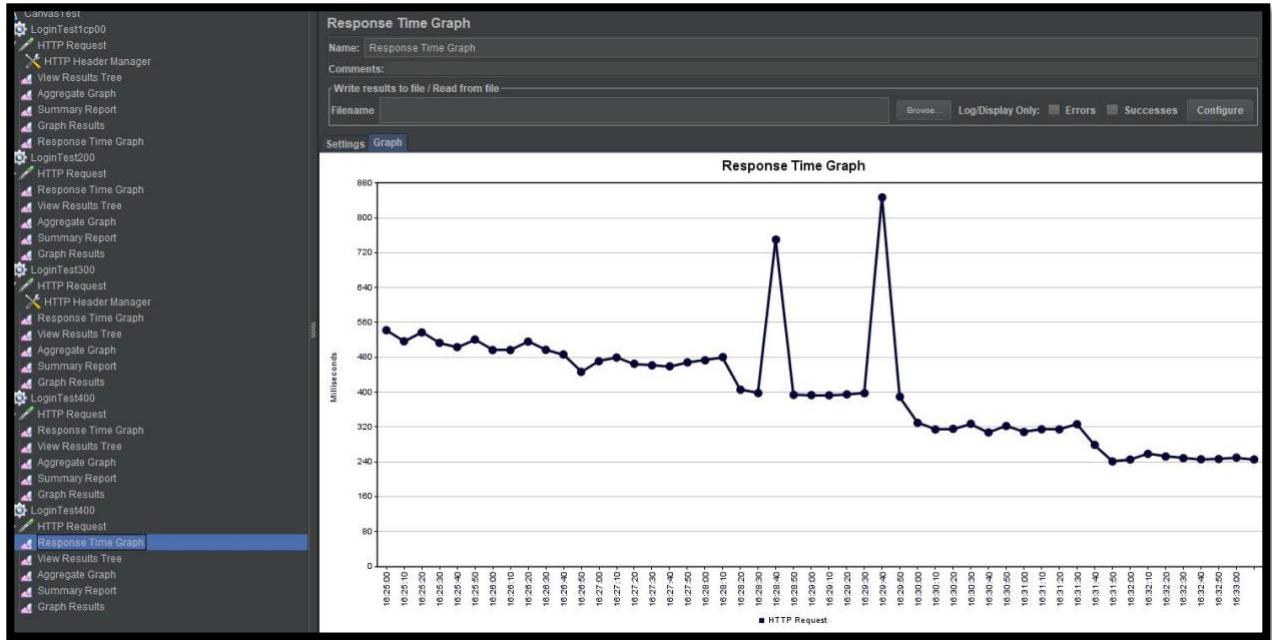
Graph size: Dynamic graph size Width: Height:

X Axis: Y Axis (milli-seconds)

Max length of x-axis label: Scale maximum value:

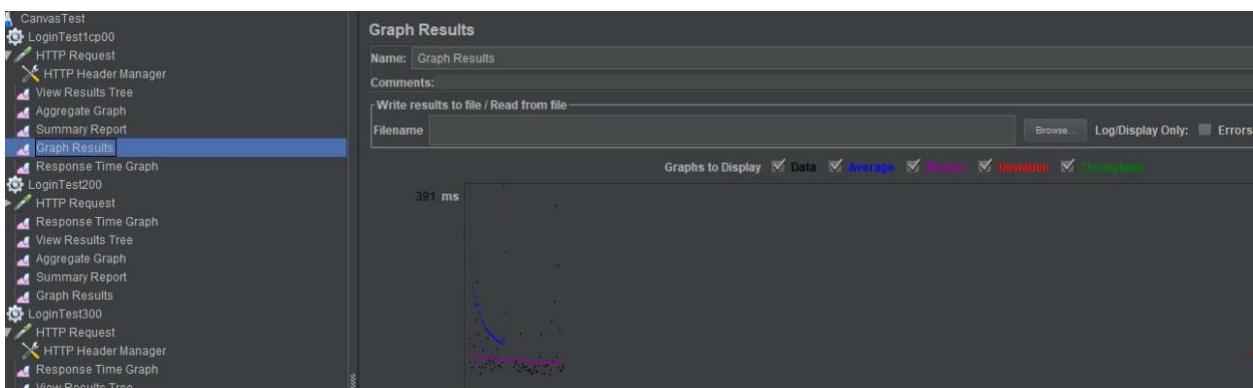
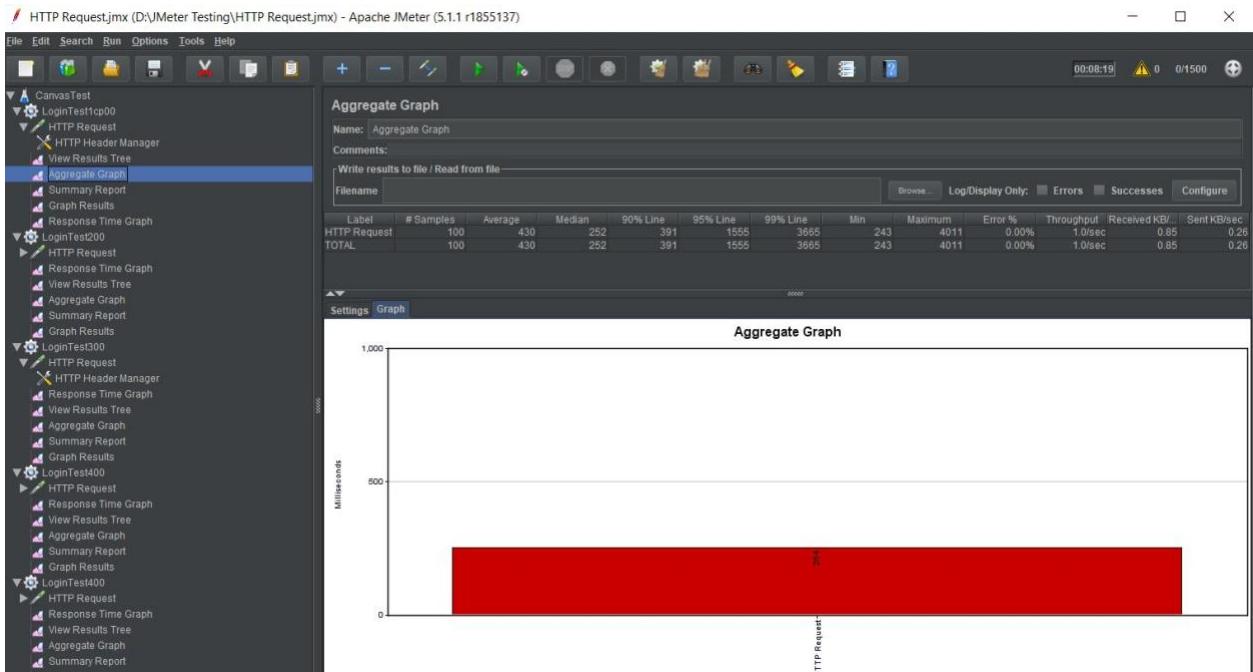
Legend: Placement: Bottom Font: Sans Serif Size: 10 Style: Normal





- JMeter With Pooling

For 100 calls,



HTTP Request

Name: HTTP Request
Comments:
Protocol [http]:
Method: POST Path: http://ec2-34-207-122-1.compute-1.amazonaws.com:3001/login
Content encoding:
Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data Browser-compatible headers

Parameters Body Data Files Upload

```
1 | "username": "s449", "password": "s449", "student": "student"
```

Response Time Graph

Name: Response Time Graph
Comments: Write results to file / Read from file
Filename:
Log/Display Only: Errors Successes Configure

Settings Graph

Response Time Graph

Milliseconds

15/5/10 15/5/10 15/5/10 15/5/10 15/5/10 15/5/10 15/5/10 15/5/10 15/5/10 15/5/10

■ HTTP Request

Summary Report

Name: Summary Report
Comments:
Write results to file / Read from file
Filename:
Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	100	430	243	4011	638.10	0.00%	1.0/sec	0.85	0.26	866.0
TOTAL	100	430	243	4011	638.10	0.00%	1.0/sec	0.85	0.26	866.0

For 200 calls,

The screenshot shows the JMeter results analysis interface with two main tabs displayed: "Aggregate Graph" and "Graph Results".

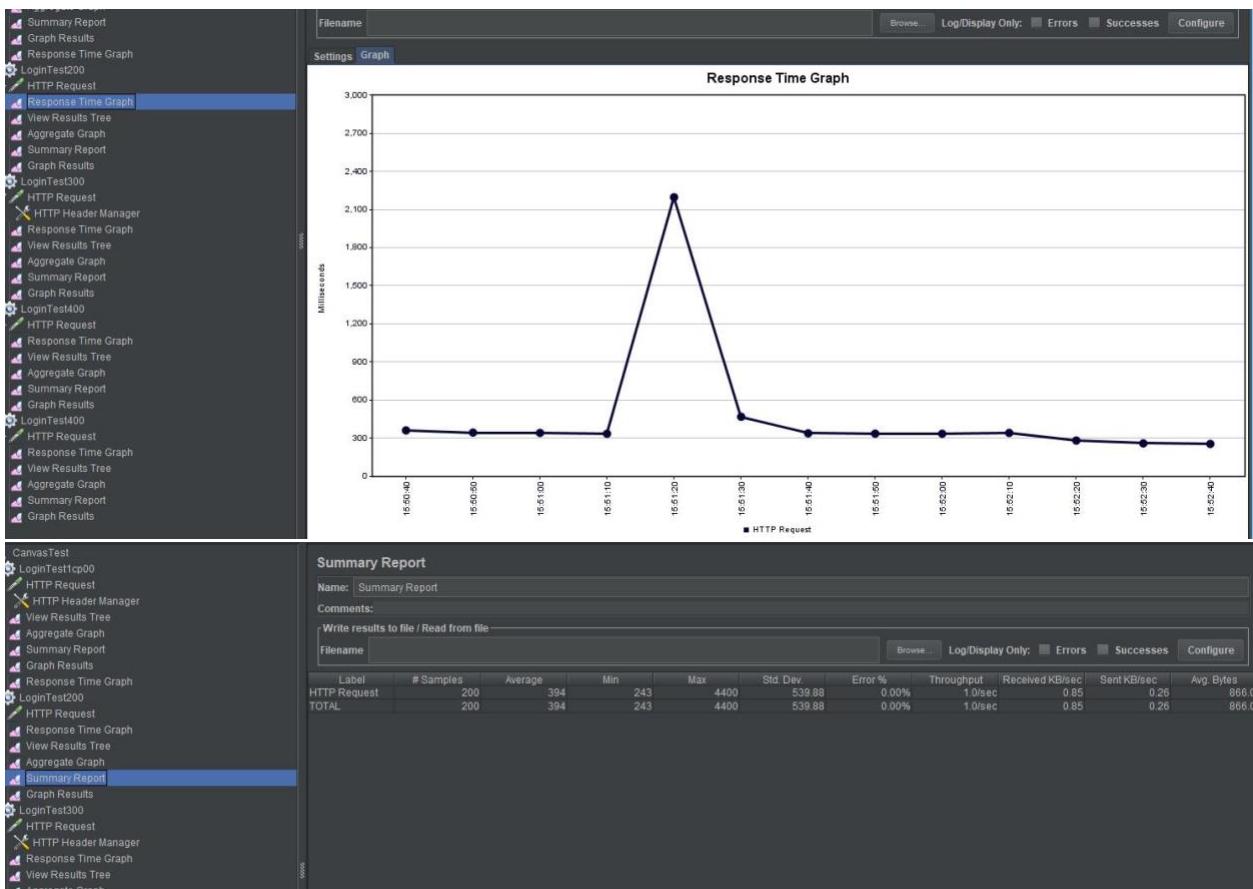
Aggregate Graph Tab:

- Name:** Aggregate Graph
- Comments:** Write results to file / Read from file
- Filename:** [empty]
- Table Data:**

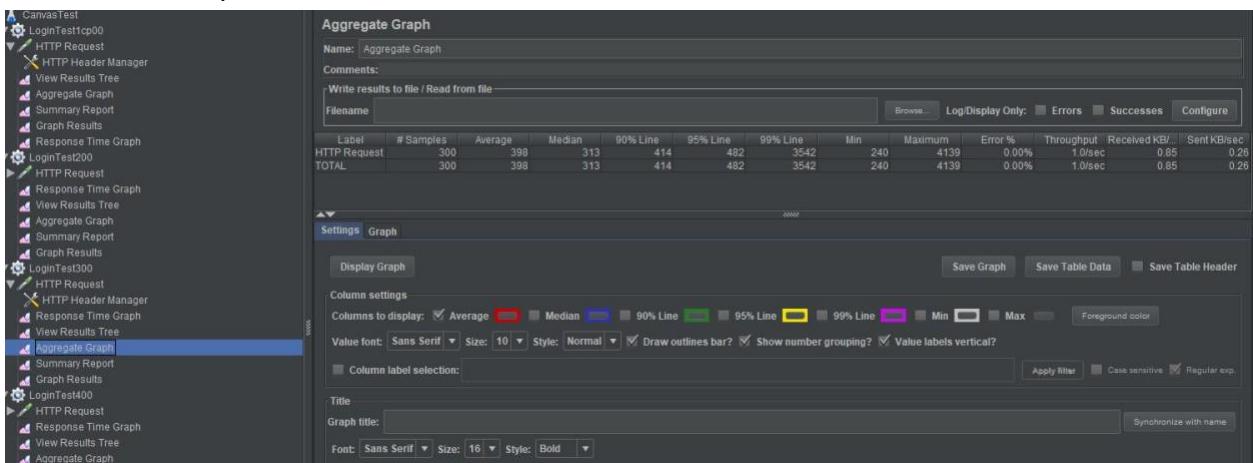
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	200	394	291	357	417	4020	243	4400	0.00%	1.0/sec	0.85	0.26
TOTAL	200	394	291	357	417	4020	243	4400	0.00%	1.0/sec	0.85	0.26

Graph Results Tab:

- Name:** Graph Results
- Comments:** Write results to file / Read from file
- Filename:** [empty]
- Graph Options:** Graphs to Display: Data, Average, Success, Deviation, Throughput
- Graph View:** A line graph showing response times over 200 samples. The Y-axis is labeled "359 ms". The graph shows a general downward trend with some fluctuations.



For 300 calls,



Graph Results

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

Graphs to Display: Data, Average, Min, Max, Deviation, Throughput
Log/Display Only: Errors, Successes
Configure

440 ms
0 ms
No of Samples: 300 Latest Sample: 245 Average: 398

Response Time Graph

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

Settings Graph

Response Time Graph

Milliseconds

HTTP Request

Time	Response Time (ms)
15:50:40	~500
15:50:50	~400
15:51:00	~400
15:51:10	~400
15:51:20	~2400
15:51:30	~600
15:51:40	~400
15:51:50	~400
15:52:00	~400
15:52:10	~400
15:52:20	~400
15:52:30	~400
15:52:40	~400
15:52:50	~400

00:08:19 0 0/1500

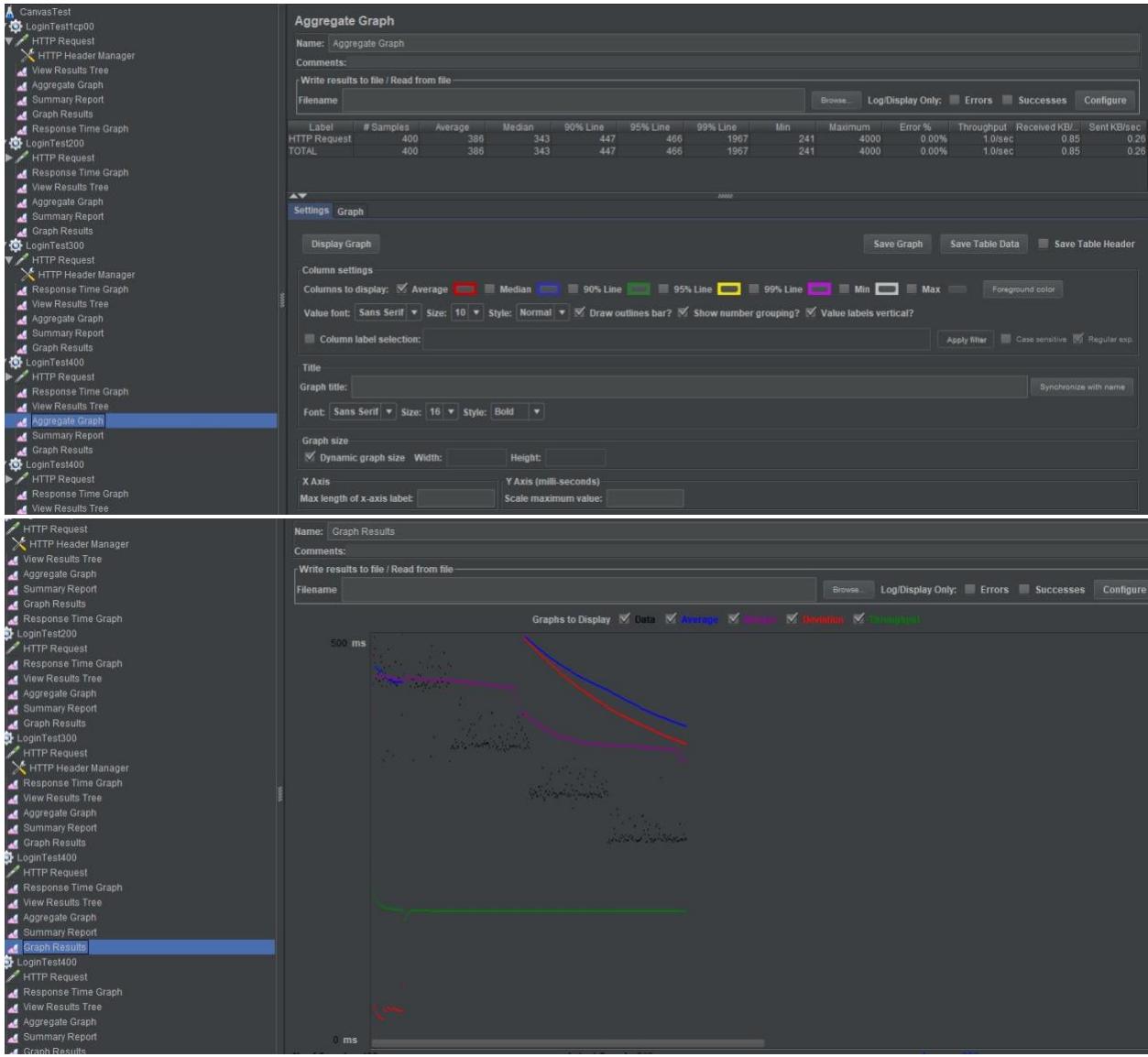
Summary Report

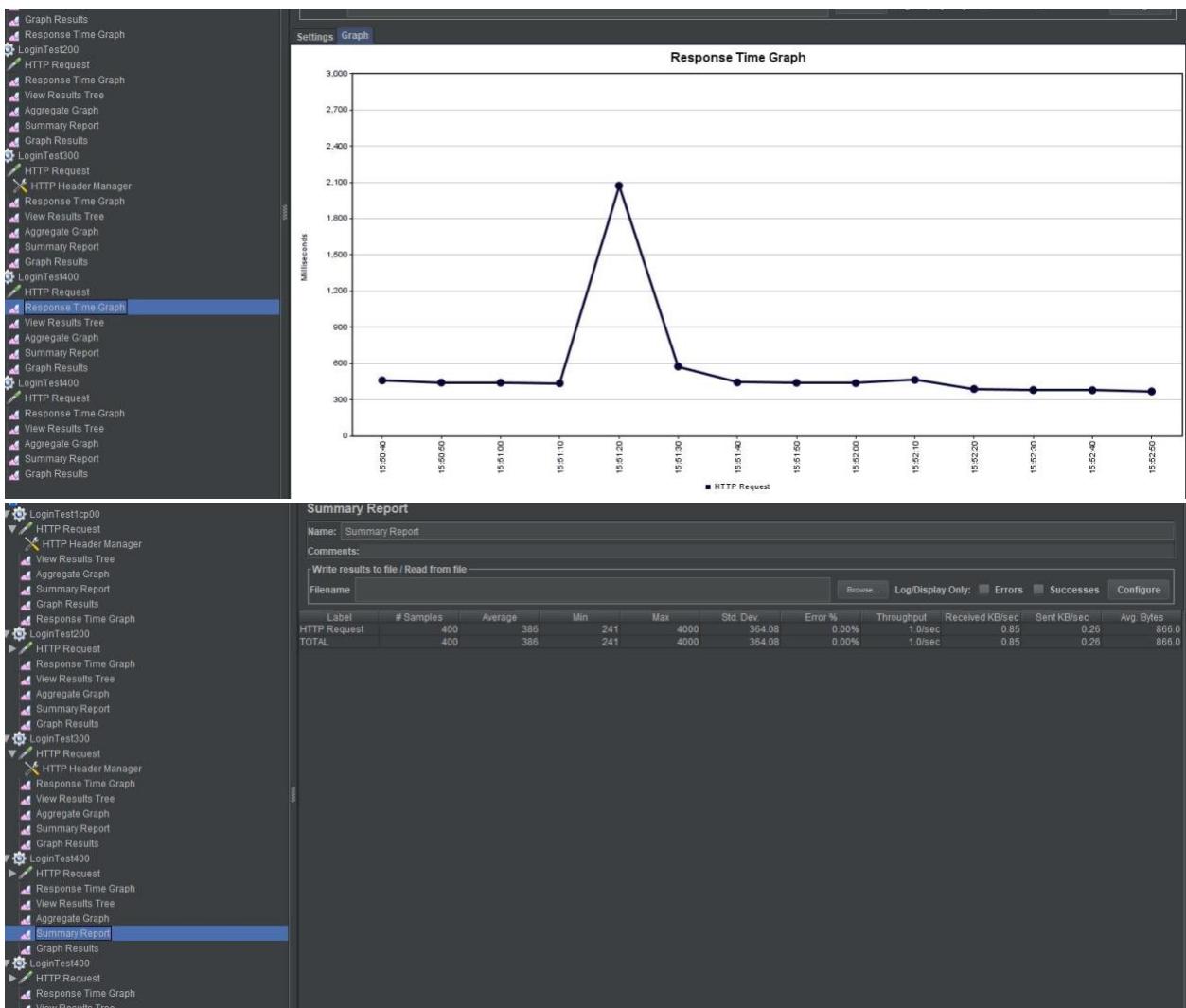
Name: Summary Report
Comments:
Write results to file / Read from file
Filename:

Log/Display Only: Errors, Successes
Configure

Label	# Samples	Average	Min	Max	Std Dev	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	300	398	240	4139	461.32	0.00%	1.0/sec	0.85	0.25	866.0
TOTAL	300	398	240	4139	461.32	0.00%	1.0/sec	0.85	0.26	866.0

For 400 calls,





For 500 calls,

Tree View Results Tree
 □ Aggregate Graph
 □ Summary Report
 □ Graph Results
 □ Response Time Graph
 □ LoginTest200
 □ HTTP Request
 □ Response Time Graph
 □ View Results Tree
 □ Aggregate Graph
 □ Summary Report
 □ Graph Results
 □ LoginTest300
 □ HTTP Request
 □ HTTP Header Manager
 □ Response Time Graph
 □ View Results Tree
 □ Aggregate Graph
 □ Summary Report
 □ Graph Results
 □ LoginTest400
 □ HTTP Request
 □ Response Time Graph
 □ View Results Tree
 □ Aggregate Graph
 □ Summary Report
 □ Graph Results
 □ LoginTest400
 □ HTTP Request
 □ Response Time Graph
 □ View Results Tree
 □ Aggregate Graph
 □ Summary Report
 □ Graph Results
 □ LoginTest400
 □ HTTP Request
 □ Response Time Graph
 □ View Results Tree
 □ Aggregate Graph
 □ Summary Report
 □ Graph Results
 □ LoginTest400
 □ HTTP Request
 □ Response Time Graph
 □ View Results Tree
 □ Aggregate Graph
 □ Summary Report
 □ Graph Results
 □ LoginTest400
 □ HTTP Request
 □ Response Time Graph
 □ View Results Tree
 □ Aggregate Graph
 □ Summary Report
 □ Graph Results

Write results to file / Read from file

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	500	394	340	483	498	1685	239	5088	0.00%	1.0/sec	0.85	0.26
TOTAL	500	394	340	483	498	1685	239	5088	0.00%	1.0/sec	0.85	0.26

Settings | Graph

Display Graph

Column settings

Columns to display: Average Median 90% Line 95% Line 99% Line Min Max Foreground color

Value font: Sans Serif Size: 10 Style: Normal Draw outlines bar? Show number grouping? Value labels vertical?

Column label selection: Apply filter Case sensitive Regular exp.

Title: Graph title: Synchronize with name

Font: Sans Serif Size: 16 Style: Bold

Graph size: Dynamic graph size Width: Height:

X Axis: Max length of x-axis label: Y Axis (milli-seconds) Scale maximum value:

Legend: Placement: Bottom Font: Sans Serif Size: 10 Style: Normal

Filename: Browse... Log/Display Only: Errors Successes Configure

Graphs to Display: Data Average Min Duration Throughput

No of Samples: 500 Latest Sample: 245 Average: 394

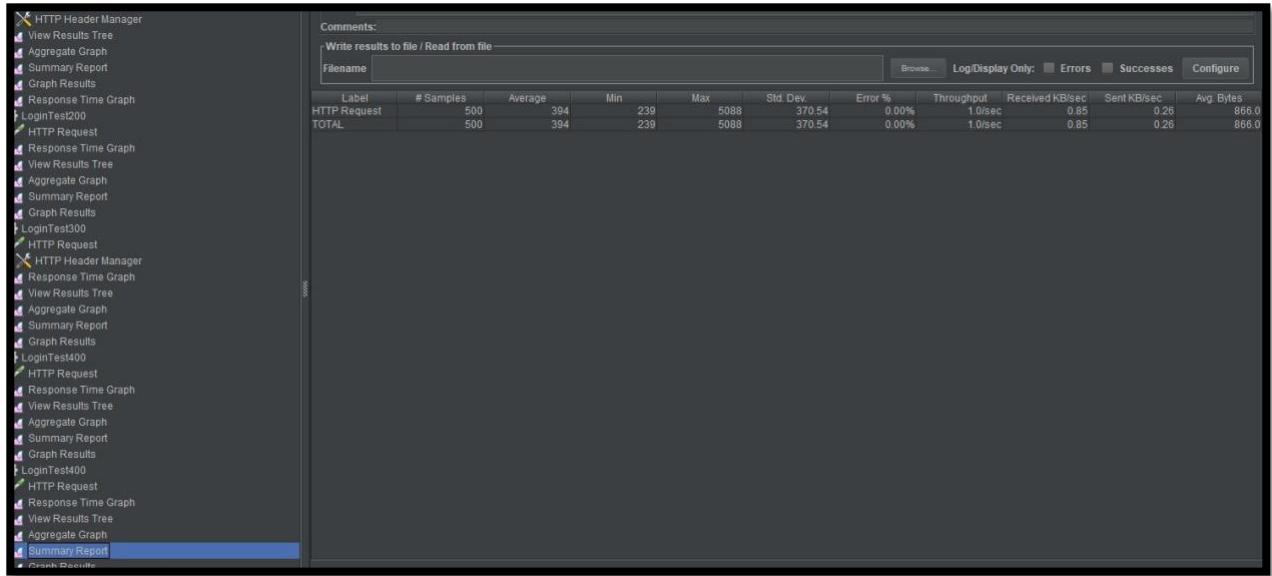
Settings | Graph

Response Time Graph

Milliseconds

■ HTTP Request

15:50:40 15:50:50 15:51:00 15:51:10 15:51:20 15:51:30 15:51:40 15:51:50 15:52:00 15:52:10 15:52:20 15:52:30 15:52:40 15:52:50



- Mocha Testing

```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
  - checking login credentials
  - validating login
  - pending...
  ✓ Login Successful
  - checking signup
  - validating signup
  - pending...
  ✓ Signup Successful
  - checking course page
  - validating course page
  - pending...
  ✓ Course page Successful
  - checking profile update
  - validating profile update
  - pending...
  ✓ Profile update Successful
  - checking search for courses
  - validating search
  - pending...
  ✓ Searching Successful
```

- Enzyme

The screenshot shows a code editor interface with several tabs at the top: 'Assignment.js', 'JS assignments.js', 'JS grades.js', 'JS assignment.test.js', and 'JS announcement.test.js'. The 'JS assignment.test.js' tab is active, displaying a Jest test script. Below the tabs is a terminal window showing the command 'yarn run v1.13.0 \$ jest' and the resulting test output:

```

yarn run v1.13.0
$ jest
PASS  src/assignment.test.js (10.391s)
  ● Console

    console.log src/components/Assignments/Assignments.js:9
      Calling constructor
    console.log src/components/Assignments/Assignments.js:28
      Inside mountnull
    console.log src/components/Assignments/Assignments.js:30
      http://localhost:4000/getEachAssignment/null/null

PASS  src/enrollment.test.js (10.879s)
PASS  src/announcement.test.js (12.464s)

Test Suites: 3 passed, 3 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        17.985s
Ran all test suites.
Done in 22.65s.

```

- Question&Answers:

1. Kafka improves the response time drastically. If there are many users making a request onto a server like 1000 requests, the server cannot handle all the requests at a time. But, Kafka, A messaging queue with its low latency, high throughput does handle the incoming huge requests. MongoDB in cloud has automated load balancing by using sharding, but locally, it is quite a tough task and has a lot of complexity.
2. MySQL is the database model which consists of rows and columns and MongoDB consists of documents. Query of data is comparatively easy in MongoDB than in MySQL as it may have triggers and joins. But in MongoDB, it is very fast. When data is extensively large, mongoDB is used as it is horizontally scalable and can handle unstructured data and also can be accessed frequently. Whereas in MySQL is scalable vertically and can be used on small datasets.

