CMPE 273 Lab2
(Splitwise Application)

https://github.com/jayanpra/splitwise1

https://www.youtube.com/watch?v=BC-2QH3fxW8

Goals:

Necessity is the mother of invention. But there are some inventions that exist in crude form, which comes to life after a groundbreaking need. Such an invention is Splitwise.

The procedure of splitting the expenses to multiple persons can sometimes turn into a complex problem. Slightly more complex would be to record these transactions into a database and query from it. Machine/servers are having the capability to solve the problem into a user friendly application which hides all complexities of calculation and storing.

My goal in this lab is to recreate this application with partial features with the help of modern web technologies like React with Redux components on FrontEnd, Express Node Js on Backend, KafkaServer for services and MongoDb as Database.

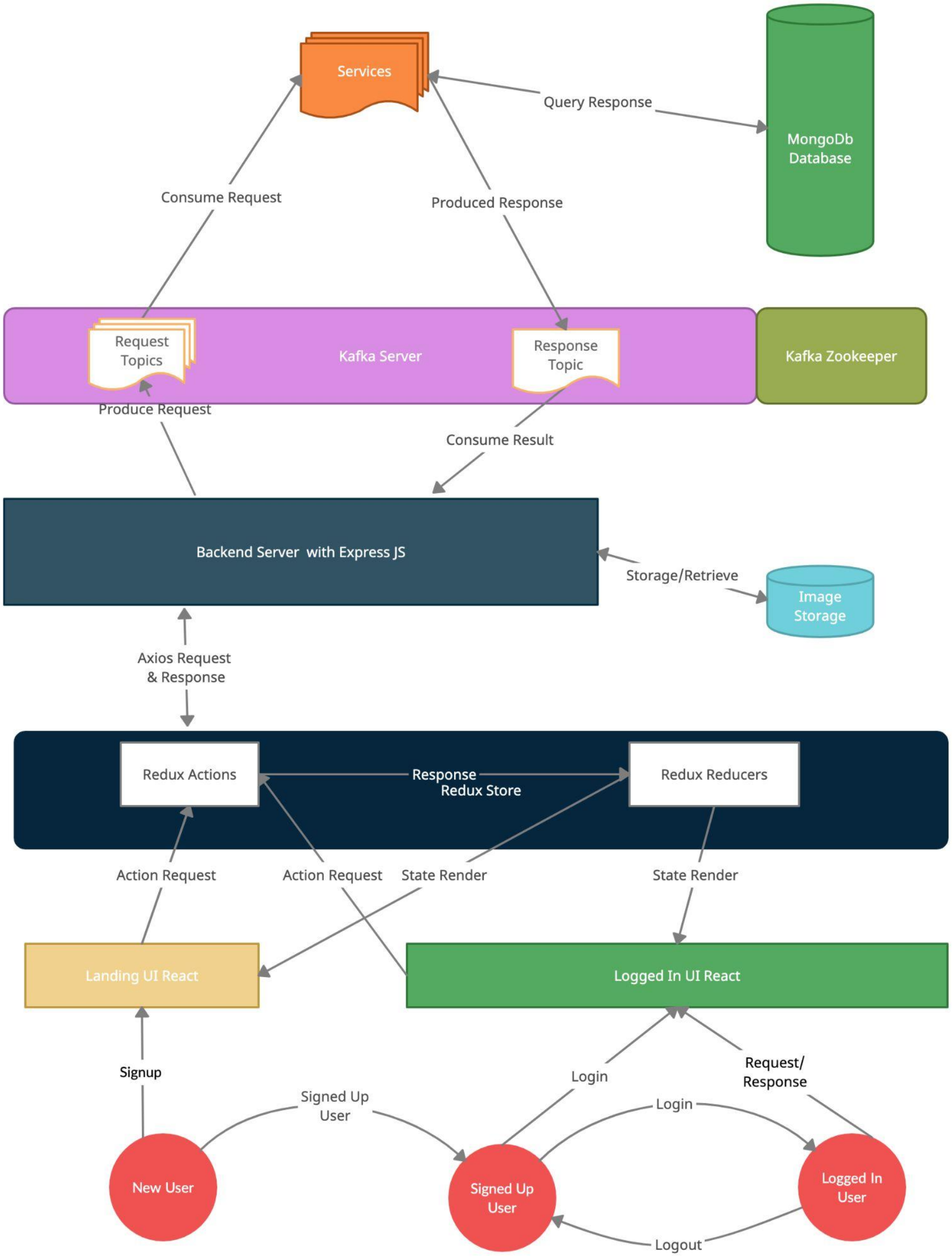
Purpose of the System:

This application is a partial implementation of the original Splitwise Application. Below are the points that can explain the purpose of the application:

- Storing the personal data and preferences of the persons using the application.
- Access control to restrict access to unauthorized users trying to view/corrupt valid transactions, the user is not authorized to do.
- Create groups of individuals, while taking care of individual’s consent to join the association.
- Adding expense in the group, viewed by other members.

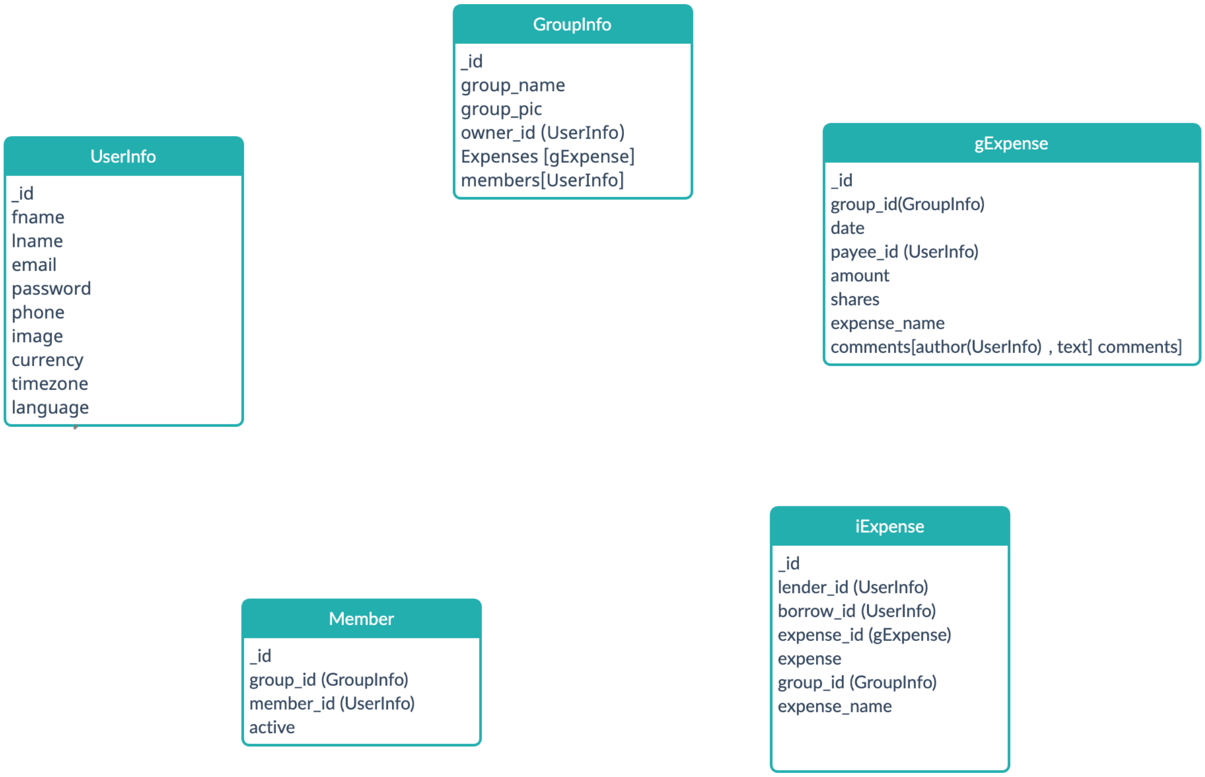
- Check balances and clear them to voluntarily exit a group.
- Add comments for expenses and discuss them.

System Design



Following steps can explain the flow of the application:

1. User signs up through a register page. The user info is stored with unique id and unique email. User is redirected to the dashboard after successful registration
2. Users can directly access the login page and enter the credentials.
3. The credentials are sent to the backend server via redux action methods, which in turn produces a request on Kafka Server. Kafka backend consumes the request and creates a JWT token for user id and sends it to client via Backend through Kafka Clusters. This token expires after some time and is used every time for accessing the server on user behalf.
4. User is redirected to the Dashboard page after successful authentication. He can switch to the profile page where he can change/add more info about himself.
5. Users can navigate to the group page to create a group and accept group invites. He can then add expenses on his active groups. Users can write comments on the expenses.
6. Users can view recent activity of all his groups on the recent activity page.
7. Users can check the expenses summary on the dashboard page, and can settle up with their expenses.



GroupInfo

_id
group_name
group_pic
owner_id (UserInfo)
Expenses [gExpense]
members[UserInfo]

UserInfo

_id
fname
lname
email
password
phone
image
currency
timezone
language

gExpense

_id
group_id(GroupInfo)
date
payee_id (UserInfo)
amount
shares
expense_name
comments[author(UserInfo) , text] comments]

iExpense

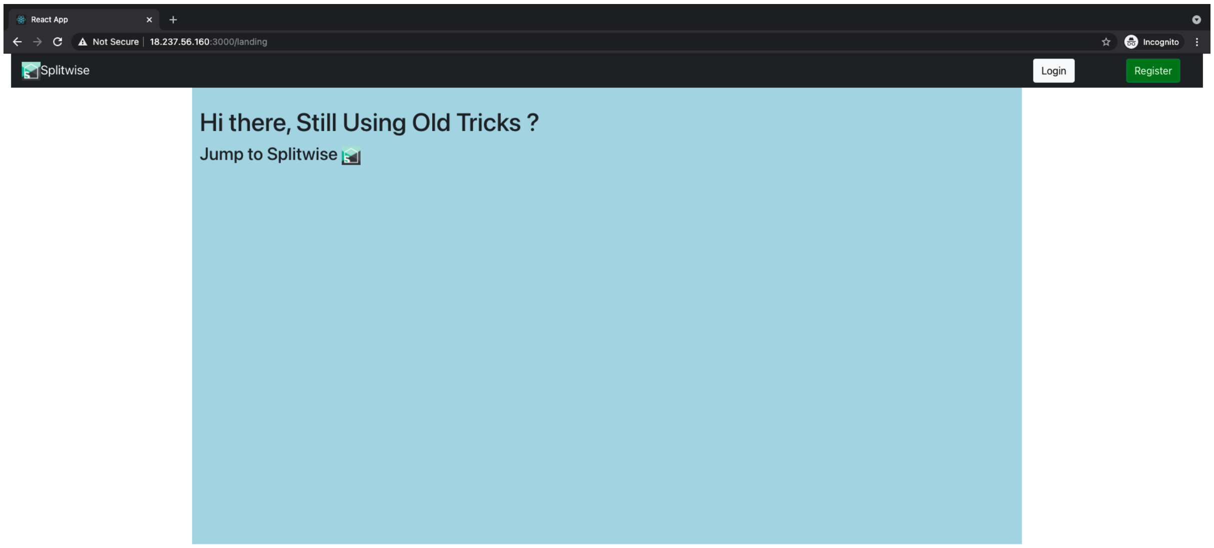
_id
lender_id (UserInfo)
borrow_id (UserInfo)
expense_id (gExpense)
expense
group_id (GroupInfo)
expense_name

Member

_id
group_id (GroupInfo)
member_id (UserInfo)
active

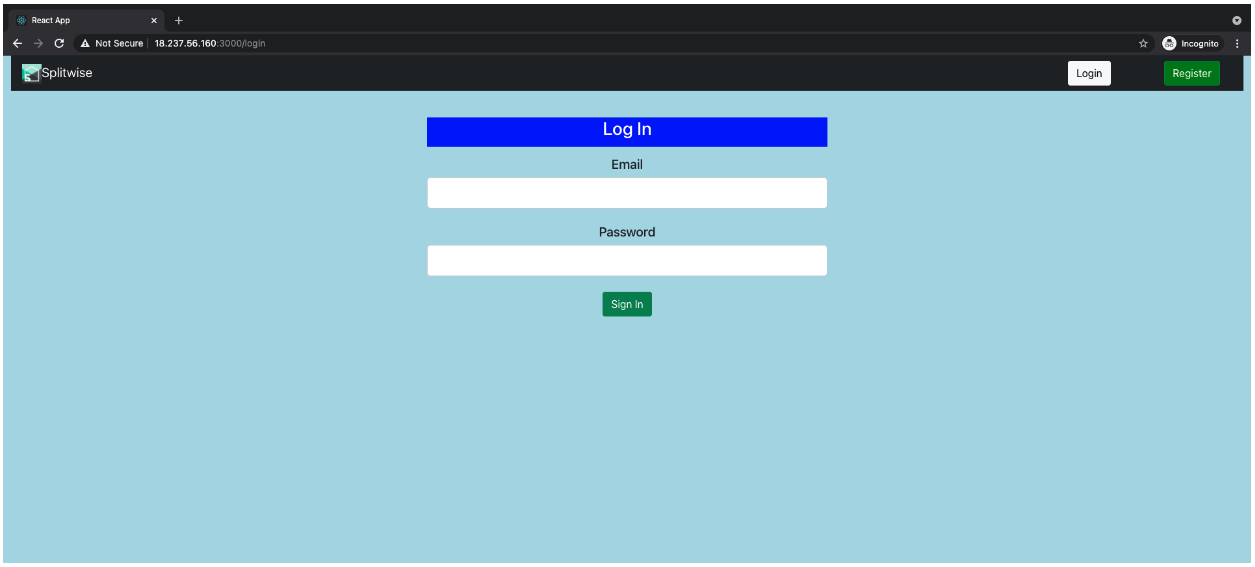
Result

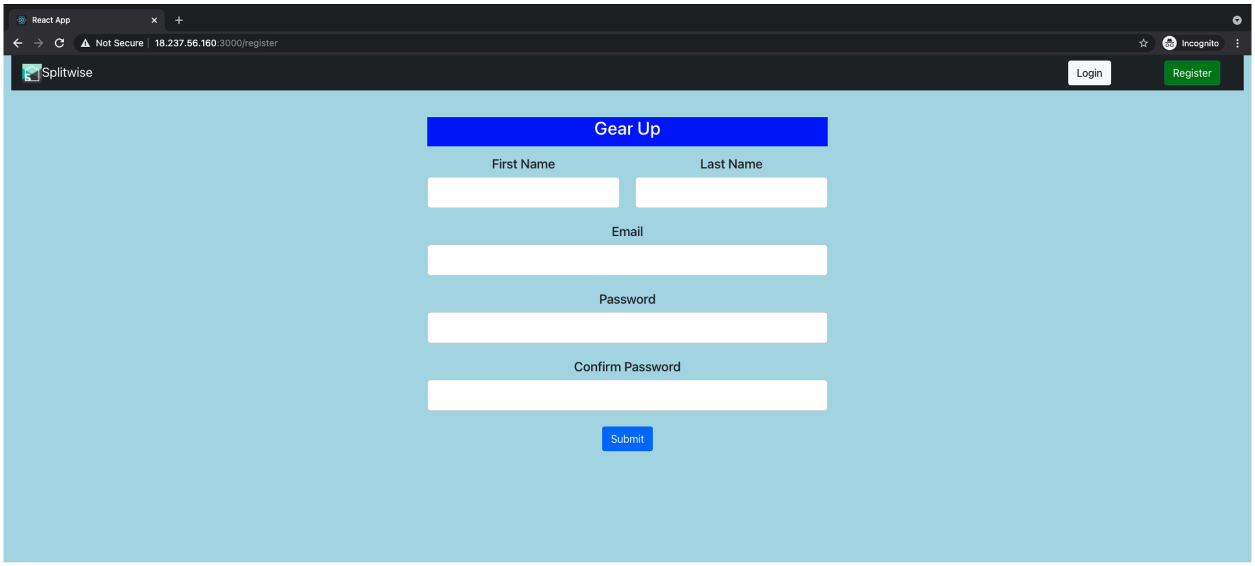
Below are the screenshots of the app:



Hi there, Still Using Old Tricks ?

Jump to Splitwise





Gear Up

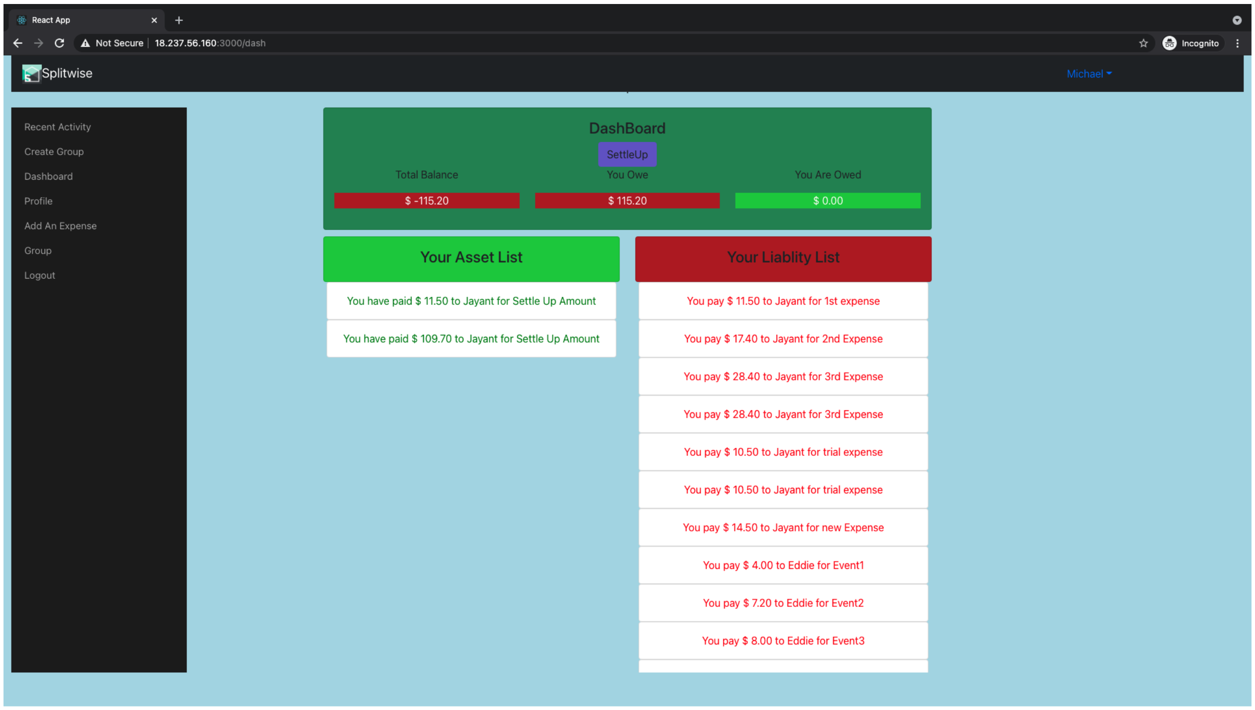
First Name

Last Name

Email

Password

Confirm Password



DashBoard

Total Balance	You Owe	You Are Owed
$ -115.20	$ 115.20	$ 0.00

Your Asset List

You have paid $ 11.50 to Jayant for Settle Up Amount

You have paid $ 109.70 to Jayant for Settle Up Amount

Your Liablity List

You pay $ 11.50 to Jayant for 1st expense

You pay $ 17.40 to Jayant for 2nd Expense

You pay $ 28.40 to Jayant for 3rd Expense

You pay $ 28.40 to Jayant for 3rd Expense

You pay $ 10.50 to Jayant for trial expense

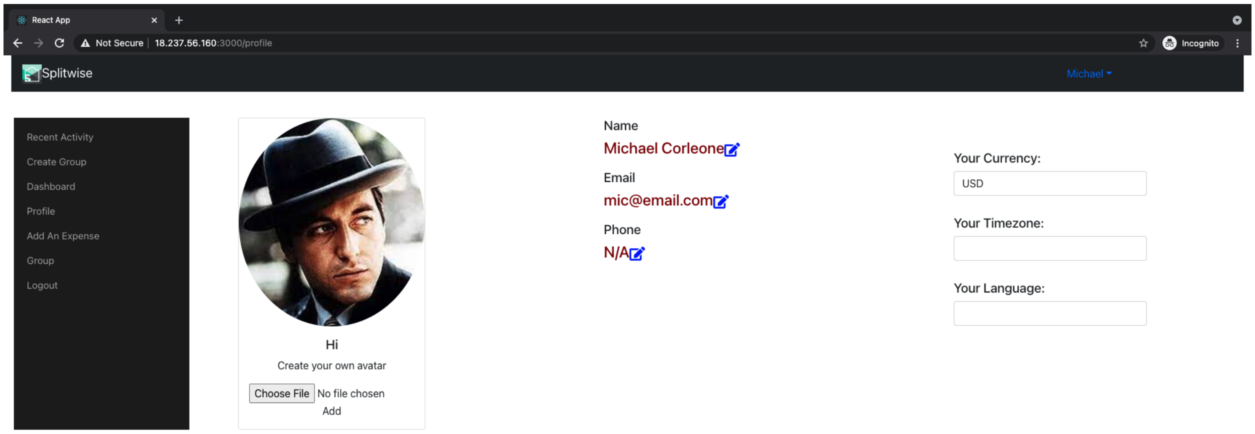
You pay $ 10.50 to Jayant for trial expense

You pay $ 14.50 to Jayant for new Expense

You pay $ 4.00 to Eddie for Event1

You pay $ 7.20 to Eddie for Event2

You pay $ 8.00 to Eddie for Event3



Hi

Create your own avatar

Choose File No file chosen

Add

Name

Michael Corleone

Email

mic@email.com

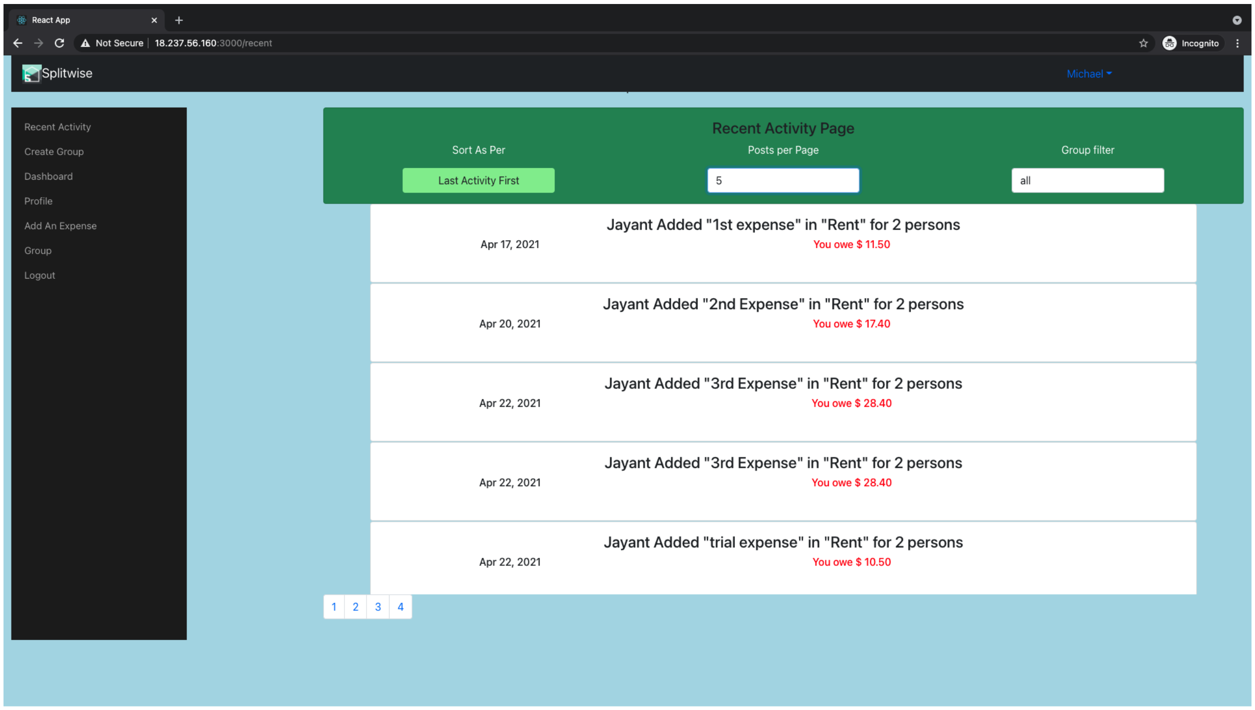
Phone

N/A

Your Currency: USD

Your Timezone:

Your Language:



Recent Activity Page

Sort As Per: Last Activity First

Posts per Page: 5

Group filter: all

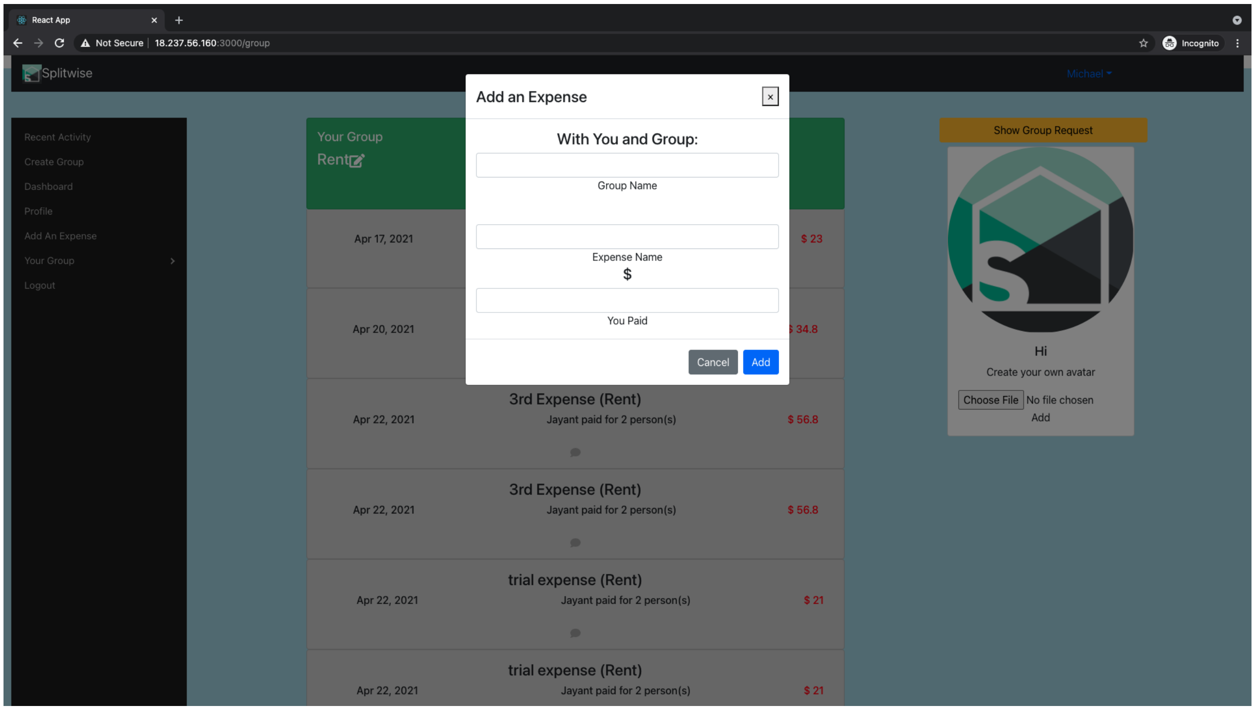
Apr 17, 2021 — Jayant Added "1st expense" in "Rent" for 2 persons — You owe $ 11.50

Apr 20, 2021 — Jayant Added "2nd Expense" in "Rent" for 2 persons — You owe $ 17.40

Apr 22, 2021 — Jayant Added "3rd Expense" in "Rent" for 2 persons — You owe $ 28.40

Apr 22, 2021 — Jayant Added "3rd Expense" in "Rent" for 2 persons — You owe $ 28.40

Apr 22, 2021 — Jayant Added "trial expense" in "Rent" for 2 persons — You owe $ 10.50



Your Group

Rent

Apr 17, 2021 $ 23

Apr 20, 2021 $ 34.8

Apr 22, 2021 3rd Expense (Rent) Jayant paid for 2 person(s) $ 56.8

Apr 22, 2021 3rd Expense (Rent) Jayant paid for 2 person(s) $ 56.8

Apr 22, 2021 trial expense (Rent) Jayant paid for 2 person(s) $ 21

Apr 22, 2021 trial expense (Rent) Jayant paid for 2 person(s) $ 21

Add an Expense

With You and Group:

Group Name

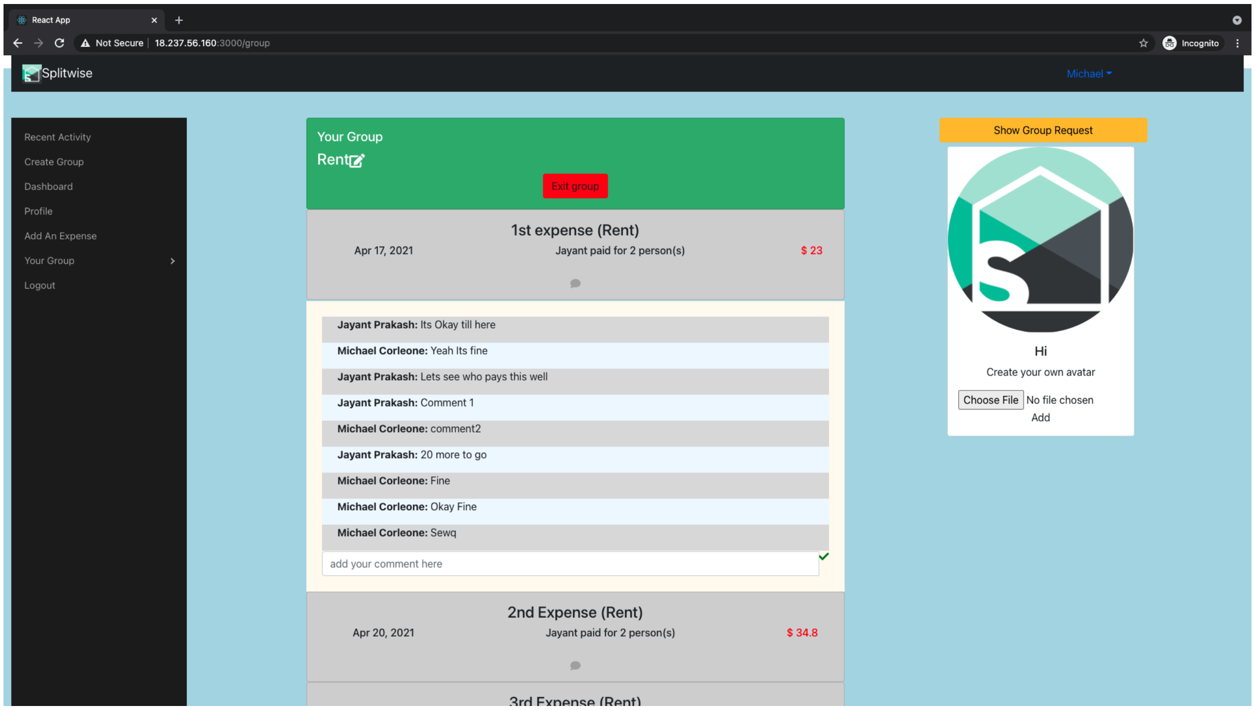
Expense Name

$

You Paid

Hi

Create your own avatar



1st expense (Rent)

Apr 17, 2021 | Jayant paid for 2 person(s) | $ 23

Jayant Prakash: Its Okay till here

Michael Corleone: Yeah Its fine

Jayant Prakash: Lets see who pays this well

Jayant Prakash: Comment 1

Michael Corleone: comment2

Jayant Prakash: 20 more to go

Michael Corleone: Fine

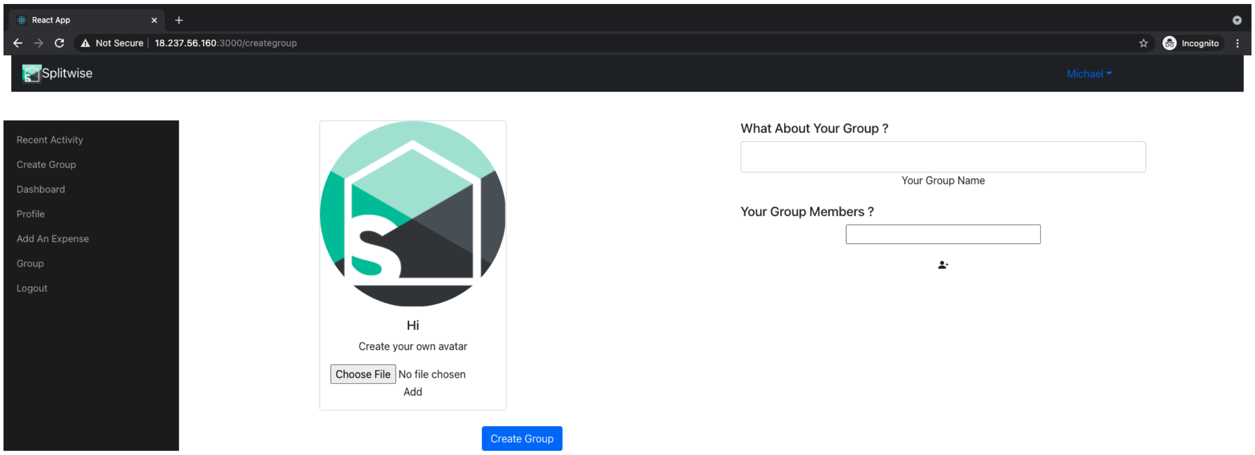
Michael Corleone: Okay Fine

Michael Corleone: Sewq

2nd Expense (Rent)

Apr 20, 2021 | Jayant paid for 2 person(s) | $ 34.8

3rd Expense (Rent)



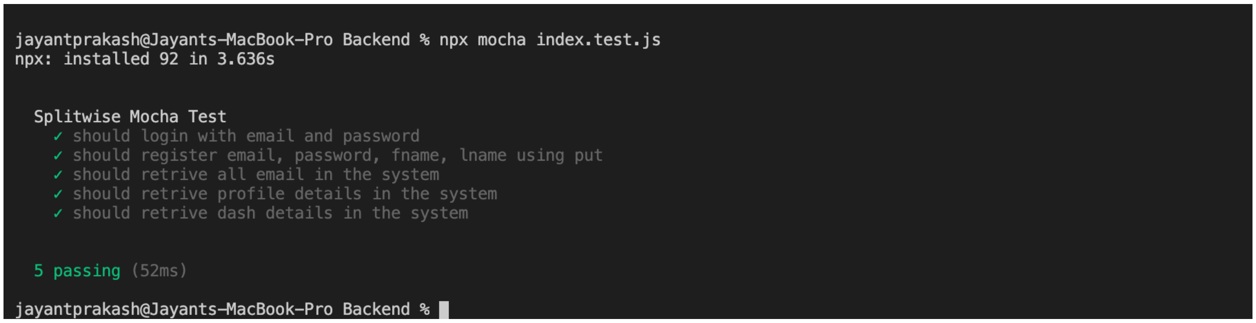
What About Your Group ?

Your Group Name

Your Group Members ?

Testing

Below are the screenshot of Mocha Backend testing result:



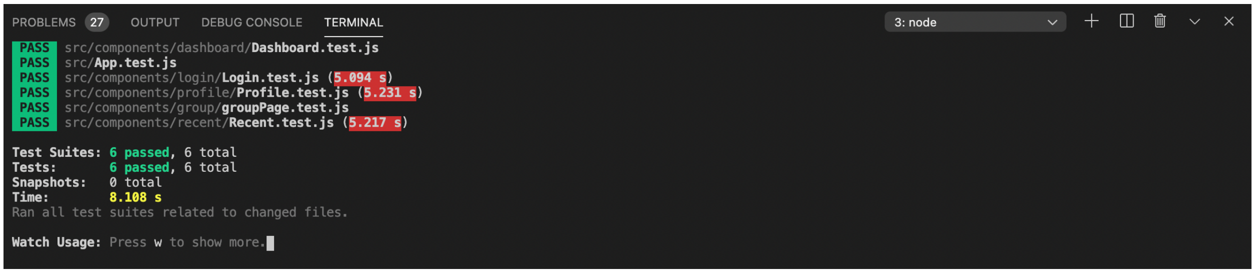
```
jayantprakash@Jayants-MacBook-Pro Backend % npx mocha index.test.js
npx: installed 92 in 3.636s


  Splitwise Mocha Test
    ✓ should login with email and password
    ✓ should register email, password, fname, lname using put
    ✓ should retrive all email in the system
    ✓ should retrive profile details in the system
    ✓ should retrive dash details in the system


  5 passing (52ms)

jayantprakash@Jayants-MacBook-Pro Backend %
```

Below is the Screen shot of the output from React Testing Library:



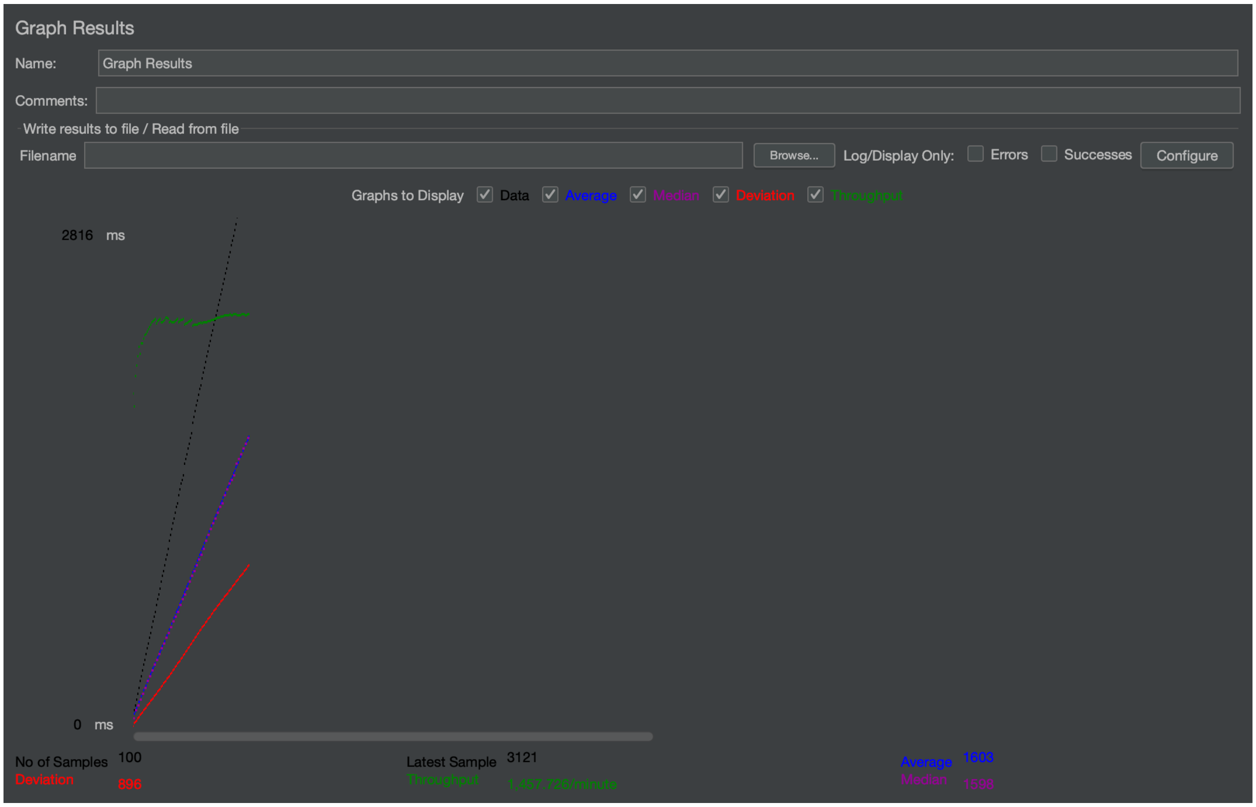
```
 PASS  src/components/dashboard/Dashboard.test.js
 PASS  src/App.test.js
 PASS  src/components/login/Login.test.js (5.094 s)
 PASS  src/components/profile/Profile.test.js (5.231 s)
 PASS  src/components/group/groupPage.test.js
 PASS  src/components/recent/Recent.test.js (5.217 s)

Test Suites: 6 passed, 6 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        8.108 s
Ran all test suites related to changed files.

Watch Usage: Press w to show more.
```

Screenshot of jMeter testing for Mongodb without connection pool:

1. 100users



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 100

Deviation 896

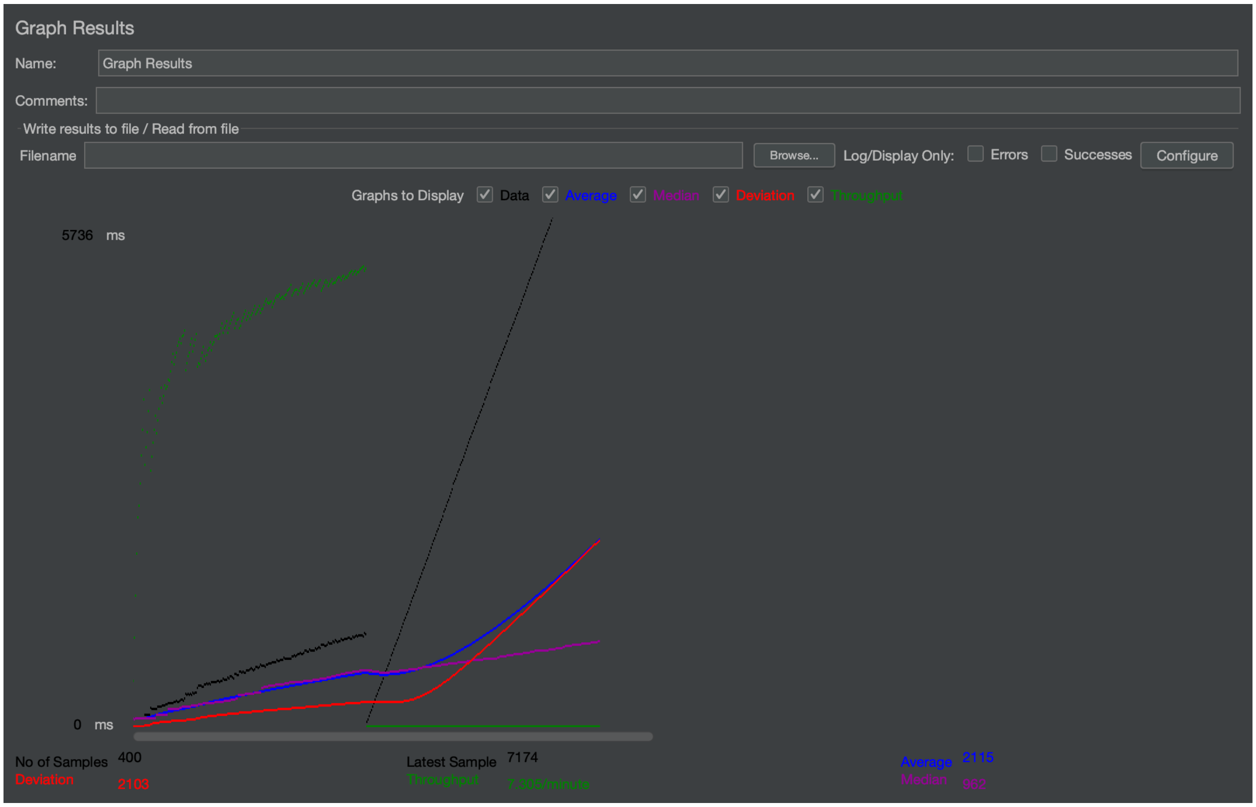
Latest Sample 3121

Throughput 1,457.726/minute

Average 1603

Median 1598

2. 200users



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

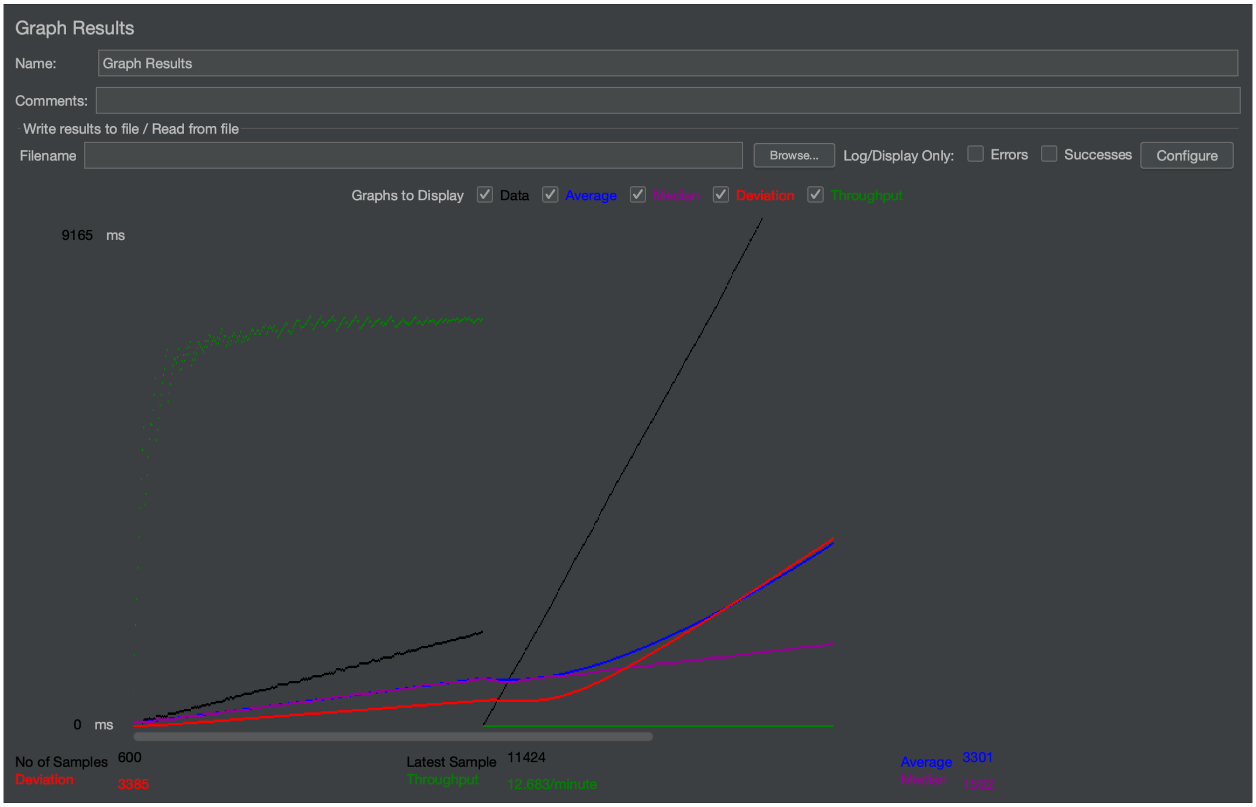
Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 400
Deviation 2103

Latest Sample 7174
Throughput 7.305/minute

Average 2115
Median 962

3. 300users



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

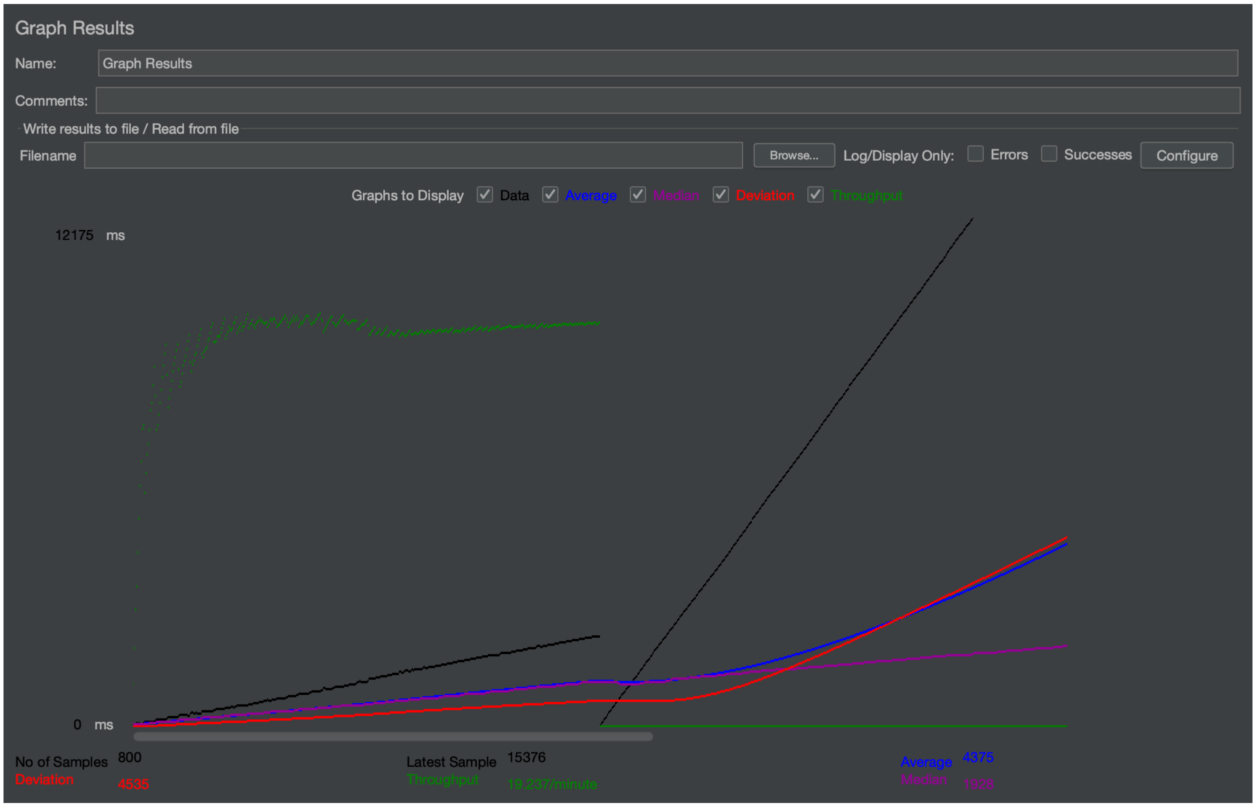
Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 600
Deviation 3385

Latest Sample 11424
Throughput 12.683/minute

Average 3301
Median 1502

4. 400users



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

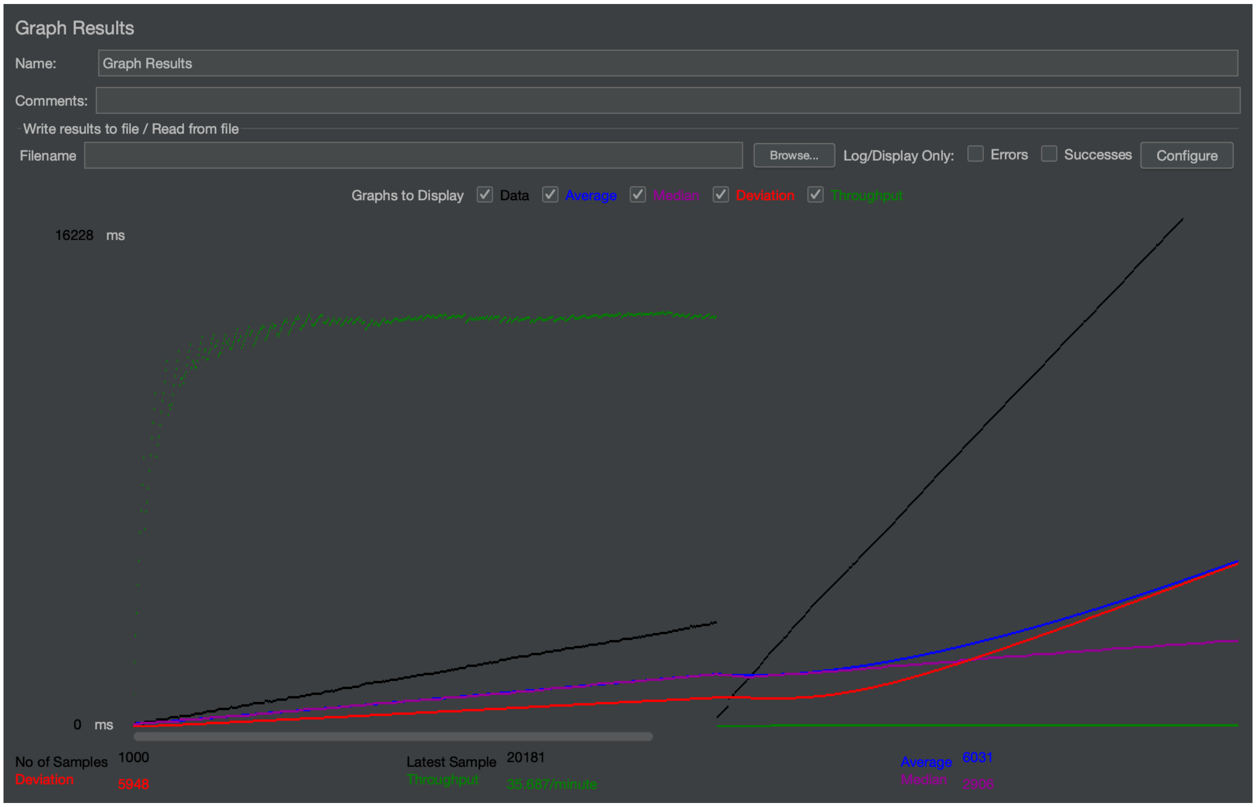
Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 800
Deviation 4535

Latest Sample 15376
Throughput 19.237/minute

Average 4375
Median 1928

5. 500users



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

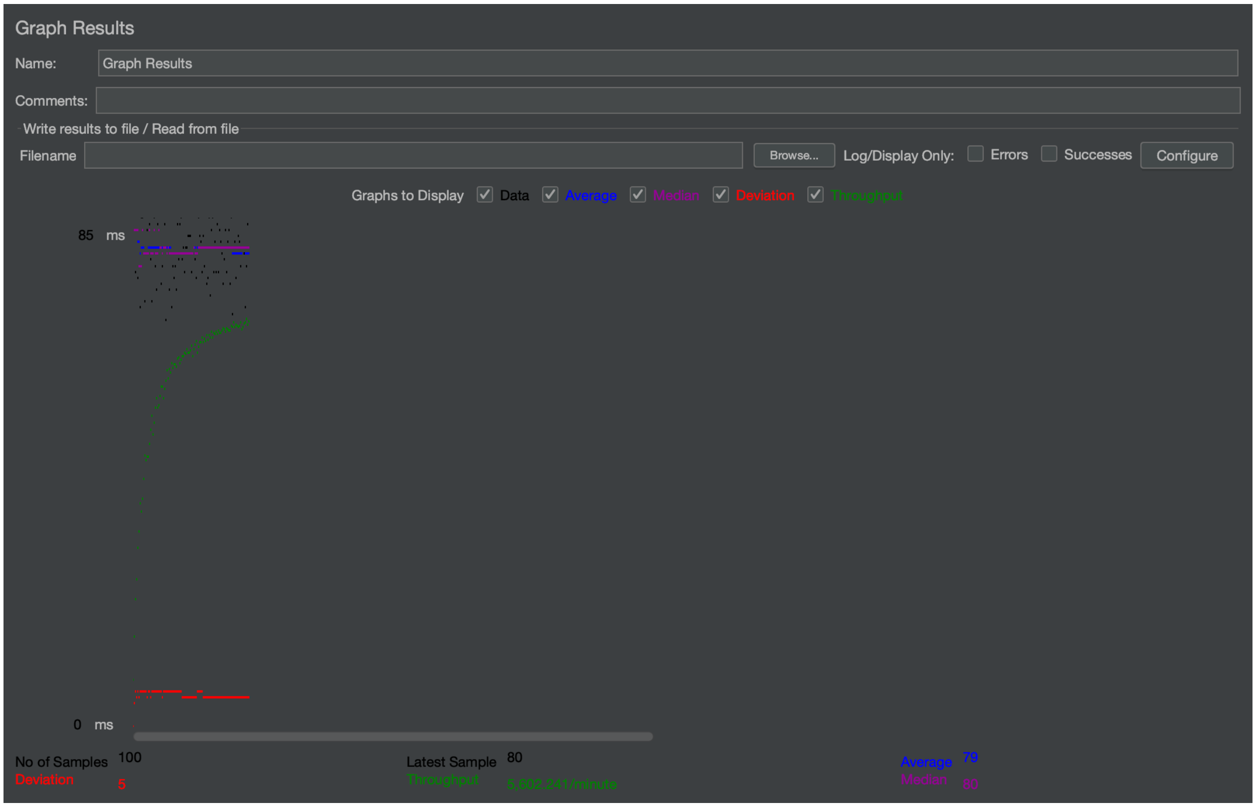
No of Samples 1000
Deviation 5948

Latest Sample 20181
Throughput 35.687/minute

Average 6031
Median 2906

Screenshots of jMeter testing for Mongodb without connection pool:

1. 100users



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

No of Samples 100

Deviation 5

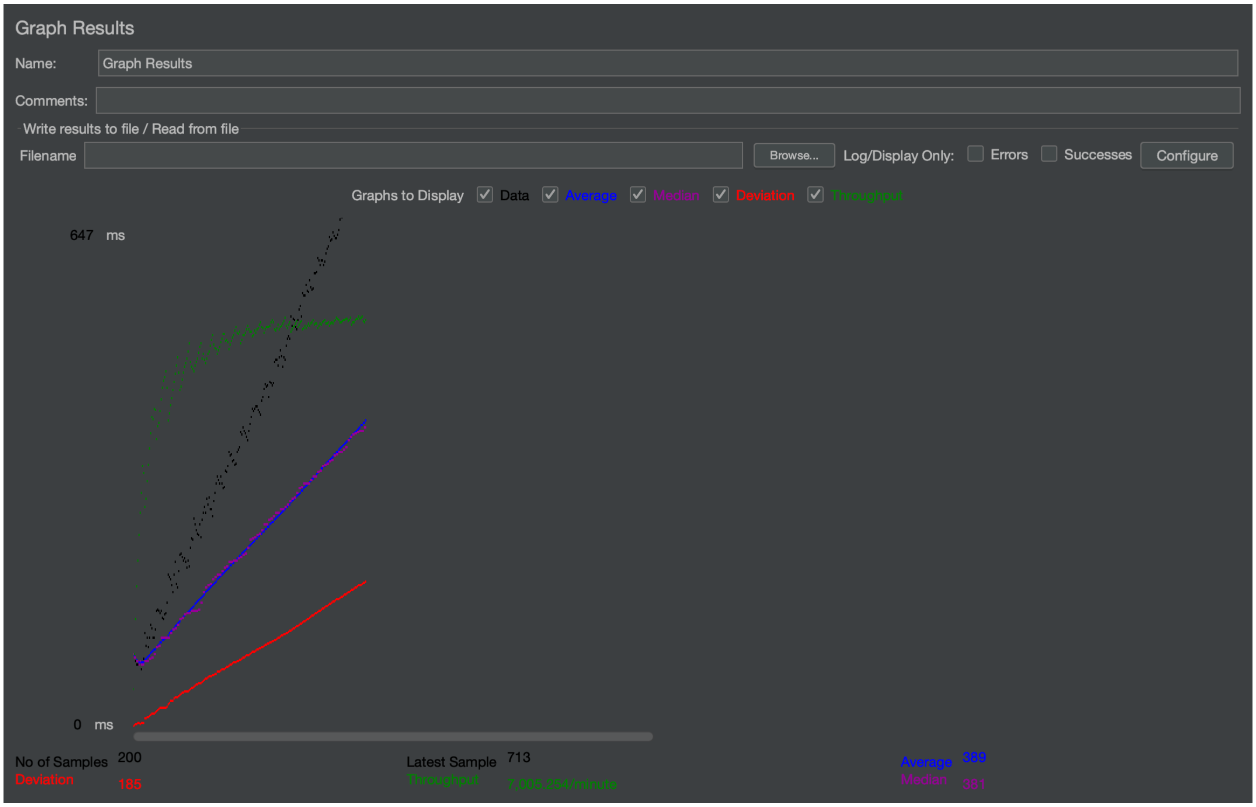
Latest Sample 80

Throughput 5,602.241/minute

Average 79

Median 80

2. 200users



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

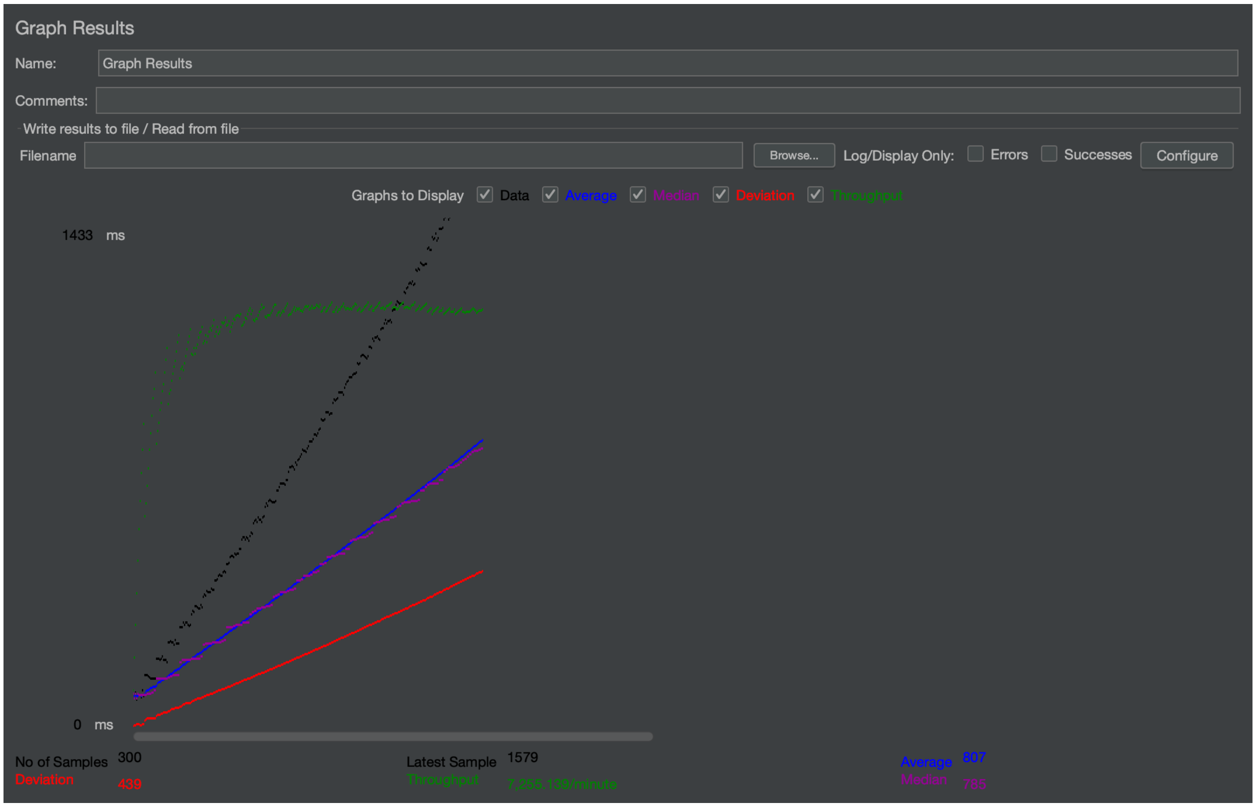
Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 200
Deviation 185

Latest Sample 713
Throughput 7,005.254/minute

Average 389
Median 381

3. 300users



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

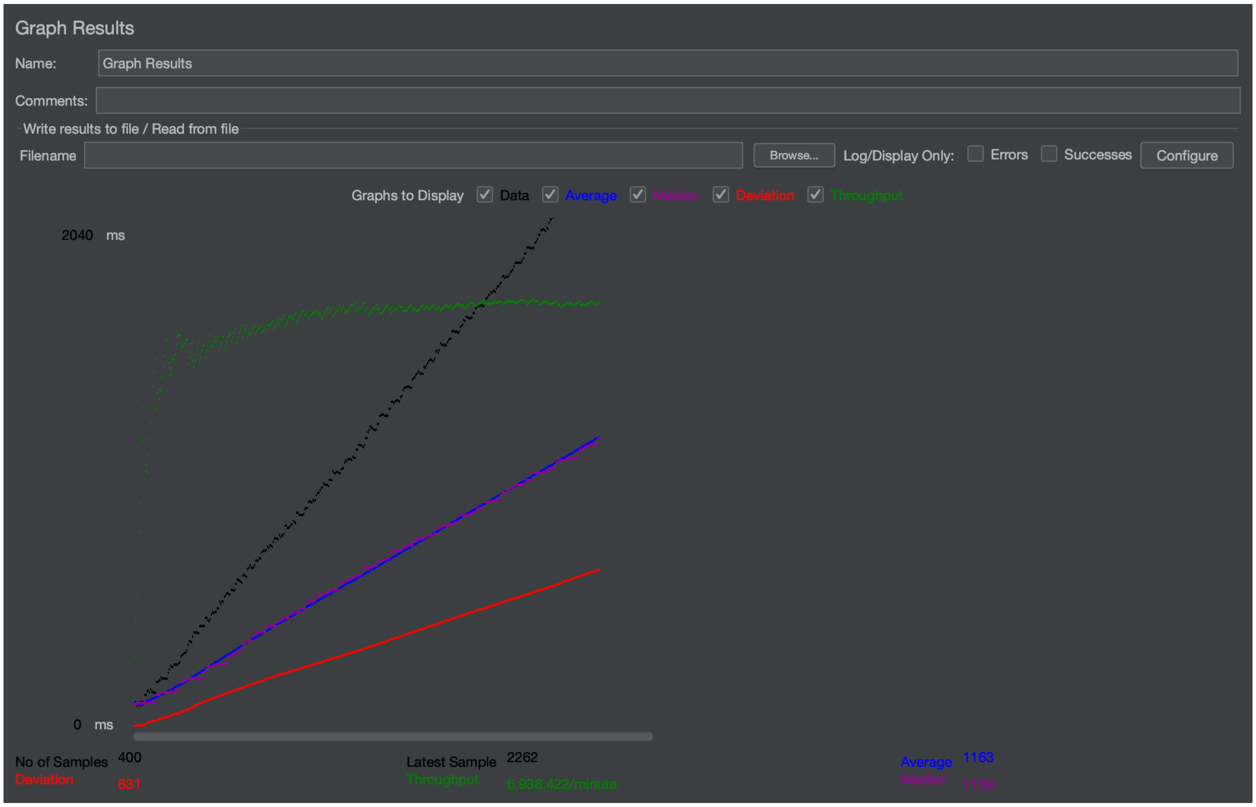
Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 300
Deviation 439

Latest Sample 1579
Throughput 7,255.139/minute

Average 807
Median 785

4. 400users



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

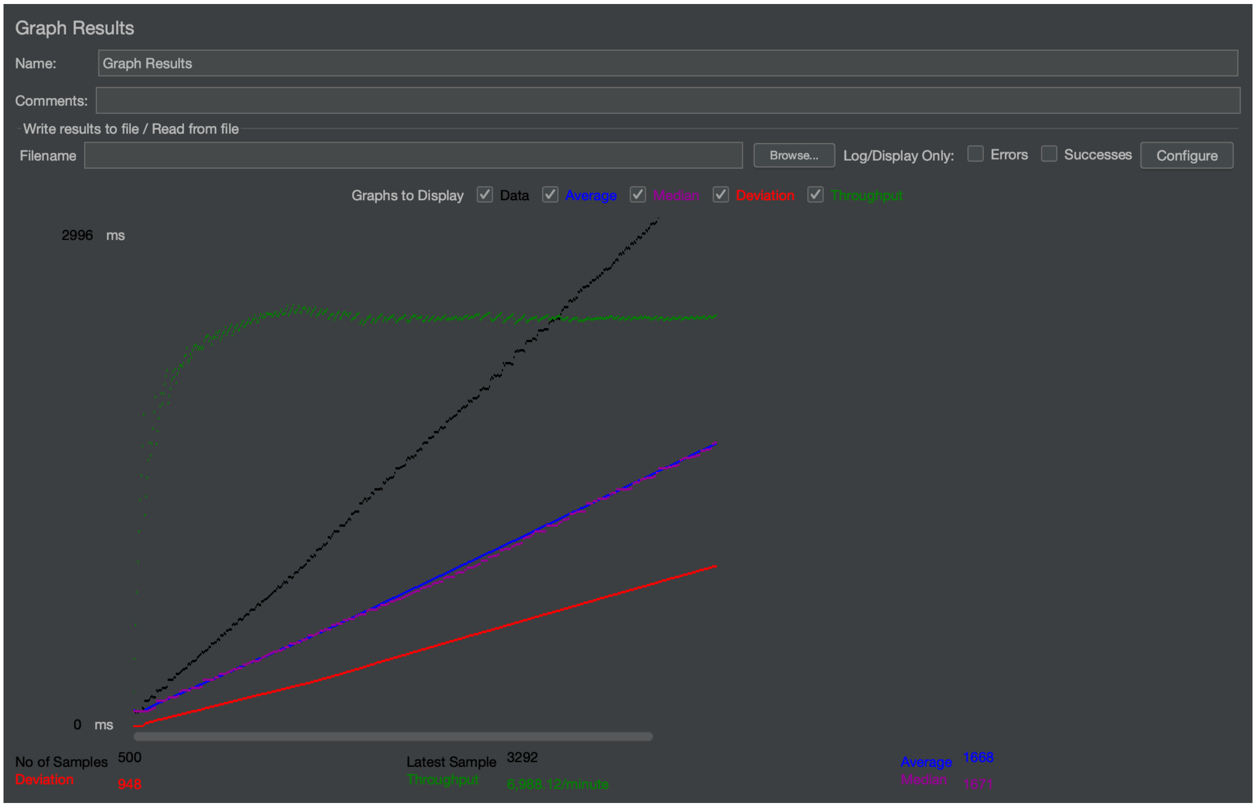
Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 400
Deviation 631

Latest Sample 2262
Throughput 6,938.422/minute

Average 1163
Median 1159

5. 500users



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 500
Deviation 948

Latest Sample 3292
Throughput 6,988.12/minute

Average 1668
Median 1671

Answer the Question

1. Compare passport authentication process with the authentication process used in Lab1.

In Lab 1 we implemented an authentication mechanism using Jwt cookie, which was generated by a backend server during register/login. The cookie was then sent to the client which passed the session id for authentication and authorization to interact with the information available.

For a more secure way of sharing token and controlling transactions, passport-jwt was used in lab2 of the project. Passport-jwt takes inspiration from jsonwebtoken to validate signed tokens. It serves as a middleware for Node for verification purposes.

Following are the steps in which describes the working Passport JWT vs traditional authentication:

- In lab1 User send credentials and it is verified and the cookie with session id is stored in local storage of client browser, whereas in Passport jwt token is stored in client browser's local storage.
- Each subsequent request in lab1 was subjected to a session id, and response is given out after its verification. Each token is encrypted with a secret key which resides in the server, and everytime a request is sent with the token, the token is decrypted and verified via checkAuth middleware provided by Passport js.
- After logging out from the session, cookies having session id needs to be destroyed from the client as well as server. So in case the server is unavailable, you technically cannot logout. Whereas in Passport jwt, the client needs to delete the jwt token, which is final.

2. Compare performance with and without Kafka. Explain in detail the reason for difference in performance ?

As the application becomes large and more and more services are getting added regularly, it is inefficient to have all end points in the backend file. For scalability reasons, it is better to have a Kafka broker cluster which maps incoming requests to a large number of scalable services present for the application.

Below points explain the difference in performance that can be seen with and without kafka:

- Performance may be better for small scale applications without a kafka broker because the significant extra processing of kafka with small sets of APIs and added wait time degrades the performance of the App.

- A traditional message broker is faster than a kafka broker but when the application scales horizontally, and more and more requests flow into the server the performance degrades for the traditional broker.
- In large scale enterprises, Kafka enhances distributed architecture as the load is less in absence of indexes. This is why no random access is available in kafka which undermines the involvement with indexes.

3. If given an option to implement MySQL and MongoDB both in your application, specify which part of data of the application you will store in MongoDB and MySQL respectively

If given an option to use both databases for an application, I will use SQL for structured data where horizontal scaling of information is less likely. And I will use MongoDb for dynamic data handling.

Following data I will prefer to store in MySQL:

- UserInfo: Since the UserInfo table has limited information and it is referenced very frequently. So this will likely to lie in MySQL bucket.
- iExpense(Individual Expense): This table is more of a register and not much changes need to be added to the table once created.
- Member: This table contains all group requests, it is only meant to read and delete in case of approval.

Following data I will prefer to store in MongoDb:

- groupInfo: GroupInfo collection is very dynamic as it contains members, adds new members and deletes existing members, in addition to that cumulative expenses are getting added to the group over time. So this will go in MongoDb.
- gExpense(Group Expense) : Group expense has to be in MySQL, but with the introduction of comments on expenses, the dynamicality of this collection heeds to the concept of Mongodb.