LAB – 1 Report

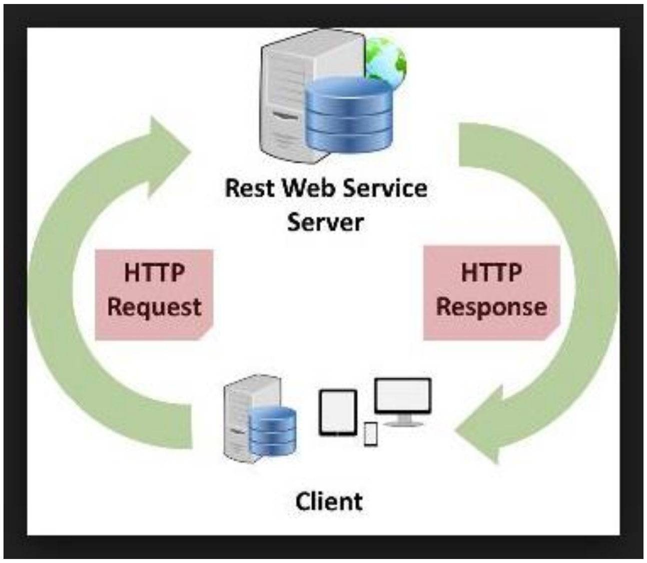
Part – 1: Calculator

Introduction:

1. Goals:
   Design and implement a simple Calculator using Restful Services. The goal is to using ReactJS for frontend and NodeJS for backend server. Calculator application should perform basic operations i.e., Addition, Subtraction, Multiplication, Division.
2. Purpose of the system:
   The purpose of the Calculator is to perform basic calculations i.e., Addition, Subtraction, Multiplication and Division. The app takes input values, sends it to the backend service and renders the answer as output. The Server will process the inputs and send output back to the client.

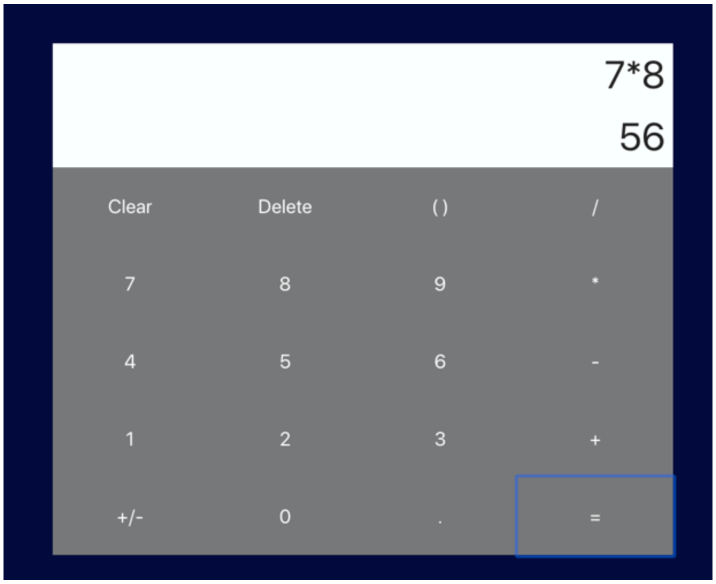
System Design:

RESTful webservice will provide interoperability between computer systems on the Internet. The Frontend interface is accessible from different client while sending HTTP request, process them by the backend server and send the corresponding response through HTTP response.

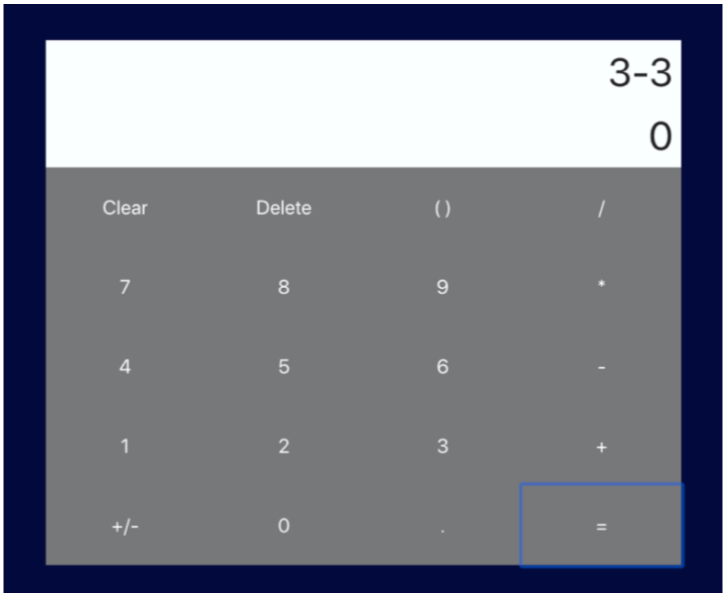


Results:

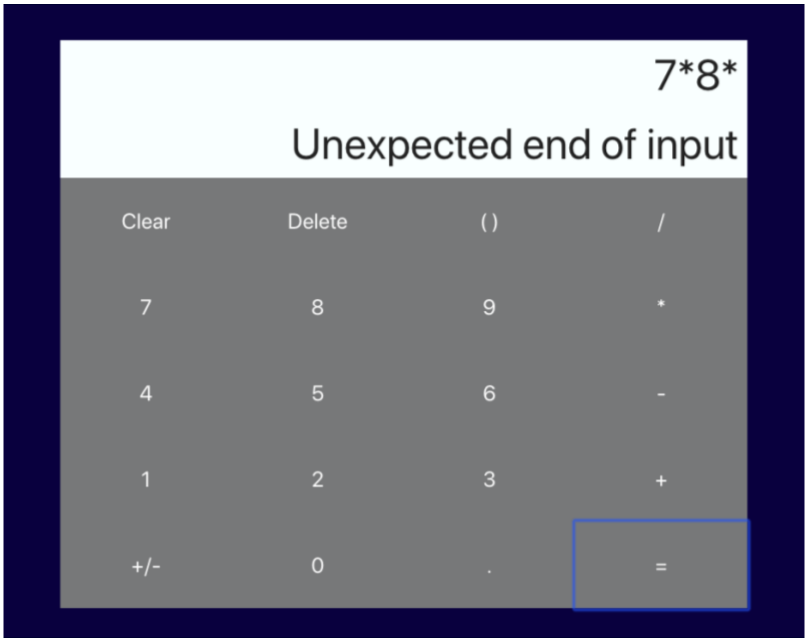
Client-side:



```
7*8
56
```



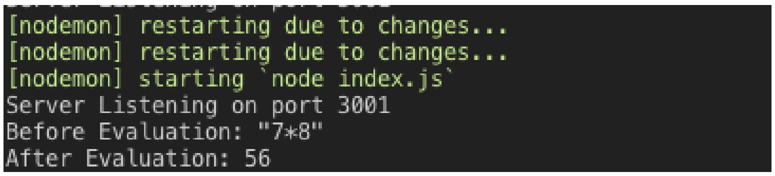
```
3-3
0
```



7*8*

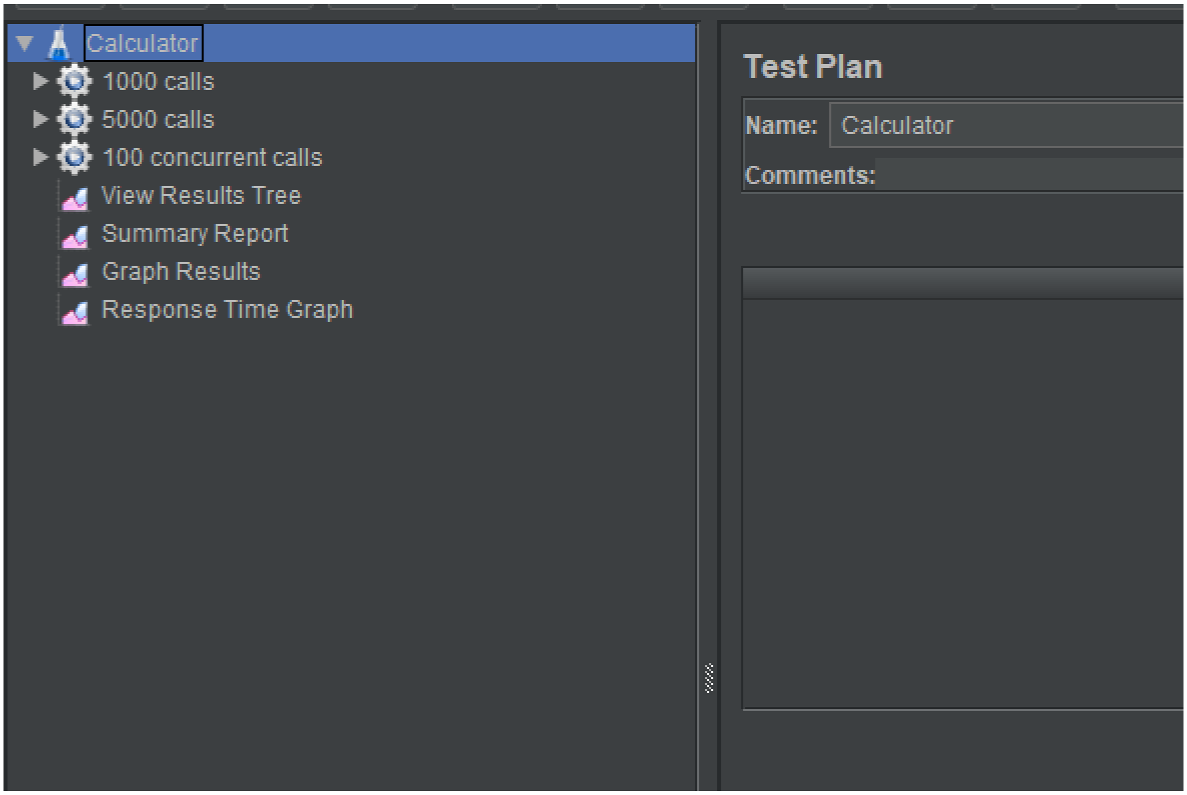
Unexpected end of input

After giving a input,



```
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server Listening on port 3001
Before Evaluation: "7*8"
After Evaluation: 56
```

Testing:



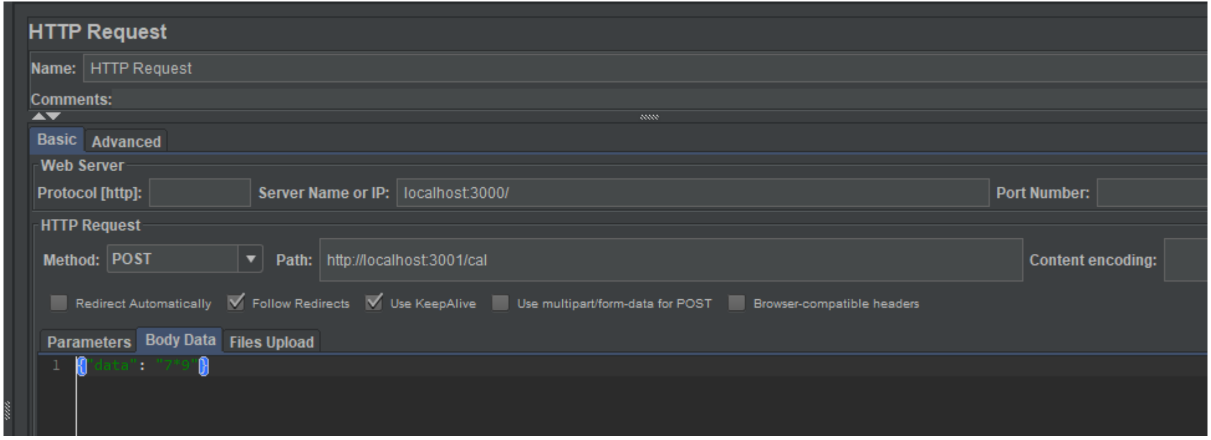
- Calculator
  - 1000 calls
  - 5000 calls
  - 100 concurrent calls
  - View Results Tree
  - Summary Report
  - Graph Results
  - Response Time Graph

Test Plan

Name: Calculator

Comments:

For 1,000 calculator calls,



HTTP Request

Name: HTTP Request

Comments:

Web Server

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

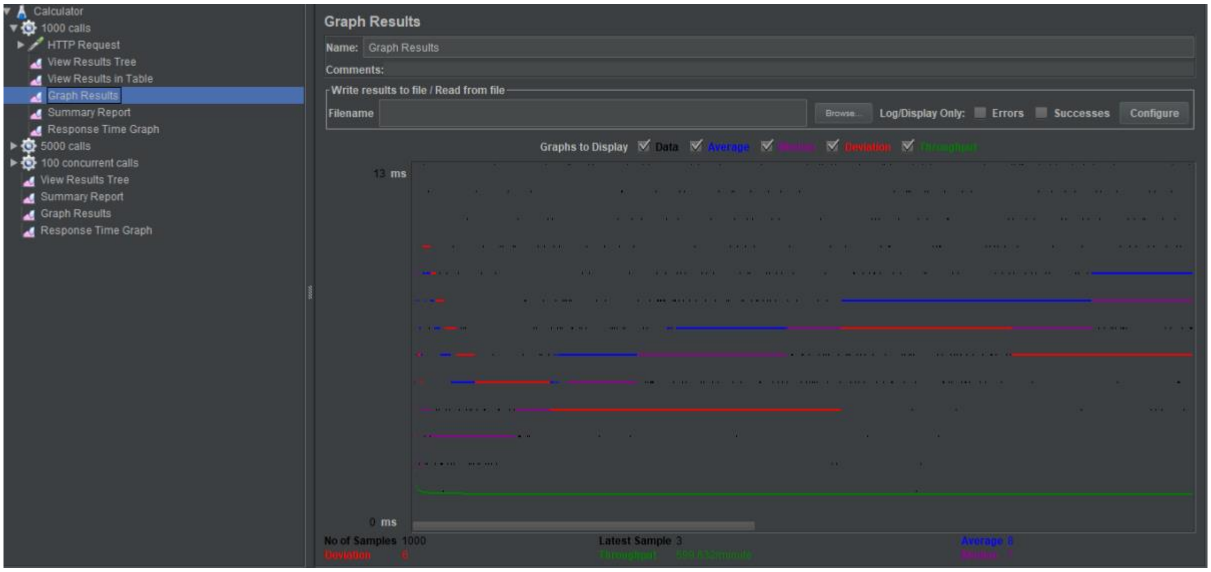
HTTP Request

Method: POST Path: http://localhost:3001/cal Content encoding:

□ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive □ Use multipart/form-data for POST □ Browser-compatible headers

```
{"data": "7*9"}
```

Graph Results



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

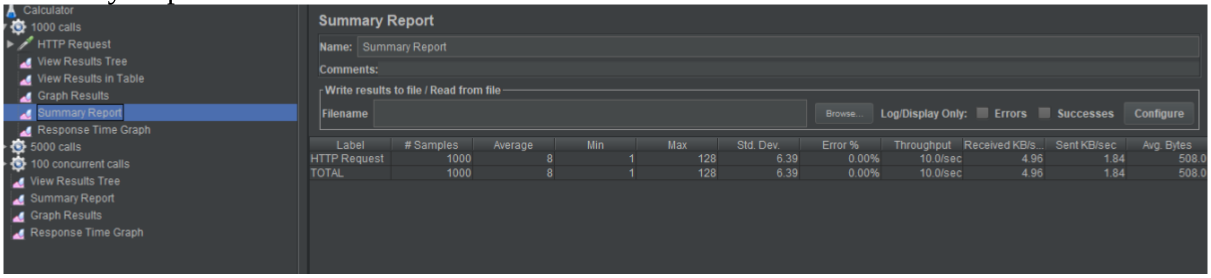
Filename

Log/Display Only: □ Errors □ Successes

Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 1000 Latest Sample 3 Average 8
Deviation 6 Throughput 599.632/minute Median 7

Summary Report



Summary Report

Name: Summary Report

Comments:

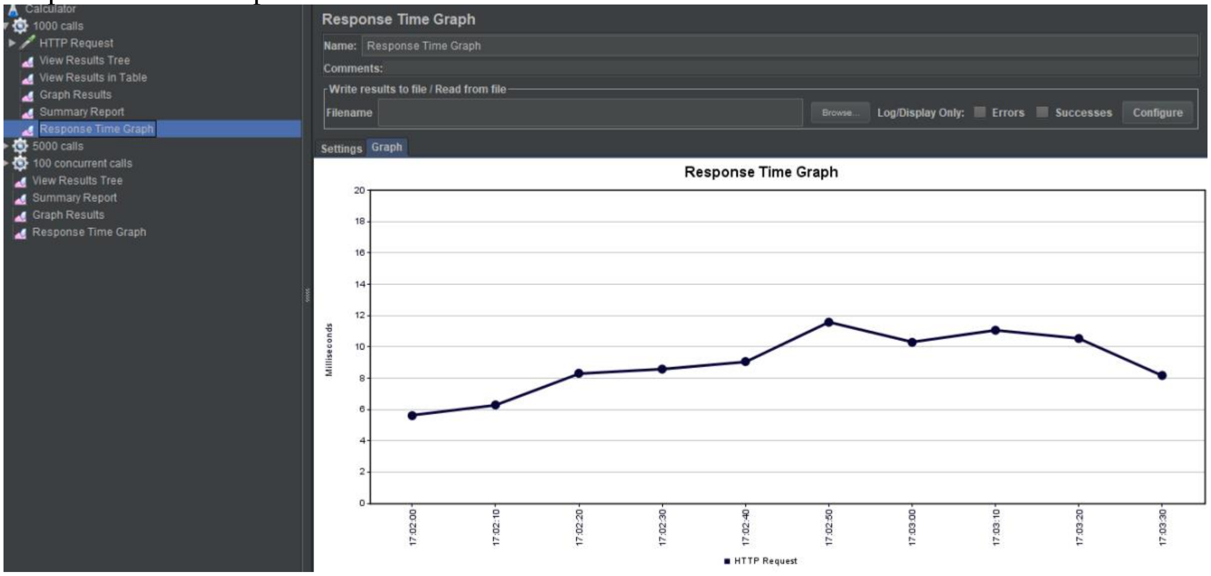
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	1000	8	1	128	6.39	0.00%	10.0/sec	4.96	1.84	508.0
TOTAL	1000	8	1	128	6.39	0.00%	10.0/sec	4.96	1.84	508.0

Response Time Graph



Response Time Graph

Name: Response Time Graph

Comments:

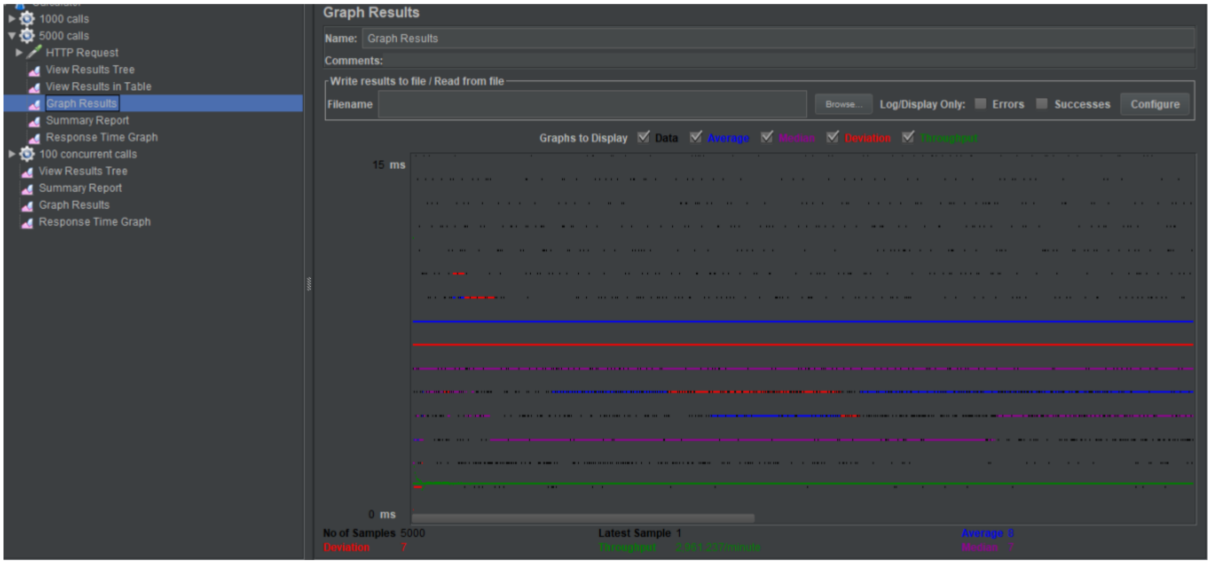
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

For 5,000 calculator calls,

Graph Results



Graph Results

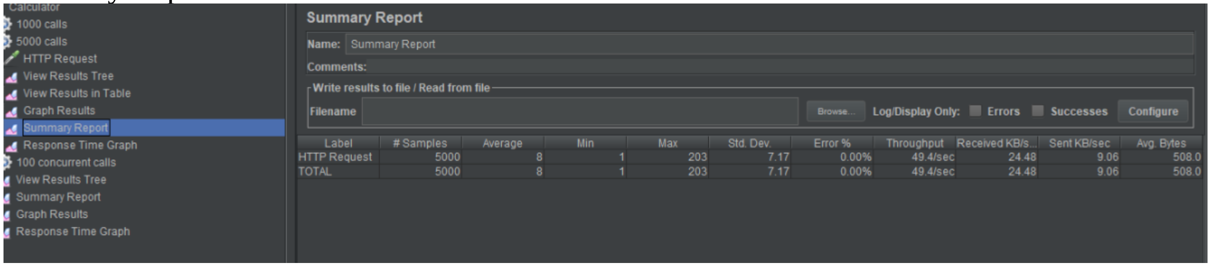
Name: Graph Results

Comments:

No of Samples 5000 Latest Sample 1 Average 8

Deviation 7 Throughput 2,961.237/minute Median 7

Summary Report



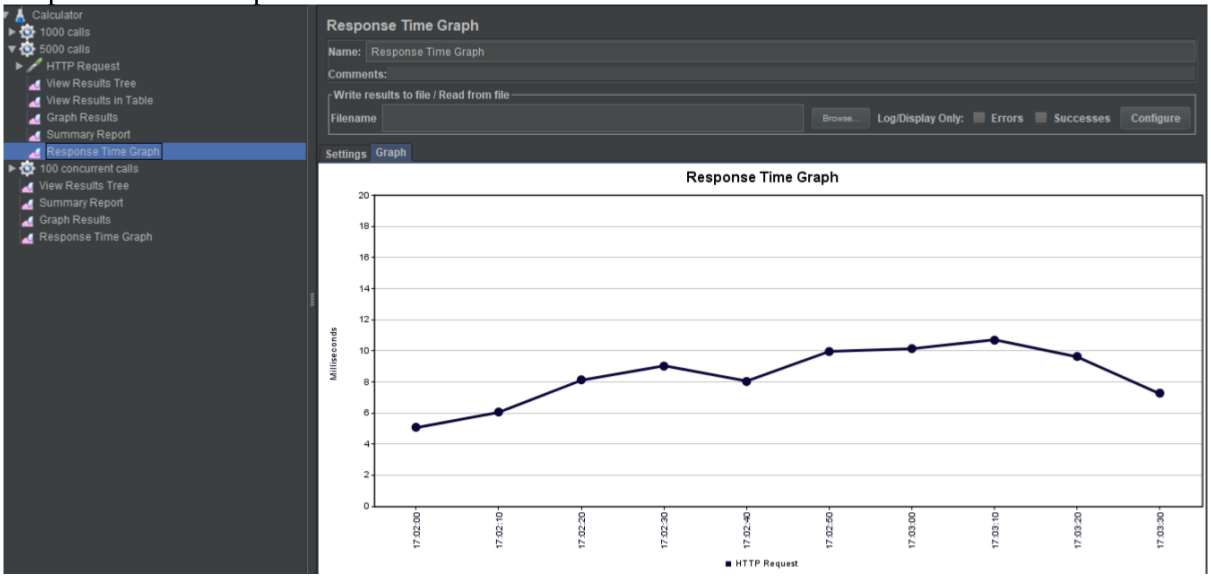
Summary Report

Name: Summary Report

Comments:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	5000	8	1	203	7.17	0.00%	49.4/sec	24.48	9.06	508.0
TOTAL	5000	8	1	203	7.17	0.00%	49.4/sec	24.48	9.06	508.0

Response Time Graph



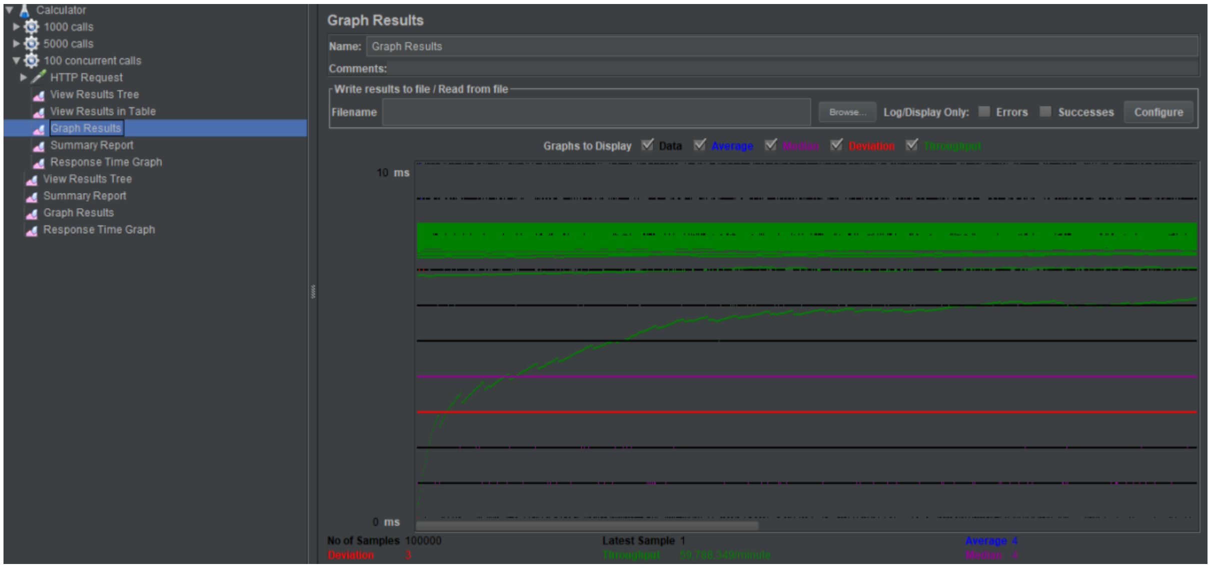
Response Time Graph

Name: Response Time Graph

Comments:

For 100 concurrent users with 1000 calls each,

Graph Results



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

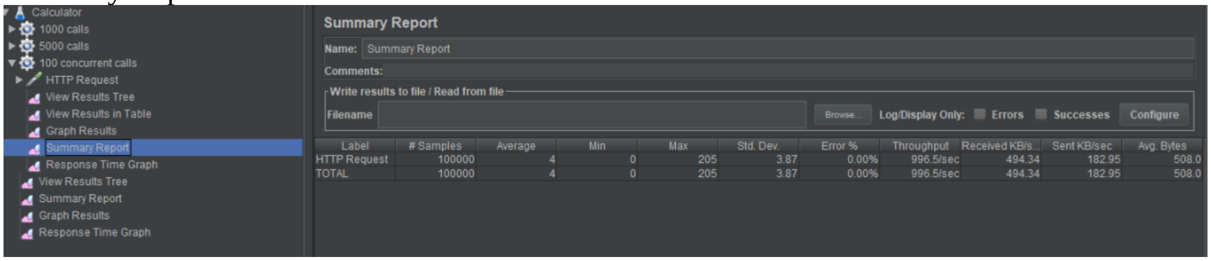
Log/Display Only: □ Errors □ Successes

Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 100000 Latest Sample 1 Average 4

Deviation 3 Throughput 59,788.349/minute Median 4

Summary Report



Summary Report

Name: Summary Report

Comments:

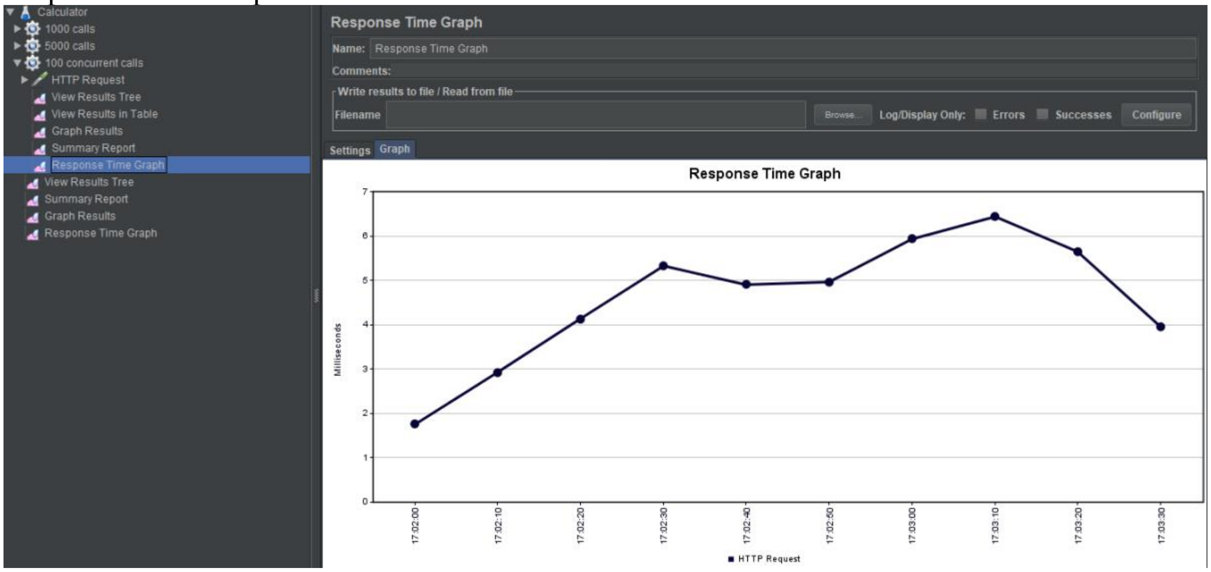
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	100000	4	0	205	3.87	0.00%	996.5/sec	494.34	182.95	508.0
TOTAL	100000	4	0	205	3.87	0.00%	996.5/sec	494.34	182.95	508.0

Response Time Graph



Response Time Graph

Name: Response Time Graph

Comments:

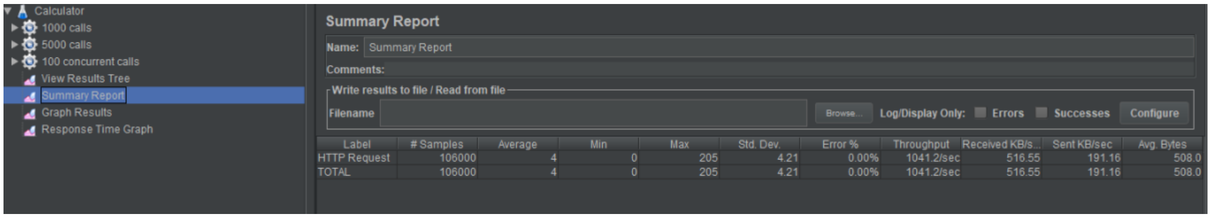
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Summarized Reports,

Graph Results



Summary Report

Name: Summary Report

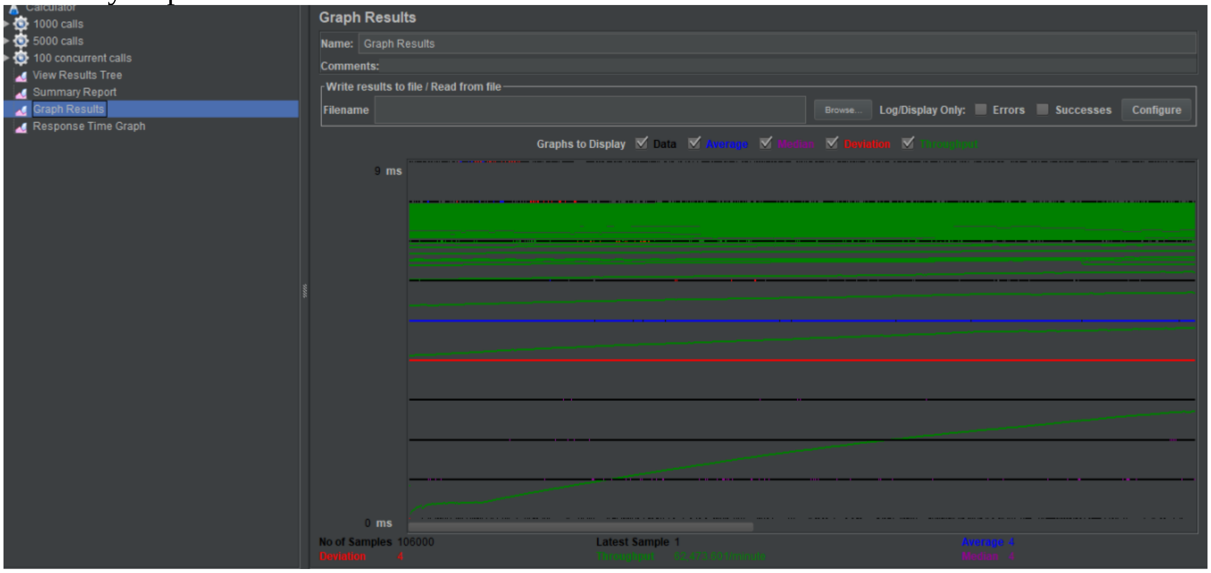
Comments:

Write results to file / Read from file

Filename

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	106000	4	0	205	4.21	0.00%	1041.2/sec	516.55	191.16	508.0
TOTAL	106000	4	0	205	4.21	0.00%	1041.2/sec	516.55	191.16	508.0

Summary Report



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Browse...
Log/Display Only: □ Errors □ Successes
Configure
Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput
9 ms
0 ms

No of Samples 106000

Latest Sample 1

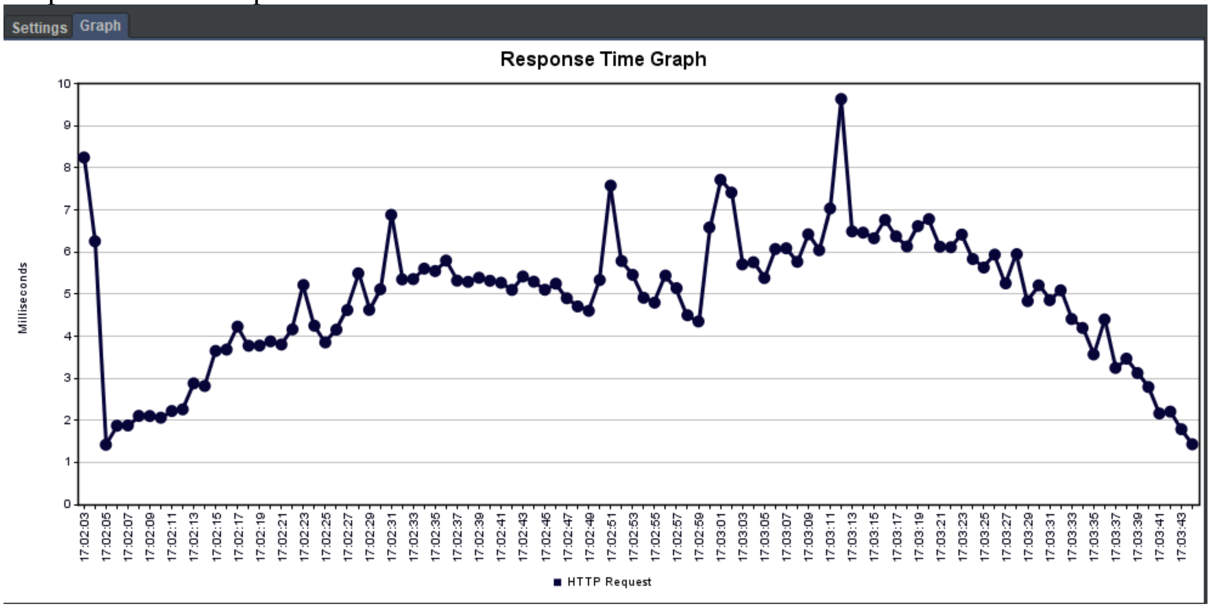
Average 4

Deviation 4

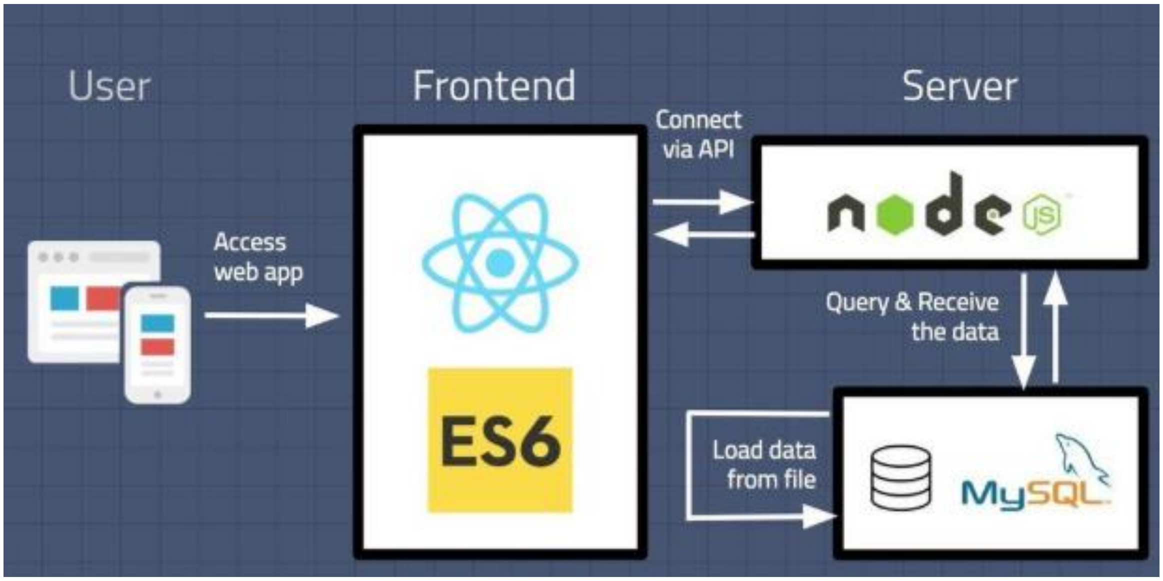
Throughput 62,473.601/minute

Median 4

Response Time Graph



Part – 2: Prototype of Canvas Application



Workflow: The canvas application is aimed for both the students and the professors, performing different academic related activities.

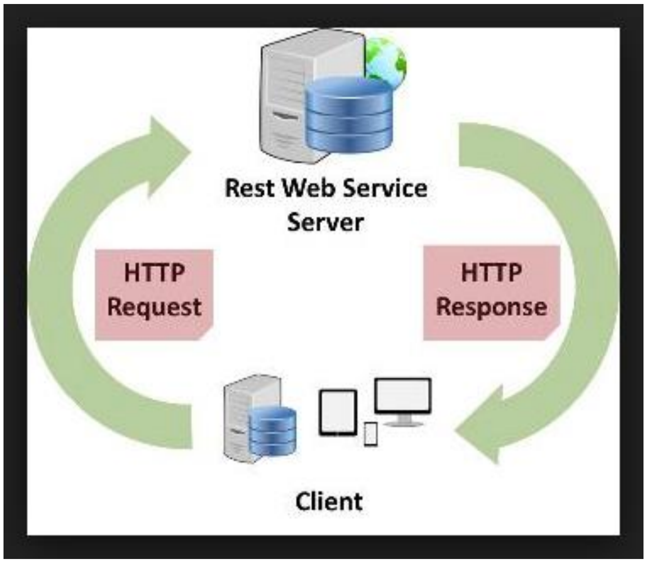
From student's side, a student can register for courses, view the courses, request for permission number, search for the courses. While from the professor's view, he can create a create for students, give permission numbers, check the waitlist, conduct quizzes, submit documents, publish assignment, make any announcements to the class or group.

Overall modules:

1. Frontend UI based on user role(Student/professor) with respective permissions, User will login in to the application, if not registered, should register and thus if has no courses, can search for courses and add or request for waiting list if full, able to view them on the dashboard. Find the respective course announcements, assignments and answer quizzes navigating form courses dashboard to a course page, with sidebar navigation tray. Here all the UI is build using **ReactJS**, provided by npm manager, which creates virtual DOM and updates the parts that are required by the user.
2. When a user makes a request, the request is sent to the backend service, implemented using **NodeJS**, provided by npm manager. Based on the user request, the backend routes the request to the respective service and thereby sends appropriate response using **Postman** for designing the backend services.
3. For all the services, we have used MySQL database, with appropriate definition of foreign keys and primary key constrains, perform join operations based on the data required.

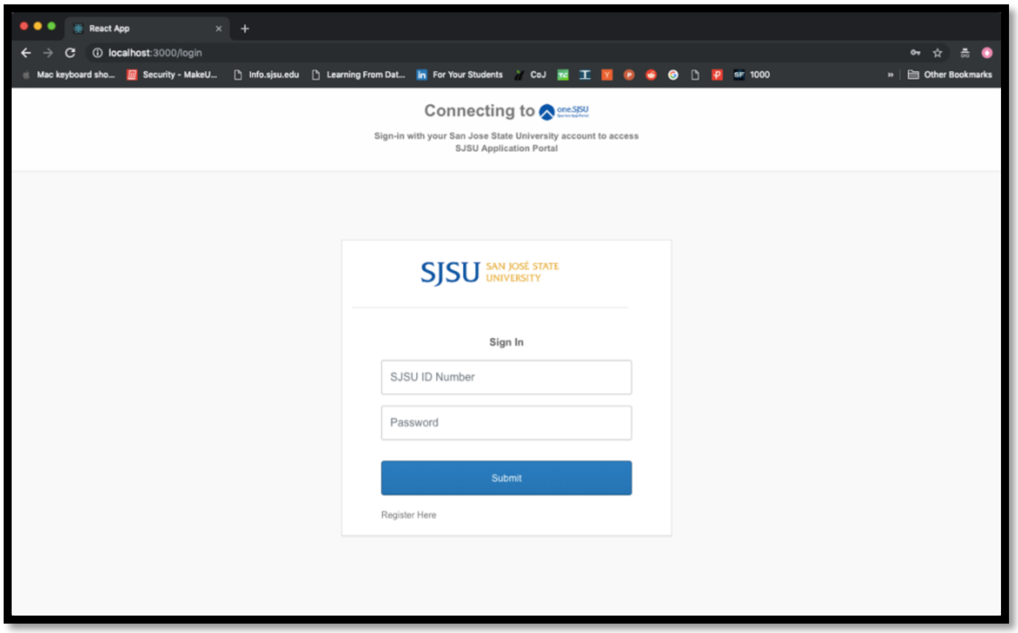
System Design:

RESTful webservice will provide interoperability between computer systems on the Internet. The Frontend interface is accessible from different client while sending HTTP request, process them by the backend server and send the corresponding response through HTTP response.



Basic User Functionalities:

Sign in page



Connecting to oneSJSU

Sign-in with your San Jose State University account to access SJSU Application Portal

SJSU SAN JOSÉ STATE UNIVERSITY

Sign In

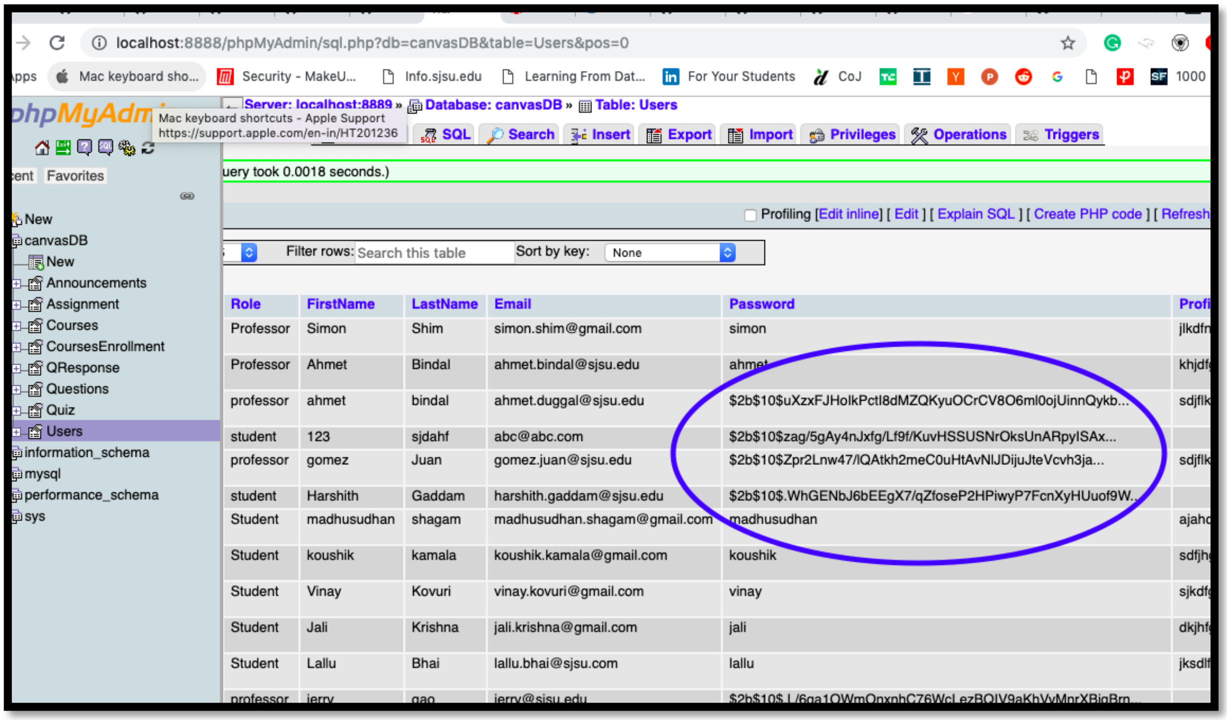
SJSU ID Number

Password

Submit

Register Here

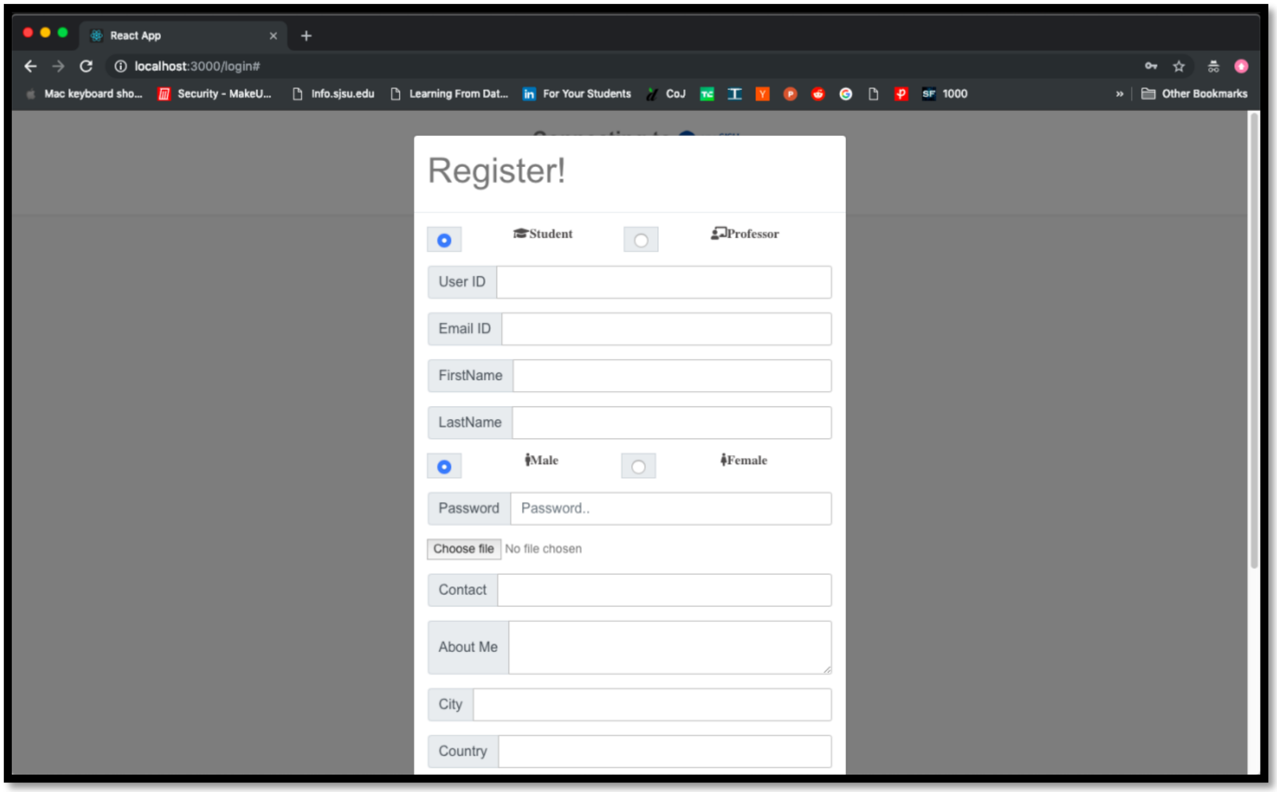
Password encryption using bcrypt algorithm:



Query took 0.0018 seconds.)

Role	FirstName	LastName	Email	Password	Profi
Professor	Simon	Shim	simon.shim@gmail.com	simon	jlkdfn
Professor	Ahmet	Bindal	ahmet.bindal@sjsu.edu	ahmet	khjdf
professor	ahmet	bindal	ahmet.duggal@sjsu.edu	$2b$10$uXzxFJHolkPctl8dMZQKyuOCrCV8O6ml0ojUinnQykb...	sdjflk
student	123	sjdahf	abc@abc.com	$2b$10$zag/5gAy4nJxfg/Lf9f/KuvHSSUSNrOksUnARpyISAx...	
professor	gomez	Juan	gomez.juan@sjsu.edu	$2b$10$Zpr2Lnw47/IQAtkh2meC0uHtAvNIJDijuJteVcvh3ja...	sdjflk
student	Harshith	Gaddam	harshith.gaddam@sjsu.edu	$2b$10$.WhGENbJ6bEEgX7/qZfoseP2HPiwyP7FcnXyHUuof9W...	
Student	madhusudhan	shagam	madhusudhan.shagam@gmail.com	madhusudhan	ajahd
Student	koushik	kamala	koushik.kamala@gmail.com	koushik	sdfjh
Student	Vinay	Kovuri	vinay.kovuri@gmail.com	vinay	sjkdf
Student	Jali	Krishna	jali.krishna@gmail.com	jali	dkjhf
Student	Lallu	Bhai	lallu.bhai@sjsu.com	lallu	jksdlf
professor	jerry	gao	jerry@sjsu.edu	$2b$10$.L/6ga1QWmQnxnhC76WcLezBQIV9aKhVyMnrXBigBrn...	

Register Page



Register!

● Student ○ Professor

User ID

Email ID

FirstName

LastName

● Male ○ Female

Password Password..

Choose file No file chosen

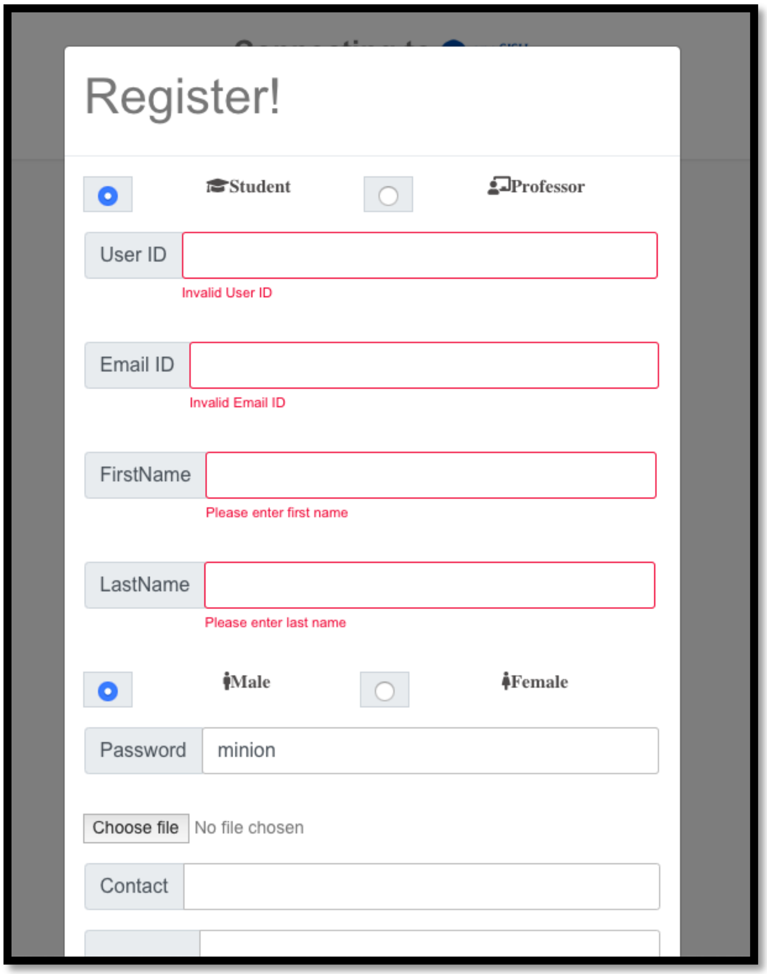
Contact

About Me

City

Country

Register Page input validation:



Register!

● Student ○ Professor

User ID

Invalid User ID

Email ID

Invalid Email ID

FirstName

Please enter first name

LastName

Please enter last name

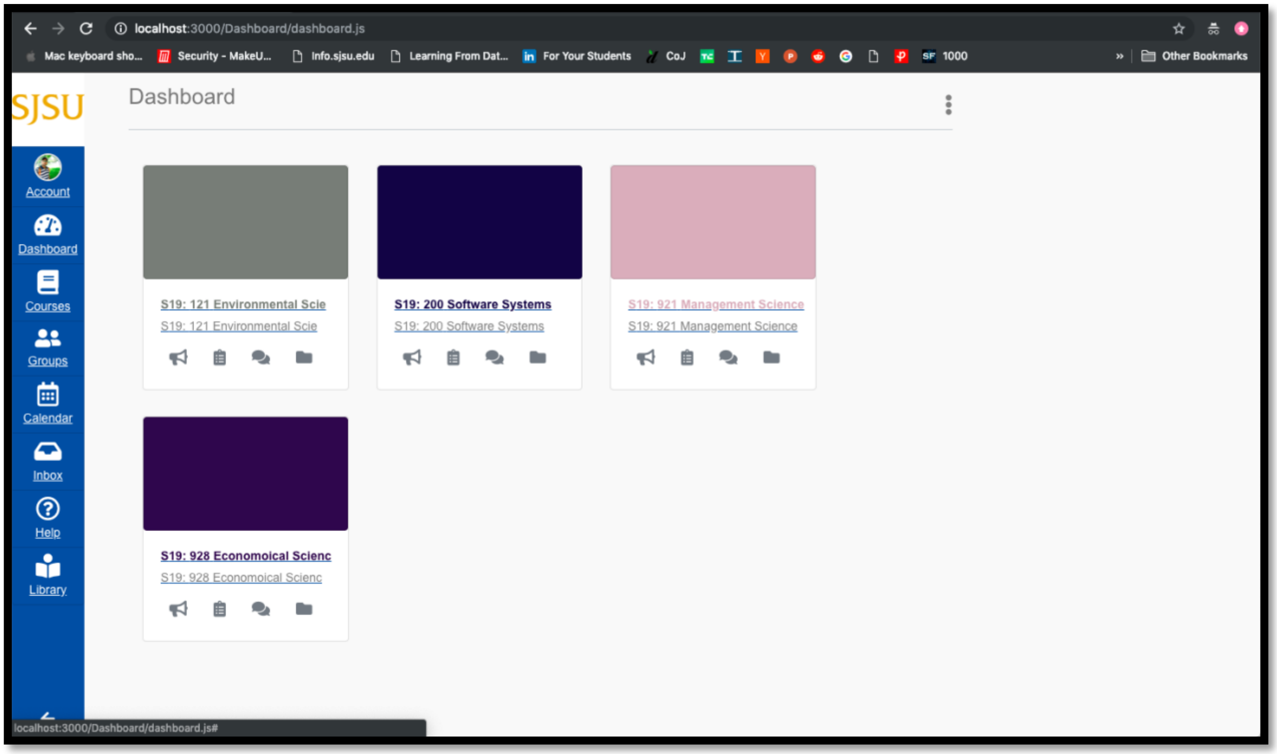
● Male ○ Female

Password: minion

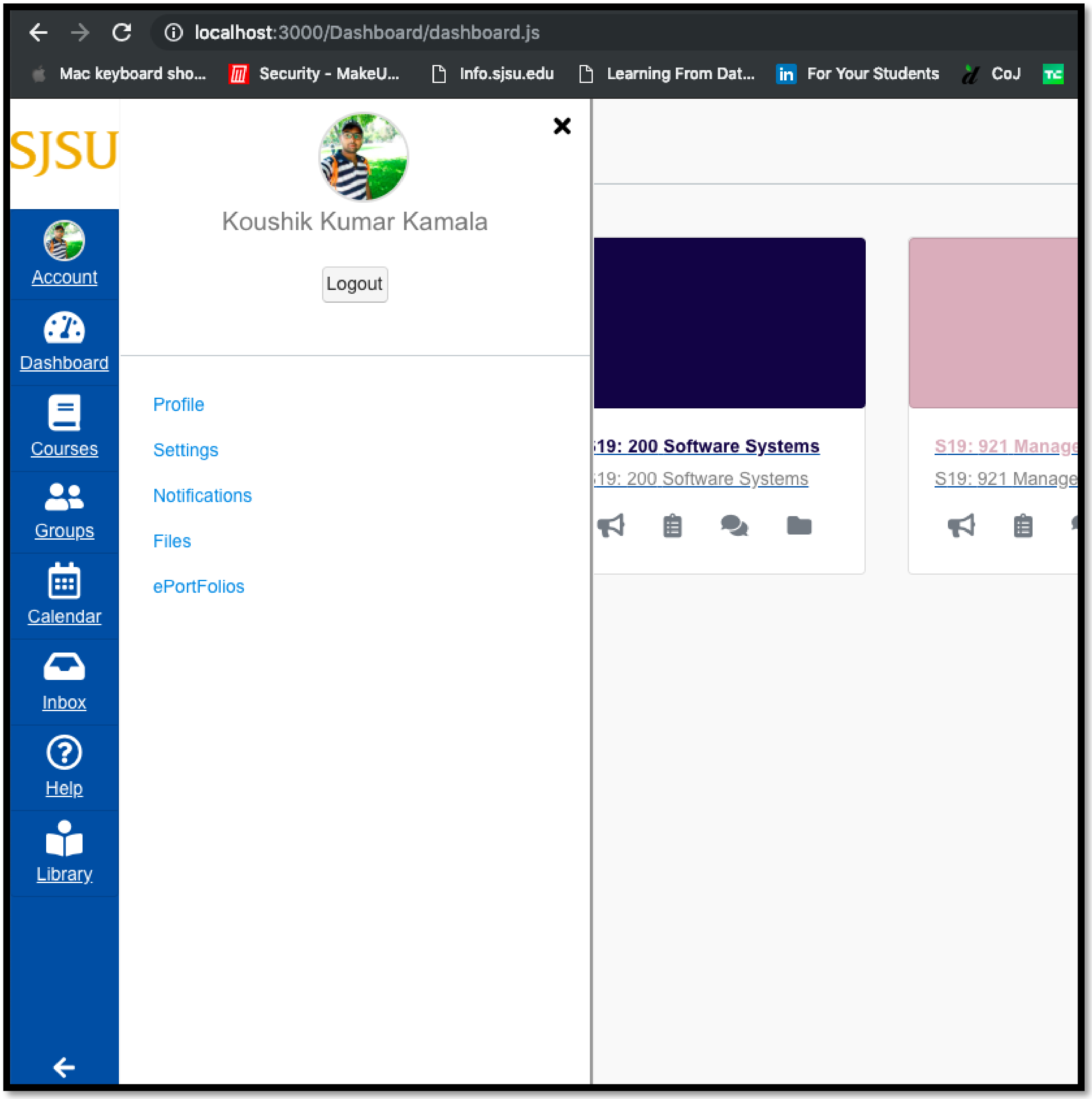
Choose file No file chosen

Contact

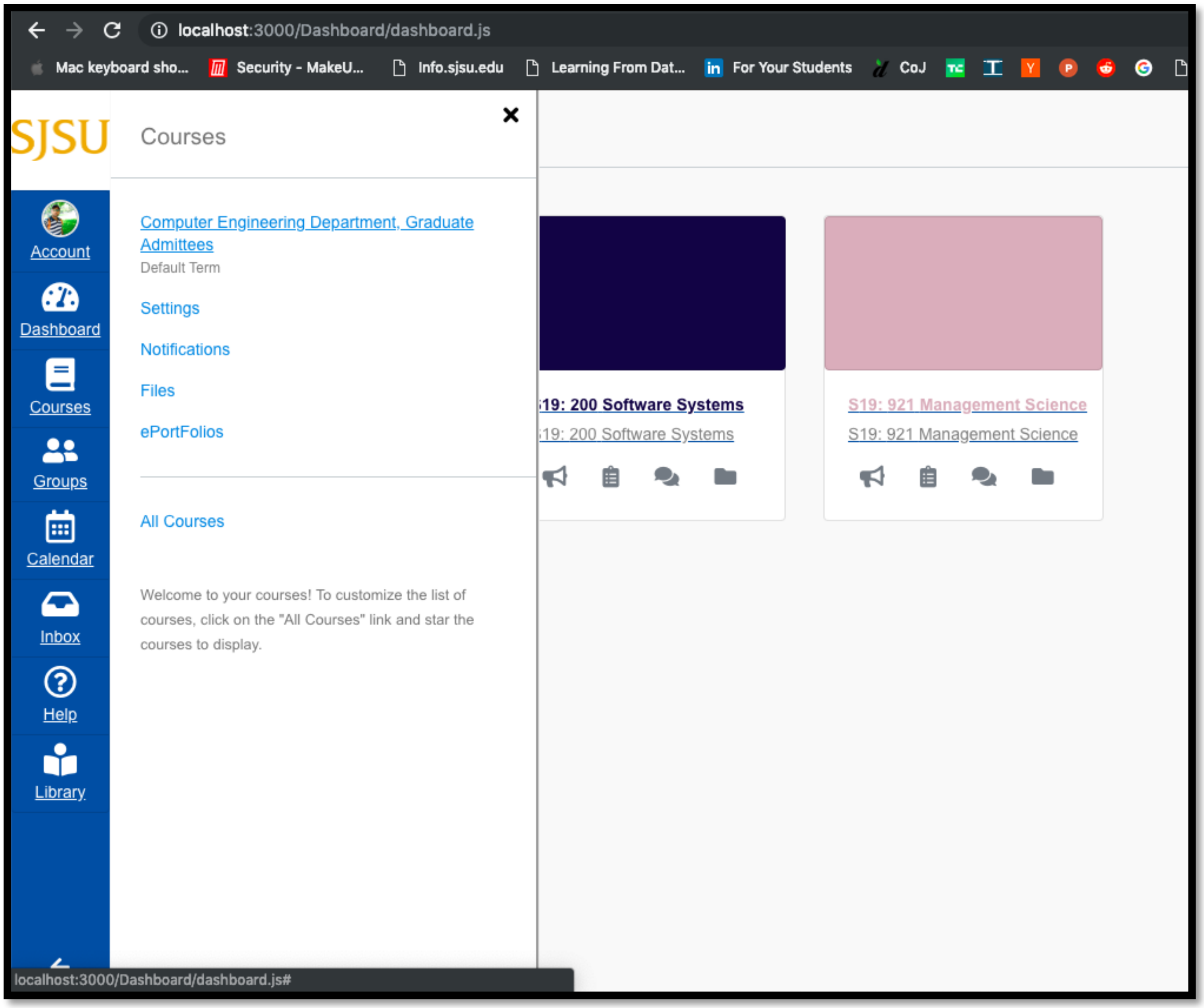
Register Page


localhost:3000/Dashboard/dashboard.js
Mac keyboard sho...
Security - MakeU...
Info.sjsu.edu
Learning From Dat...
For Your Students
CoJ
1000
Other Bookmarks
SJSU
Dashboard
Account
Dashboard
Courses
Groups
Calendar
Inbox
Help
Library
S19: 121 Environmental Scie
S19: 121 Environmental Scie
S19: 200 Software Systems
S19: 200 Software Systems
S19: 921 Management Science
S19: 921 Management Science
S19: 928 Economoical Scienc
S19: 928 Economoical Scienc
localhost:3000/Dashboard/dashboard.js#

Profile Tray:



Courses Tray:



Courses

Computer Engineering Department, Graduate Admittees

Default Term

Settings

Notifications

Files

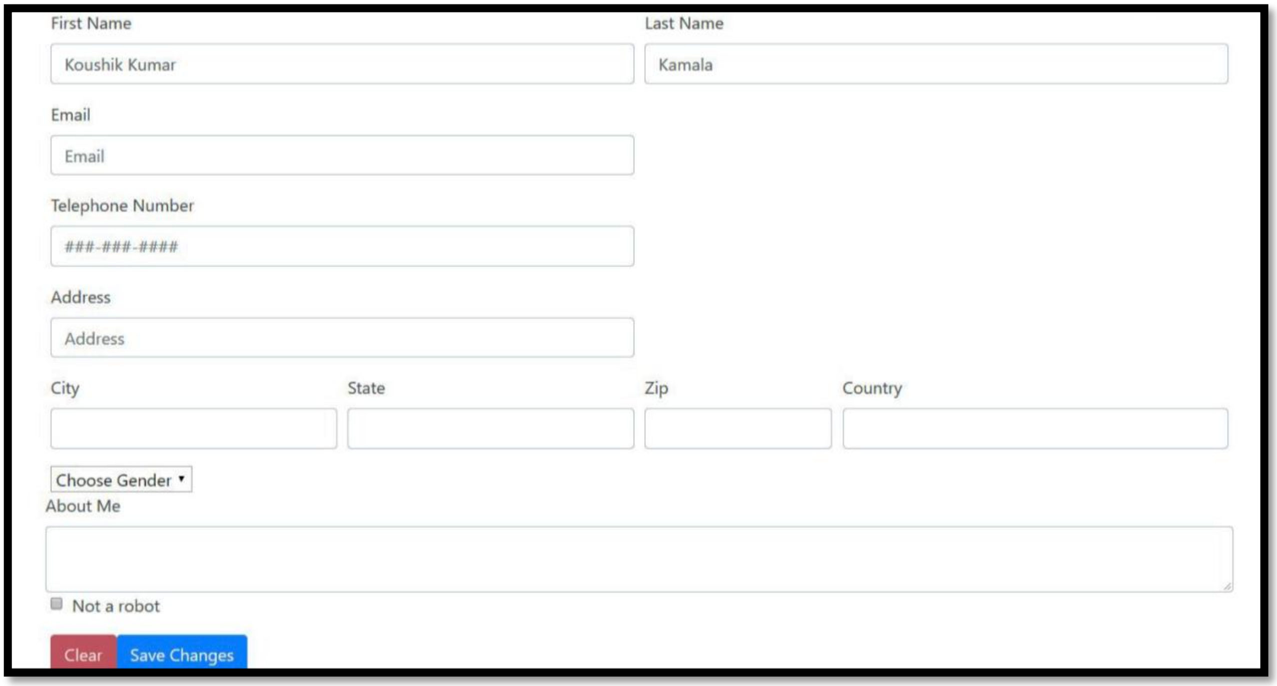
ePortFolios

All Courses

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

19: 200 Software Systems
19: 200 Software Systems
S19: 921 Management Science
S19: 921 Management Science
localhost:3000/Dashboard/dashboard.js#

Add a course:



First Name: Koushik Kumar

Last Name: Kamala

Email: Email

Telephone Number: ###-###-####

Address: Address

City:

State:

Zip:

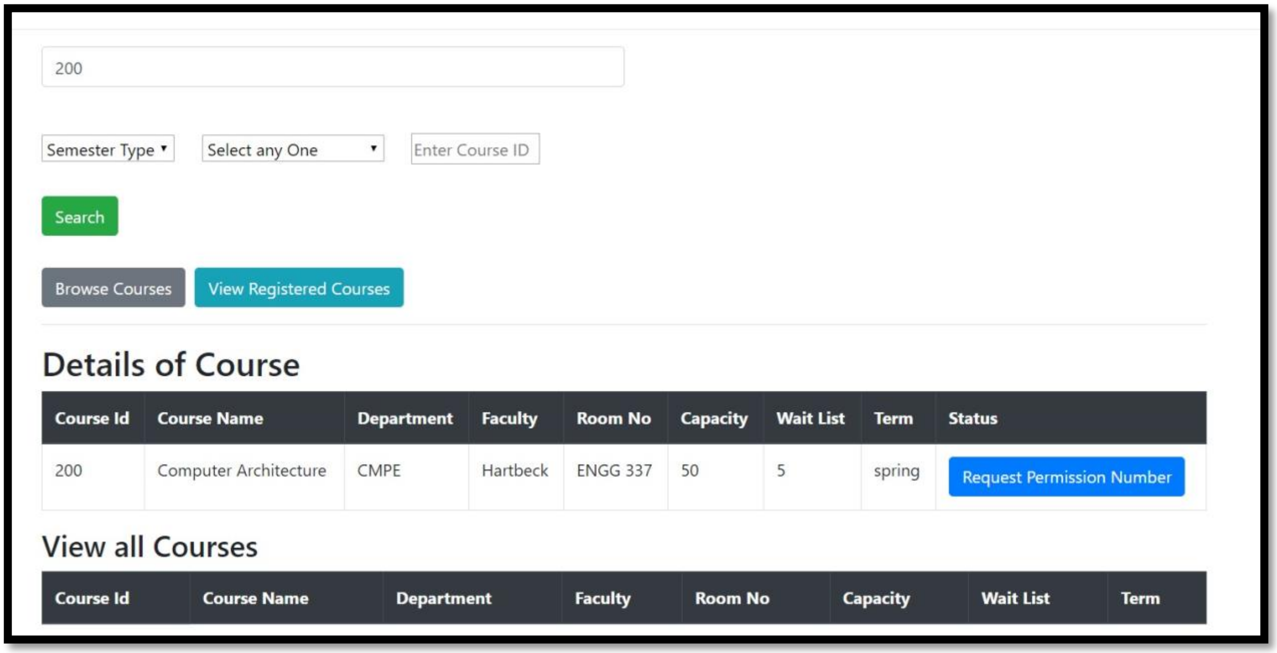
Country:

Choose Gender ▾

About Me:

□ Not a robot

Search courses:



200

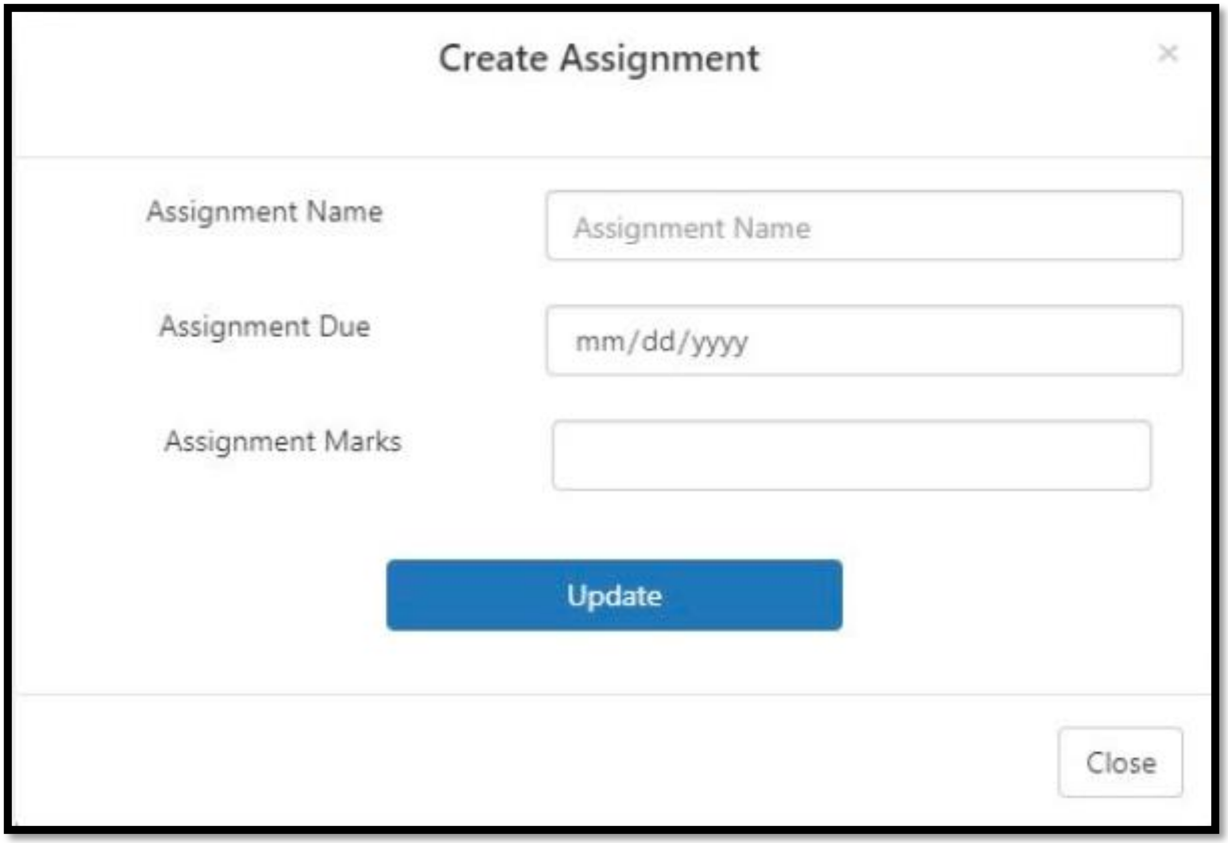
Details of Course

Course Id	Course Name	Department	Faculty	Room No	Capacity	Wait List	Term	Status
200	Computer Architecture	CMPE	Hartbeck	ENGG 337	50	5	spring	Request Permission Number

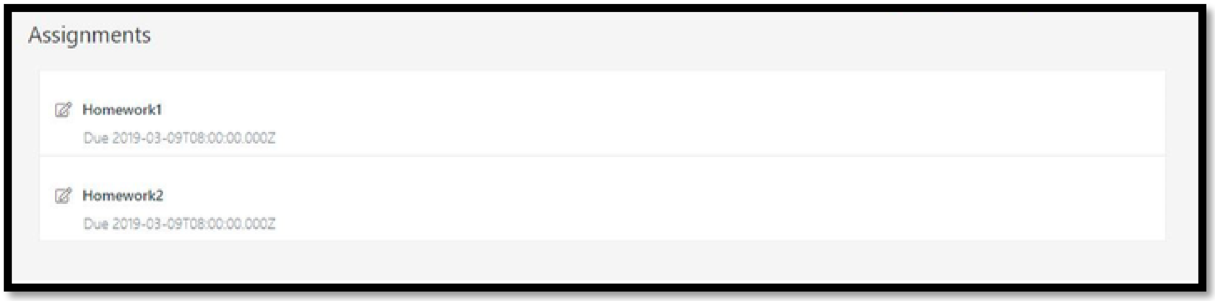
View all Courses

Course Id	Course Name	Department	Faculty	Room No	Capacity	Wait List	Term

Assignments creation:



Assignments List



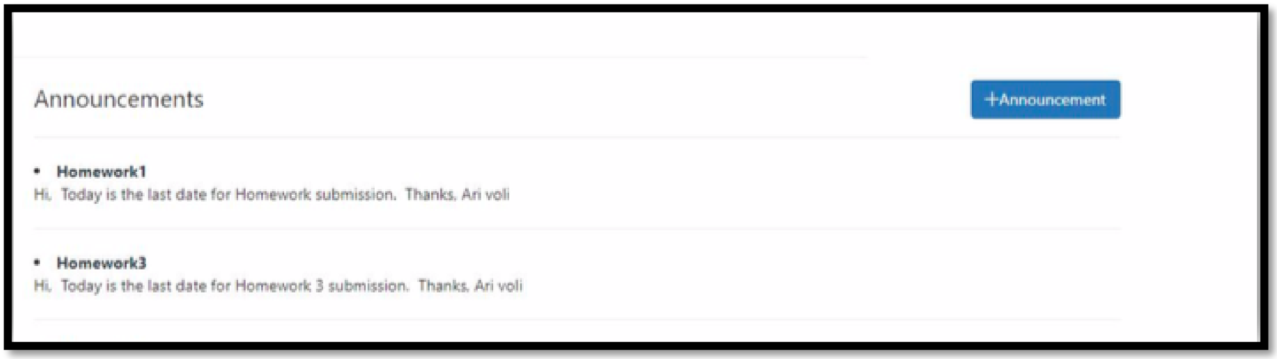
Homework1

Due 2019-03-09T08:00:00.000Z

Homework2

Due 2019-03-09T08:00:00.000Z

Announcements:



Announcements

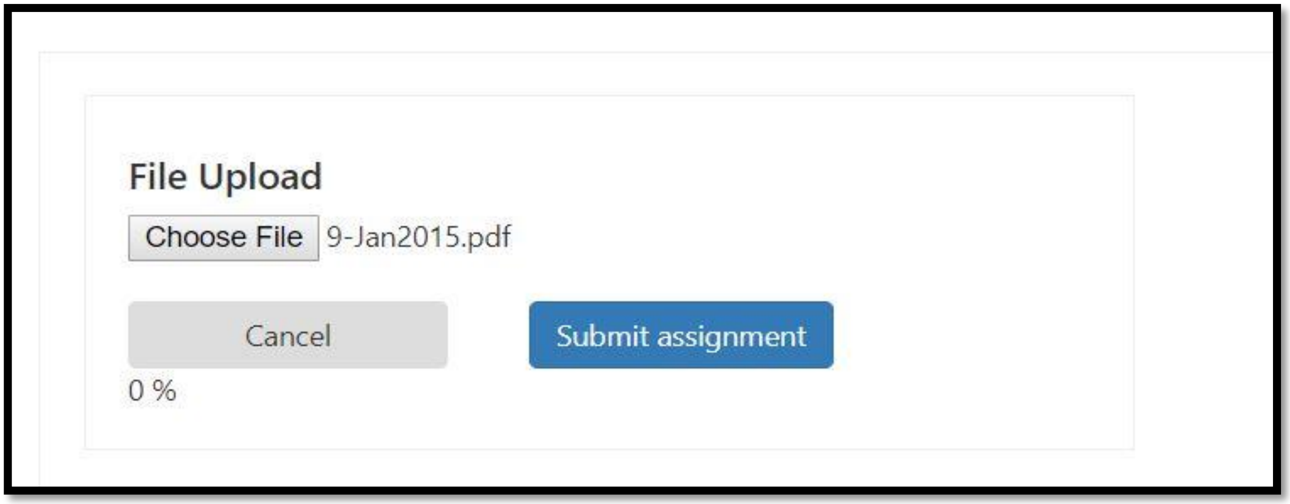
- **Homework1**

Hi, Today is the last date for Homework submission. Thanks, Ari voli

- **Homework3**

Hi, Today is the last date for Homework 3 submission. Thanks, Ari voli

Files Upload:

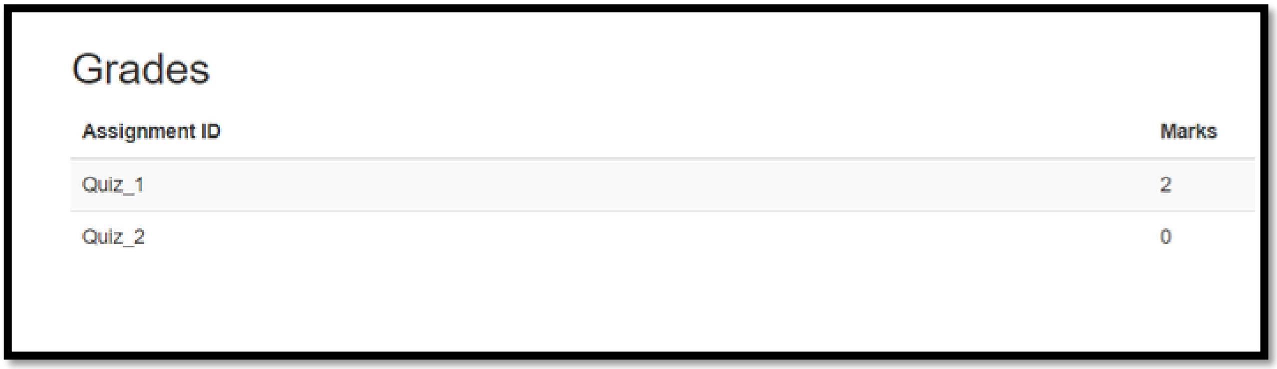


File Upload

9-Jan2015.pdf

0 %

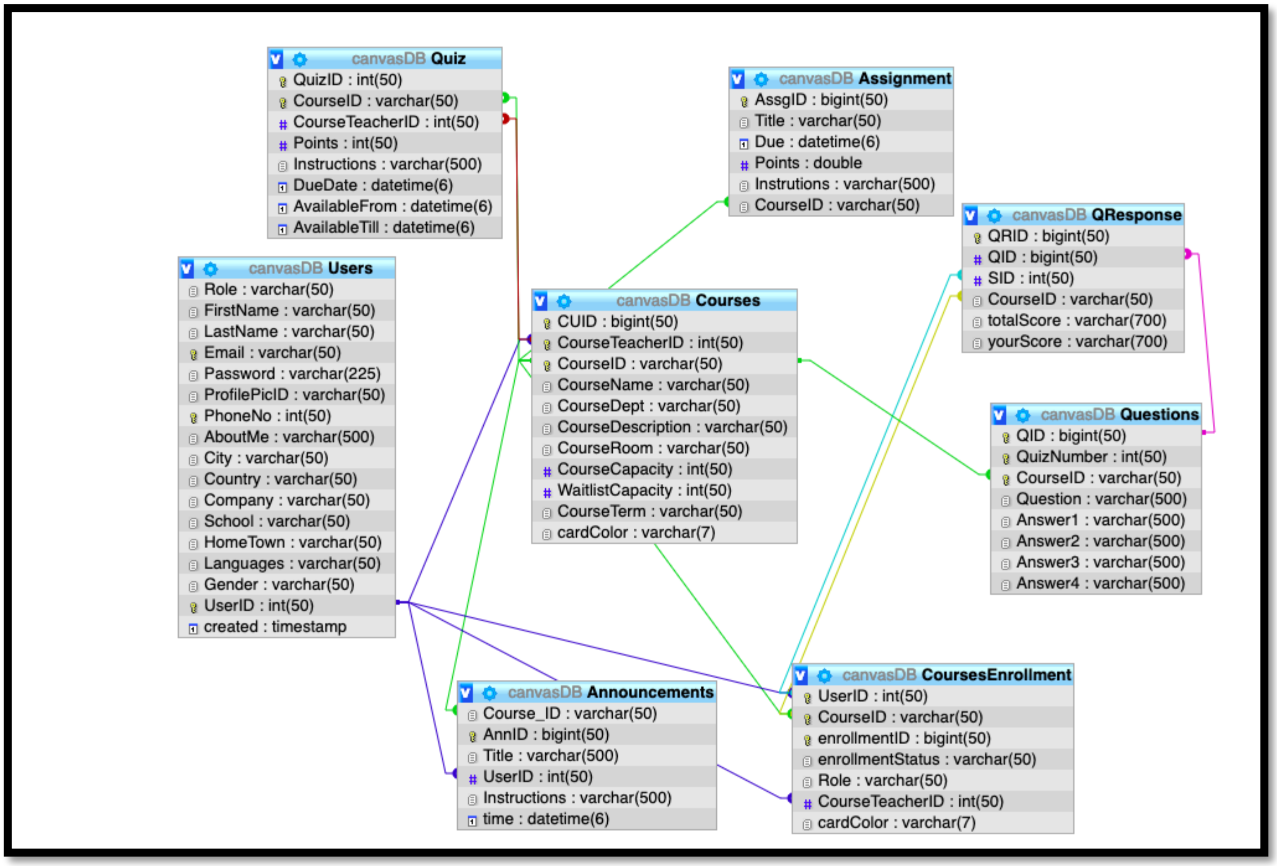
Grades:



Grades

Assignment ID	Marks
Quiz_1	2
Quiz_2	0

Database Design:



canvasDB **Quiz**
QuizID : int(50)
CourseID : varchar(50)
CourseTeacherID : int(50)
Points : int(50)
Instructions : varchar(500)
DueDate : datetime(6)
AvailableFrom : datetime(6)
AvailableTill : datetime(6)

canvasDB **Assignment**
AssgID : bigint(50)
Title : varchar(50)
Due : datetime(6)
Points : double
Instrutions : varchar(500)
CourseID : varchar(50)

canvasDB **QResponse**
QRID : bigint(50)
QID : bigint(50)
SID : int(50)
CourseID : varchar(50)
totalScore : varchar(700)
yourScore : varchar(700)

canvasDB **Users**
Role : varchar(50)
FirstName : varchar(50)
LastName : varchar(50)
Email : varchar(50)
Password : varchar(225)
ProfilePicID : varchar(50)
PhoneNo : int(50)
AboutMe : varchar(500)
City : varchar(50)
Country : varchar(50)
Company : varchar(50)
School : varchar(50)
HomeTown : varchar(50)
Languages : varchar(50)
Gender : varchar(50)
UserID : int(50)
created : timestamp

canvasDB **Courses**
CUID : bigint(50)
CourseTeacherID : int(50)
CourseID : varchar(50)
CourseName : varchar(50)
CourseDept : varchar(50)
CourseDescription : varchar(50)
CourseRoom : varchar(50)
CourseCapacity : int(50)
WaitlistCapacity : int(50)
CourseTerm : varchar(50)
cardColor : varchar(7)

canvasDB **Questions**
QID : bigint(50)
QuizNumber : int(50)
CourseID : varchar(50)
Question : varchar(500)
Answer1 : varchar(500)
Answer2 : varchar(500)
Answer3 : varchar(500)
Answer4 : varchar(500)

canvasDB **Announcements**
Course_ID : varchar(50)
AnnID : bigint(50)
Title : varchar(500)
UserID : int(50)
Instructions : varchar(500)
time : datetime(6)

canvasDB **CoursesEnrollment**
UserID : int(50)
CourseID : varchar(50)
enrollmentID : bigint(50)
enrollmentStatus : varchar(500)
Role : varchar(50)
CourseTeacherID : int(50)
cardColor : varchar(7)

QUESTION & ANSWERS:

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

The algorithm used in my application is bcrypt algorithm with 10 as salts count. Basically, this is one of the secure algorithms available and takes huge time in decrypting with proper salts count.

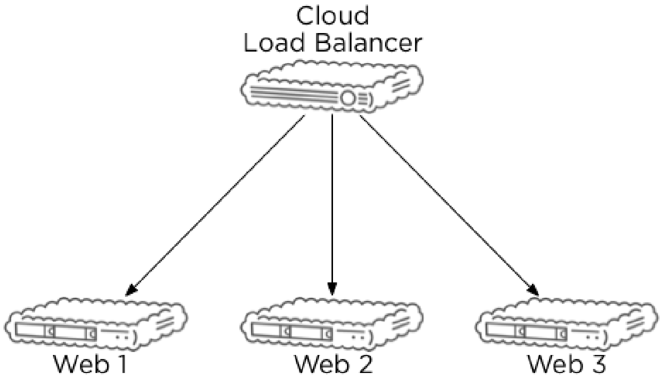
2. Compare the results of graphs with and without In-built mysql connection pooling of Database. Explain the result in detail and describe the connection pooling algorithm if you need to implement connection pooling on your own.

A free stack, where a connection to be added to the pool, push it into the stack, and when is requested, pop it from the stack, and then according to the request availability, respective actions can be defines either to wait or build a new one, make changes to the existing etc.

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

SQL caching refers to the process of caching the result of a process into an object instead of rendering every time from the server thereby accounting to performance. In MySQL, we gonna use, query_cache_size and query_cache_type for the configuration.

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



Horizontal scaling is always useful since we don't get caught in resources deficit. Here we can horizontally scale in three ways,

a. Cloning (Forming cluster – 2 servers, same DB, divide the workload based on the load amount) *b. Splitting (Here the application is split into instances, each being responsible for specific functionality of the application, this strategy also known as* ***horizontal portioning or sharding****)*

c. decomposing (Multiple and different applications, sometimes even with separate codebase and specific UIs)