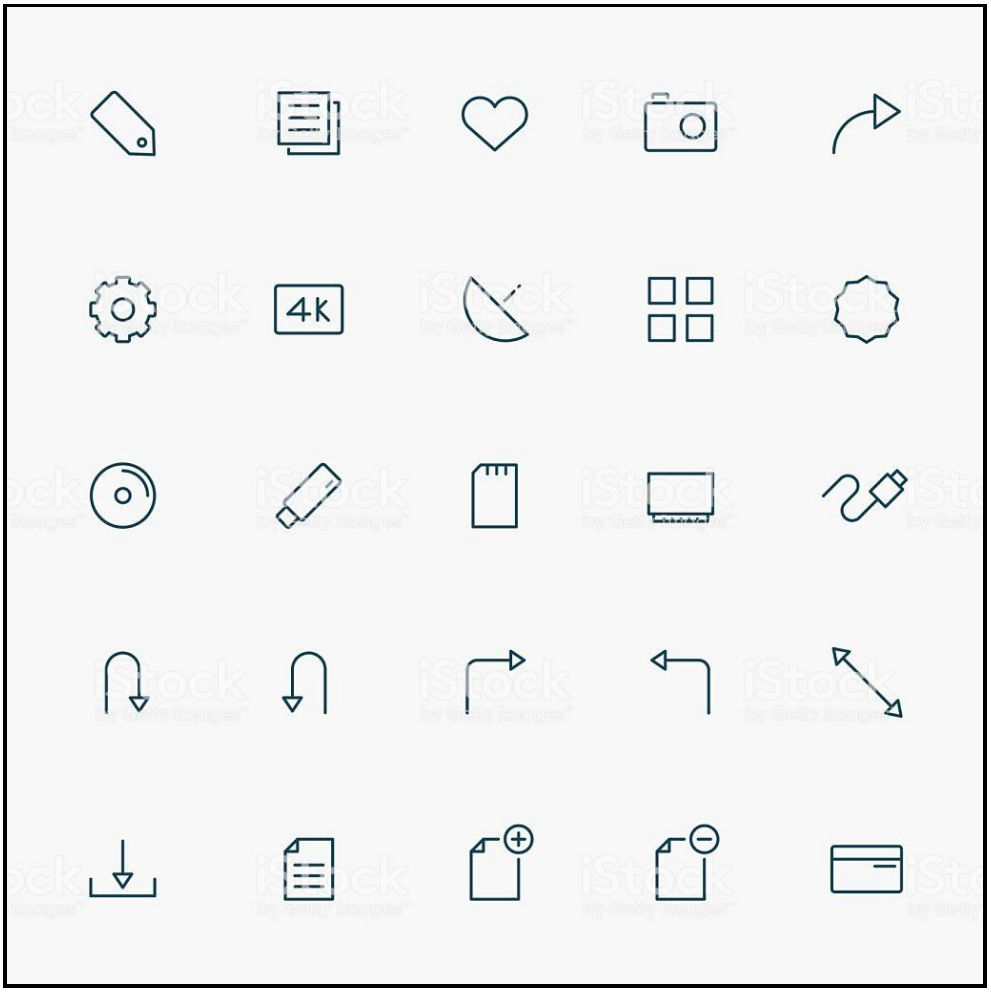


Project Report - Lab 1

ENTERPRISE DISTRIBUTED SYSTEMS

Shivang Mistry | 013823108 | 17th March 2019

Part 1 - Calculator

Introduction

This application implements a calculator with basic functionalities like addition, subtraction, multiplication and division. The goal is to respond to the requests efficiently with optimum performance.

System Design

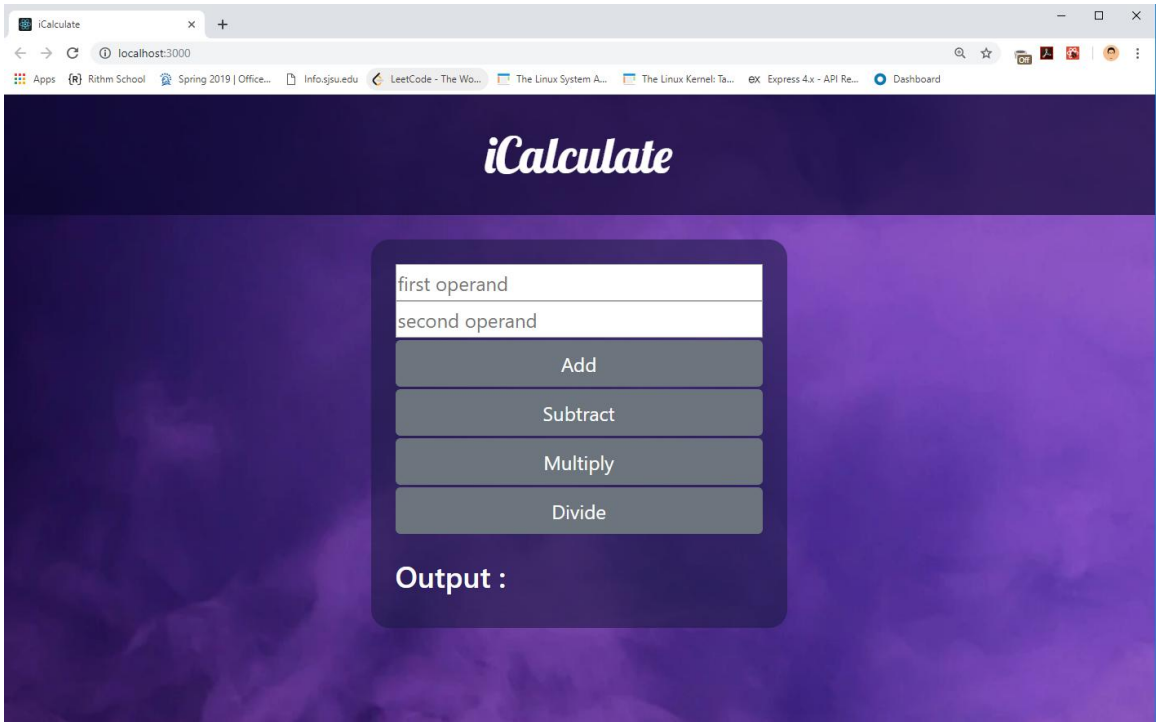
The system is composed of a server and multiple clients. The server is responsible for performing basic operations like addition, subtraction, multiplication and division, using RESTful services. It maintains a minimal design and simple layout for the client to enter data and get the result. The input fields are validated to accept a valid number. All the elements are responsive to varying browser width and hence appears compatible with mobile/tablet view.

Software: HTML, Javascipt, CSS, Node.js, React, Bootstrap.

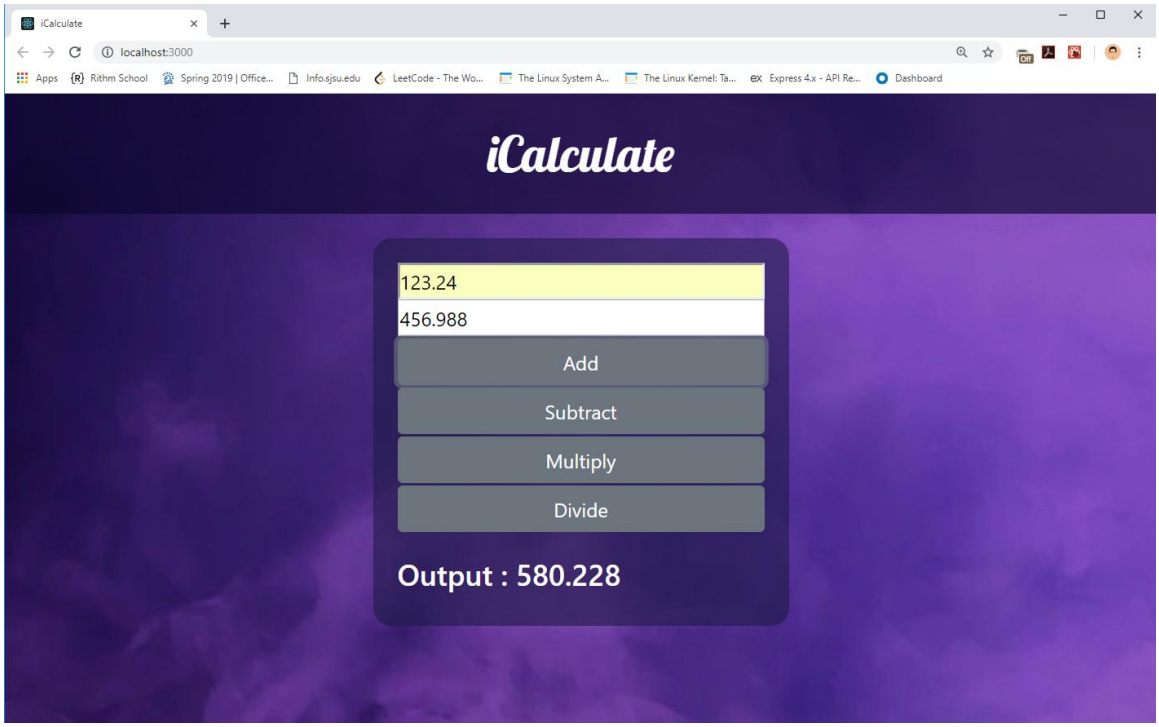
Hardware: Any computer having Chrome and required packages installed.

Results

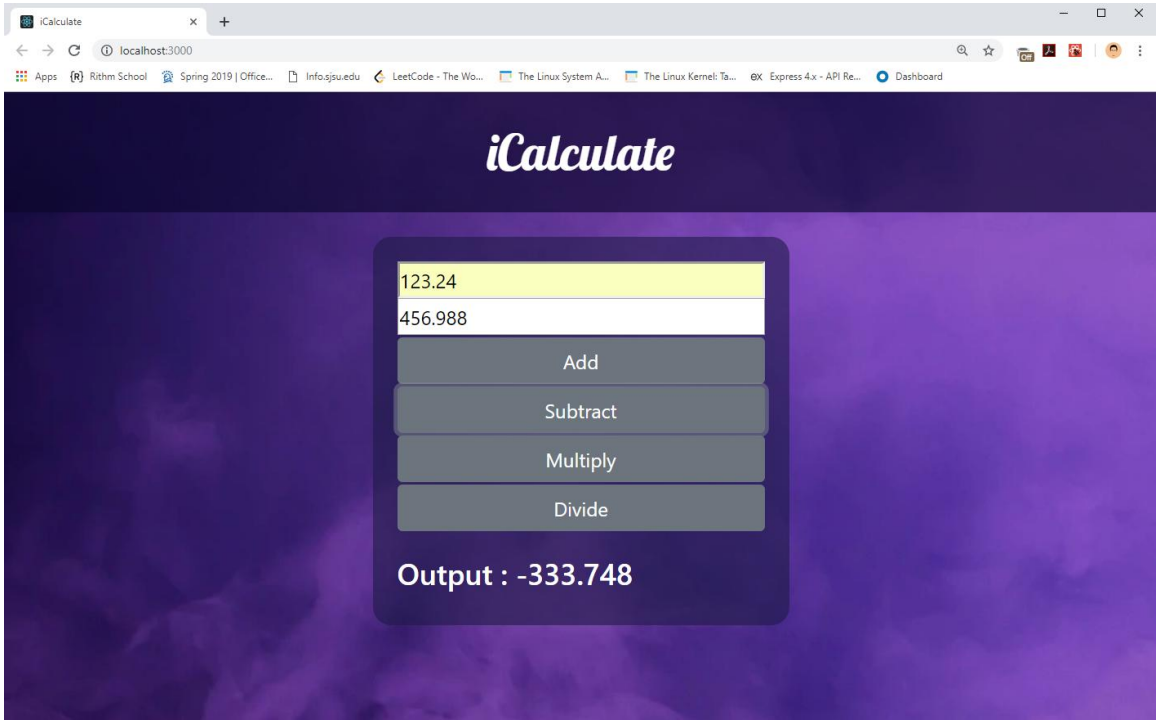
Homepage



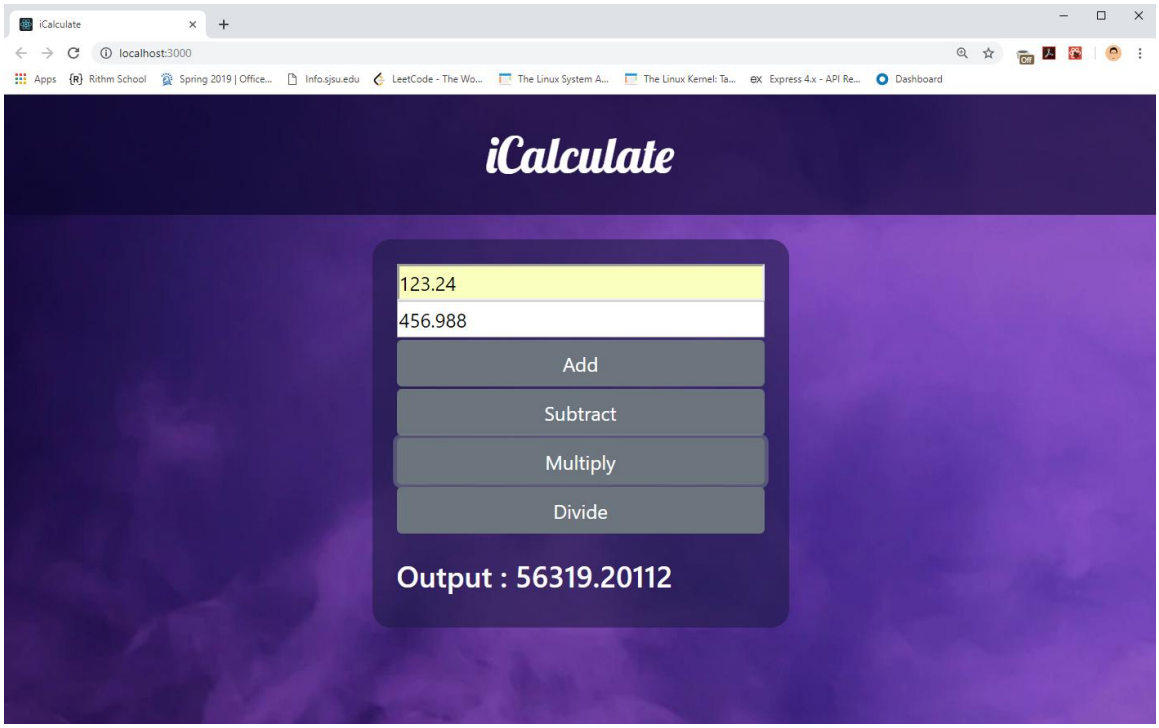
User input and corresponding output for addition operation:



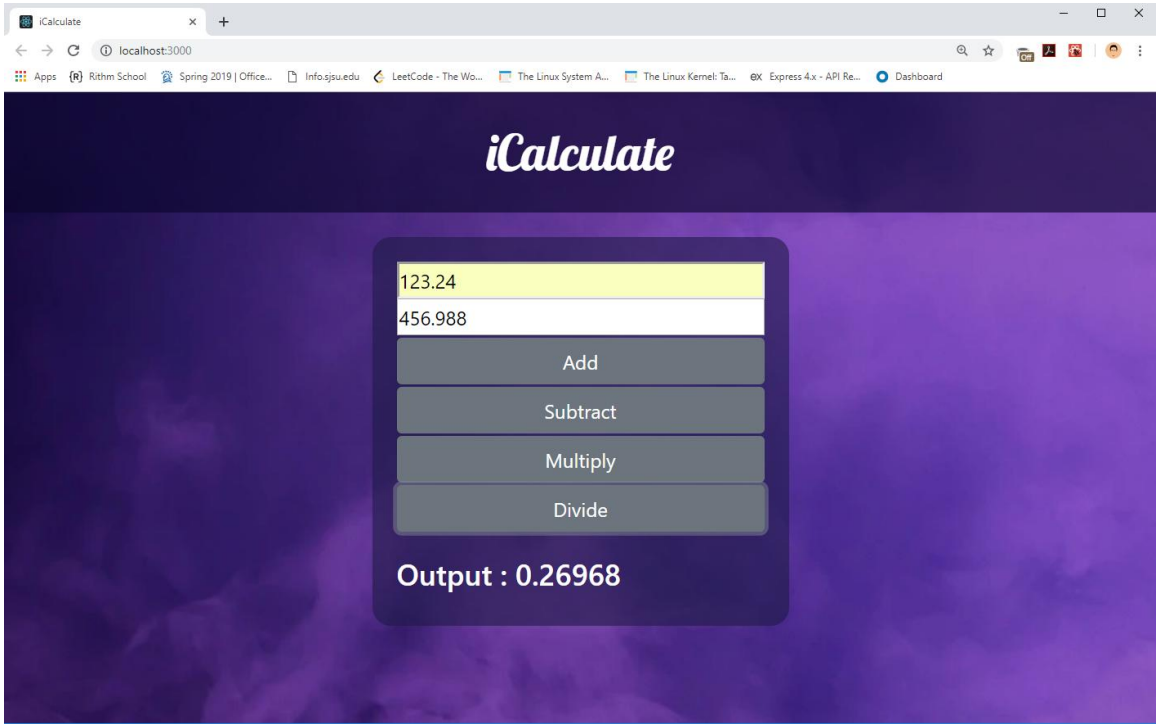
User input and corresponding output for subtraction operation:



User input and corresponding output for multiplication operation:



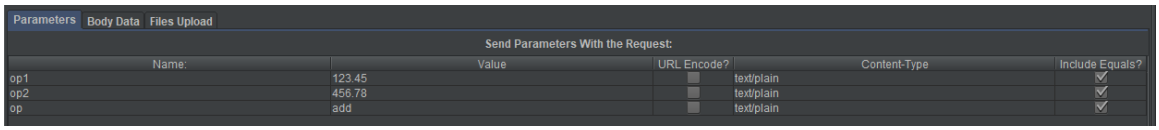
User input and corresponding output for division operation:



Performance

Test Operations:

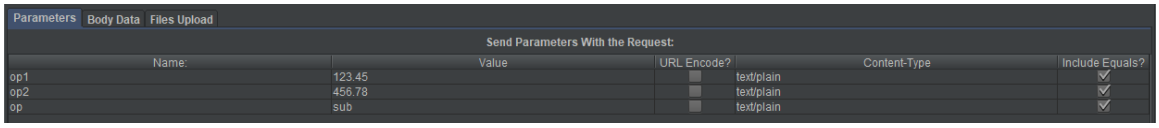
Addition


Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
op1	123.45	☐	text/plain	☑
op2	456.78	☐	text/plain	☑
op	add	☐	text/plain	☑

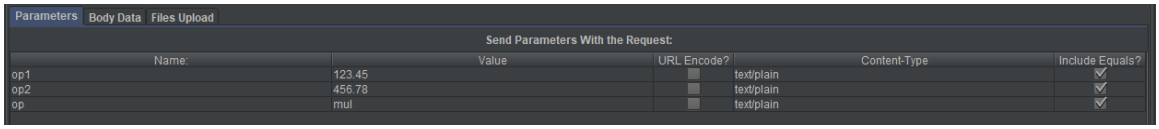
Subtraction


Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
op1	123.45	☐	text/plain	☑
op2	456.78	☐	text/plain	☑
op	sub	☐	text/plain	☑

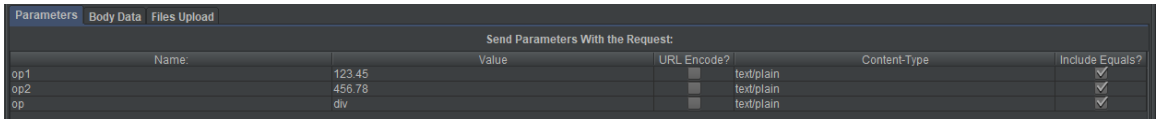
Multiplication


Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
op1	123.45	☐	text/plain	☑
op2	456.78	☐	text/plain	☑
op	mul	☐	text/plain	☑

Division

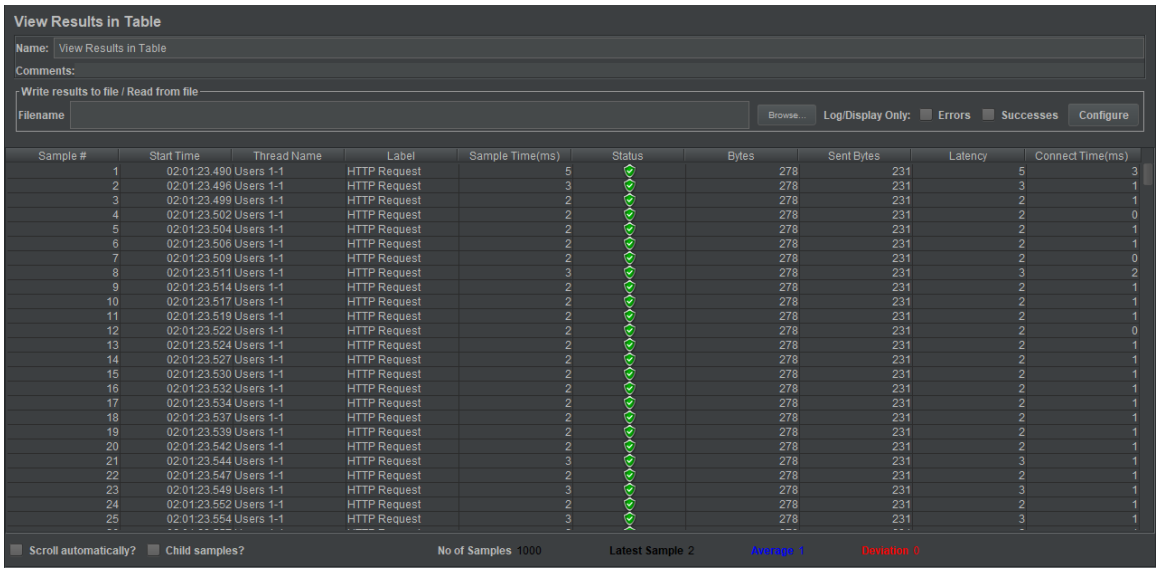

Parameters | Body Data | Files Upload

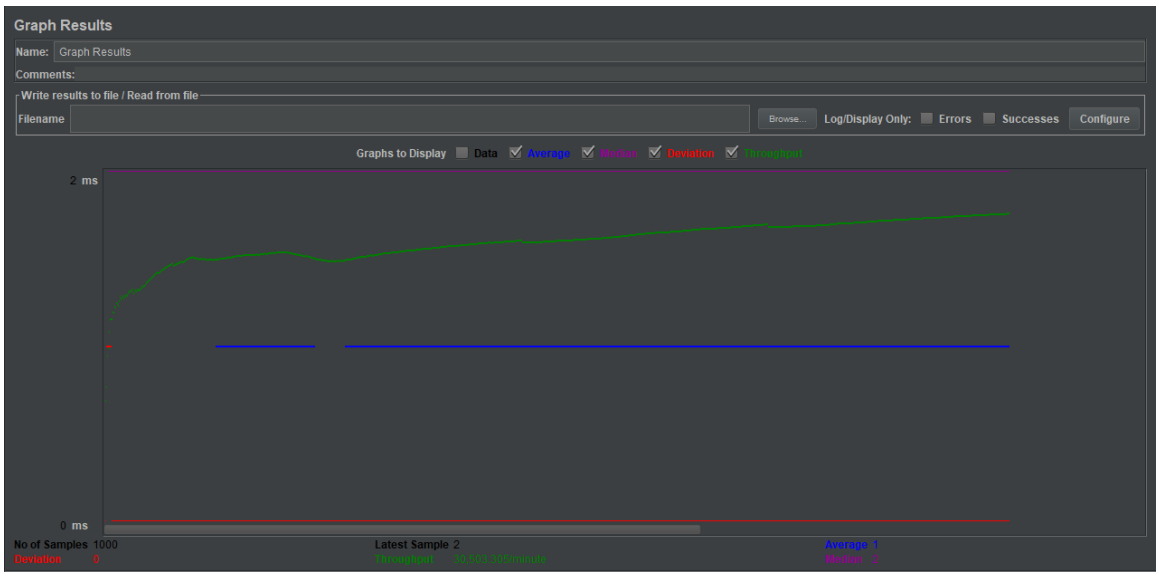
Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
op1	123.45	☐	text/plain	☑
op2	456.78	☐	text/plain	☑
op	div	☐	text/plain	☑

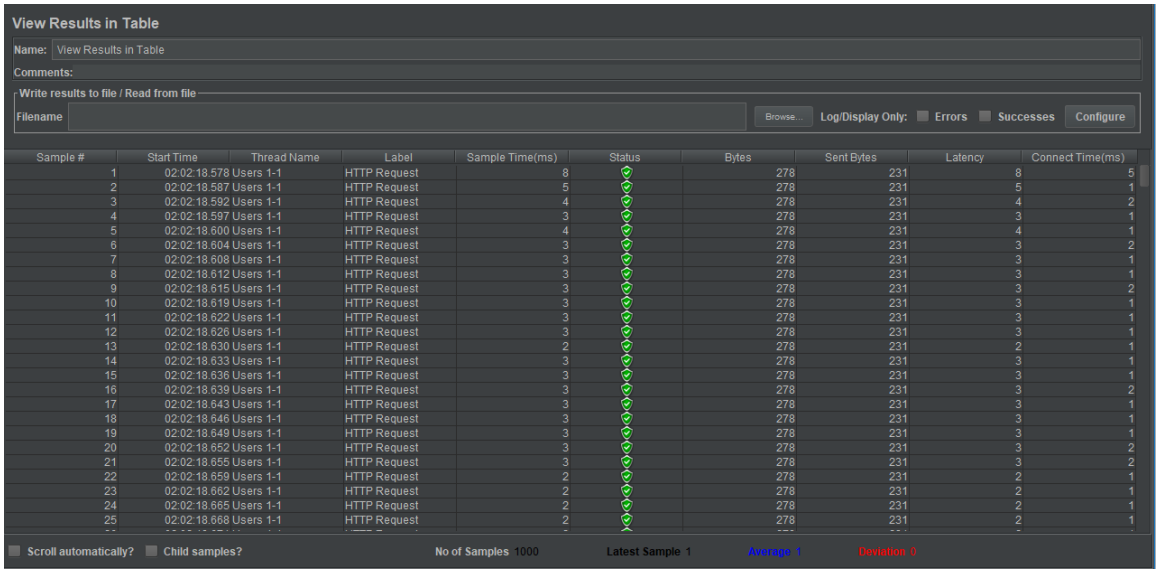
Test Case 1: Invoking 1000 random calls:

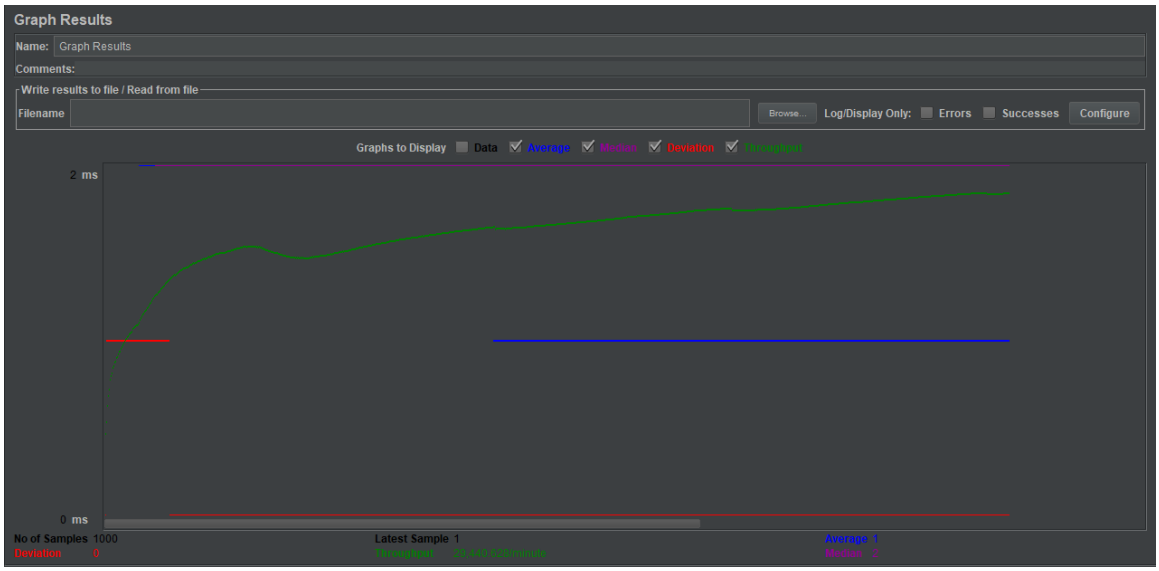
Operation = addition, average = 1 ms



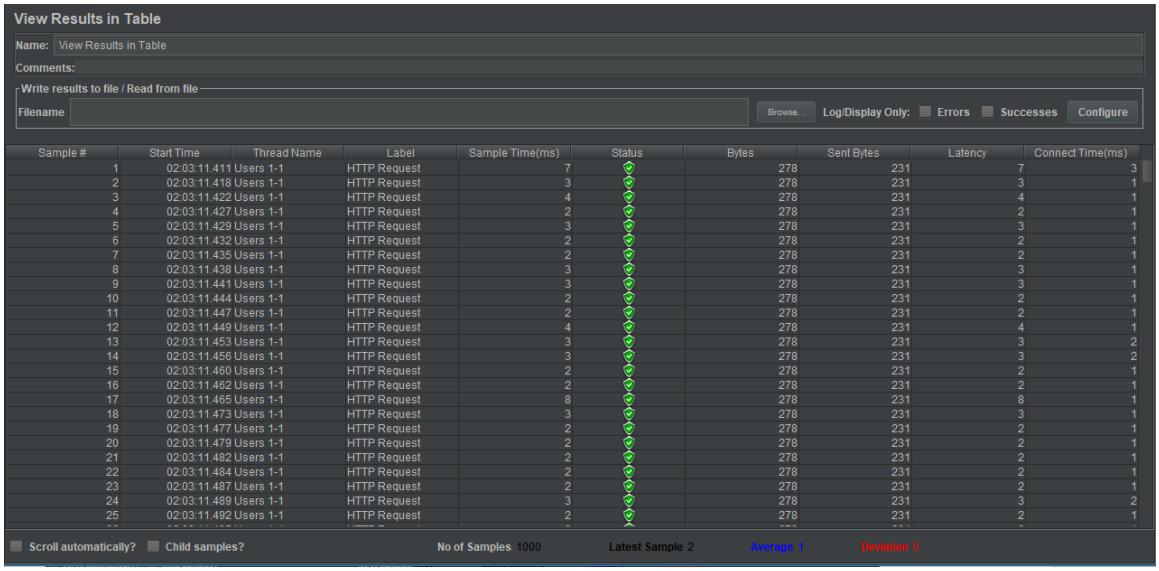


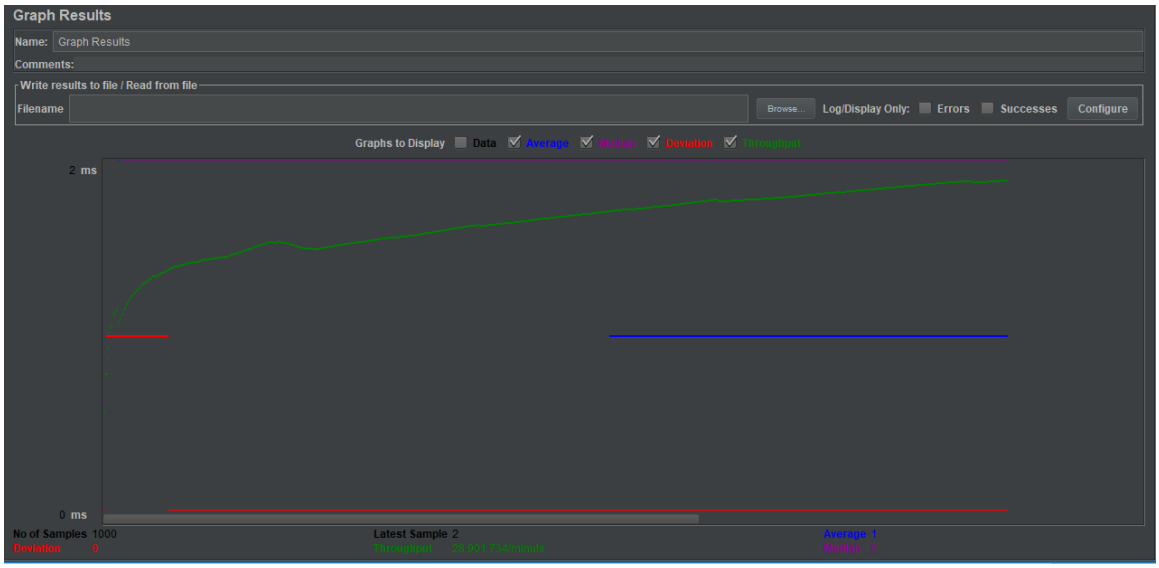
Operation = subtraction, average = 1 ms



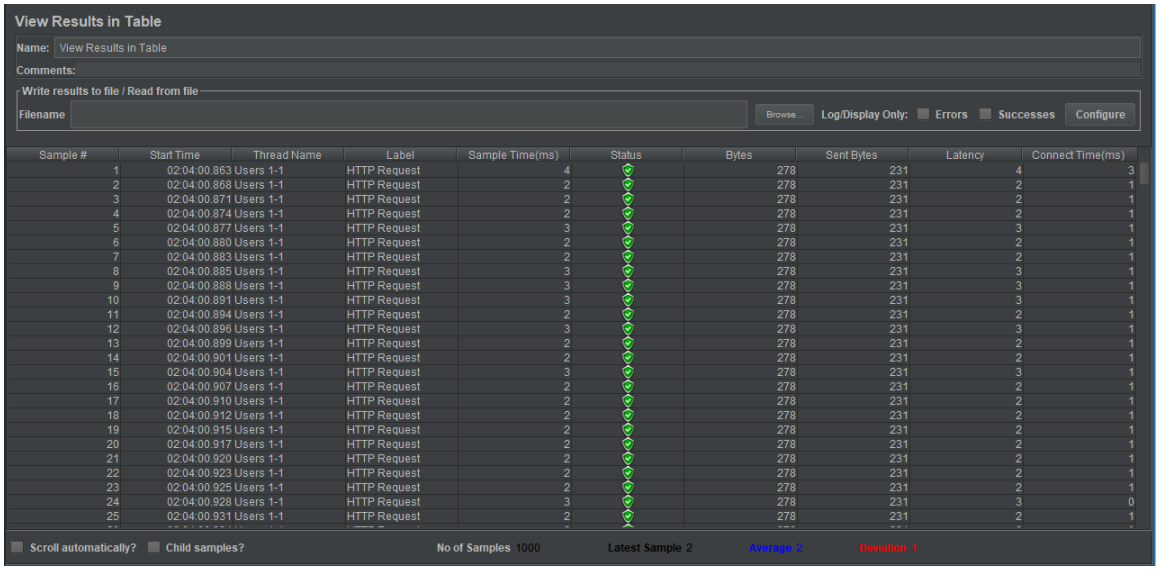


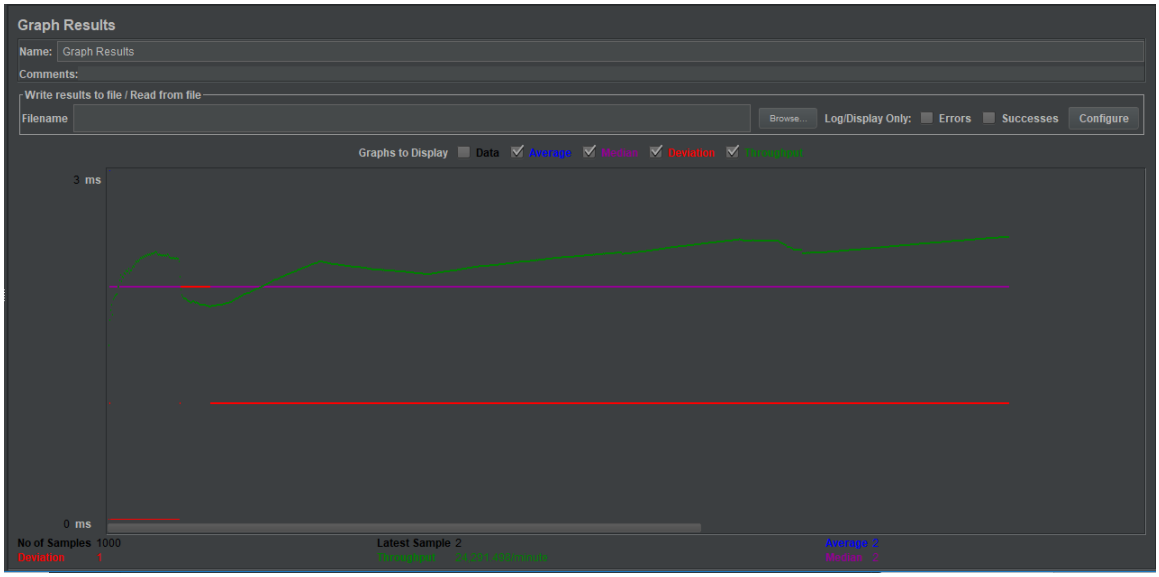
Operation = multiplication, average = 1 ms





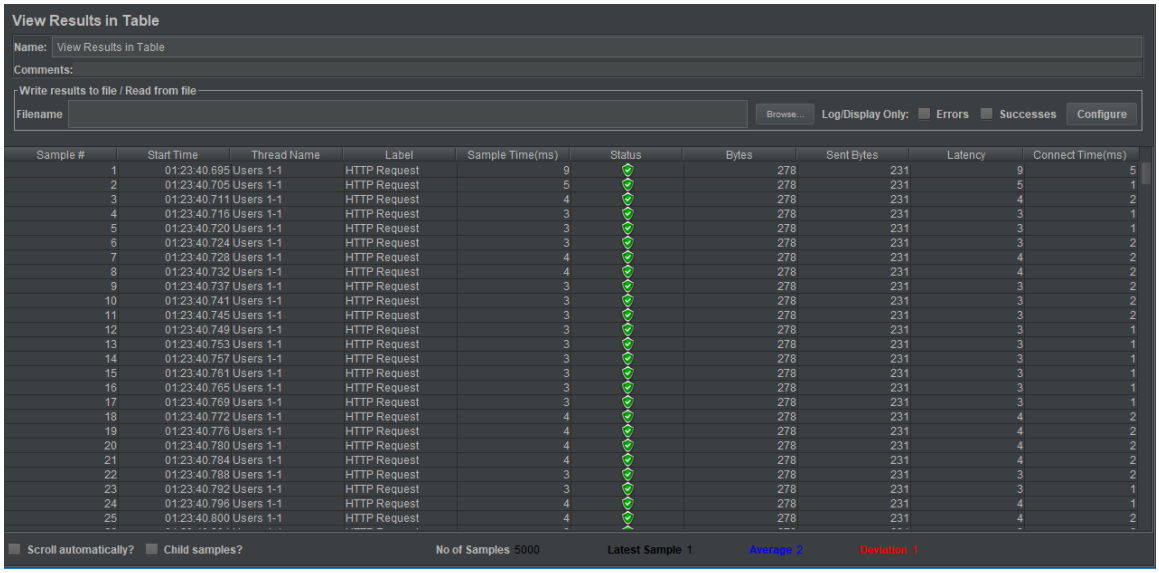
Operation = division, average = 2 ms

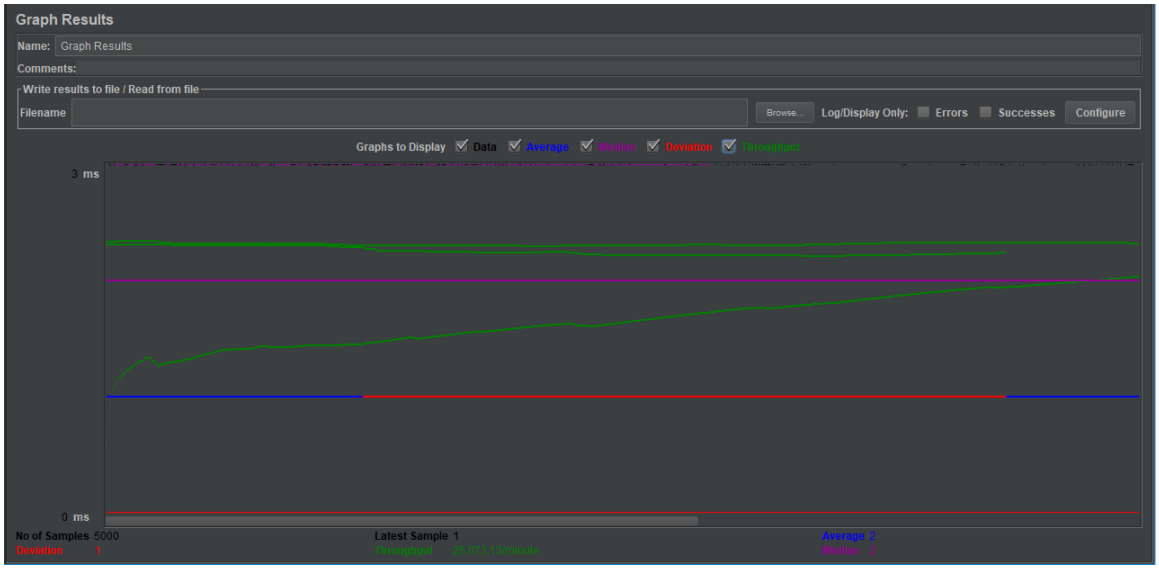




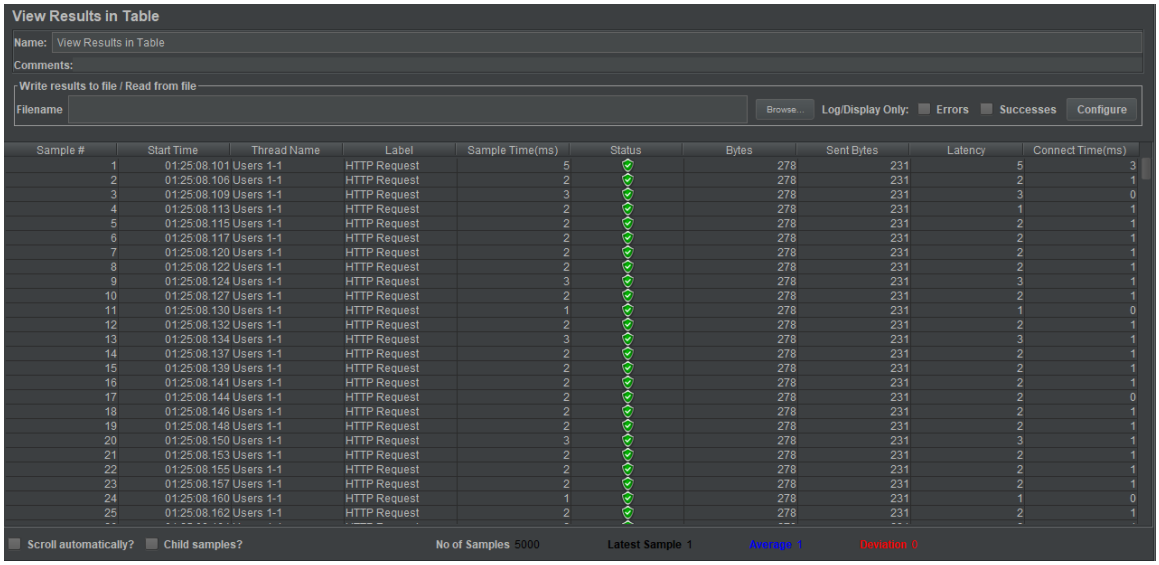
Test Case 2: Invoking 5000 random calls:

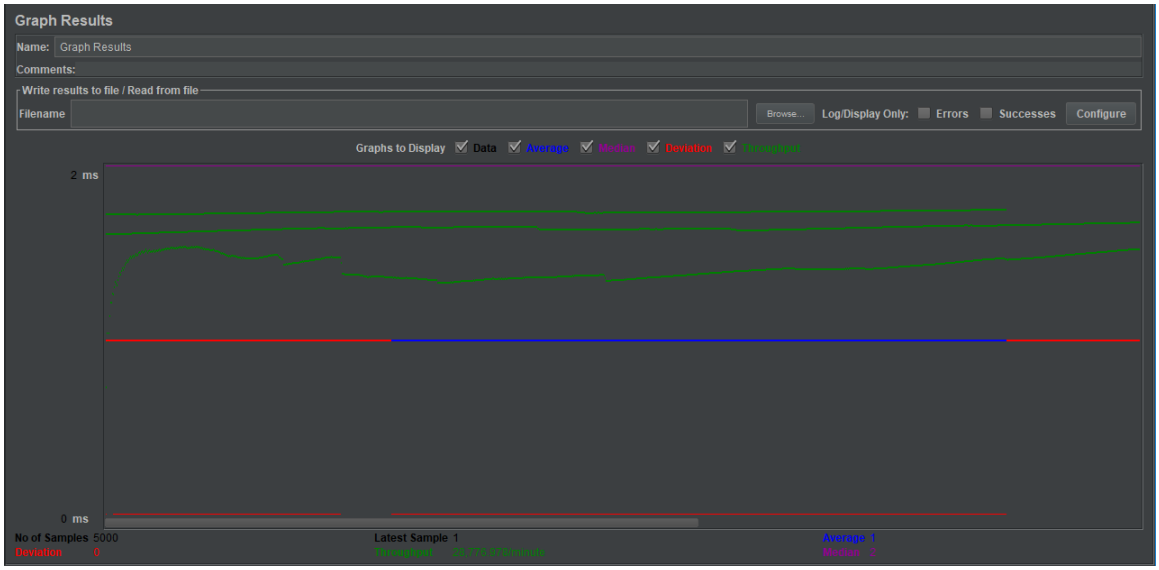
Operation = addition, average = 2 ms



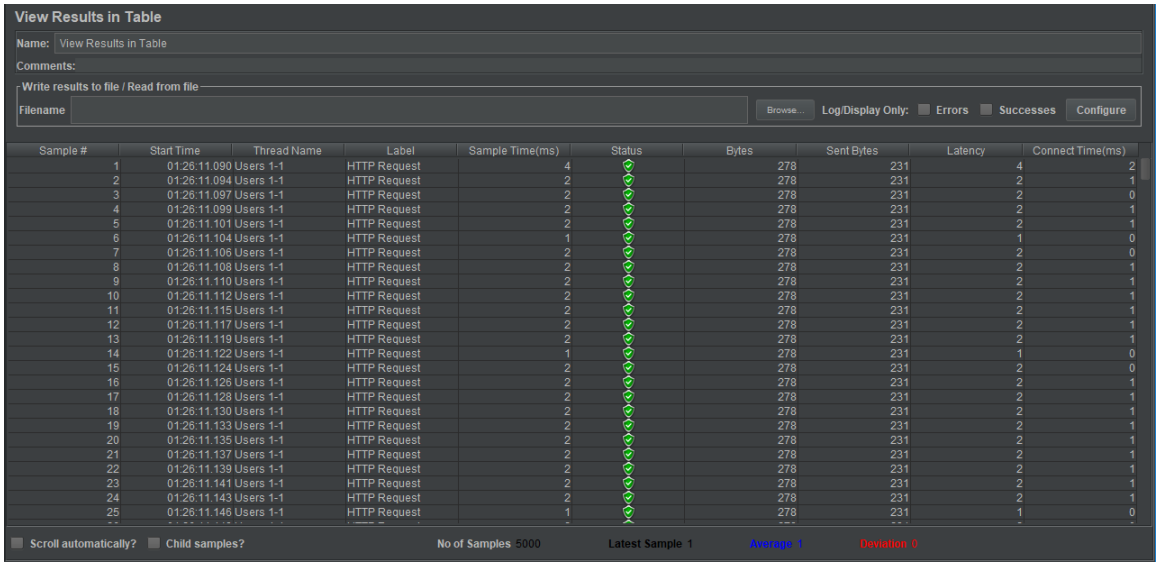


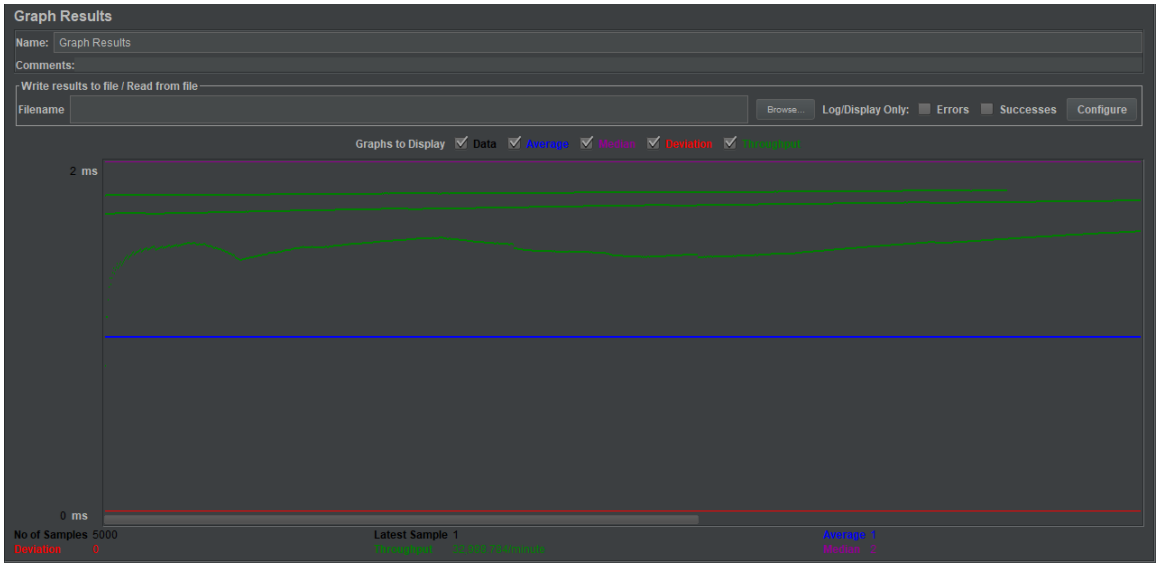
Operation = subtraction, average = 2 ms



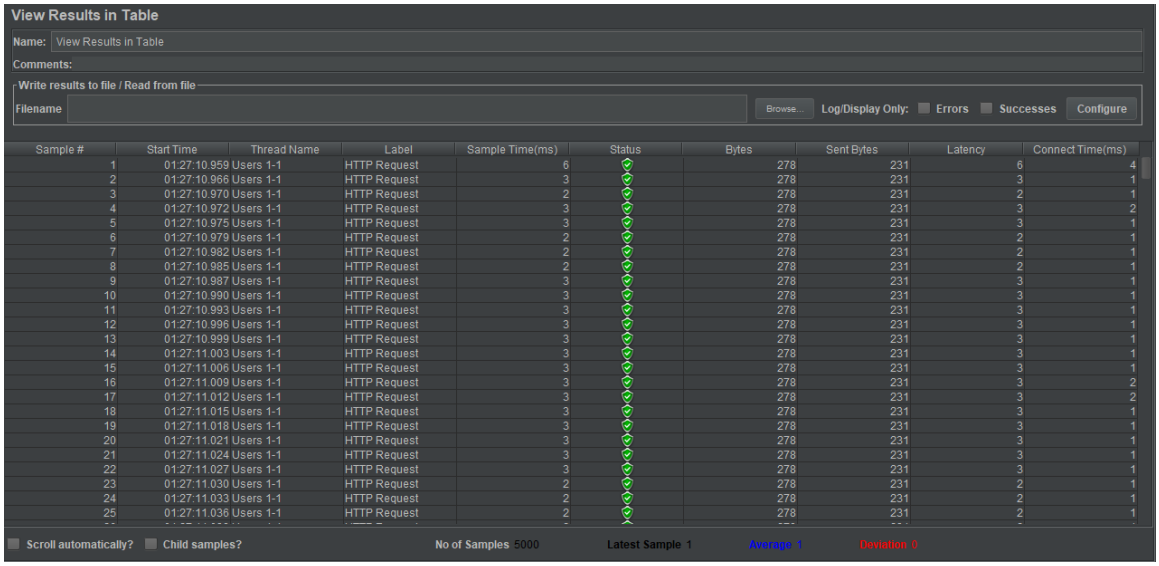


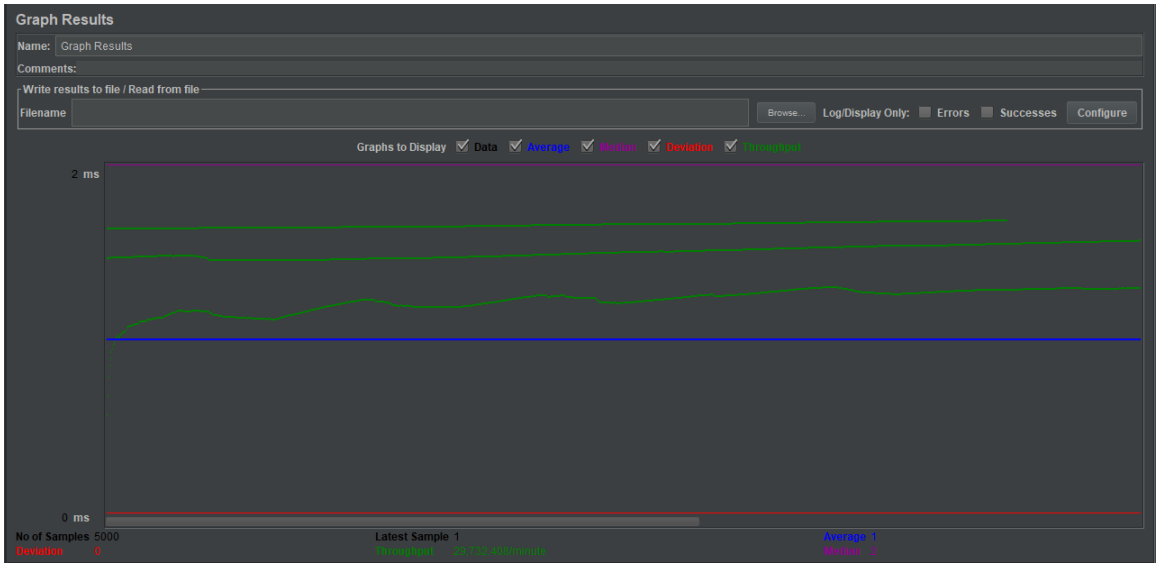
Operation = multiplication, average = 1 ms



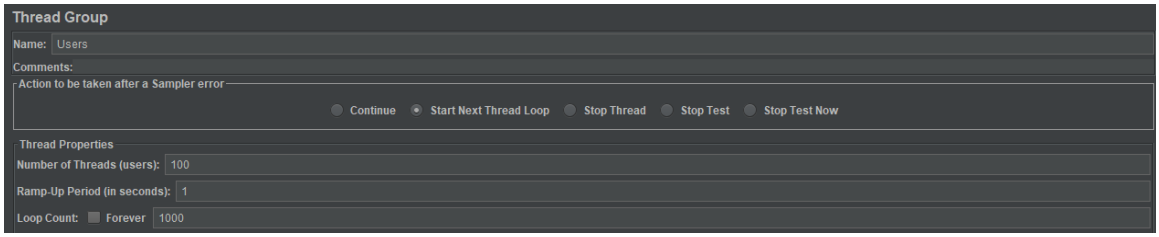


Operation = division, average = 1 ms

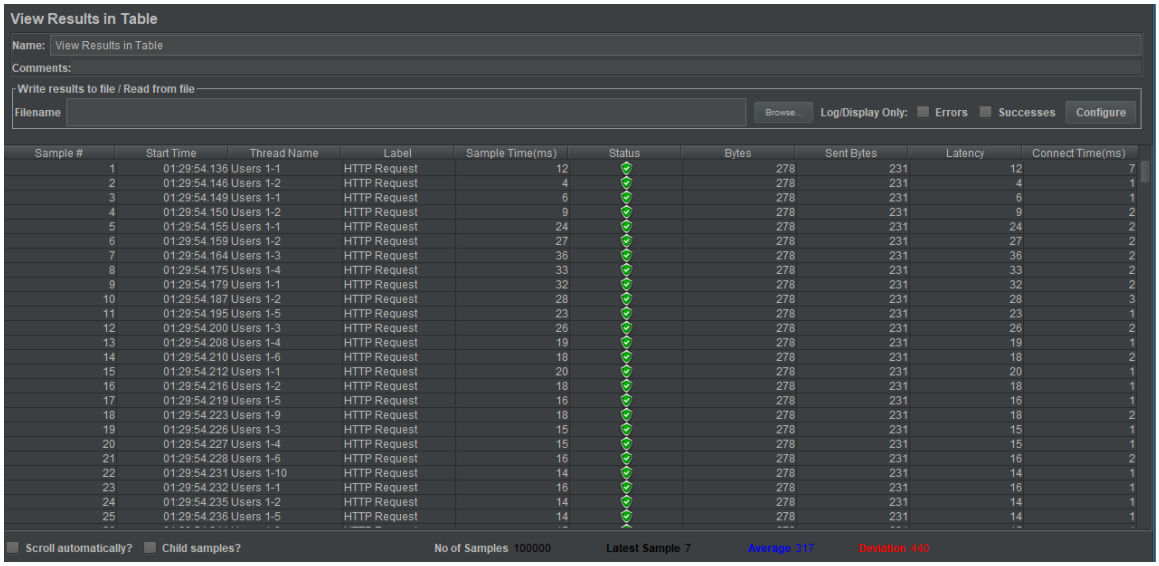


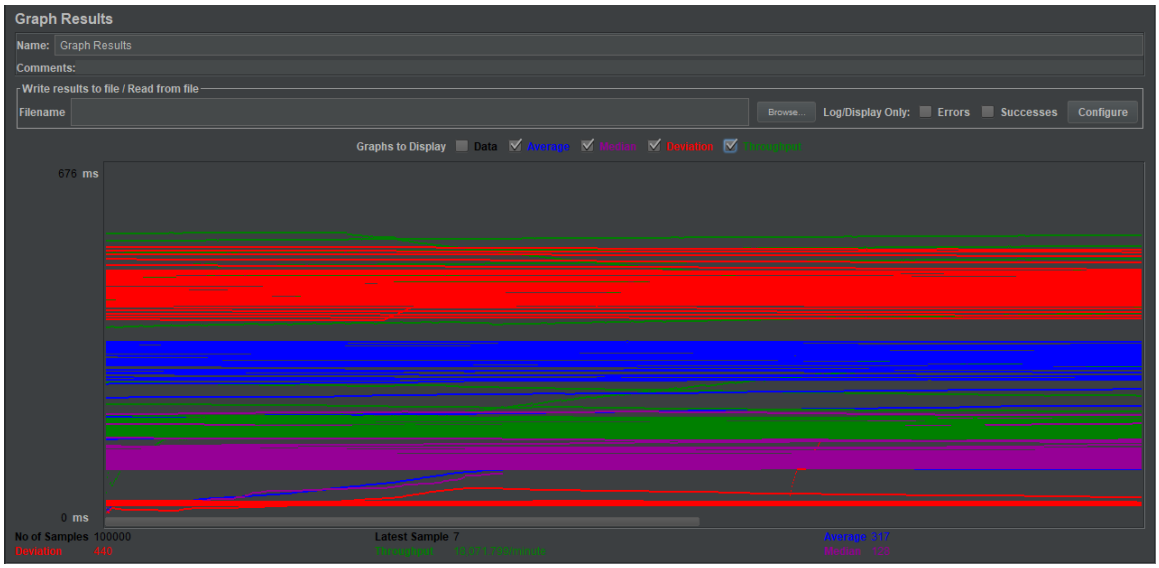


Test Case 3: Invoking 1000 calls from 100 concurrent users:

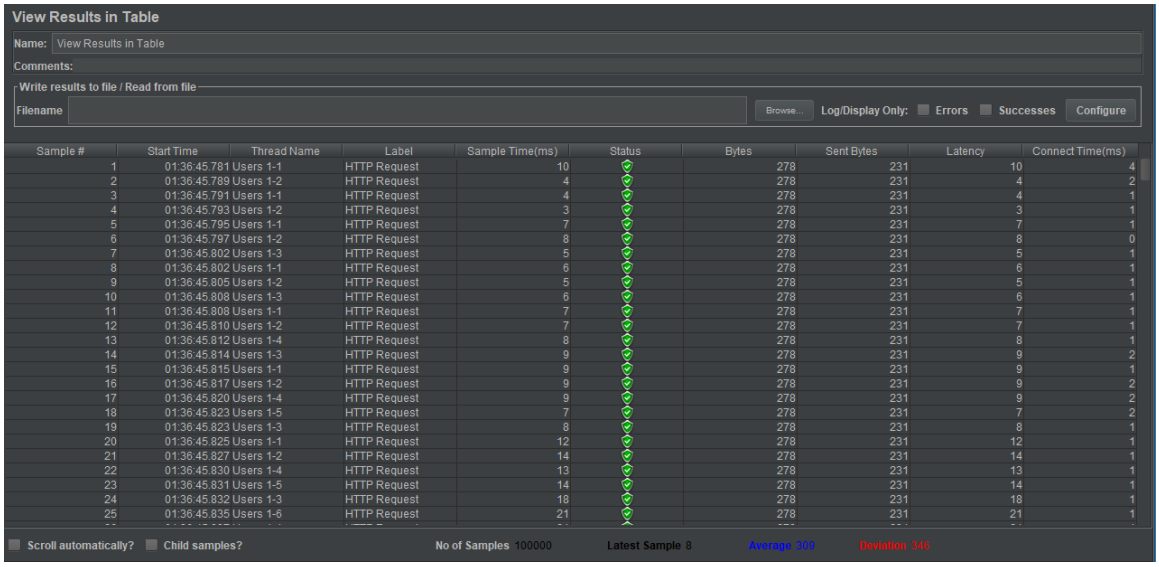


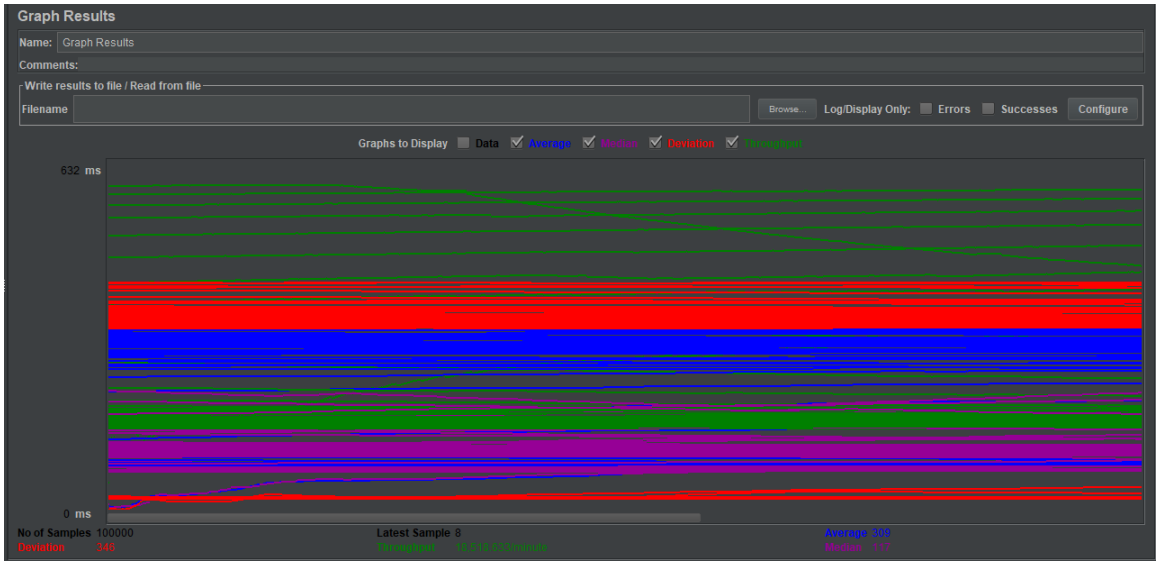
Operation = addition, average = 317 ms



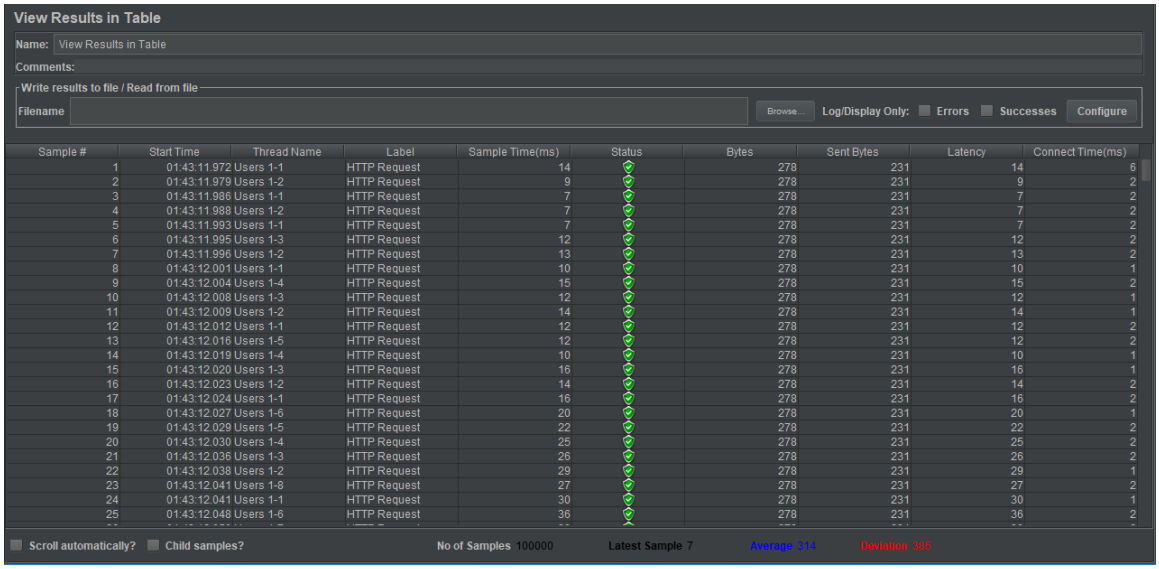


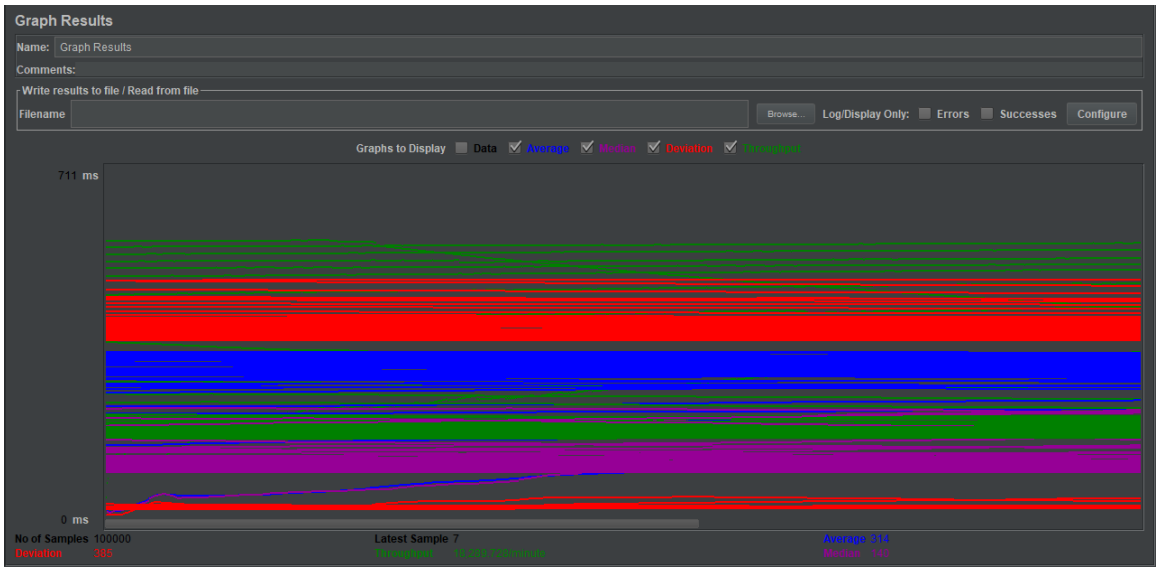
Operation = subtraction, average = 309 ms



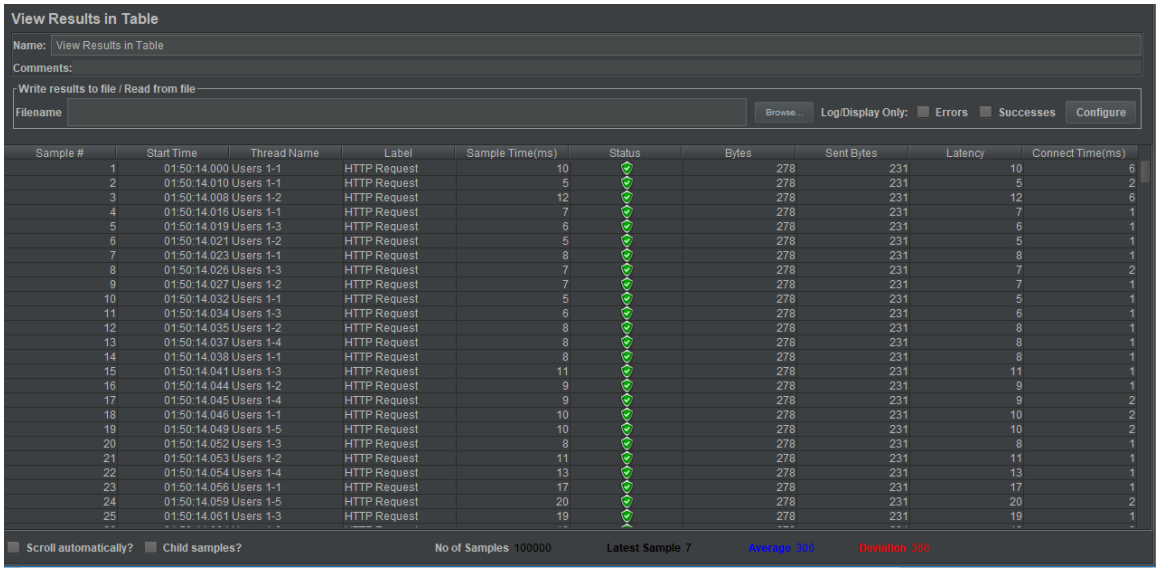


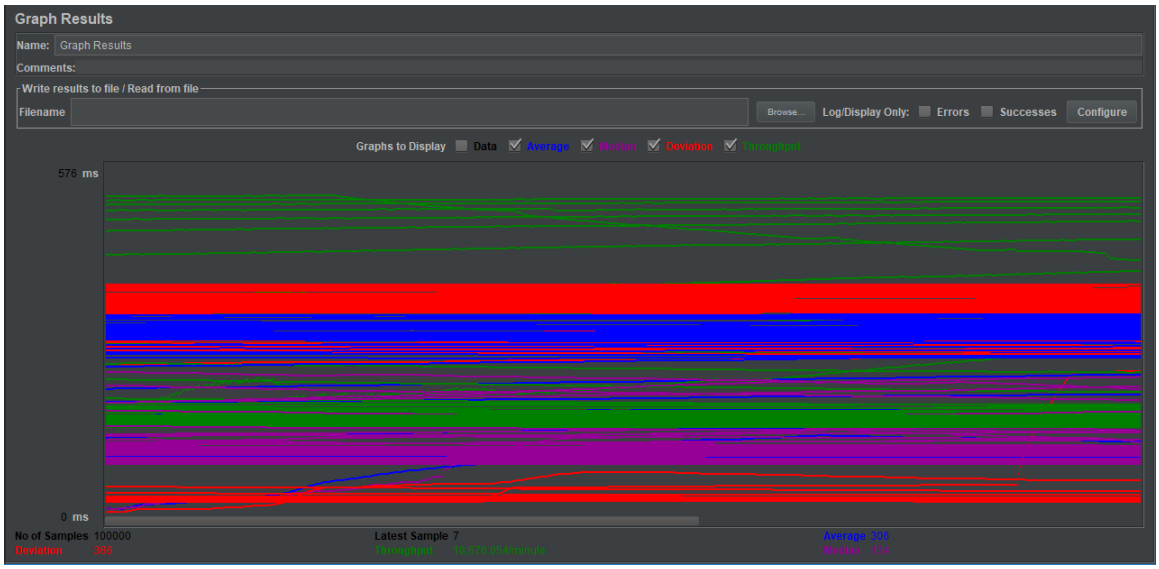
Operation = multiplication, average = 314 ms





Operation = division, average = 306ms





Conclusion

The throughput is almost same for test cases 1,2 while that for test case 3 is less due to more concurrent requests.

The response time is being between 1-2 ms for test case 1 and 2. This implies the despite increasing the number of requests in thousands, the change in average response time is nearly the same. The average rises drastically for concurrent users. Hence it can be said that it highly affected by concurrency.

The type of operation does not matter in any case as all operations produce nearly same average.

Part 2 - Application

Introduction

This project aims to implement the major functionalities of *Canvas* application which is a web portal accessible to students and the faculty of San Jose State University. It will basically be a virtual classroom involving process like basic sign up, searching and enrolling courses, submitting quiz and assignments, grading of the same, making announcements, thus connecting the student and faculty's world to a common accessible platform.

System Design

The application is made to revolve around two types of its users which are faculty and student. Detailed information and functionalities can be inferred from the following: -

Assumptions

- Every user must have a unique SJSU ID prior to using this application.
- All input fields are validated.
- Profile image can be updated by an image URL.
- Passwords are encrypted using bcrypt.
- Permission code is based on timestamp so that it is unique.

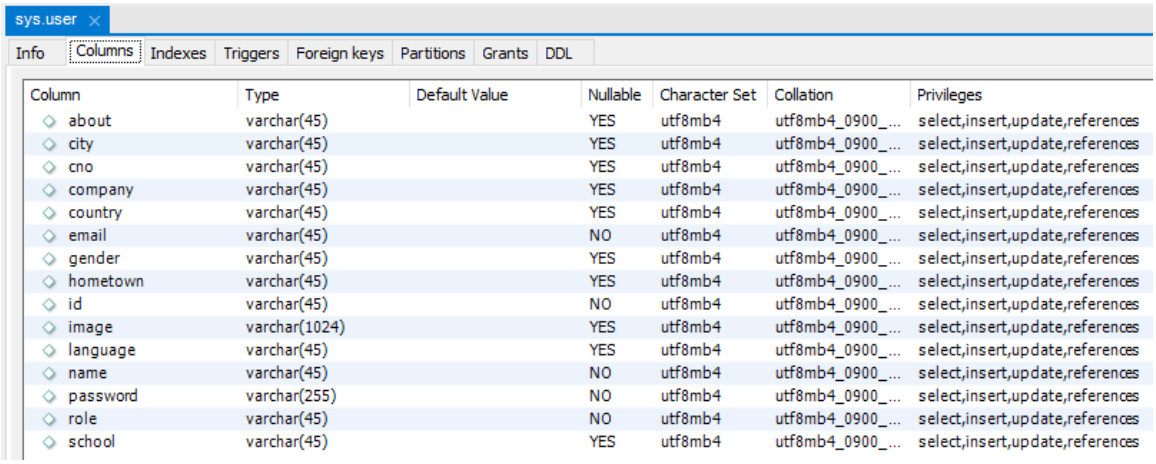
Functionalities

- Setting up and maintaining user account and profile information.
- Faculty can create a course.
- Student can search among all the available courses.
- Student can enroll, waitlist or drop from a course.
- User can view students enrolled in a class.
- Faculty can remove a student from a class; generate unique permission code for waitlisted students.
- Faculty can make and view announcement for corresponding course; students can view them.
- Faculties can create and view quiz. Students can take the quiz and have their grades displayed under Grades.
- Faculty can upload lecture notes as files, which the student can download.

- Faculty can upload assignments. Student can submit the assignment which will be graded by Faculty and the score will be displayed under Grades.

Database Design

User

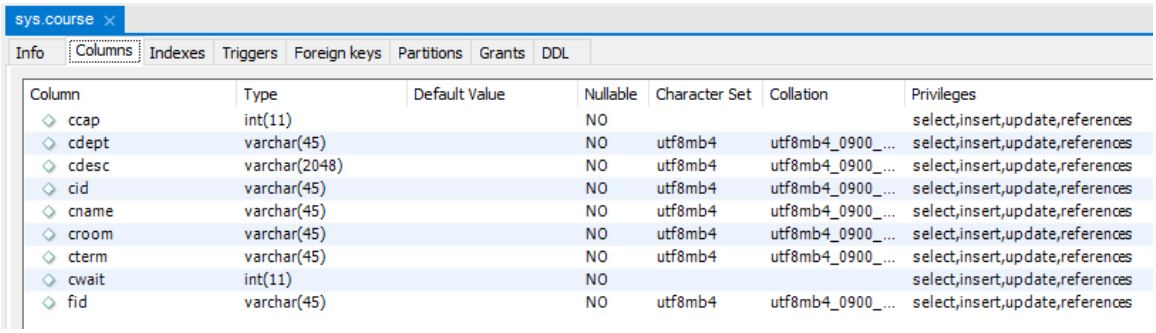


sys.user

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
about	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
city	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cno	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
company	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
country	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
email	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
gender	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
hometown	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
id	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
image	varchar(1024)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
language	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
name	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
password	varchar(255)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
role	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
school	varchar(45)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references

Course

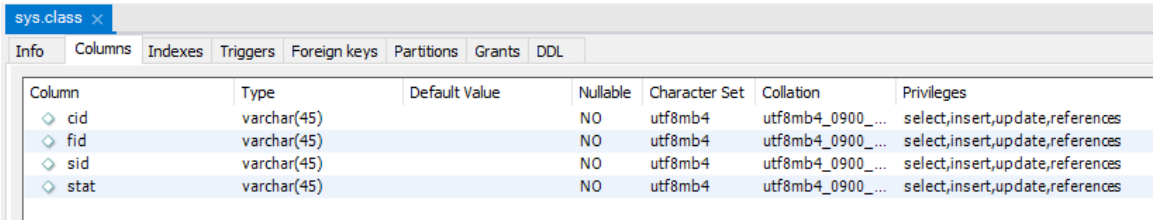


sys.course

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
ccap	int(11)		NO			select,insert,update,references
cdept	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cdesc	varchar(2048)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cname	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
croom	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cterm	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cwait	int(11)		NO			select,insert,update,references
fid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references

Class



sys.class

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
cid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
fid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
sid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
stat	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references

Announcement

sys.announcement

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
adesc	varchar(2048)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
aname	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
atime	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references

Quiz

sys.quiz

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
cid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cor1	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cor2	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
d1	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
d2	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
op11	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
op12	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
op13	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
op14	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
op21	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
op22	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
op23	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
op24	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
q1	varchar(255)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
q2	varchar(255)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
qid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
qname	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references

Grade

sys.grade

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
cid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
grade	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
sid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
typeid	varchar(255)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
typeof	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references

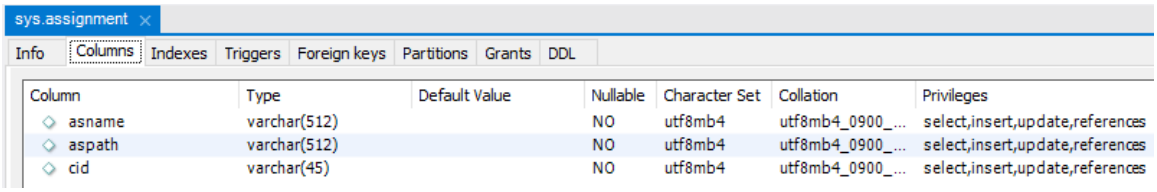
Lecture

sys.lecture

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
cid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
fname	varchar(512)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
fpath	varchar(512)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references

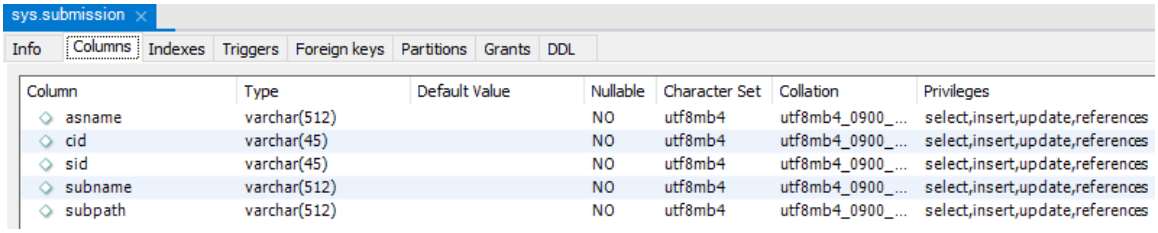
Assignment


sys.assignment

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
asname	varchar(512)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
aspath	varchar(512)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references

Submission


sys.submission

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
asname	varchar(512)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
cid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
sid	varchar(45)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
subname	varchar(512)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references
subpath	varchar(512)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references

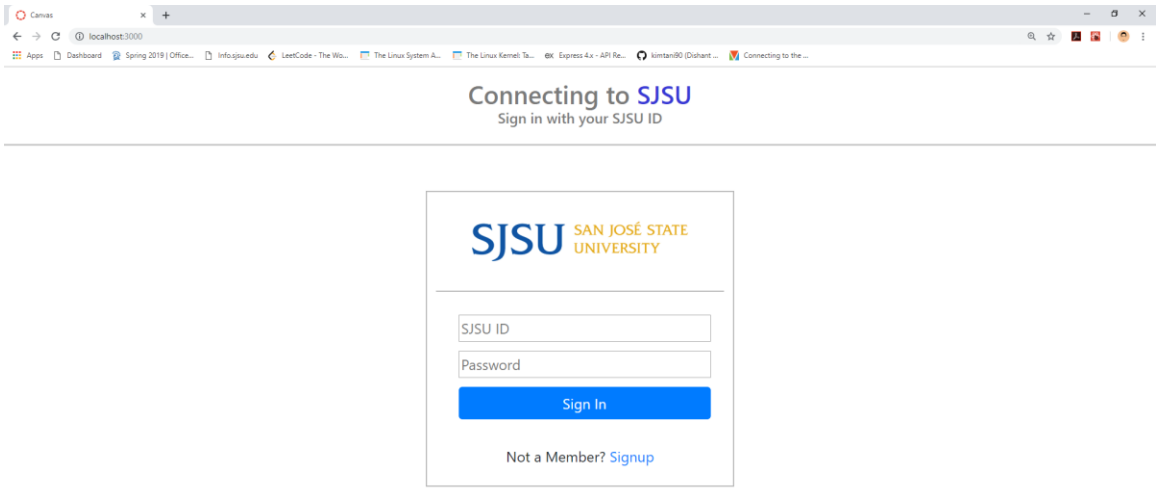
Technologies

Software: HTML, CSS, Javascript, Node, React, MySQL, Material UI, Bootstrap, Font Awesome.

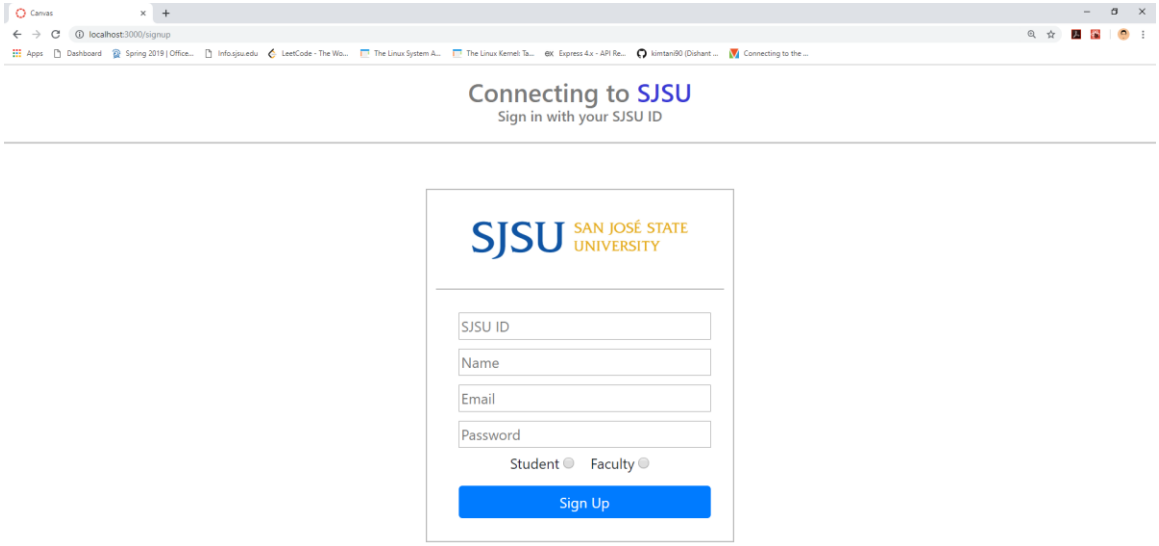
Hardware: Any computer having Chrome along with required packages installed.

Results

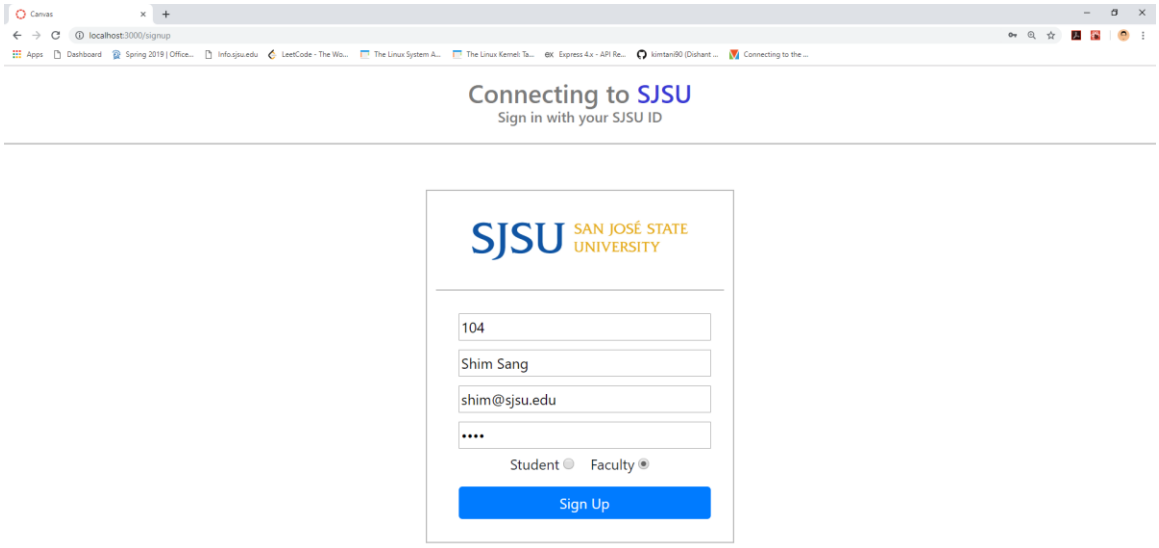
Login page:



Signup page:

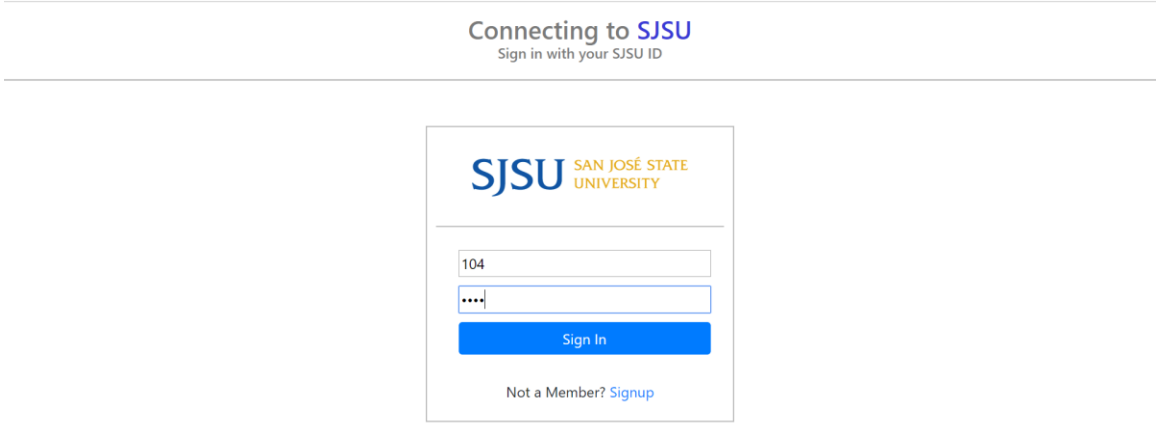


Faculty signup:



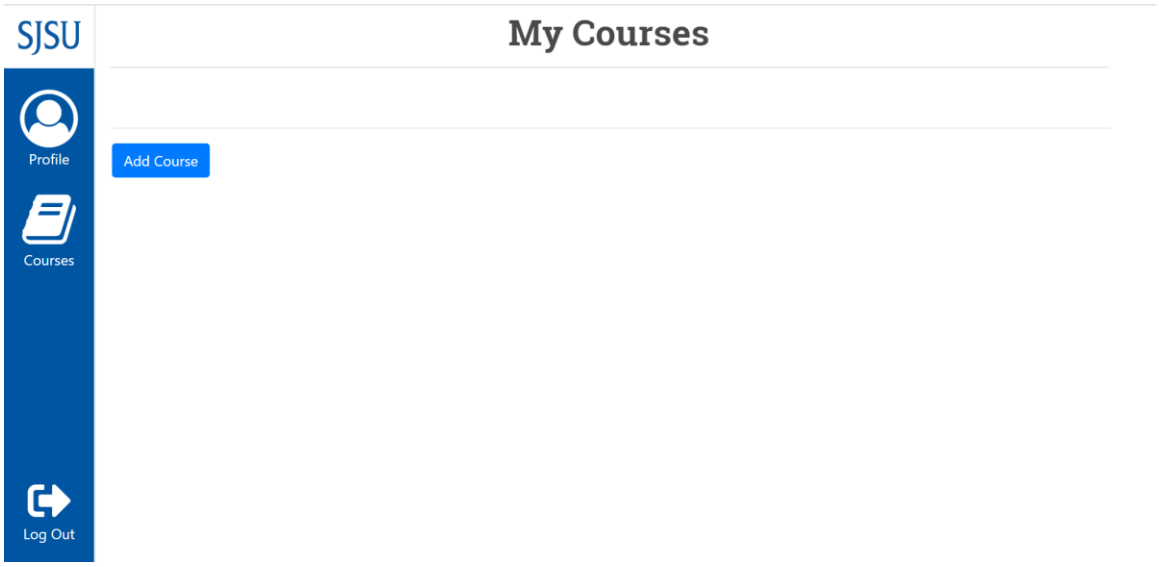

```
Server started on port 3001
Database connection successful.
104 Shim Sang shim@sjsu.edu $2b$10$68YT.9U9XAyVLypa.GkssOrUhO73ofGGA2GA4ETIUnouAhAHeTuNm faculty
Row affected: 1
```

Faculty login:

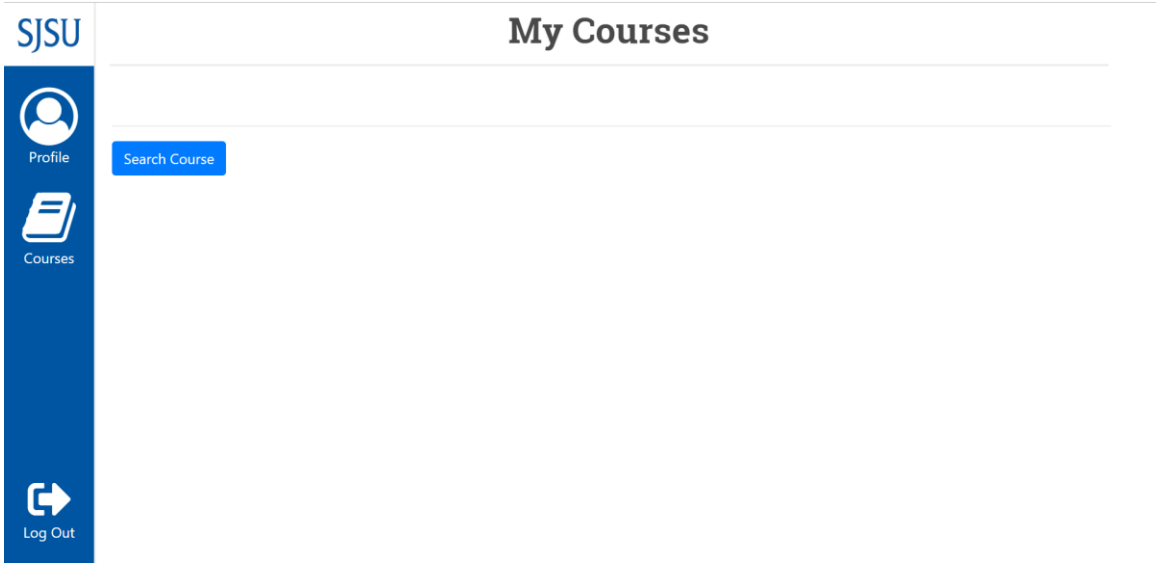



```
Server started on port 3001
Database connection successful.
[ RowDataPacket {
    password:
     '$2b$10$68YT.9U9XAyVLypa.GkssOrUhO73ofGGA2GA4ETIUnouAhAHeTuNm',
    role: 'faculty' } ]
```

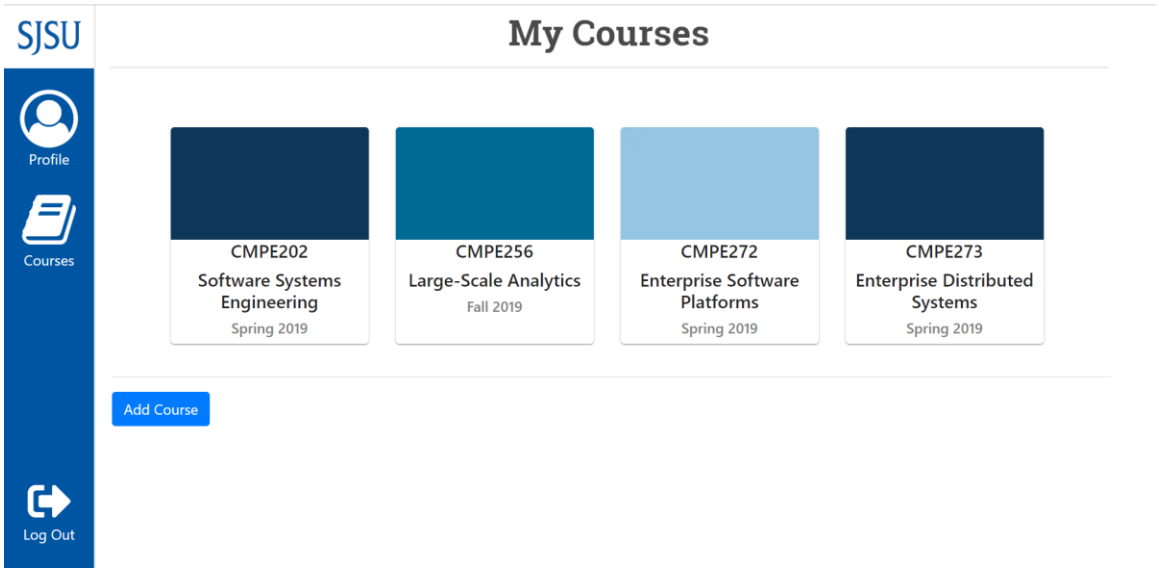
Faculty homepage:



Student homepage:

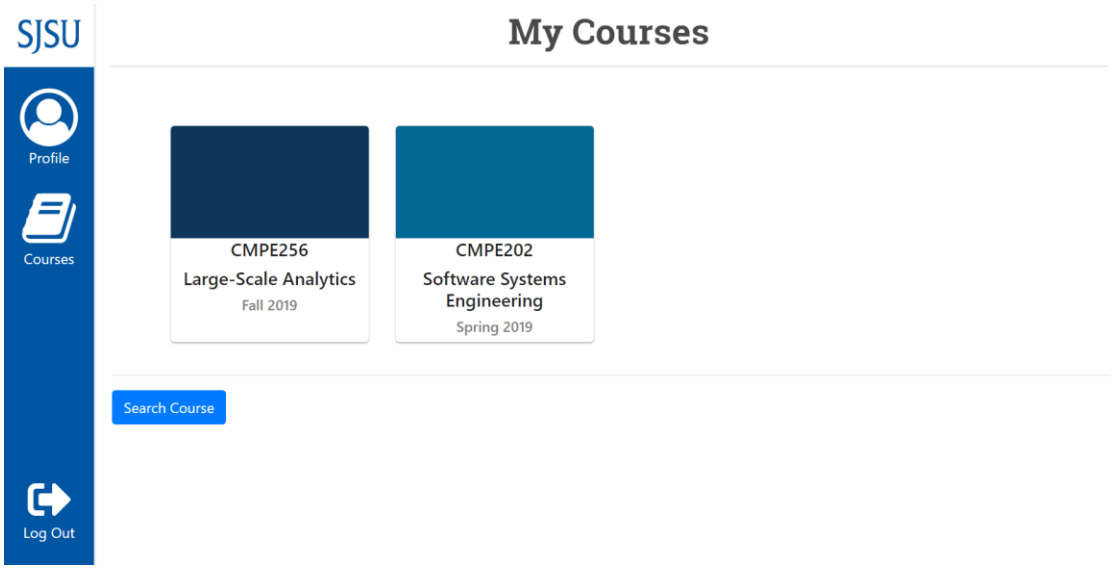


Faculty homepage after several courses are created by them:



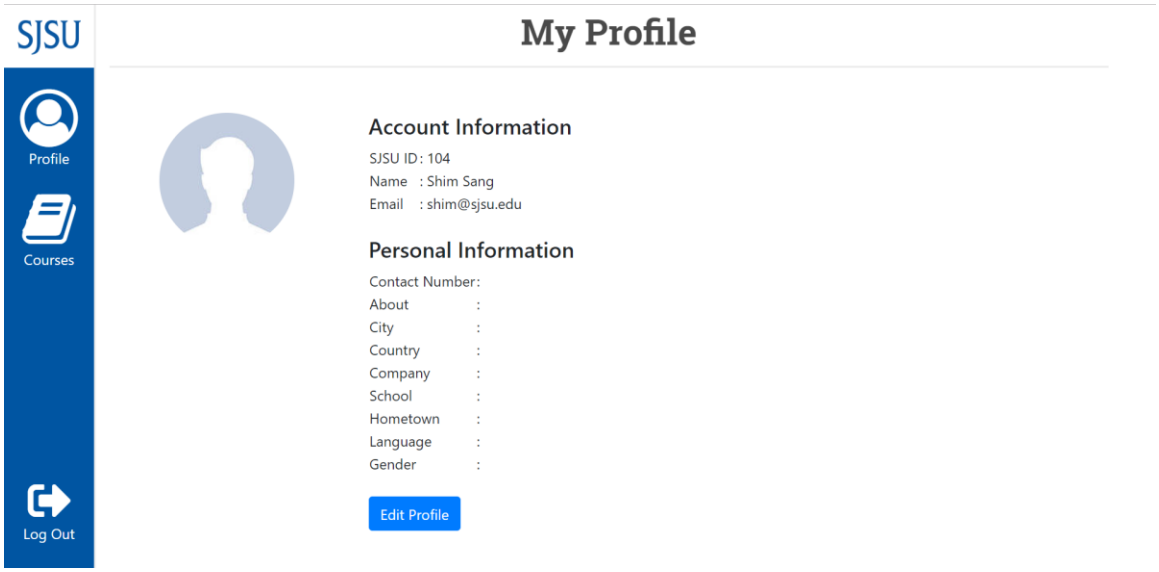

```
[ RowDataPacket {
    cid: 'CMPE202',
    cname: 'Software Systems Engineering',
    cterm: 'Spring 2019' },
  RowDataPacket {
    cid: 'CMPE256',
    cname: 'Large-Scale Analytics',
    cterm: 'Fall 2019' },
  RowDataPacket {
    cid: 'CMPE272',
    cname: 'Enterprise Software Platforms',
    cterm: 'Spring 2019' },
  RowDataPacket {
    cid: 'CMPE273',
    cname: 'Enterprise Distributed Systems',
    cterm: 'Spring 2019' } ]
```

Student homepage after enrolling/wait listing for course:



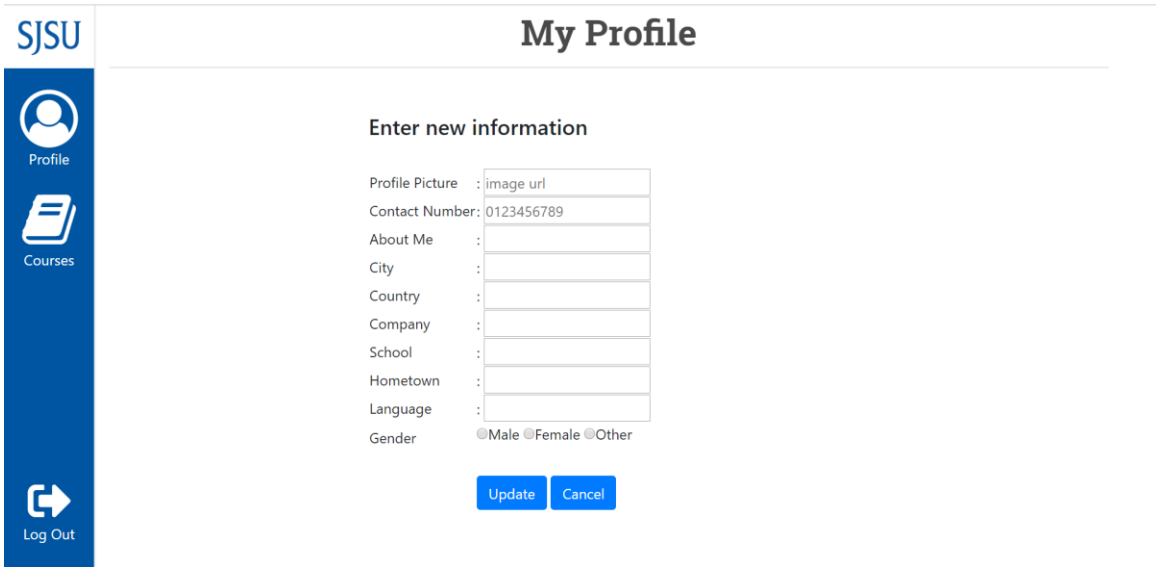

```
[ RowDataPacket {
    cid: 'CMPE256',
    cname: 'Large-Scale Analytics',
    cterm: 'Fall 2019' },
  RowDataPacket {
    cid: 'CMPE202',
    cname: 'Software Systems Engineering',
    cterm: 'Spring 2019' } ]
```

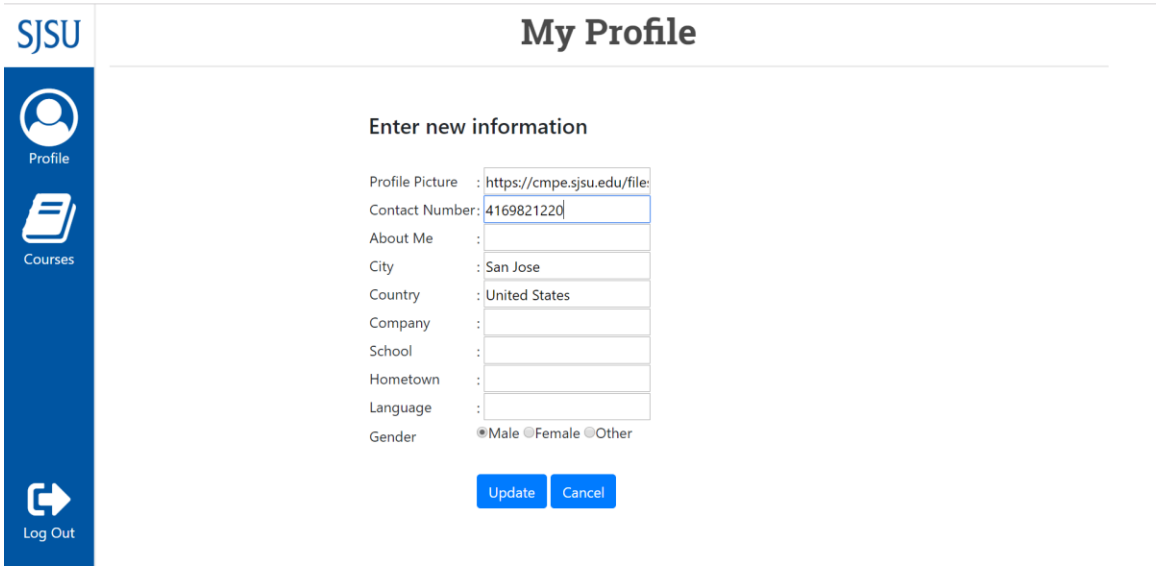
User Profile:



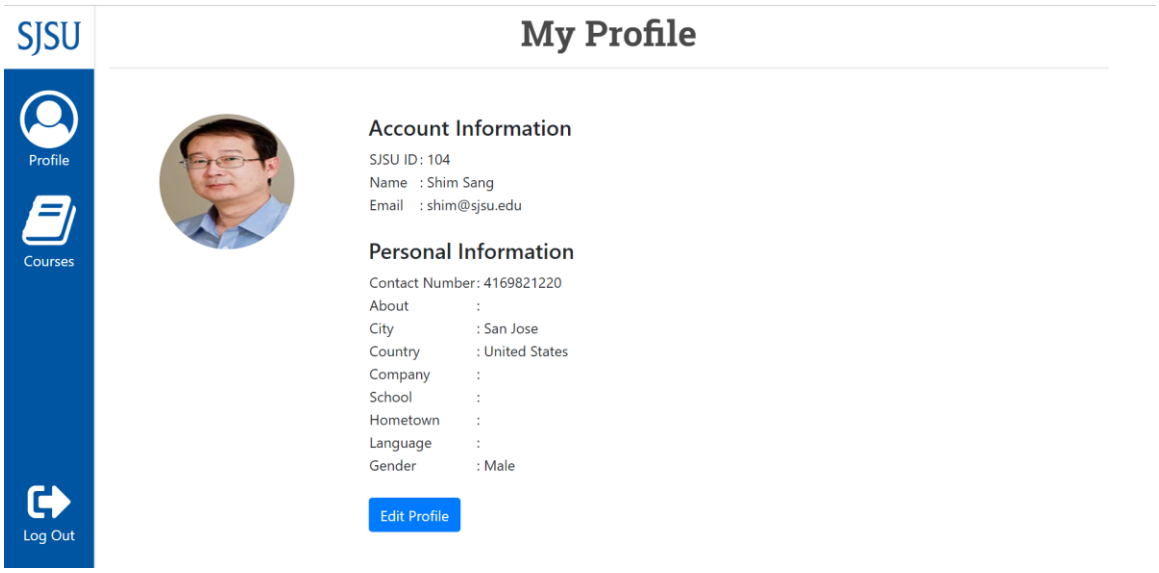

```
RowDataPacket {
  id: '104',
  name: 'Shim Sang',
  email: 'shim@sjsu.edu',
  password:
   '$2b$10$68YT.9U9XAyVLypa.GkssOrUhO73ofGGA2GA4ETIUnouAhAHeTuNm',
  cno: null,
  city: null,
  country: null,
  company: null,
  school: null,
  hometown: null,
  gender: null,
  about: null,
  role: 'faculty',
  image: null,
  language: null }
```

Profile edit page:



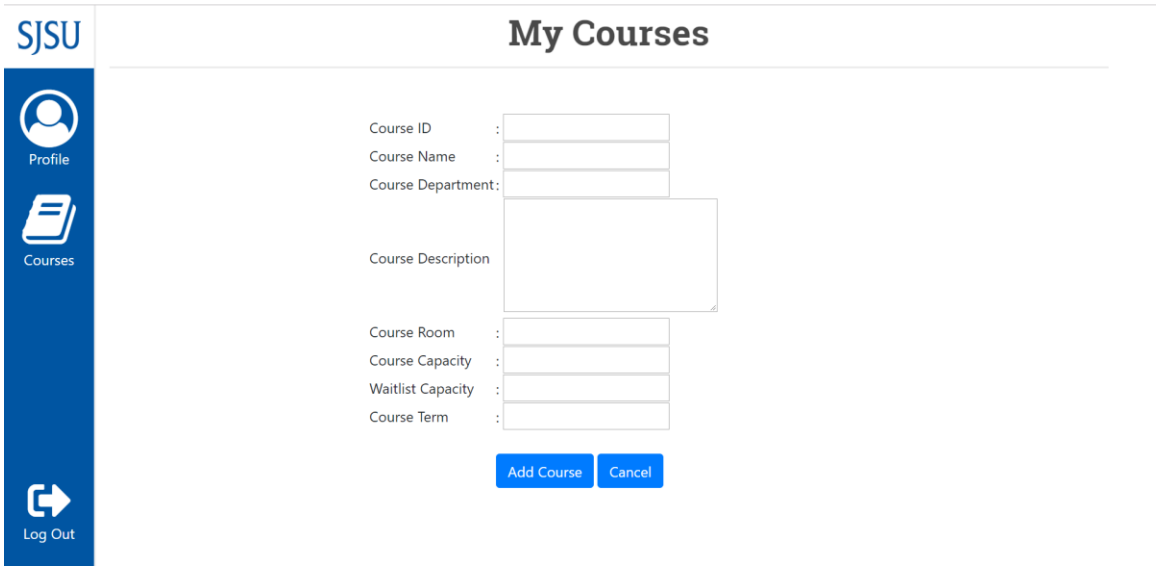


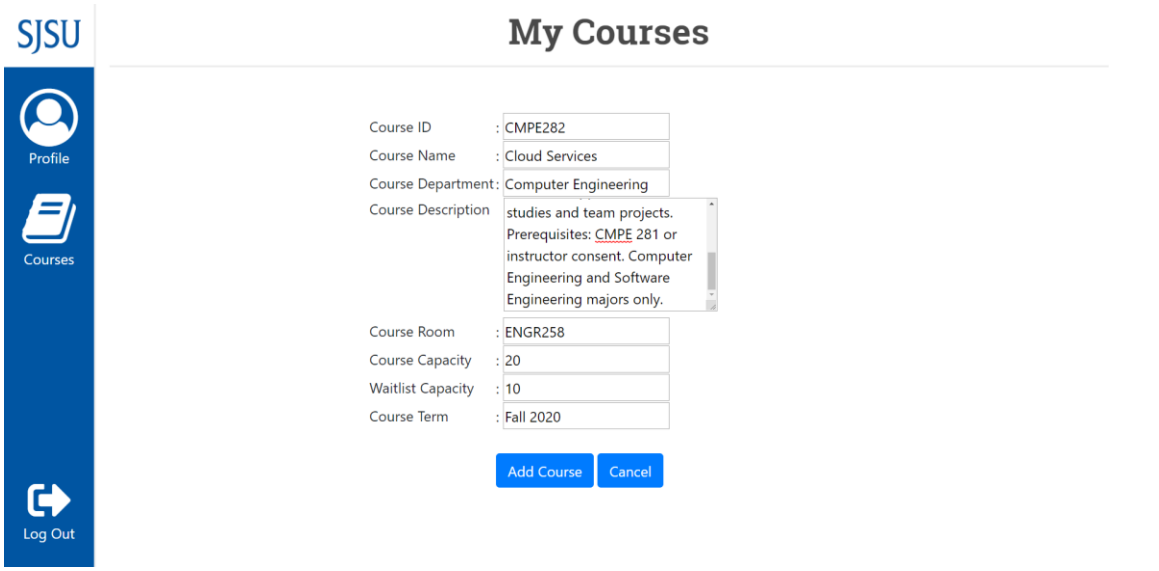

```
https://cmpe.sjsu.edu/files/public/styles/mugshot/public/shim-simon.jpg?itok=lITTcFqp 4169821220  San Jose United States     Male
Rows affected: 1
```

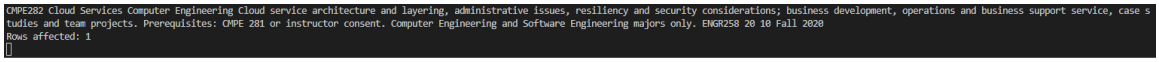



```
RowDataPacket {
  id: '104',
  name: 'Shim Sang',
  email: 'shim@sjsu.edu',
  password:
   '$2b$10$68YT.9U9XAyVLypa.GkssOrUhO73ofGGA2GA4ETIUnouAhAHeTuNm',
  cno: '4169821220',
  city: 'San Jose',
  country: 'United States',
  company: '',
  school: '',
  hometown: '',
  gender: 'Male',
  about: '',
  role: 'faculty',
  image:
   'https://cmpe.sjsu.edu/files/public/styles/mugshot/public/shim-simon.jpg?itok=lITTcFqp',
  language: '' }
```

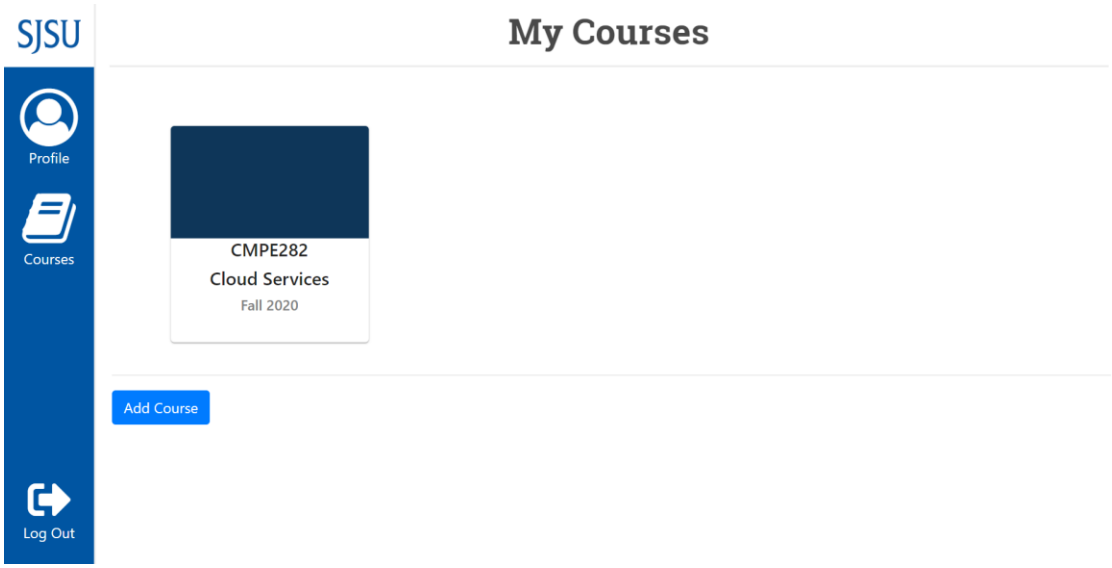
Course creation by faculty:





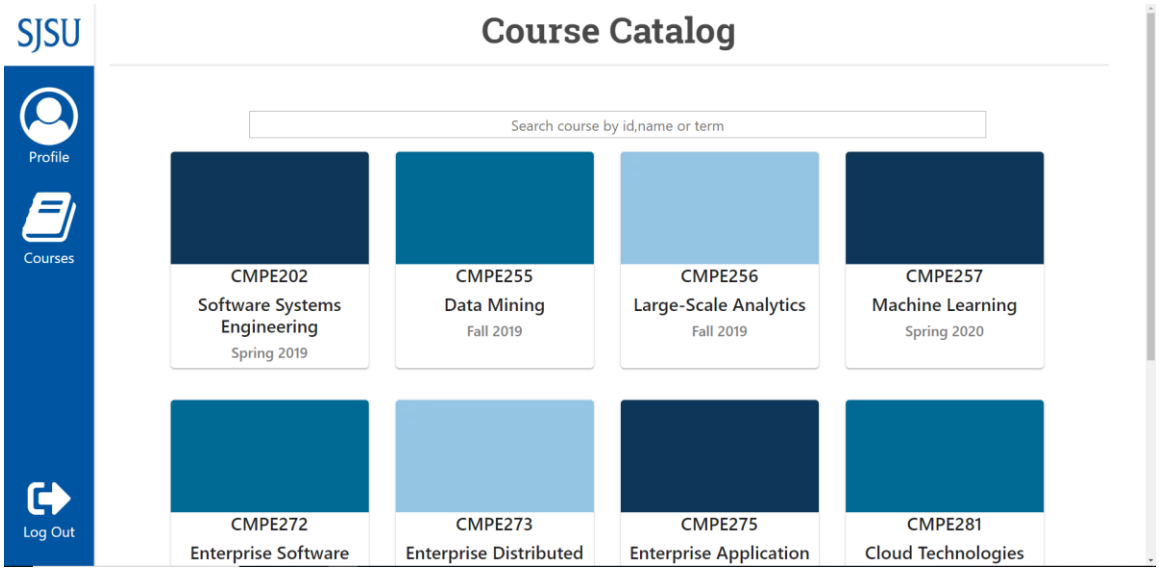


Navigation to home page after creating course:

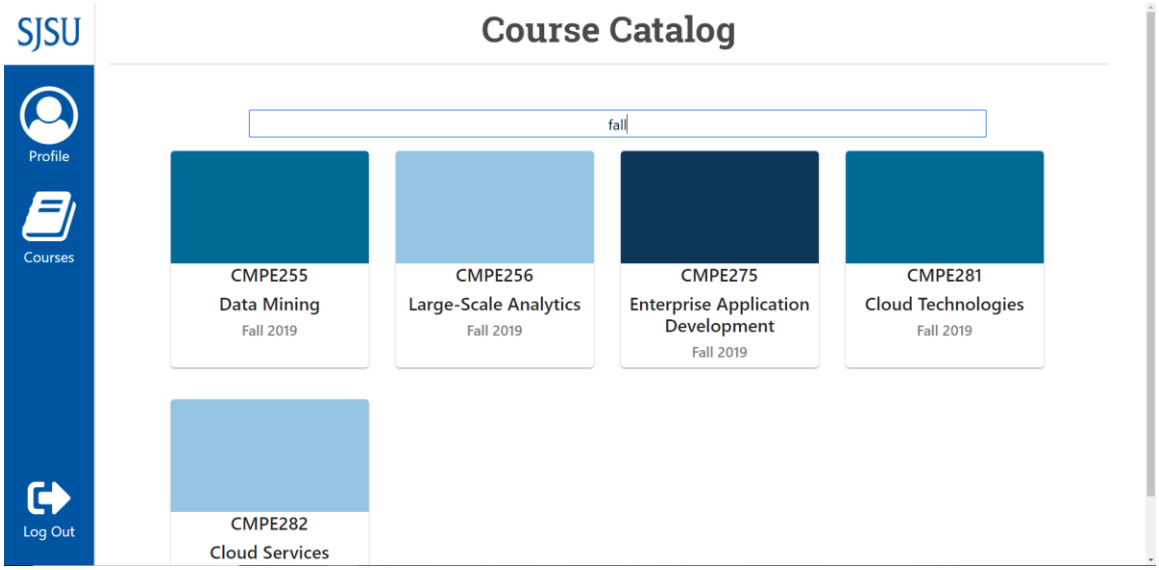


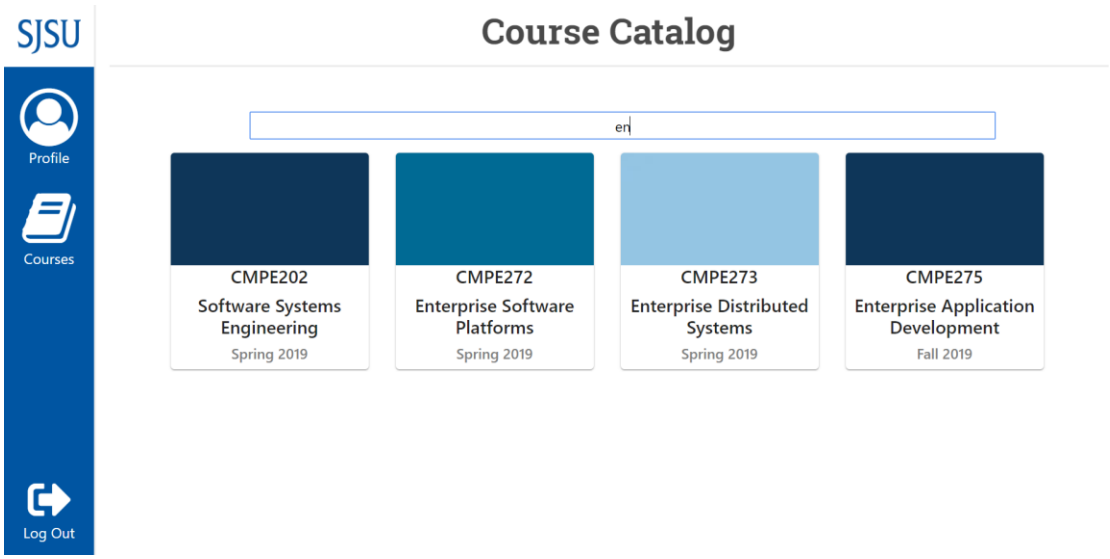

```
[ RowDataPacket { cid: 'CMPE282', cname: 'Cloud Services', cterm: 'Fall 2020' } ]
```

Course searching by student:



```
[ RowDataPacket {
    cid: 'CMPE202',
    cname: 'Software Systems Engineering',
    cterm: 'Spring 2019' },
  RowDataPacket { cid: 'CMPE255', cname: 'Data Mining', cterm: 'Fall 2019' },
  RowDataPacket {
    cid: 'CMPE256',
    cname: 'Large-Scale Analytics',
    cterm: 'Fall 2019' },
  RowDataPacket {
    cid: 'CMPE257',
    cname: 'Machine Learning',
    cterm: 'Spring 2020' },
  RowDataPacket {
    cid: 'CMPE272',
    cname: 'Enterprise Software Platforms',
    cterm: 'Spring 2019' },
  RowDataPacket {
    cid: 'CMPE273',
    cname: 'Enterprise Distributed Systems',
    cterm: 'Spring 2019' },
  RowDataPacket {
    cid: 'CMPE275',
    cname: 'Enterprise Application Development',
    cterm: 'Fall 2019' },
  RowDataPacket {
    cid: 'CMPE281',
    cname: 'Cloud Technologies',
    cterm: 'Fall 2019' },
  RowDataPacket { cid: 'CMPE282', cname: 'Cloud Services', cterm: 'Fall 2020' } ]
```

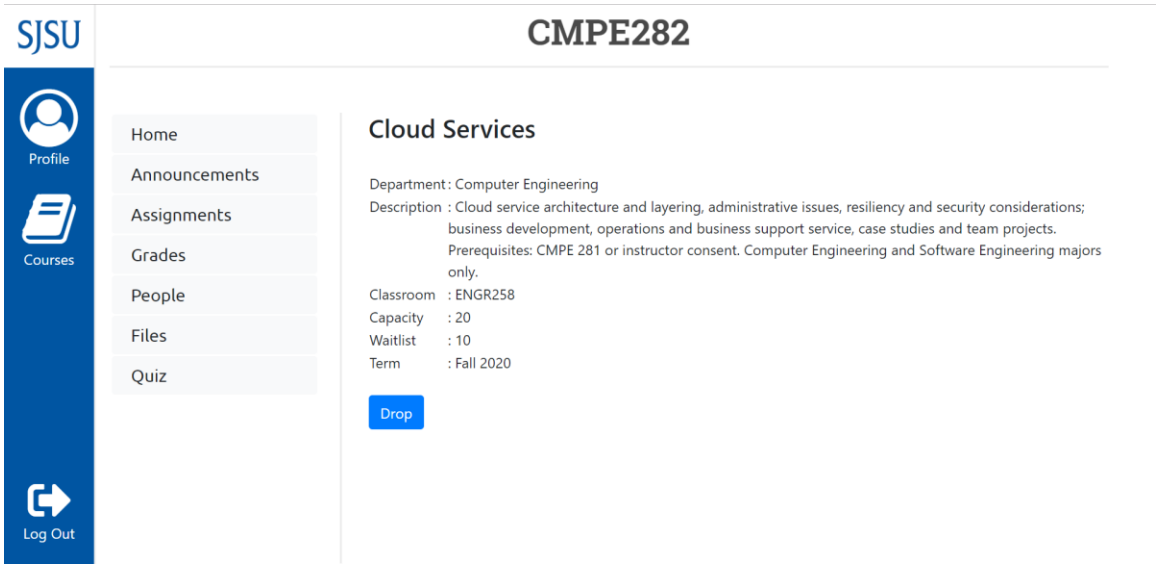


Course Homepage for faculty:

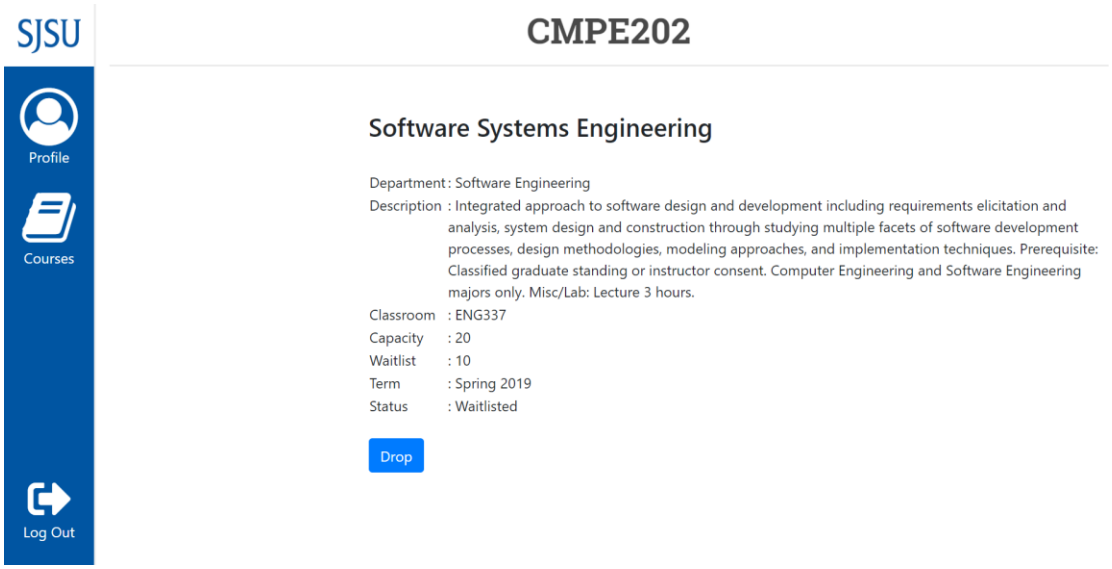
Course Homepage for student, having the option to enroll/wait-list:


```
[ RowDataPacket {
    cid: 'CMPE282',
    fid: '104',
    cname: 'Cloud Services',
    cdept: 'Computer Engineering',
    cdesc:
     'Cloud service architecture and layering, administrative issues, resiliency and security considerations; business development, operations and business support service, case studies and team projects. Prerequisite
s: CMPE 281 or instructor consent. Computer Engineering and Software Engineering majors only.',
    croom: 'ENGR258',
    ccap: 20,
    cwait: 10,
    cterm: 'Fall 2020' } ]
```


Student course view if enrolled:

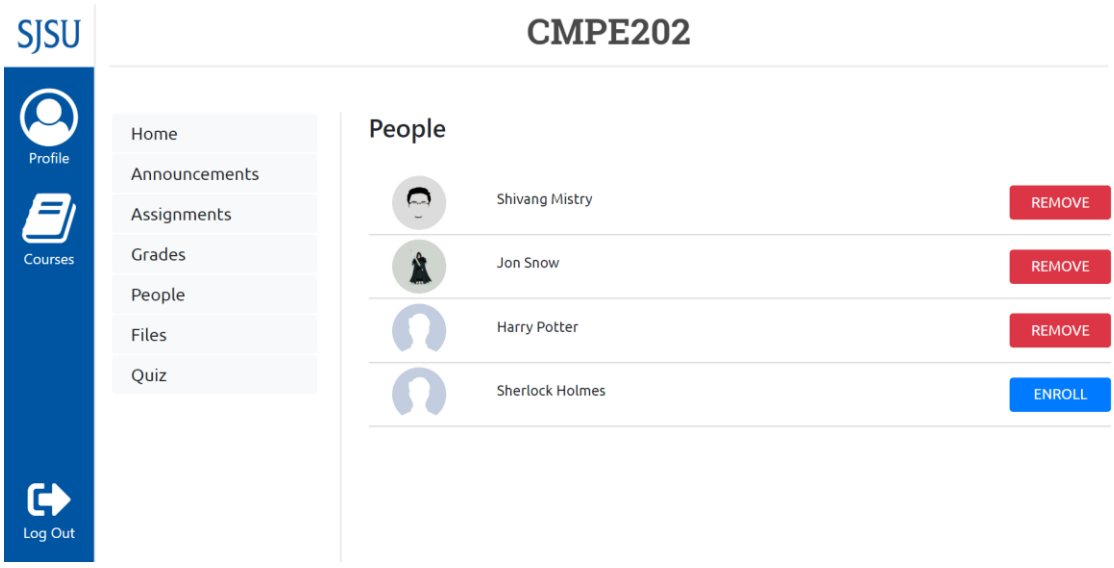


Student course view if waitlisted:

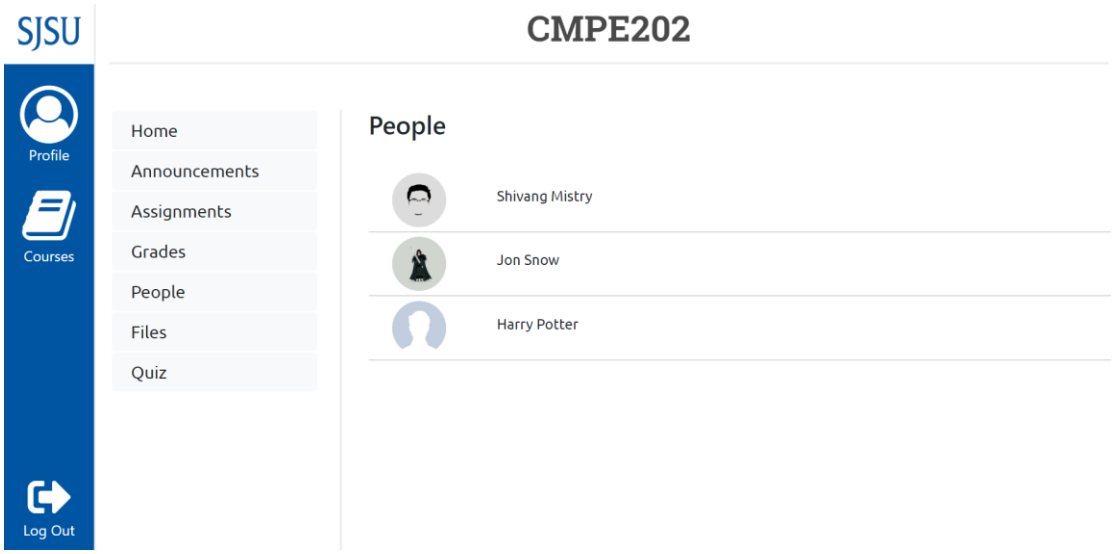


Enrolled/ wait-listed course will be shown on student's homepage in form of cards as shown before.

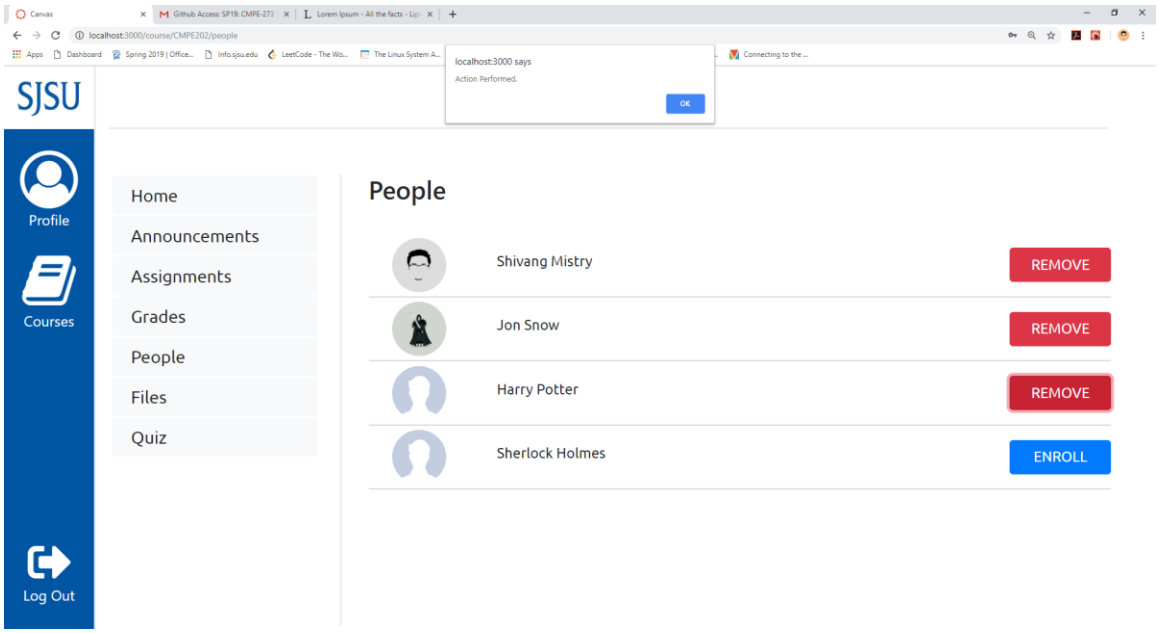
People view for faculty:



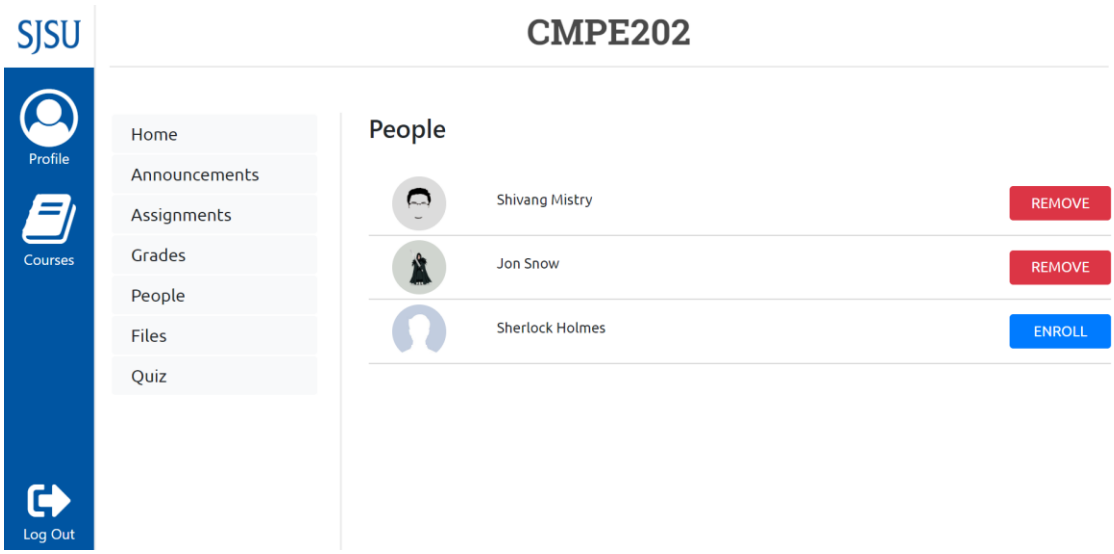
People view for student:



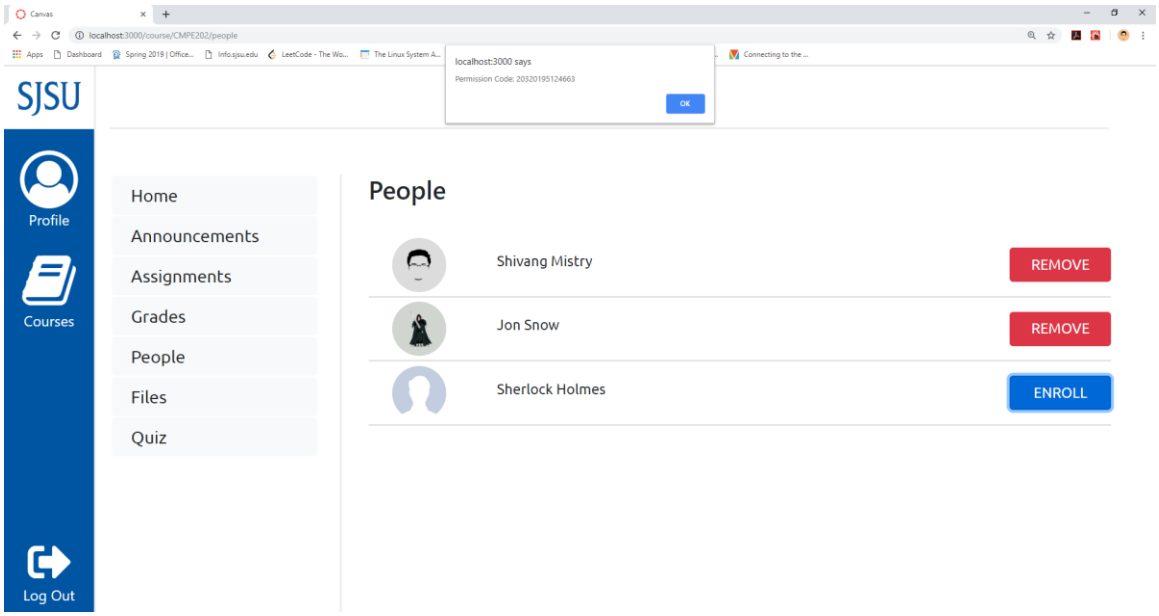
Faculty removing a student (Harry Potter):



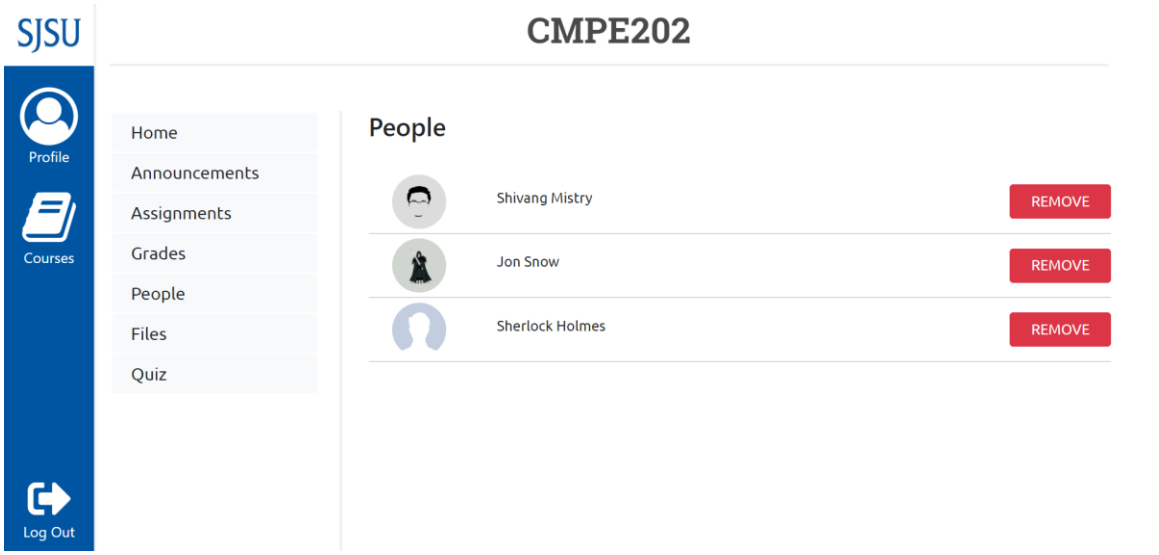

```
 RowDataPacket { id: '202', name: 'Harry Potter', image: '', stat: 'enroll' },
 RowDataPacket {
   id: '204',
   name: 'Sherlock Holmes',
   image: null,
   stat: 'waitlist' } ]
kPacket {
 fieldCount: 0,
 affectedRows: 1,
 insertId: 0,
 serverStatus: 2,
 warningCount: 0,
 message: '',
 protocol41: true,
 changedRows: 0 }
```



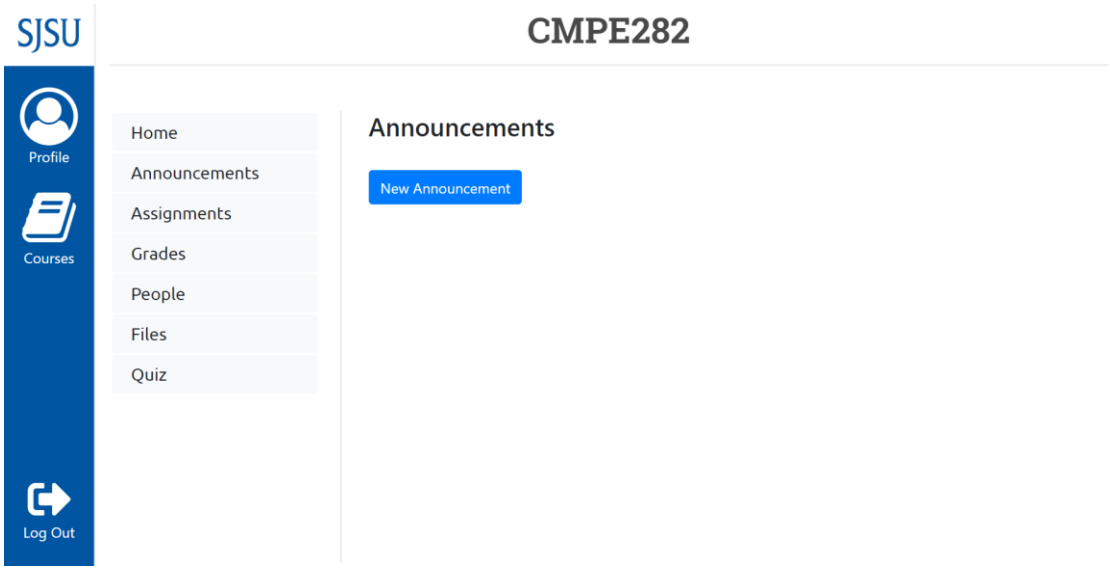
Faculty generating permission code for waitlisted student (Sherlock Holmes):



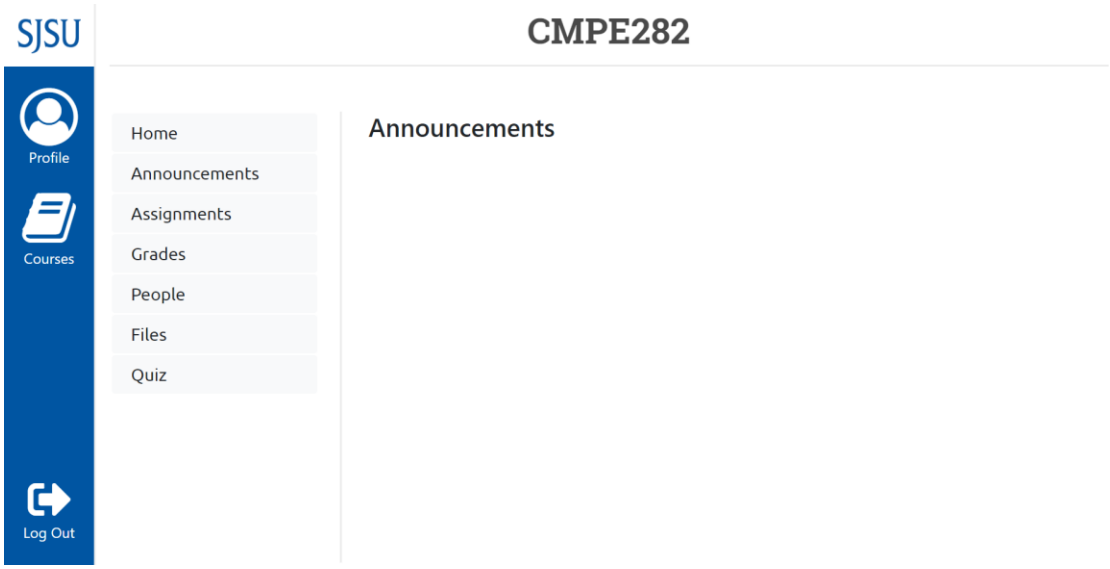

```
  RowDataPacket {
    id: '204',
    name: 'Sherlock Holmes',
    image: null,
    stat: 'waitlist' } ]
OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '(Rows matched: 1  Changed: 1  Warnings: 0',
  protocol41: true,
  changedRows: 1 }
```



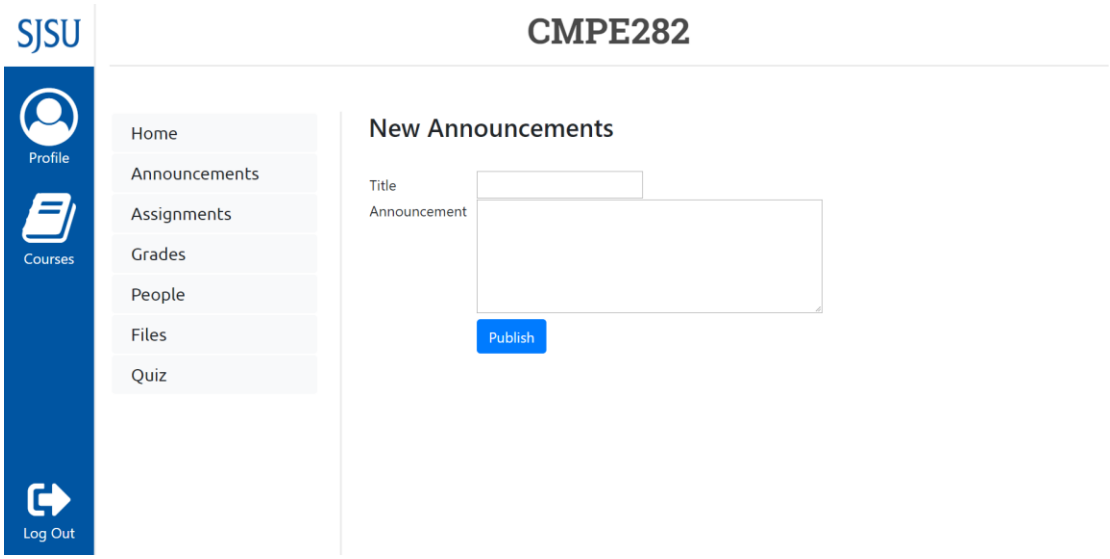
Faculty announcement view, initially:

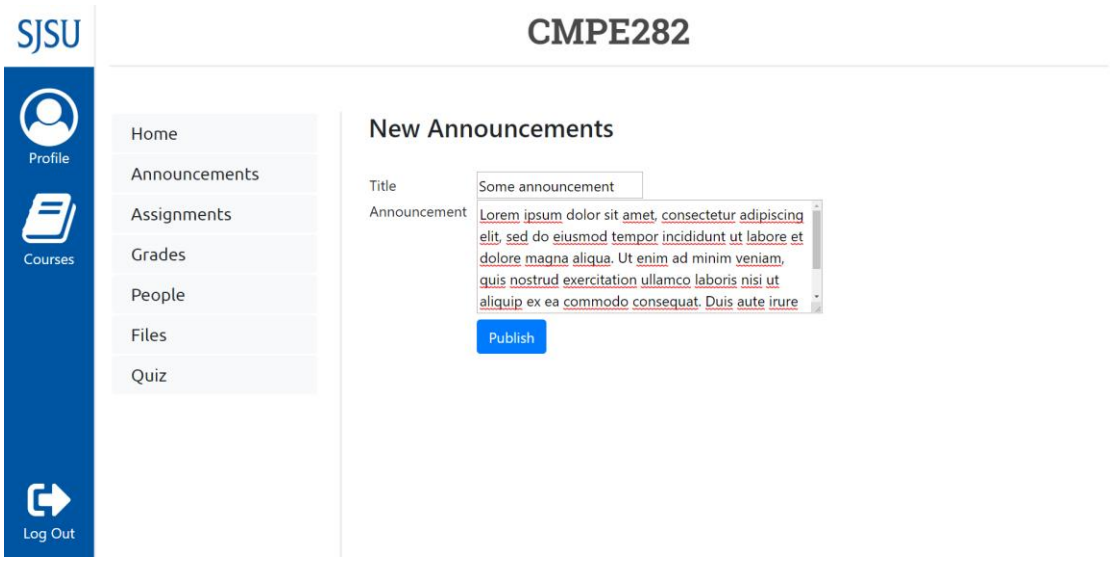


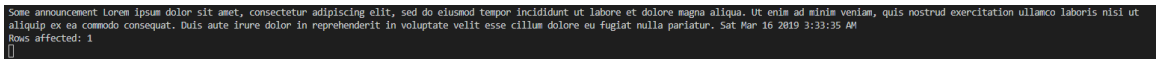
Student announcement view, initially:

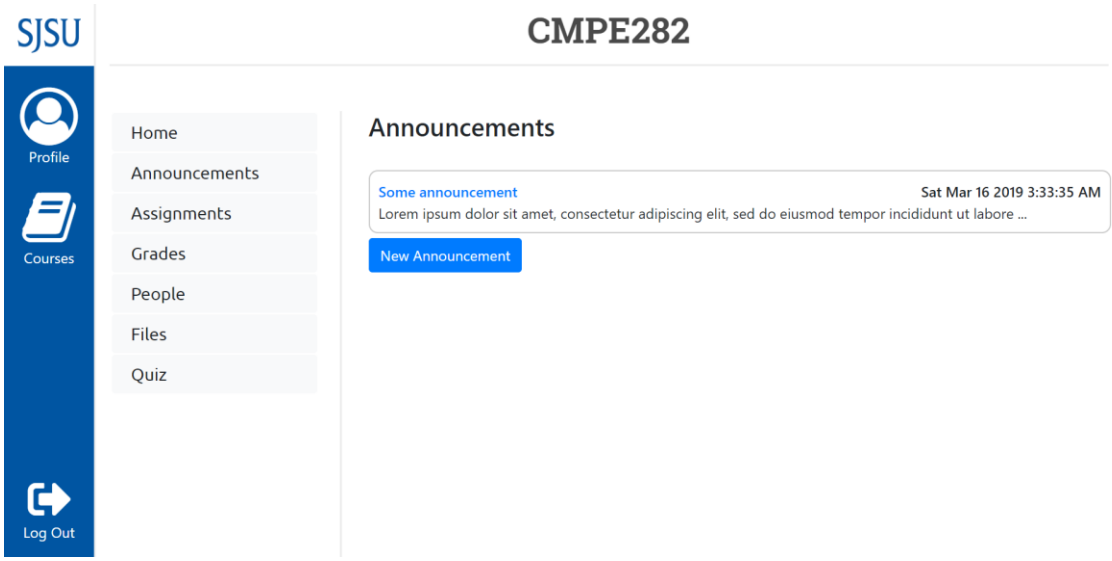


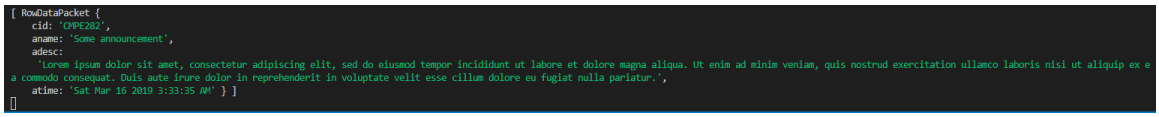
Faculty adding announcement:


SJSU
CMPE282
Profile
Courses
Log Out
Home
Announcements
Assignments
Grades
People
Files
Quiz
New Announcements
Title
Announcement
Publish

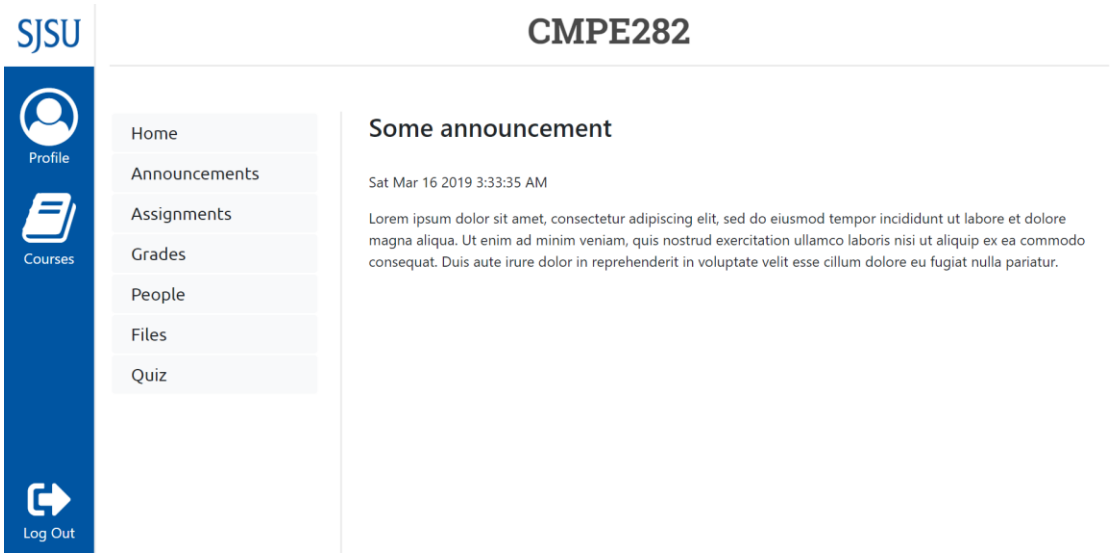

SJSU
CMPE282
Profile
Courses
Log Out
Home
Announcements
Assignments
Grades
People
Files
Quiz
New Announcements
Title
Some announcement
Announcement
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
Publish


Some announcement Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Sat Mar 16 2019 3:33:35 AM
Rows affected: 1

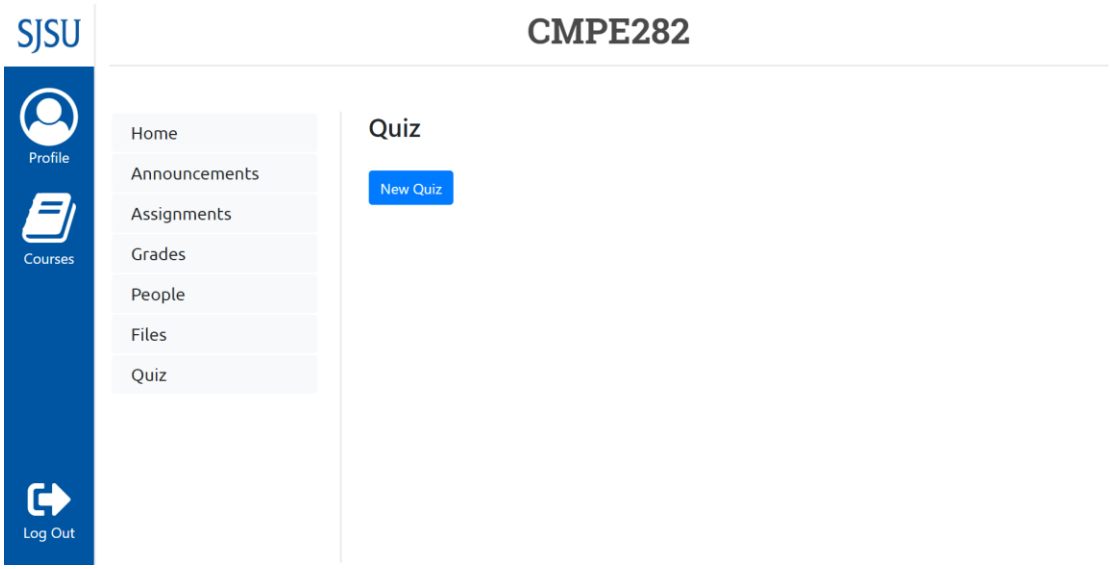




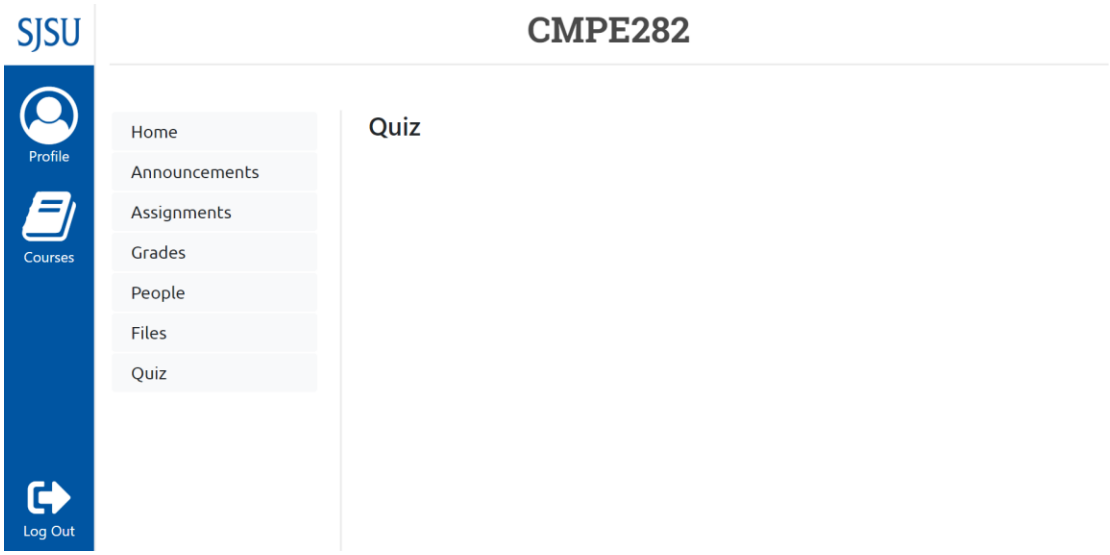
Opening an announcement:



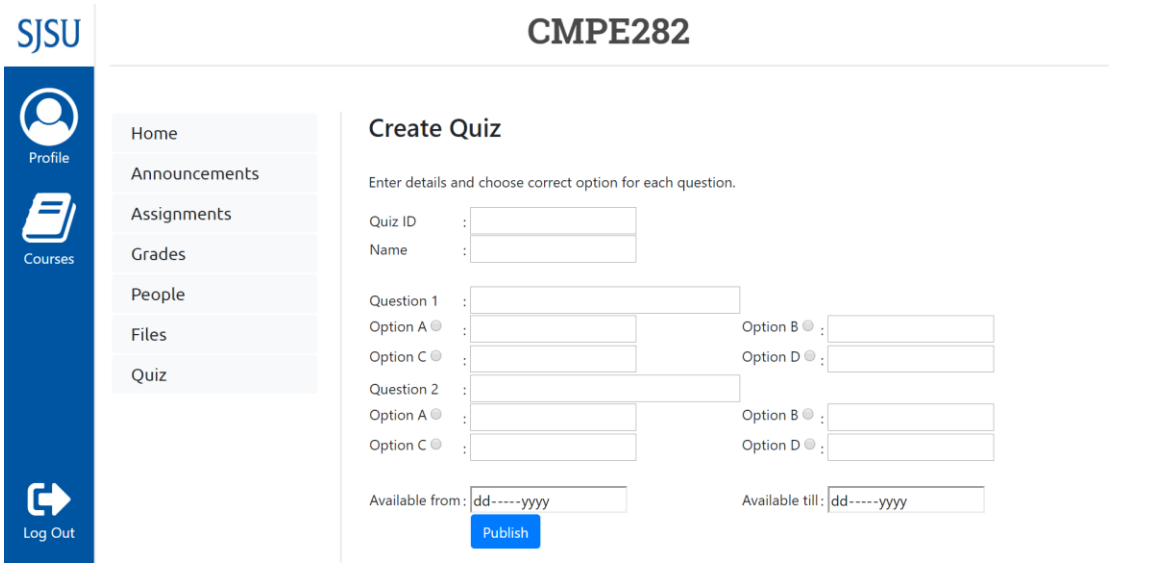
Faculty quiz view initially:

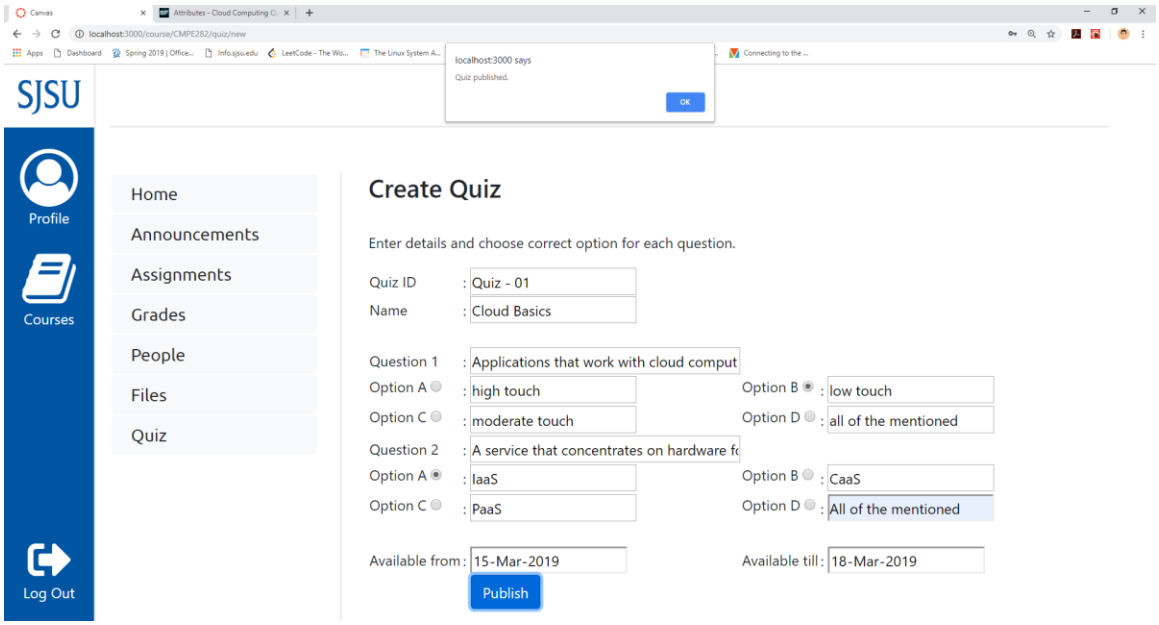


Student Quiz view initially:



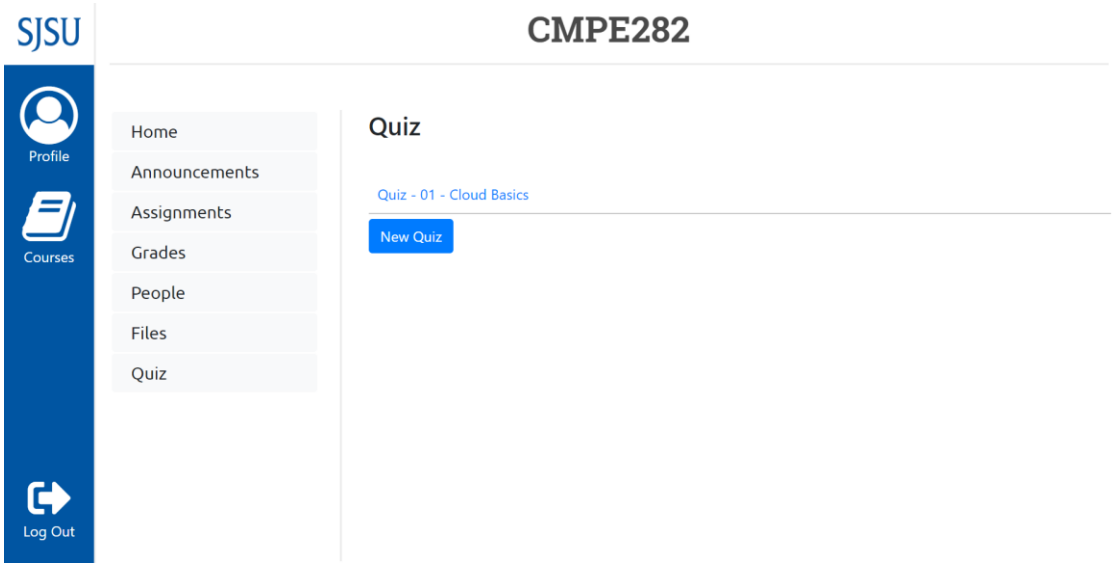
Faculty creating quiz:



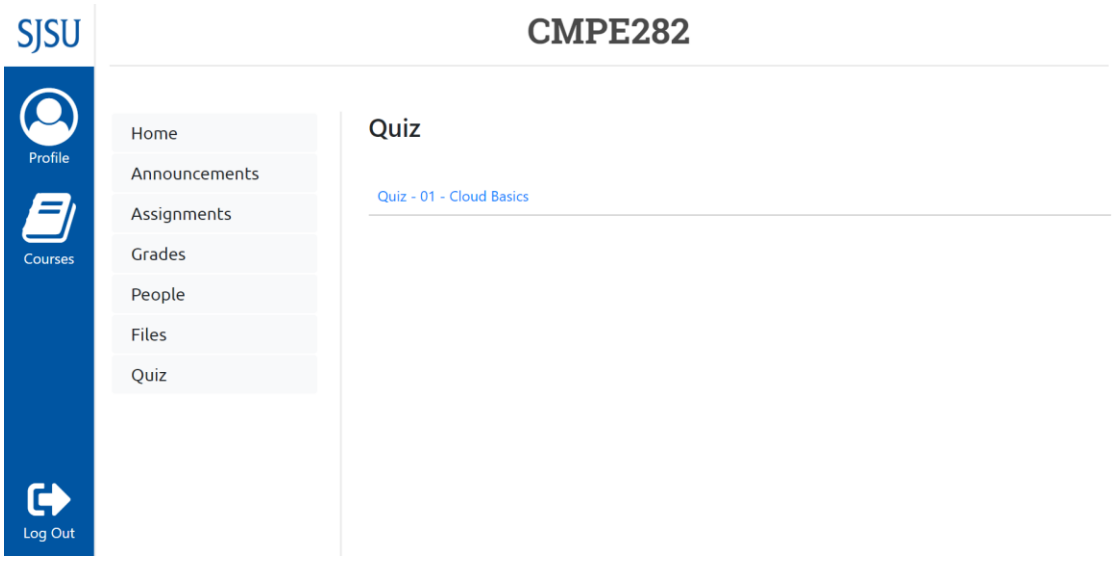



```
OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0 }
```

Faculty view:

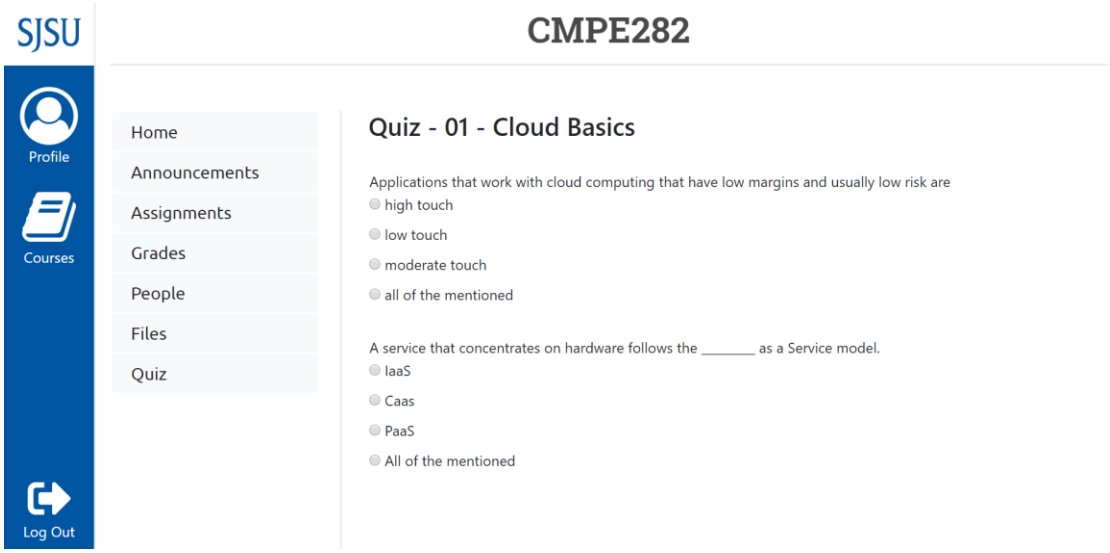


Student view:



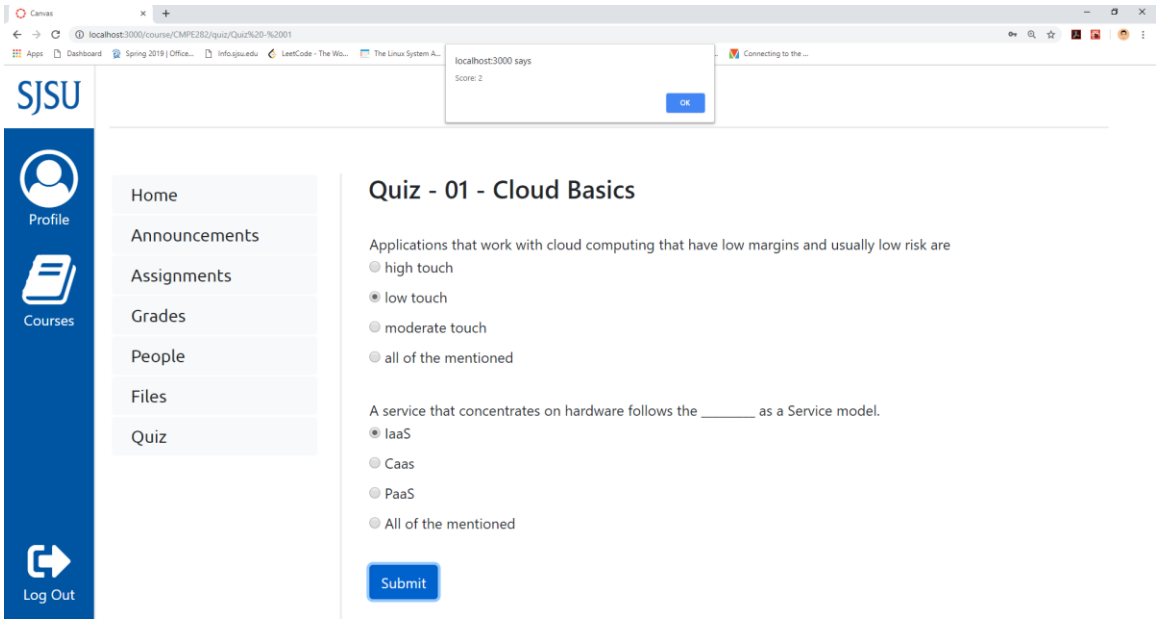

```
[ RowDataPacket { qid: 'Quiz - 01', qname: 'Cloud Basics' } ]
```

Faculty quiz info:



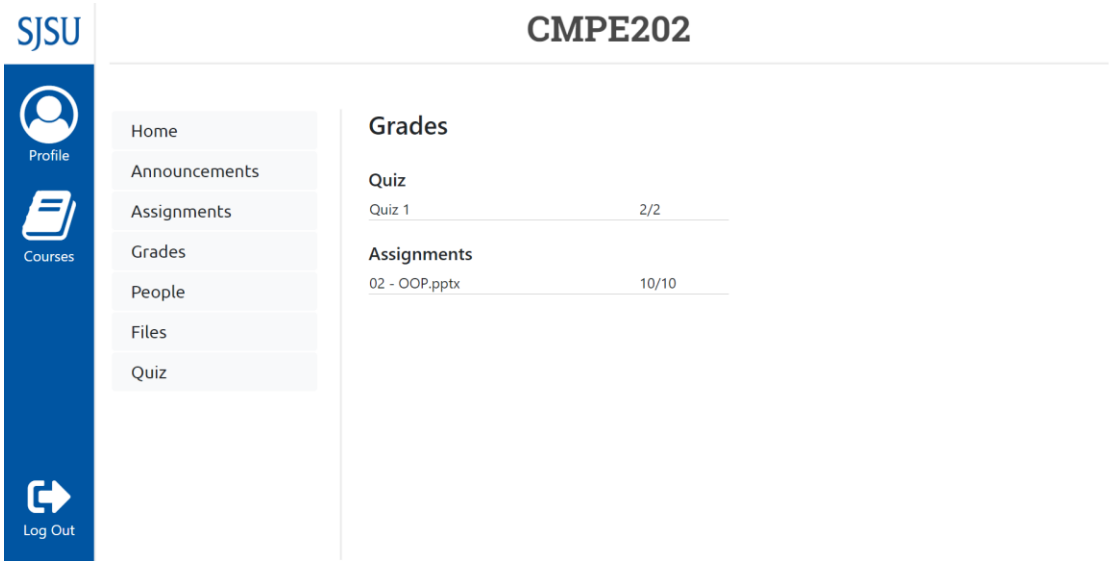

```
[ RowDataPacket {
    cid: 'CMPE282',
    qid: 'Quiz - 01',
    qname: 'Cloud Basics',
    q1:
     'Applications that work with cloud computing that have low margins and usually low risk are',
    op11: 'high touch',
    op12: 'low touch',
    op13: 'moderate touch',
    op14: 'all of the mentioned',
    cor1: 'b',
    q2:
     'A service that concentrates on hardware follows the _____ as a Service model.',
    op21: 'IaaS',
    op22: 'Caas',
    op23: 'PaaS',
    op24: 'All of the mentioned',
    cor2: 'a',
    d1: '2019-03-15',
    d2: '2019-03-18' } ]
```

Student quiz info and submission:



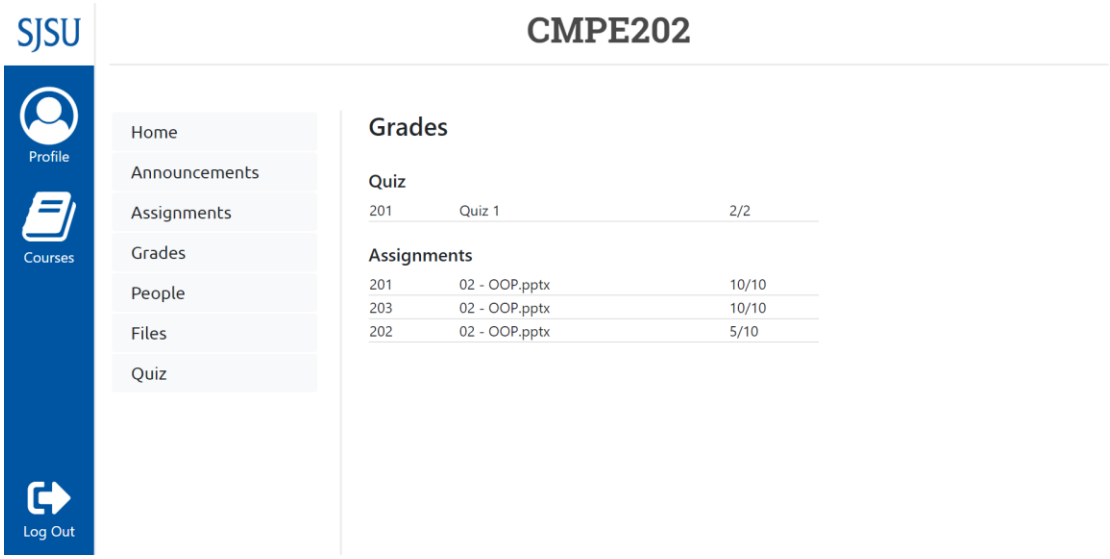

```
[ RowDataPacket {
    cid: 'CMPE282',
    qid: 'Quiz - 01',
    qname: 'Cloud Basics',
    q1:
     'Applications that work with cloud computing that have low margins and usually low risk are',
    op11: 'high touch',
    op12: 'low touch',
    op13: 'moderate touch',
    op14: 'all of the mentioned',
    cor1: 'b',
    q2:
     'A service that concentrates on hardware follows the _________ as a Service model.',
    op21: 'IaaS',
    op22: 'Caas',
    op23: 'PaaS',
    op24: 'All of the mentioned',
    cor2: 'a',
    d1: '2019-03-15',
    d2: '2019-03-18' } ]
Answers: b a
[ RowDataPacket { cor1: 'b', cor2: 'a' } ]
```

Student Grade view:



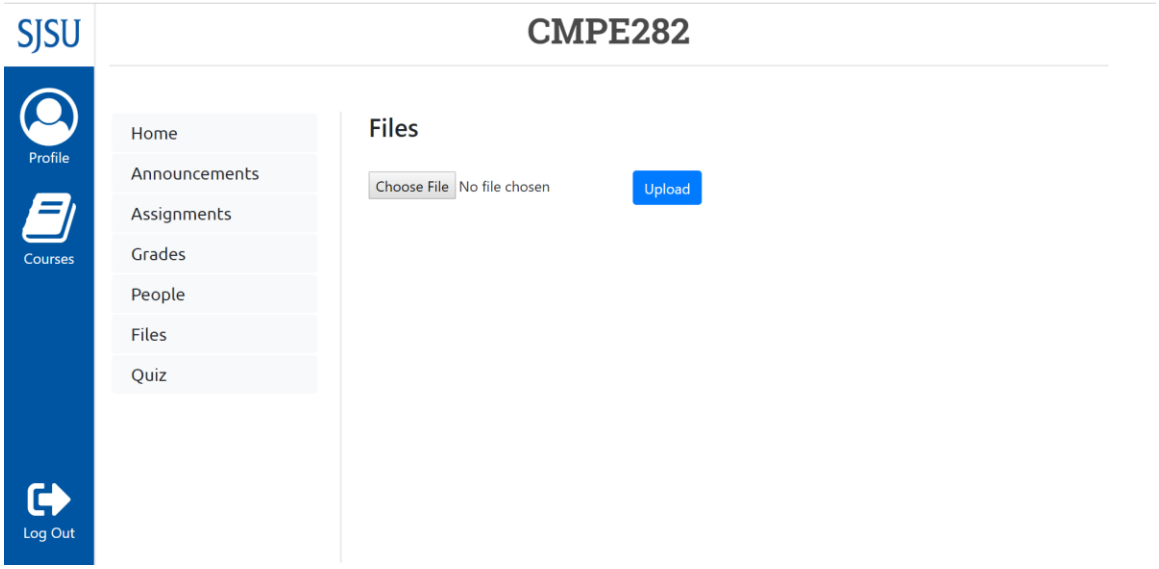

```
[ RowDataPacket {
    cid: 'CMPE202',
    sid: '201',
    typeof: 'quiz',
    typeid: 'Quiz 1',
    grade: '2' },
  RowDataPacket {
    cid: 'CMPE202',
    sid: '201',
    typeof: 'ass',
    typeid: '02 - OOP.pptx',
    grade: '10' } ]
```

Faculty Grade view:

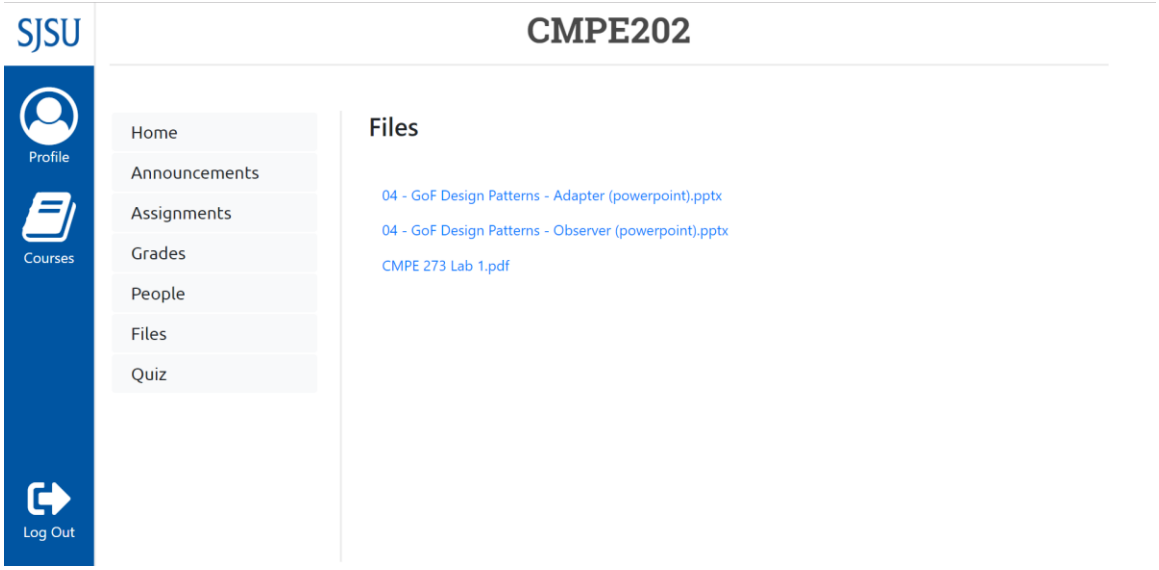



```
[ RowDataPacket {
    cid: 'CMPE202',
    sid: '201',
    typeof: 'quiz',
    typeid: 'Quiz 1',
    grade: '2' },
  RowDataPacket {
    cid: 'CMPE202',
    sid: '201',
    typeof: 'ass',
    typeid: '02 - OOP.pptx',
    grade: '10' },
  RowDataPacket {
    cid: 'CMPE202',
    sid: '203',
    typeof: 'ass',
    typeid: '02 - OOP.pptx',
    grade: '10' },
  RowDataPacket {
    cid: 'CMPE202',
    sid: '202',
    typeof: 'ass',
    typeid: '02 - OOP.pptx',
    grade: '5' } ]
```

Faculty File view:

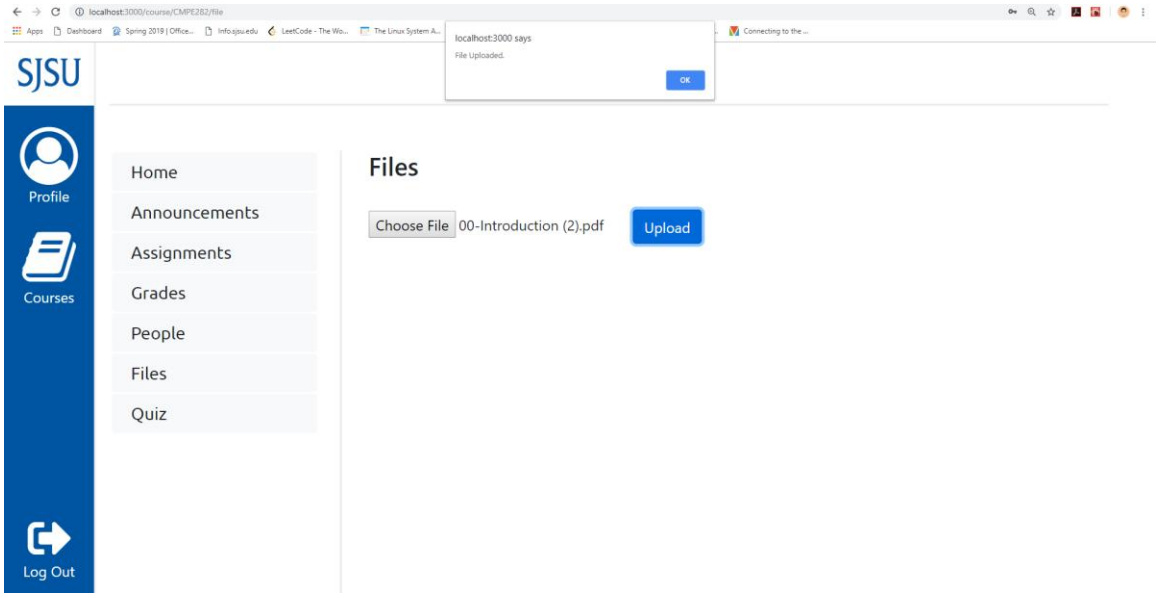


Student view for some lecture files uploaded:

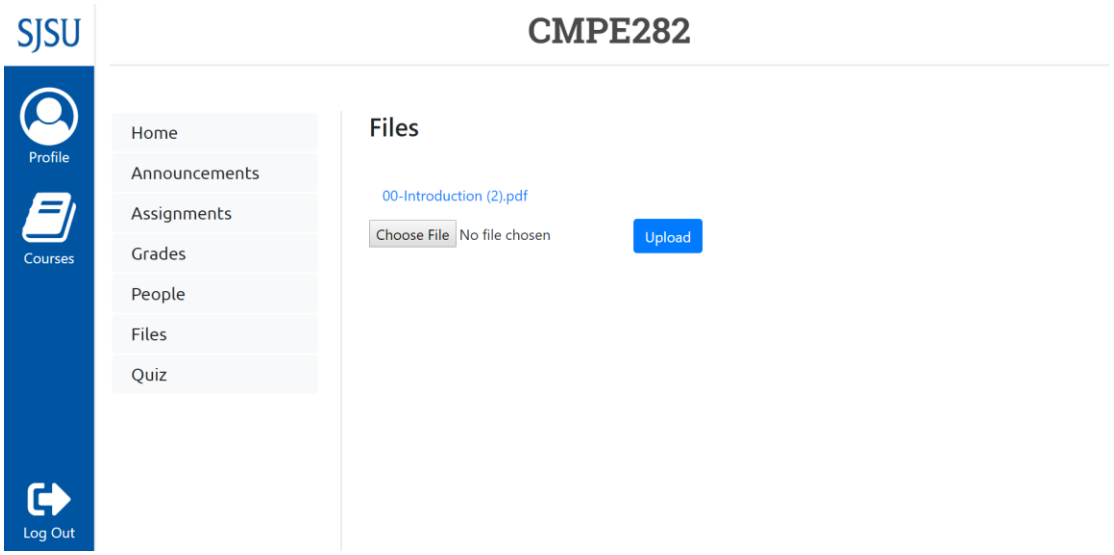


Clicking on the file name will download the file:

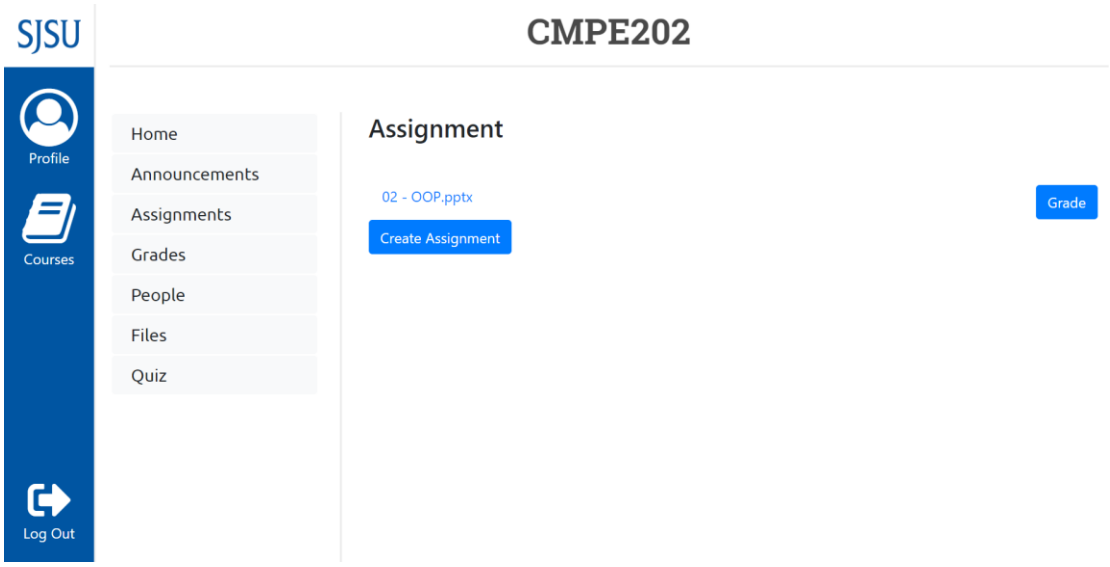
Faculty can upload new file:



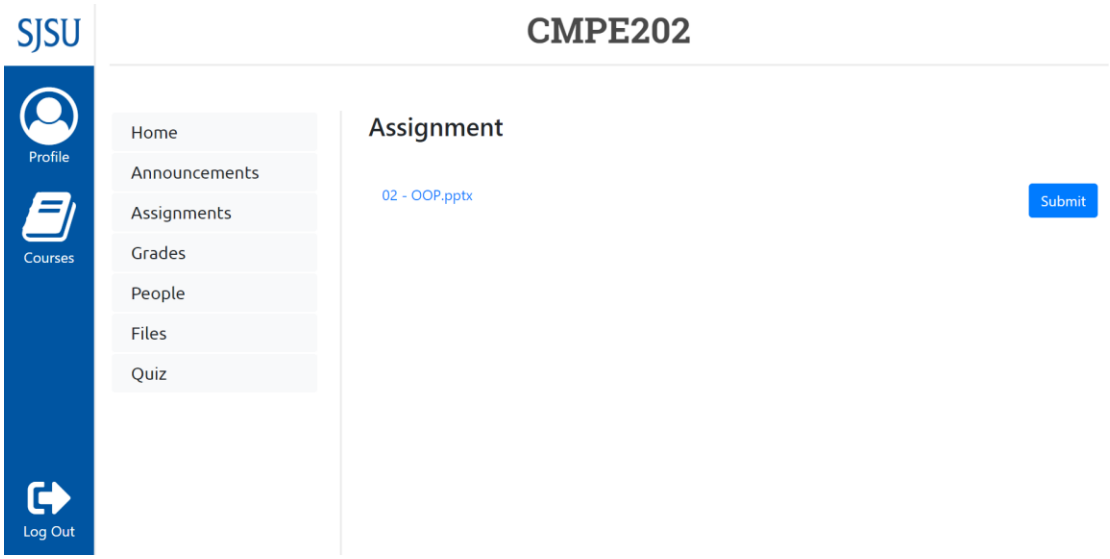
```
{ fieldname: 'lecturefile',
  originalname: '00-Introduction (2).pdf',
  encoding: '7bit',
  mimetype: 'application/pdf',
  destination: './public/uploads/',
  filename: '00-Introduction (2).pdf1552820224155.pdf',
  path: 'public\\uploads\\00-Introduction (2).pdf1552820224155.pdf',
  size: 413526 }
OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0 }
```



Faculty Assignment view:

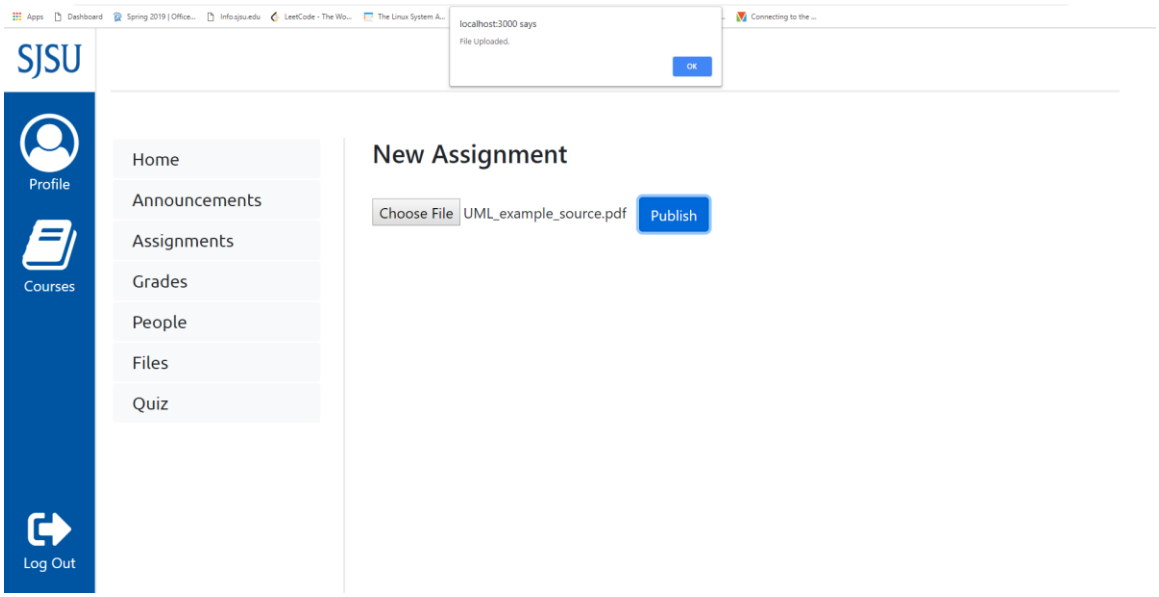


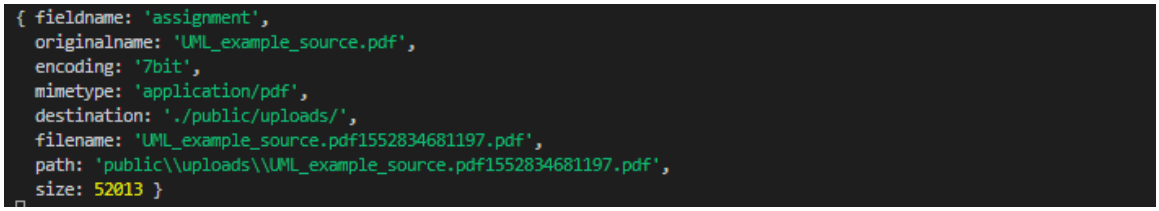
Student assignment view:

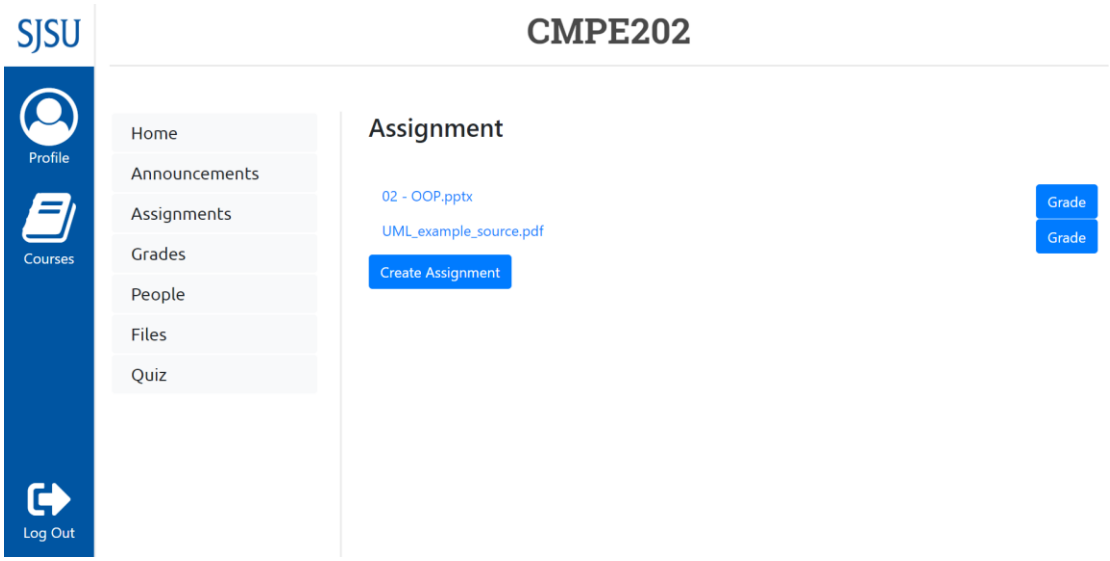



```
[ RowDataPacket {
    cid: 'CMPE202',
    asname: '02 - OOP.pptx',
    aspath: '02 - OOP.pptx1552744257657.pptx' } ]
```

Faculty posting assignment:

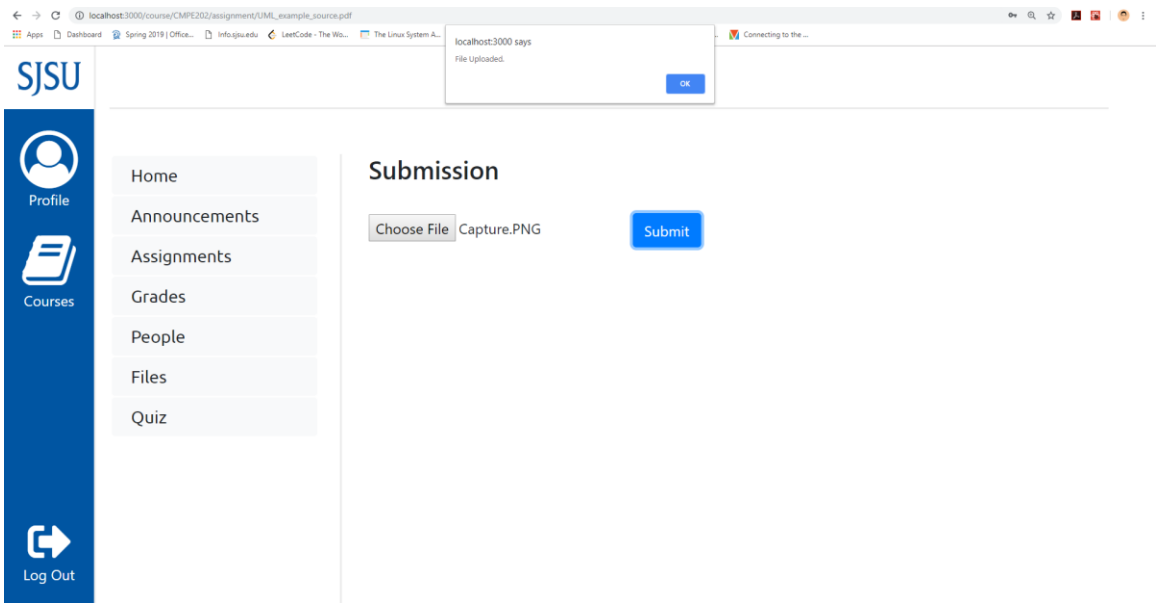






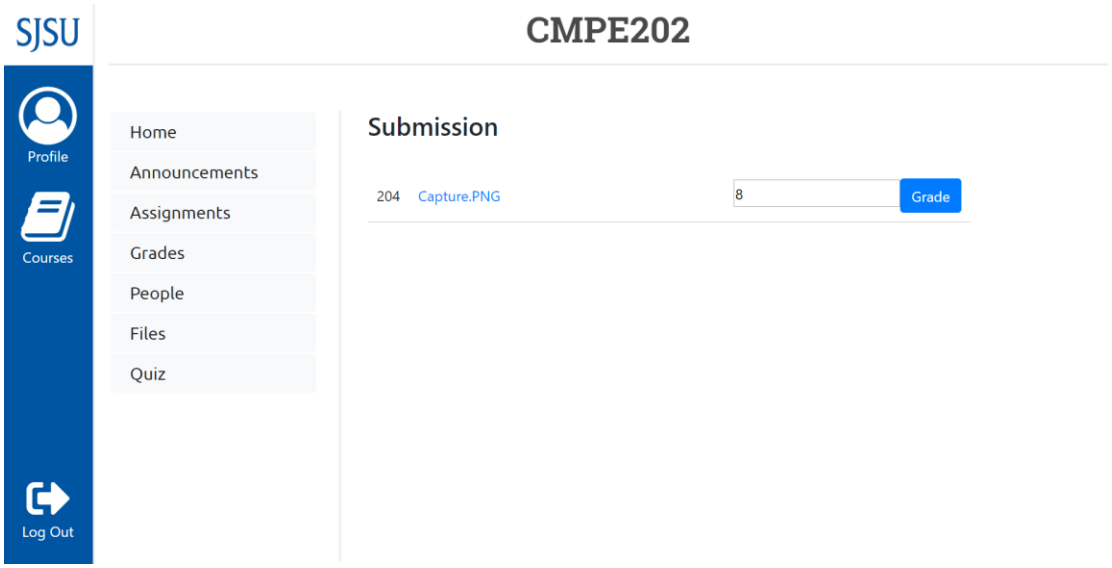
```
[ RowDataPacket {
    cid: 'CMPE202',
    asname: '02 - OOP.pptx',
    aspath: '02 - OOP.pptx1552744257657.pptx' },
  RowDataPacket {
    cid: 'CMPE202',
    asname: 'UML_example_source.pdf',
    aspath: 'UML_example_source.pdf1552834681197.pdf' } ]
```

Student submitting an assignment:



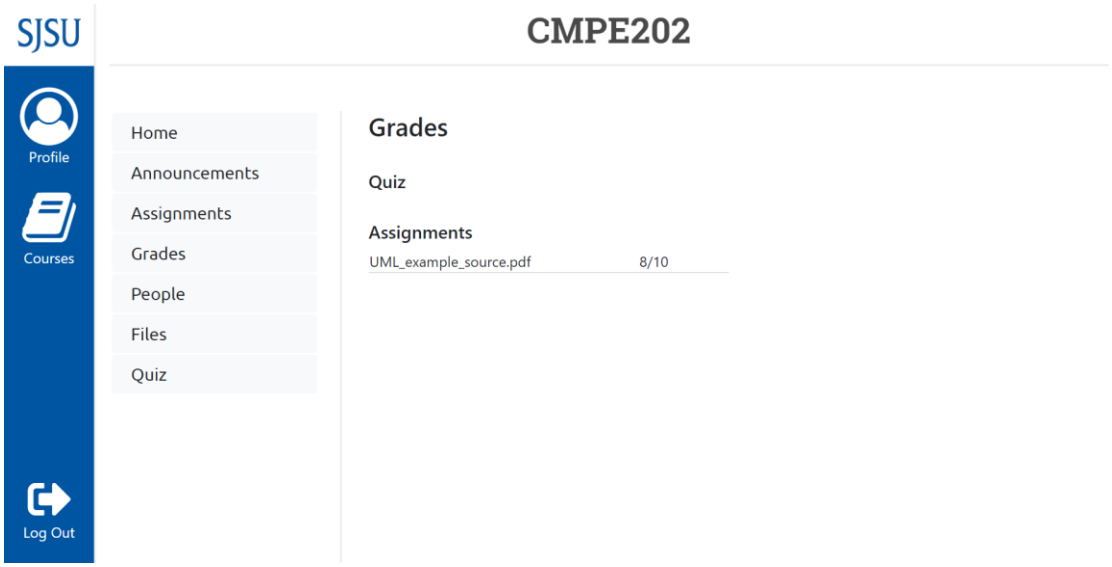

```
{ fieldname: 'submission',
  originalname: 'Capture.PNG',
  encoding: '7bit',
  mimetype: 'image/png',
  destination: './public/uploads/',
  filename: 'Capture.PNG1552835316218.PNG',
  path: 'public\\uploads\\Capture.PNG1552835316218.PNG',
  size: 120266 }
OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0 }
```

Faculty grading an assignment:




```
{ sid: '204', grade: '8' }
OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0 }
```

Student grades being updated:



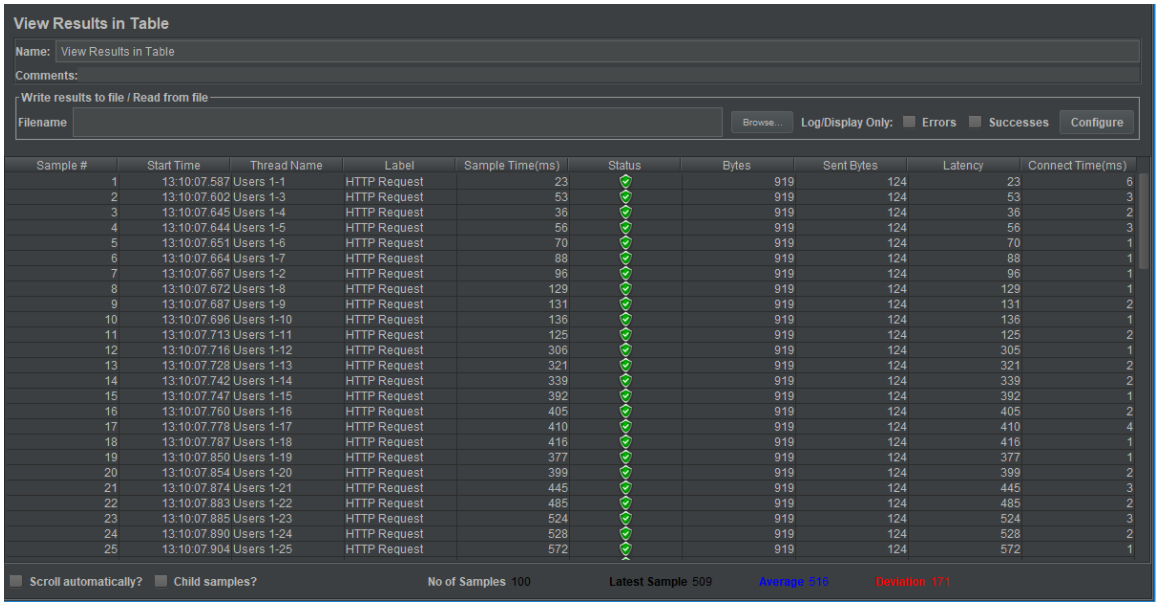
```
[ RowDataPacket {
    cid: 'CMPE202',
    sid: '204',
    typeof: 'ass',
    typeid: 'UML_example_source.pdf',
    grade: '8' } ]
```

- On any page where the file is uploaded like as a lecture note, assignment or submission, clicking on the file name will download the file.

Performance

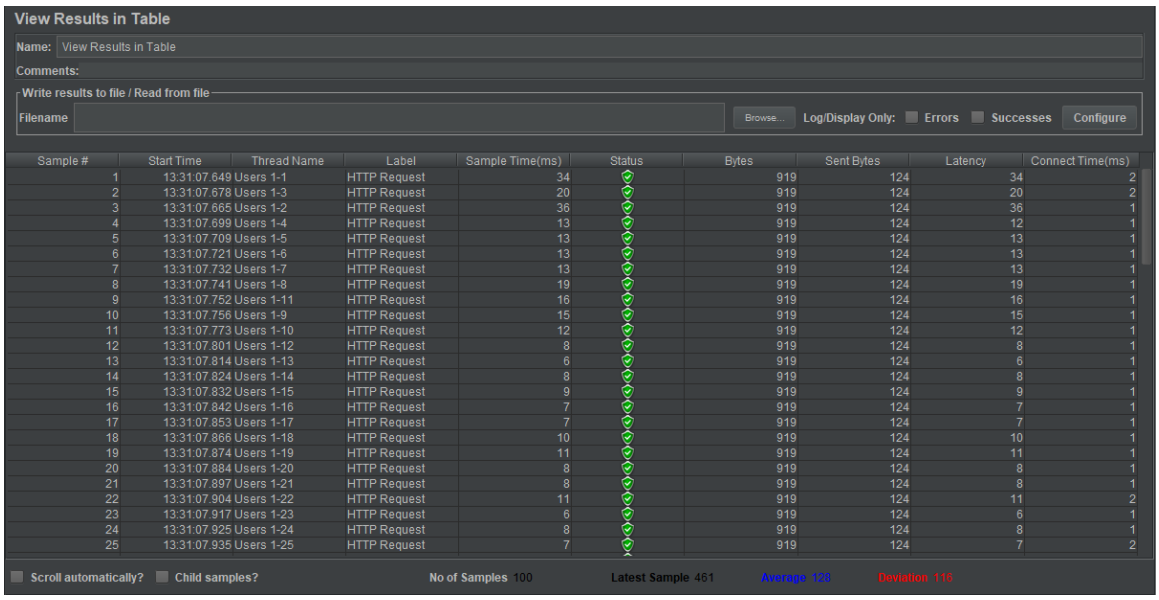
Jmeter Testing

100 concurrent users, without connection pooling, avg time = 516ms



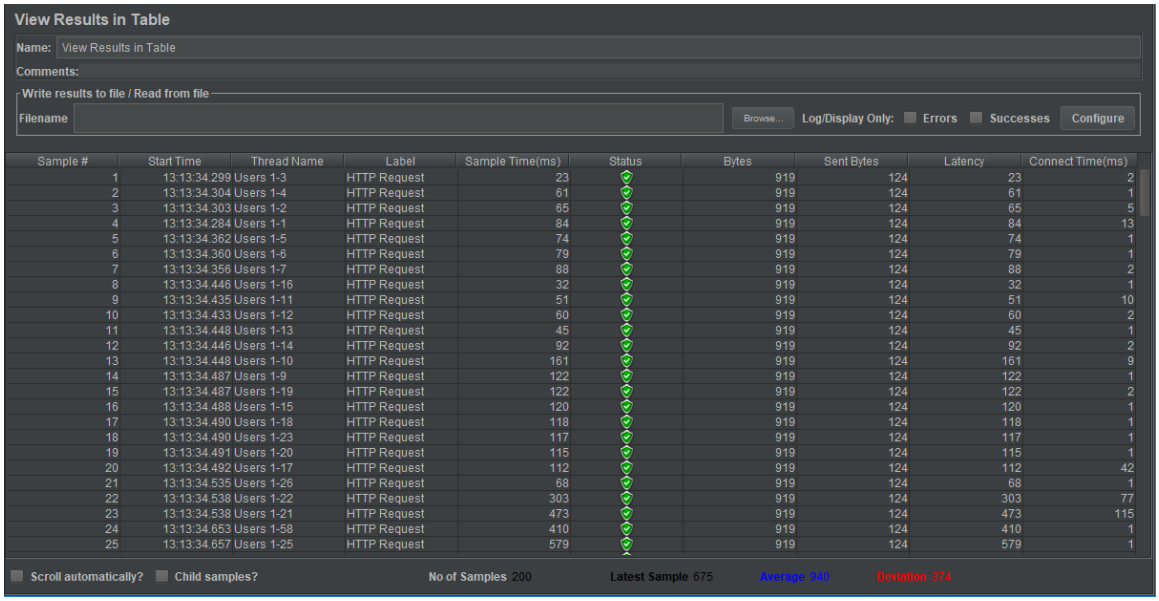
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:10:07.587	Users 1-1	HTTP Request	23		919	124	23	6
2	13:10:07.602	Users 1-3	HTTP Request	53		919	124	53	3
3	13:10:07.645	Users 1-4	HTTP Request	36		919	124	36	2
4	13:10:07.644	Users 1-5	HTTP Request	56		919	124	56	3
5	13:10:07.651	Users 1-6	HTTP Request	70		919	124	70	1
6	13:10:07.664	Users 1-7	HTTP Request	88		919	124	88	1
7	13:10:07.667	Users 1-2	HTTP Request	96		919	124	96	1
8	13:10:07.672	Users 1-8	HTTP Request	129		919	124	129	1
9	13:10:07.687	Users 1-9	HTTP Request	131		919	124	131	2
10	13:10:07.696	Users 1-10	HTTP Request	136		919	124	136	1
11	13:10:07.713	Users 1-11	HTTP Request	125		919	124	125	2
12	13:10:07.716	Users 1-12	HTTP Request	306		919	124	305	1
13	13:10:07.728	Users 1-13	HTTP Request	321		919	124	321	2
14	13:10:07.742	Users 1-14	HTTP Request	339		919	124	339	2
15	13:10:07.747	Users 1-15	HTTP Request	392		919	124	392	1
16	13:10:07.760	Users 1-16	HTTP Request	405		919	124	405	2
17	13:10:07.778	Users 1-17	HTTP Request	410		919	124	410	4
18	13:10:07.787	Users 1-18	HTTP Request	416		919	124	416	1
19	13:10:07.850	Users 1-19	HTTP Request	377		919	124	377	1
20	13:10:07.854	Users 1-20	HTTP Request	399		919	124	399	2
21	13:10:07.874	Users 1-21	HTTP Request	445		919	124	445	3
22	13:10:07.883	Users 1-22	HTTP Request	485		919	124	485	2
23	13:10:07.885	Users 1-23	HTTP Request	524		919	124	524	3
24	13:10:07.890	Users 1-24	HTTP Request	528		919	124	528	2
25	13:10:07.904	Users 1-25	HTTP Request	572		919	124	572	1

100 concurrent users, with connection pooling, avg time = 128ms



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:31:07.649	Users 1-1	HTTP Request	34		919	124	34	2
2	13:31:07.678	Users 1-3	HTTP Request	20		919	124	20	2
3	13:31:07.665	Users 1-2	HTTP Request	36		919	124	36	1
4	13:31:07.699	Users 1-4	HTTP Request	13		919	124	12	1
5	13:31:07.709	Users 1-5	HTTP Request	13		919	124	13	1
6	13:31:07.721	Users 1-6	HTTP Request	13		919	124	13	1
7	13:31:07.732	Users 1-7	HTTP Request	13		919	124	13	1
8	13:31:07.741	Users 1-8	HTTP Request	19		919	124	19	1
9	13:31:07.752	Users 1-11	HTTP Request	16		919	124	16	1
10	13:31:07.756	Users 1-9	HTTP Request	15		919	124	15	1
11	13:31:07.773	Users 1-10	HTTP Request	12		919	124	12	1
12	13:31:07.801	Users 1-12	HTTP Request	8		919	124	8	1
13	13:31:07.814	Users 1-13	HTTP Request	6		919	124	6	1
14	13:31:07.824	Users 1-14	HTTP Request	8		919	124	8	1
15	13:31:07.832	Users 1-15	HTTP Request	9		919	124	9	1
16	13:31:07.842	Users 1-16	HTTP Request	7		919	124	7	1
17	13:31:07.853	Users 1-17	HTTP Request	7		919	124	7	1
18	13:31:07.866	Users 1-18	HTTP Request	10		919	124	10	1
19	13:31:07.874	Users 1-19	HTTP Request	11		919	124	11	1
20	13:31:07.884	Users 1-20	HTTP Request	8		919	124	8	1
21	13:31:07.897	Users 1-21	HTTP Request	8		919	124	8	1
22	13:31:07.904	Users 1-22	HTTP Request	11		919	124	11	2
23	13:31:07.917	Users 1-23	HTTP Request	6		919	124	6	1
24	13:31:07.925	Users 1-24	HTTP Request	8		919	124	8	1
25	13:31:07.935	Users 1-25	HTTP Request	7		919	124	7	2

200 concurrent users, without connection pooling, avg time = 940ms



View Results in Table

Name: View Results in Table

Comments:

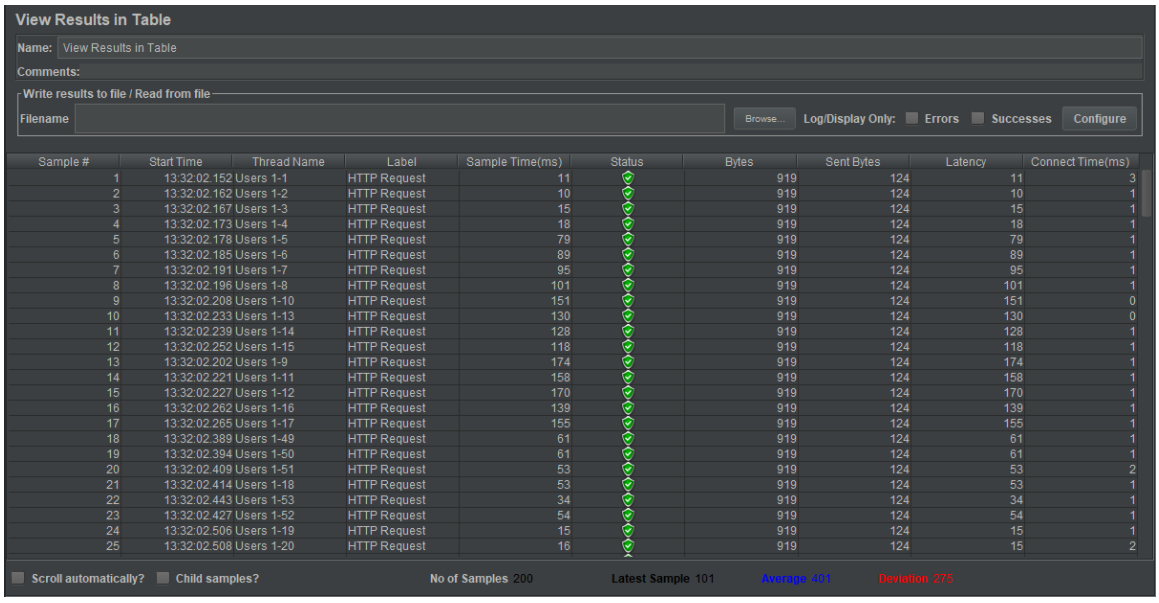
Write results to file / Read from file

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

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:13:34.299	Users 1-3	HTTP Request	23		919	124	23	2
2	13:13:34.304	Users 1-4	HTTP Request	61		919	124	61	1
3	13:13:34.303	Users 1-2	HTTP Request	65		919	124	65	5
4	13:13:34.284	Users 1-1	HTTP Request	84		919	124	84	13
5	13:13:34.362	Users 1-5	HTTP Request	74		919	124	74	1
6	13:13:34.360	Users 1-6	HTTP Request	79		919	124	79	1
7	13:13:34.356	Users 1-7	HTTP Request	88		919	124	88	2
8	13:13:34.446	Users 1-16	HTTP Request	32		919	124	32	1
9	13:13:34.435	Users 1-11	HTTP Request	51		919	124	51	10
10	13:13:34.433	Users 1-12	HTTP Request	60		919	124	60	2
11	13:13:34.448	Users 1-13	HTTP Request	45		919	124	45	1
12	13:13:34.446	Users 1-14	HTTP Request	92		919	124	92	2
13	13:13:34.448	Users 1-10	HTTP Request	161		919	124	161	9
14	13:13:34.487	Users 1-9	HTTP Request	122		919	124	122	1
15	13:13:34.487	Users 1-19	HTTP Request	122		919	124	122	2
16	13:13:34.488	Users 1-15	HTTP Request	120		919	124	120	1
17	13:13:34.490	Users 1-18	HTTP Request	118		919	124	118	1
18	13:13:34.490	Users 1-23	HTTP Request	117		919	124	117	1
19	13:13:34.491	Users 1-20	HTTP Request	115		919	124	115	1
20	13:13:34.492	Users 1-17	HTTP Request	112		919	124	112	42
21	13:13:34.535	Users 1-26	HTTP Request	68		919	124	68	1
22	13:13:34.538	Users 1-22	HTTP Request	303		919	124	303	77
23	13:13:34.538	Users 1-21	HTTP Request	473		919	124	473	115
24	13:13:34.653	Users 1-58	HTTP Request	410		919	124	410	1
25	13:13:34.657	Users 1-25	HTTP Request	579		919	124	579	1

Scroll automatically? Child samples? No of Samples 200 Latest Sample 675 Average 940 Deviation 374

200 concurrent users, with connection pooling, avg time = 401ms



View Results in Table

Name: View Results in Table

Comments:

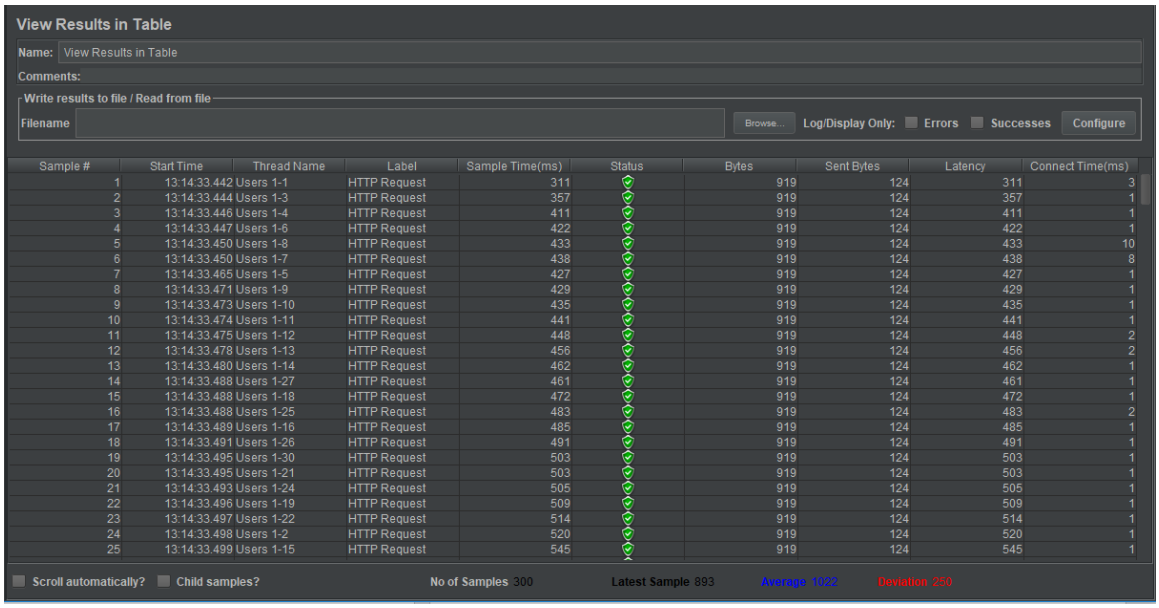
Write results to file / Read from file

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

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:32:02.152	Users 1-1	HTTP Request	11		919	124	11	3
2	13:32:02.162	Users 1-2	HTTP Request	10		919	124	10	1
3	13:32:02.167	Users 1-3	HTTP Request	15		919	124	15	1
4	13:32:02.173	Users 1-4	HTTP Request	18		919	124	18	1
5	13:32:02.178	Users 1-5	HTTP Request	79		919	124	79	1
6	13:32:02.185	Users 1-6	HTTP Request	89		919	124	89	1
7	13:32:02.191	Users 1-7	HTTP Request	95		919	124	95	1
8	13:32:02.196	Users 1-8	HTTP Request	101		919	124	101	1
9	13:32:02.208	Users 1-10	HTTP Request	151		919	124	151	0
10	13:32:02.233	Users 1-13	HTTP Request	130		919	124	130	0
11	13:32:02.239	Users 1-14	HTTP Request	128		919	124	128	1
12	13:32:02.252	Users 1-15	HTTP Request	118		919	124	118	1
13	13:32:02.202	Users 1-9	HTTP Request	174		919	124	174	1
14	13:32:02.221	Users 1-11	HTTP Request	158		919	124	158	1
15	13:32:02.227	Users 1-12	HTTP Request	170		919	124	170	1
16	13:32:02.262	Users 1-16	HTTP Request	139		919	124	139	1
17	13:32:02.265	Users 1-17	HTTP Request	155		919	124	155	1
18	13:32:02.389	Users 1-49	HTTP Request	61		919	124	61	1
19	13:32:02.394	Users 1-50	HTTP Request	61		919	124	61	1
20	13:32:02.409	Users 1-51	HTTP Request	53		919	124	53	2
21	13:32:02.414	Users 1-18	HTTP Request	53		919	124	53	1
22	13:32:02.443	Users 1-53	HTTP Request	34		919	124	34	1
23	13:32:02.427	Users 1-52	HTTP Request	54		919	124	54	1
24	13:32:02.506	Users 1-19	HTTP Request	15		919	124	15	1
25	13:32:02.508	Users 1-20	HTTP Request	16		919	124	15	2

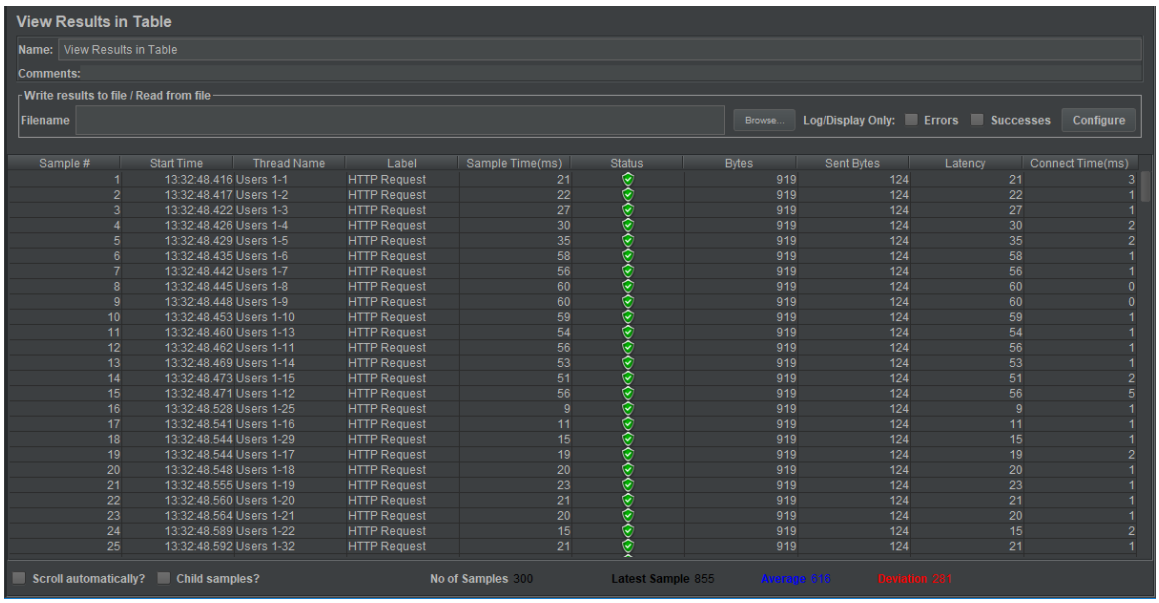
Scroll automatically? Child samples? No of Samples 200 Latest Sample 101 Average 401 Deviation 275

300 concurrent users, without connection pooling, avg time = 1022ms



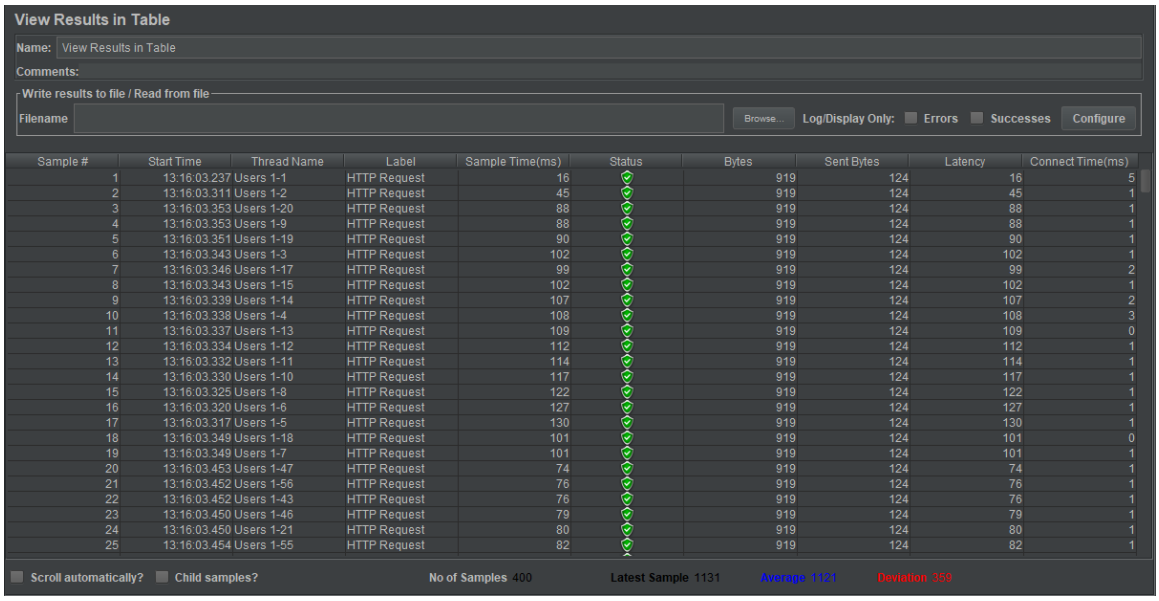
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:14:33.442	Users 1-1	HTTP Request	311		919	124	311	3
2	13:14:33.444	Users 1-3	HTTP Request	357		919	124	357	1
3	13:14:33.446	Users 1-4	HTTP Request	411		919	124	411	1
4	13:14:33.447	Users 1-6	HTTP Request	422		919	124	422	1
5	13:14:33.450	Users 1-8	HTTP Request	433		919	124	433	10
6	13:14:33.450	Users 1-7	HTTP Request	438		919	124	438	8
7	13:14:33.465	Users 1-5	HTTP Request	427		919	124	427	1
8	13:14:33.471	Users 1-9	HTTP Request	429		919	124	429	1
9	13:14:33.473	Users 1-10	HTTP Request	435		919	124	435	1
10	13:14:33.474	Users 1-11	HTTP Request	441		919	124	441	1
11	13:14:33.475	Users 1-12	HTTP Request	448		919	124	448	2
12	13:14:33.478	Users 1-13	HTTP Request	456		919	124	456	2
13	13:14:33.480	Users 1-14	HTTP Request	462		919	124	462	1
14	13:14:33.488	Users 1-27	HTTP Request	461		919	124	461	1
15	13:14:33.488	Users 1-18	HTTP Request	472		919	124	472	1
16	13:14:33.488	Users 1-25	HTTP Request	483		919	124	483	2
17	13:14:33.489	Users 1-16	HTTP Request	485		919	124	485	1
18	13:14:33.491	Users 1-26	HTTP Request	491		919	124	491	1
19	13:14:33.495	Users 1-30	HTTP Request	503		919	124	503	1
20	13:14:33.495	Users 1-21	HTTP Request	503		919	124	503	1
21	13:14:33.493	Users 1-24	HTTP Request	505		919	124	505	1
22	13:14:33.496	Users 1-19	HTTP Request	509		919	124	509	1
23	13:14:33.497	Users 1-22	HTTP Request	514		919	124	514	1
24	13:14:33.498	Users 1-2	HTTP Request	520		919	124	520	1
25	13:14:33.499	Users 1-15	HTTP Request	545		919	124	545	1

300 concurrent users, with connection pooling, avg time = 616ms



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:32:48.416	Users 1-1	HTTP Request	21		919	124	21	3
2	13:32:48.417	Users 1-2	HTTP Request	22		919	124	22	1
3	13:32:48.422	Users 1-3	HTTP Request	27		919	124	27	1
4	13:32:48.426	Users 1-4	HTTP Request	30		919	124	30	2
5	13:32:48.429	Users 1-5	HTTP Request	35		919	124	35	2
6	13:32:48.435	Users 1-6	HTTP Request	58		919	124	58	1
7	13:32:48.442	Users 1-7	HTTP Request	56		919	124	56	1
8	13:32:48.445	Users 1-8	HTTP Request	60		919	124	60	0
9	13:32:48.448	Users 1-9	HTTP Request	60		919	124	60	0
10	13:32:48.453	Users 1-10	HTTP Request	59		919	124	59	1
11	13:32:48.460	Users 1-13	HTTP Request	54		919	124	54	1
12	13:32:48.462	Users 1-11	HTTP Request	56		919	124	56	1
13	13:32:48.469	Users 1-14	HTTP Request	53		919	124	53	1
14	13:32:48.473	Users 1-15	HTTP Request	51		919	124	51	2
15	13:32:48.471	Users 1-12	HTTP Request	56		919	124	56	5
16	13:32:48.528	Users 1-25	HTTP Request	9		919	124	9	1
17	13:32:48.541	Users 1-16	HTTP Request	11		919	124	11	1
18	13:32:48.544	Users 1-29	HTTP Request	15		919	124	15	1
19	13:32:48.544	Users 1-17	HTTP Request	19		919	124	19	2
20	13:32:48.548	Users 1-18	HTTP Request	20		919	124	20	1
21	13:32:48.555	Users 1-19	HTTP Request	23		919	124	23	1
22	13:32:48.560	Users 1-20	HTTP Request	21		919	124	21	1
23	13:32:48.564	Users 1-21	HTTP Request	20		919	124	20	1
24	13:32:48.589	Users 1-22	HTTP Request	15		919	124	15	2
25	13:32:48.592	Users 1-32	HTTP Request	21		919	124	21	1

400 concurrent users, without connection pooling, avg time = 1121ms



View Results in Table

Name: View Results in Table

Comments:

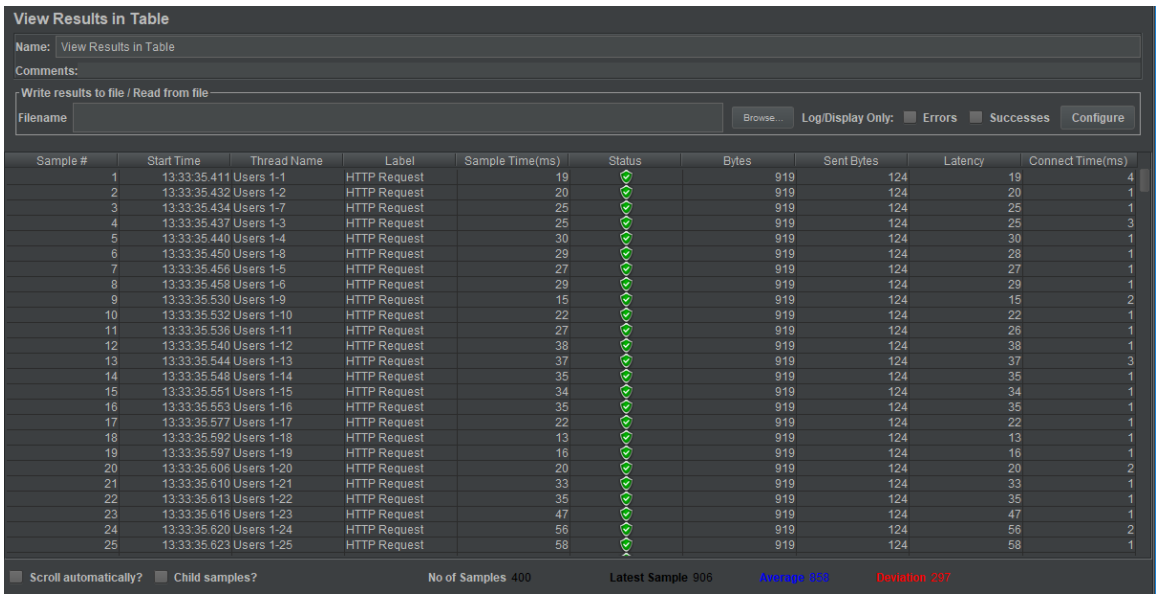
Write results to file / Read from file

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

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:16:03.237	Users 1-1	HTTP Request	16		919	124	16	5
2	13:16:03.311	Users 1-2	HTTP Request	45		919	124	45	1
3	13:16:03.353	Users 1-20	HTTP Request	88		919	124	88	1
4	13:16:03.353	Users 1-9	HTTP Request	88		919	124	88	1
5	13:16:03.351	Users 1-19	HTTP Request	90		919	124	90	1
6	13:16:03.343	Users 1-3	HTTP Request	102		919	124	102	1
7	13:16:03.346	Users 1-17	HTTP Request	99		919	124	99	2
8	13:16:03.343	Users 1-15	HTTP Request	102		919	124	102	1
9	13:16:03.339	Users 1-14	HTTP Request	107		919	124	107	2
10	13:16:03.338	Users 1-4	HTTP Request	108		919	124	108	3
11	13:16:03.337	Users 1-13	HTTP Request	109		919	124	109	0
12	13:16:03.334	Users 1-12	HTTP Request	112		919	124	112	1
13	13:16:03.332	Users 1-11	HTTP Request	114		919	124	114	1
14	13:16:03.330	Users 1-10	HTTP Request	117		919	124	117	1
15	13:16:03.325	Users 1-8	HTTP Request	122		919	124	122	1
16	13:16:03.320	Users 1-6	HTTP Request	127		919	124	127	1
17	13:16:03.317	Users 1-5	HTTP Request	130		919	124	130	1
18	13:16:03.349	Users 1-18	HTTP Request	101		919	124	101	0
19	13:16:03.349	Users 1-7	HTTP Request	101		919	124	101	1
20	13:16:03.453	Users 1-47	HTTP Request	74		919	124	74	1
21	13:16:03.452	Users 1-56	HTTP Request	76		919	124	76	1
22	13:16:03.452	Users 1-43	HTTP Request	76		919	124	76	1
23	13:16:03.450	Users 1-46	HTTP Request	79		919	124	79	1
24	13:16:03.450	Users 1-21	HTTP Request	80		919	124	80	1
25	13:16:03.454	Users 1-55	HTTP Request	82		919	124	82	1

Scroll automatically? Child samples? No of Samples 400 Latest Sample 1131 Average 1121 Deviation 359

400 concurrent users, with connection pooling, avg time = 858ms



View Results in Table

Name: View Results in Table

Comments:

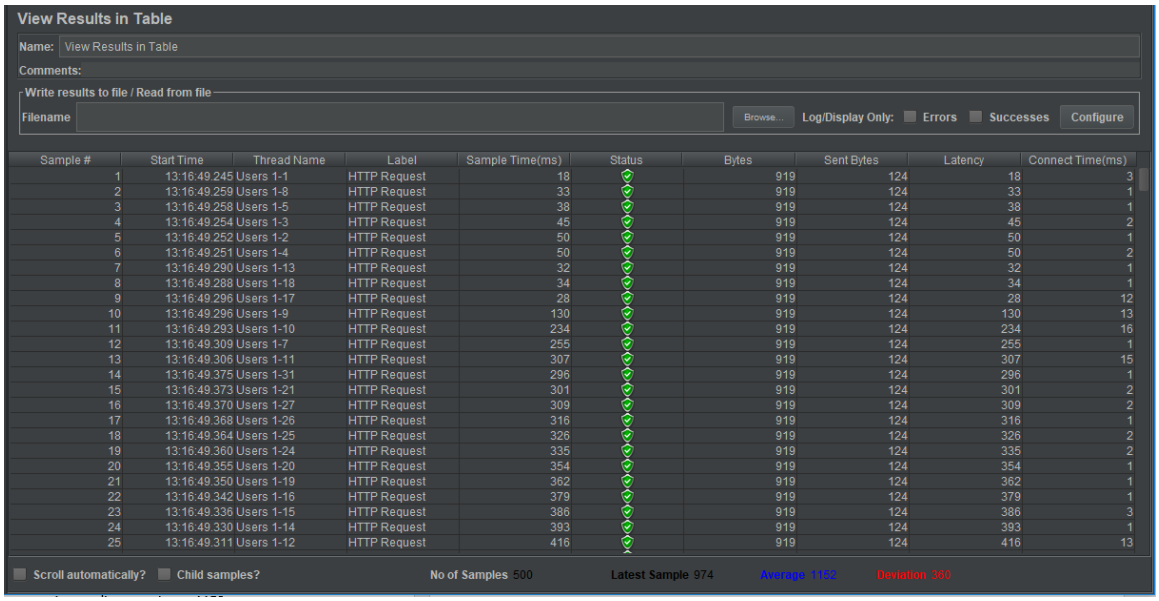
Write results to file / Read from file

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

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:33:35.411	Users 1-1	HTTP Request	19		919	124	19	4
2	13:33:35.432	Users 1-2	HTTP Request	20		919	124	20	1
3	13:33:35.434	Users 1-7	HTTP Request	25		919	124	25	1
4	13:33:35.437	Users 1-3	HTTP Request	25		919	124	25	3
5	13:33:35.440	Users 1-4	HTTP Request	30		919	124	30	1
6	13:33:35.450	Users 1-8	HTTP Request	29		919	124	28	1
7	13:33:35.456	Users 1-5	HTTP Request	27		919	124	27	1
8	13:33:35.458	Users 1-6	HTTP Request	29		919	124	29	1
9	13:33:35.530	Users 1-9	HTTP Request	15		919	124	15	2
10	13:33:35.532	Users 1-10	HTTP Request	22		919	124	22	1
11	13:33:35.536	Users 1-11	HTTP Request	27		919	124	26	1
12	13:33:35.540	Users 1-12	HTTP Request	38		919	124	38	1
13	13:33:35.544	Users 1-13	HTTP Request	37		919	124	37	3
14	13:33:35.548	Users 1-14	HTTP Request	35		919	124	35	1
15	13:33:35.551	Users 1-15	HTTP Request	34		919	124	34	1
16	13:33:35.553	Users 1-16	HTTP Request	35		919	124	35	1
17	13:33:35.577	Users 1-17	HTTP Request	22		919	124	22	1
18	13:33:35.592	Users 1-18	HTTP Request	13		919	124	13	1
19	13:33:35.597	Users 1-19	HTTP Request	16		919	124	16	1
20	13:33:35.606	Users 1-20	HTTP Request	20		919	124	20	2
21	13:33:35.610	Users 1-21	HTTP Request	33		919	124	33	1
22	13:33:35.613	Users 1-22	HTTP Request	35		919	124	35	1
23	13:33:35.616	Users 1-23	HTTP Request	47		919	124	47	1
24	13:33:35.620	Users 1-24	HTTP Request	56		919	124	56	2
25	13:33:35.623	Users 1-25	HTTP Request	58		919	124	58	1

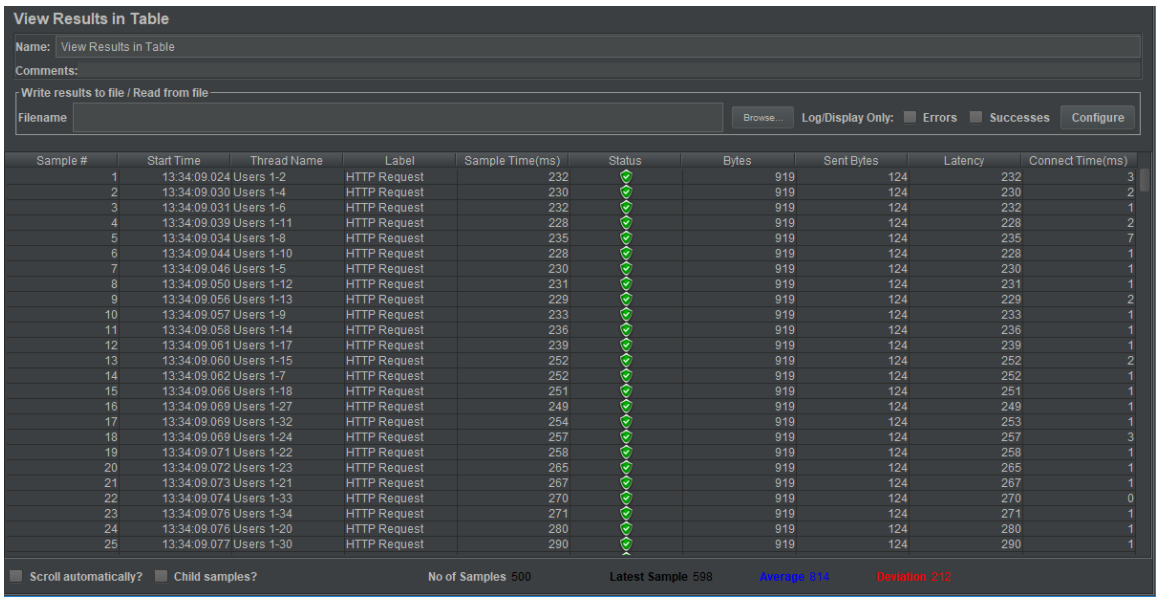
Scroll automatically? Child samples? No of Samples 400 Latest Sample 906 Average 858 Deviation 207

500 concurrent users, without connection pooling, avg time = 1152ms



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:16:49.245	Users 1-1	HTTP Request	18		919	124	18	3
2	13:16:49.259	Users 1-8	HTTP Request	33		919	124	33	1
3	13:16:49.258	Users 1-5	HTTP Request	38		919	124	38	1
4	13:16:49.254	Users 1-3	HTTP Request	45		919	124	45	2
5	13:16:49.252	Users 1-2	HTTP Request	50		919	124	50	1
6	13:16:49.251	Users 1-4	HTTP Request	50		919	124	50	2
7	13:16:49.290	Users 1-13	HTTP Request	32		919	124	32	1
8	13:16:49.288	Users 1-18	HTTP Request	34		919	124	34	1
9	13:16:49.296	Users 1-17	HTTP Request	28		919	124	28	12
10	13:16:49.296	Users 1-9	HTTP Request	130		919	124	130	13
11	13:16:49.293	Users 1-10	HTTP Request	234		919	124	234	16
12	13:16:49.309	Users 1-7	HTTP Request	255		919	124	255	1
13	13:16:49.306	Users 1-11	HTTP Request	307		919	124	307	15
14	13:16:49.375	Users 1-31	HTTP Request	296		919	124	296	1
15	13:16:49.373	Users 1-21	HTTP Request	301		919	124	301	2
16	13:16:49.370	Users 1-27	HTTP Request	309		919	124	309	2
17	13:16:49.368	Users 1-26	HTTP Request	316		919	124	316	1
18	13:16:49.364	Users 1-25	HTTP Request	326		919	124	326	2
19	13:16:49.360	Users 1-24	HTTP Request	335		919	124	335	2
20	13:16:49.355	Users 1-20	HTTP Request	354		919	124	354	1
21	13:16:49.350	Users 1-19	HTTP Request	362		919	124	362	1
22	13:16:49.342	Users 1-16	HTTP Request	379		919	124	379	1
23	13:16:49.336	Users 1-15	HTTP Request	386		919	124	386	3
24	13:16:49.330	Users 1-14	HTTP Request	393		919	124	393	1
25	13:16:49.311	Users 1-12	HTTP Request	416		919	124	416	13

500 concurrent users, with connection pooling, avg time = 814ms



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:34:09.024	Users 1-2	HTTP Request	232		919	124	232	3
2	13:34:09.030	Users 1-4	HTTP Request	230		919	124	230	2
3	13:34:09.031	Users 1-6	HTTP Request	232		919	124	232	1
4	13:34:09.039	Users 1-11	HTTP Request	228		919	124	228	2
5	13:34:09.034	Users 1-8	HTTP Request	235		919	124	235	7
6	13:34:09.044	Users 1-10	HTTP Request	228		919	124	228	1
7	13:34:09.046	Users 1-5	HTTP Request	230		919	124	230	1
8	13:34:09.050	Users 1-12	HTTP Request	231		919	124	231	1
9	13:34:09.056	Users 1-13	HTTP Request	229		919	124	229	2
10	13:34:09.057	Users 1-9	HTTP Request	233		919	124	233	1
11	13:34:09.058	Users 1-14	HTTP Request	236		919	124	236	1
12	13:34:09.061	Users 1-17	HTTP Request	239		919	124	239	1
13	13:34:09.060	Users 1-15	HTTP Request	252		919	124	252	2
14	13:34:09.062	Users 1-7	HTTP Request	252		919	124	252	1
15	13:34:09.066	Users 1-18	HTTP Request	251		919	124	251	1
16	13:34:09.069	Users 1-27	HTTP Request	249		919	124	249	1
17	13:34:09.069	Users 1-32	HTTP Request	254		919	124	253	1
18	13:34:09.069	Users 1-24	HTTP Request	257		919	124	257	3
19	13:34:09.071	Users 1-22	HTTP Request	258		919	124	258	1
20	13:34:09.072	Users 1-23	HTTP Request	265		919	124	265	1
21	13:34:09.073	Users 1-21	HTTP Request	267		919	124	267	1
22	13:34:09.074	Users 1-33	HTTP Request	270		919	124	270	0
23	13:34:09.076	Users 1-34	HTTP Request	271		919	124	271	1
24	13:34:09.076	Users 1-20	HTTP Request	280		919	124	280	1
25	13:34:09.077	Users 1-30	HTTP Request	290		919	124	290	1

Conclusion

The average response time increases gradually along with number of users till the users are 400, then it decreases slightly in both cases.

The response time for connection pooling is always less than without pooling.

The difference of average between with and without pooling decreases as the number of users increases. Setting a higher pool limit may decrease the average response time.

Mocha Test

```
Canvas
  √ GET /course
  √ GET /course/search
  √ GET /profile
  √ GET /course/:id/people
  √ GET /course/:id/home
```

```
it('GET /course', () => {
    agent.get('/course').query({ currentUser:"101"})
        .then((res)=>{
            let expect = chai.expect;
            expect(res.body.courses).to.have.lengthOf(4);
        });
});

it('GET /course/search', () => {
    agent.get('/course/search').query({ currentUser:"101"})
        .then((res)=>{
            let expect = chai.expect;
            expect(res.body.courses).to.have.lengthOf(9);
        });
});

it('GET /profile', () => {
    agent.get('/profile').query({ currentUser:"101"})
        .then((res)=>{
            let expect = chai.expect;
            expect(res.body.name).to.equal("Colt Steele");
        });
});

it('GET /course/:id/people', () => {
    agent.get('/course/:id/people').query({ currentUser:"101", cid:"CMPE202"})
        .then((res)=>{
            let expect = chai.expect;
            expect(res.body.people).to.have.lengthOf(4);
        });
});

it('GET /course/:id/home', () => {
    agent.get('/course/:id/home').query({ currentUser:"101", cid:"CMPE202"})
        .then((res)=>{
            let expect = chai.expect;
            expect(res.body.course.fid).to.equal("101");
        });
});
```

Questions

1) Broadly speaking, passwords can be stored in any of the following ways: -

Plaintext: The exact password entered by user is stored in database which offers no security against attacks.

Encryption: It converts a plaintext to ciphertext using encryption algorithm and a key. Decryption follows decryption algorithm using the same key. It requires a key for both processes.

Hashing: It requires no key. Passwords are encrypted using a one-way hash function which are computationally very expensive to crack. Two users cannot have same salt implies their passwords can never be same.

Hash and Salt: Salt is a randomly generated string of characters which can be append or prepend to the passwords. Incorporating salts and hashing the same makes many attacks like dictionary or brute-force attacks futile.

Modern hashing algorithm are MD-5, SHA-1, SHA-2 and SHA-3. There are currently 3 main password hashing algorithms:

- PBKDF2
- bcrypt
- scrypt.

The passwords in this application are encrypted using bcrypt algorithm which is based on Blowfish Cipher. Since it's a type of Blowfish cipher, it provides good encryption rate and currently no effective cryptanalysis of it has been found. To generate hash, it uses look-up table which is loaded into memory; so it will utilize memory space. It uses salt iteratively to resist rainbow and brute-force attack even with increasing computational power. It uses timestamp so even the same password will be hashed to different value at different instance of time and generates long hash containing numbers, letters and special characters.

This algorithm is programmed by the bcrypt node module available to generate salt, hash it and compare password with its hash. Higher salt rounds can achieve stronger passwords but it is memory consuming.

2) Connection pooling is a technique where database connections are cached so that they can be reused in future instead of establishing and releasing a connection every time data needs to be accessed. It is used to improve performance of executing codes in database. It also cuts down time when a user must wait to establish a new connection.

Comparing the results of jmeter test, it is clear that the average time increases in a polynomial manner as the concurrent users increases. The time is always less for pooled connection and the difference between pooled and without pooled connection decrease as number of concurrent users increases. In the testing case, the poolLimit is set to 100. As the number of concurrent users grow, the response time slope gradually decreases. This declination can be assumed to decrease to a lesser extent if the poolLimit is set higher which I turn will give better performance. But eventually, as the number of users approach or cross the poolLimit, the graph gets shallow.

If I were to implement a connection pooling technique, it would work as follows:

Since threads are light-weight processes, they can be used to create and release a connection to the database. We can have a pool of reusable threads. The pool data structure provides with a free thread to an incoming request. If the number of requests passes the number of threads, the system will facilitate by adding a new thread to the pool. The number of threads added this way must have a certain limit. When the server id less busy, the newly created threads can be released. This way, connection to database will be established only once, hence reducing overhead due to creating new connections.

As long as data is being read, the threads can work concurrently. Similarly, when threads are being used to write data to different locations they will work concurrently. When threads are writing to same data, they must wait in a queue. The concept of semaphores can be helpful for the same. Threads will be maintained in first-come-first-serve bases written accordingly. We can also use thread priority to surpass or condone the operation. For example, a system admin's thread priority will be higher than normal user and it must access the database first. Trivial processes which doesn't affect much to the database can have less thread priority.

3) Caching refers to storing frequently used data so that it can be retrieved quickly, which otherwise is a resource intensive process. The idea is same in SQL caching. SQL has a buffer cache which it uses to store table or index data pages once they are modified or read from the disk. These pages are same as they appear on the disk. If the application has to access data, it will search in cache and if it is not found, only then it will search for the same in database. The cached data can be read as long as it is not changed. Once the data changes, the system must update the cache. Thus, SQL caching aims to reduce I/O tasks and hence reduce data retrieval and response time.

There are several caching strategies, which depends on data and data access pattern They are: -

- Cache-aside
- Read-through Cache
- Write-through Cache
- Write-around
- Write-back

According to me, combination of Read-through and Write-through technique will be the most efficient way to cache data. These techniques have a direct access to cache and indirect access to database, through cache. Database is only accessed if a cache-miss occurs in case of read; in case of write, the data is always written to cache before writing to database.

Since the number of students is way more than faculty, the number or read/write operations for students will always be more than that for faculty. Suppose a faculty makes a new announcement or uploads some files, then the students under those faculty and his course will read the same data in large number. Therefore, new announcement or data common for students can be stored in the cache for faster access. Similarly, submitting assignments will be in large number and in case of quiz it can be concurrent. In this case, the write-through cache will store the data in cache prior to writing it to database. By the time it starts caching new write data, the previous ones can be written to database simultaneously. For faculty, the read/write data can be stored in cache but only temporarily. Also, it is assumed the database connection is made through pooling which is much lighter than the traditional way.

4) Scalability means a system, network or process capable of handling growing workload or the potential to accommodate the same.

Vertically scalability refers to extending the capacity of server by adding CPU, memory, I/O and networking components to address workload requirements.

When the server gets overloaded with incoming requests and does not has enough resources, they can be allocated dynamically. Horizontally scalable allows adding multiple hardware which work as a single entity. If the server is clustered, it can be scaled horizontally to provide availability and performance. It allows scalability on the fly. Some examples of horizontally scalable are MongoDB, Cassandra, Google Cloud Spanner, etc.

My current application's session strategy is not horizontally scalable because sessions are stored as states. The session is maintained using cookies for each user in their browser as well as in the server's database. Which means authenticating user for every request requires accessing the database and validating the same. This can be very expensive process. This can be overcome by using what is known as JSON Web Token (JWT). JWT is basically a string that is sent as part of http request to validate authenticity of user. This string is not stored in database but can be stored on client side. JWT is made with a secret which is private to the user. Every request can be verified for authenticity through this secret key. Attempts to change the JWT will lead to verification failure. Hence the application can be made horizontally scalable with help of JWT. This resolves overhead due to accessing stored session cookie for every user for every request.

Github commit history