CMPE 273 – Enterprise Distributed Systems
Lab2: Splitwise Assignment

YouTube Link for Splitwise Application: https://youtu.be/UHoztbcOd1s

AWS Link for Splitwise Application: http://3.238.230.69:3000/

Github Link : https://github.com/Swes-sjsu/Final_Lab2_Splitwise

Name: Swetha Singi Reddy
SJSU ID: 015354015

Introduction

Splitwise is a widely used application that facilitates bill management where the bills are split among the members of a group. It also keeps track of bills paid and who owes how much. In this Lab2 assignment we will be building a prototype similar to Splitwise application where bills are split equally among the members of a group once they become part of the group. We will be using NodeJs, React, MongoDb, Redux and Kafka for implementation.

Goal

The goal of this project is to build a prototype like Splitwise using Nodejs, React and MongoDb.

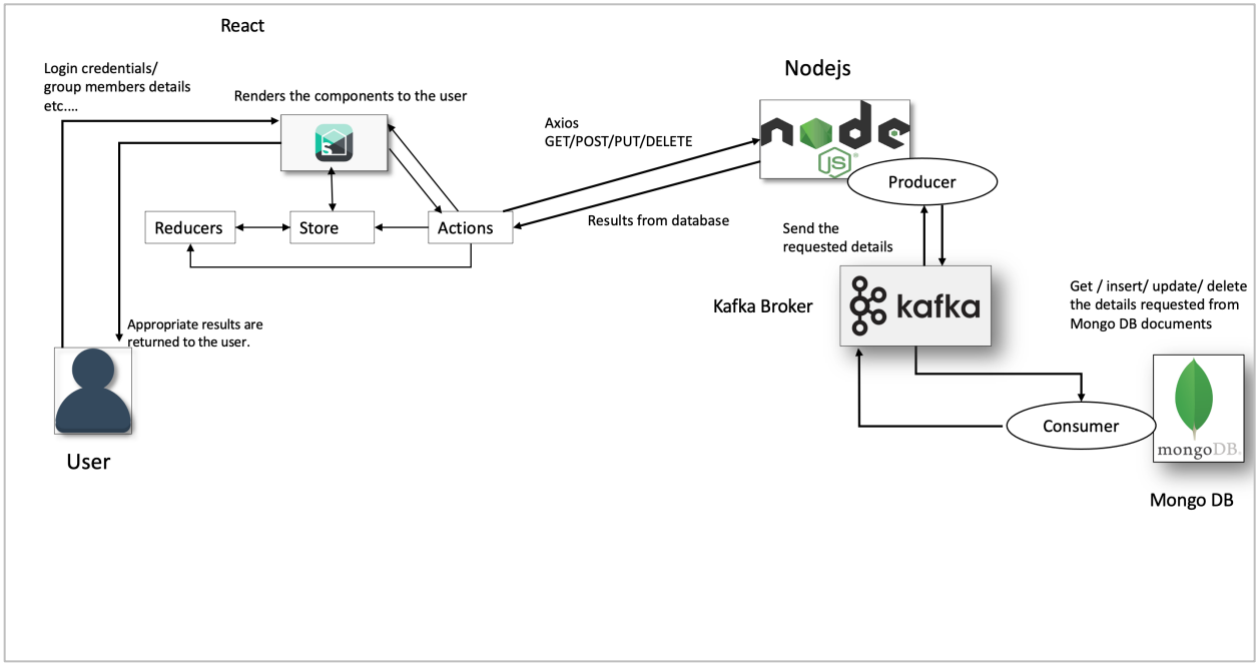
The application will have the following features:

- Users will be able to signup to the application by providing their full name, email address and password.
- Users then will be able to login to the application.
- Users can update their account details.
- Users will have access to a dashboard page as soon they are logged in which provides users with a summary of "who owes who" in total and across the groups.
- Users can create a group where they can add a group member by selecting from a list of dropdown members who are registered users.
- Users will be sending an invitation to other users to join the group.
- Users can either accept or deny the invitation. If they accept, they become part of the group.
- "My Groups" page is an addition to the original application where it displays the list of groups, they are part of and the list of groups they are invited to. They can navigate to the groups they are part of or create a new group. They can also accept or reject an invitation.
- Once the invite is accepted the users will be navigated to the group page which displays the list of transaction in the group along with the summary of "who owes who" in the group.
- Multiple users will be able to add a bill that gets split equally. Additionally, the users can add comments to the expenses. Users can remove the comments they have added as well.
- Recent activity page will give you all the details of the transactions for that user. Pagination is implemented making it more readable.
- From dashboard the users can settle up the balances between them.

» Users will be able to log out of the application.

System Design

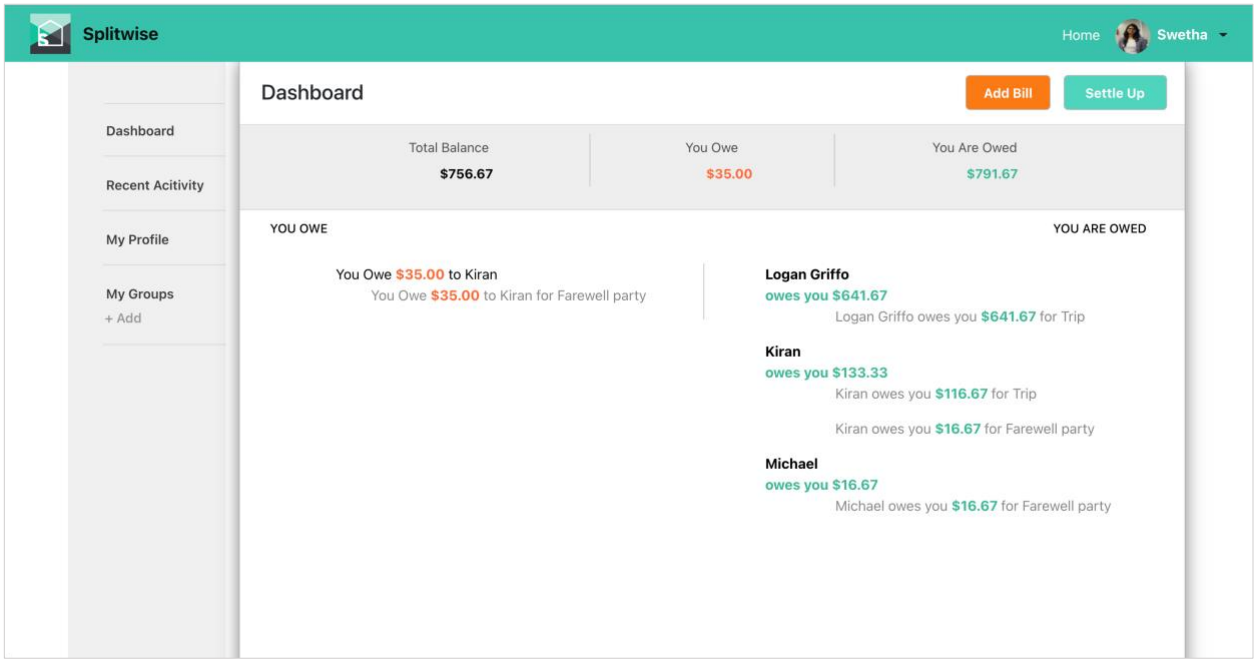
This Application uses a simple distributed architecture with React as the front end consisting of around 10 components and 18+ Kafka topics and APIs.



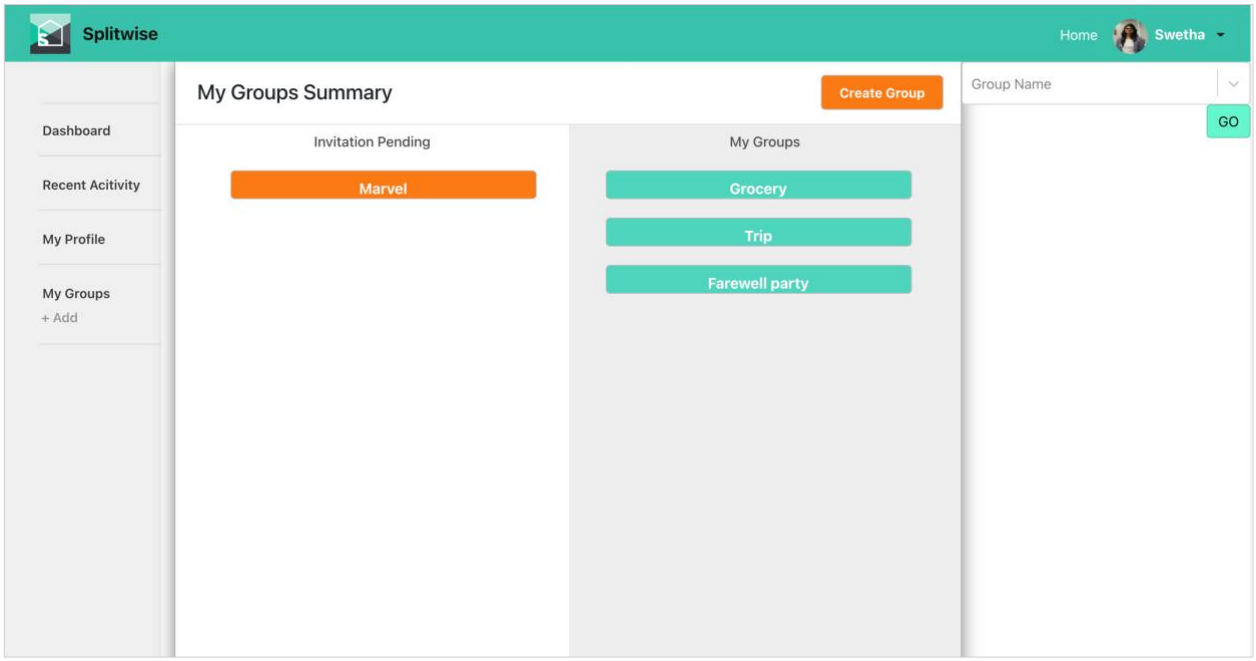
React, Redux, React Router, JavaScript is used for frontend. Express and Node Js for backend, Kafka as the messaging queue. Passport and JWT are used as authentication middleware. Mongo DB is used as the database in this application.

Pages in the application are as follows:

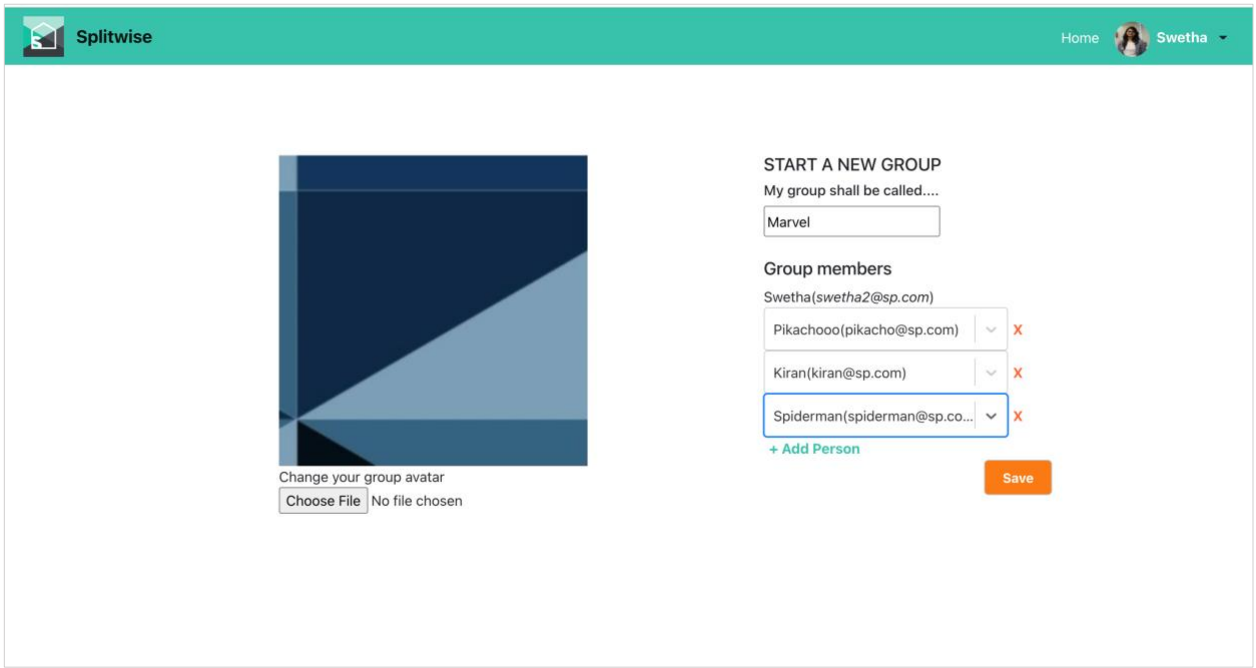
Dashboard Page



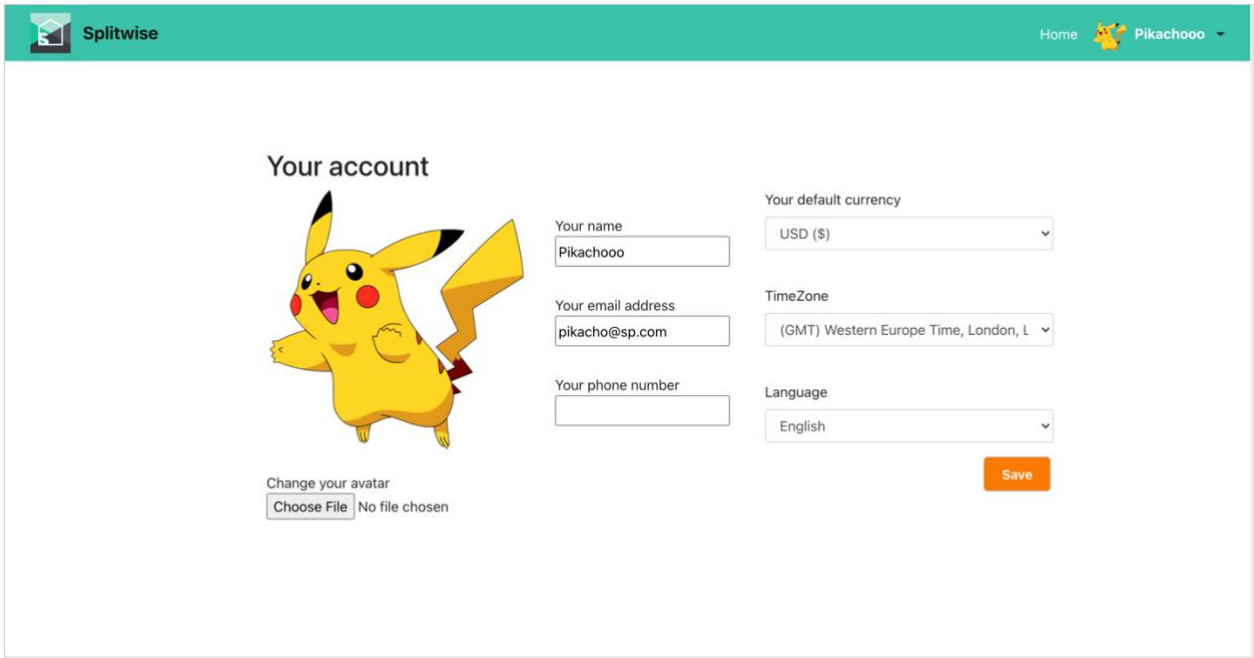
My Groups



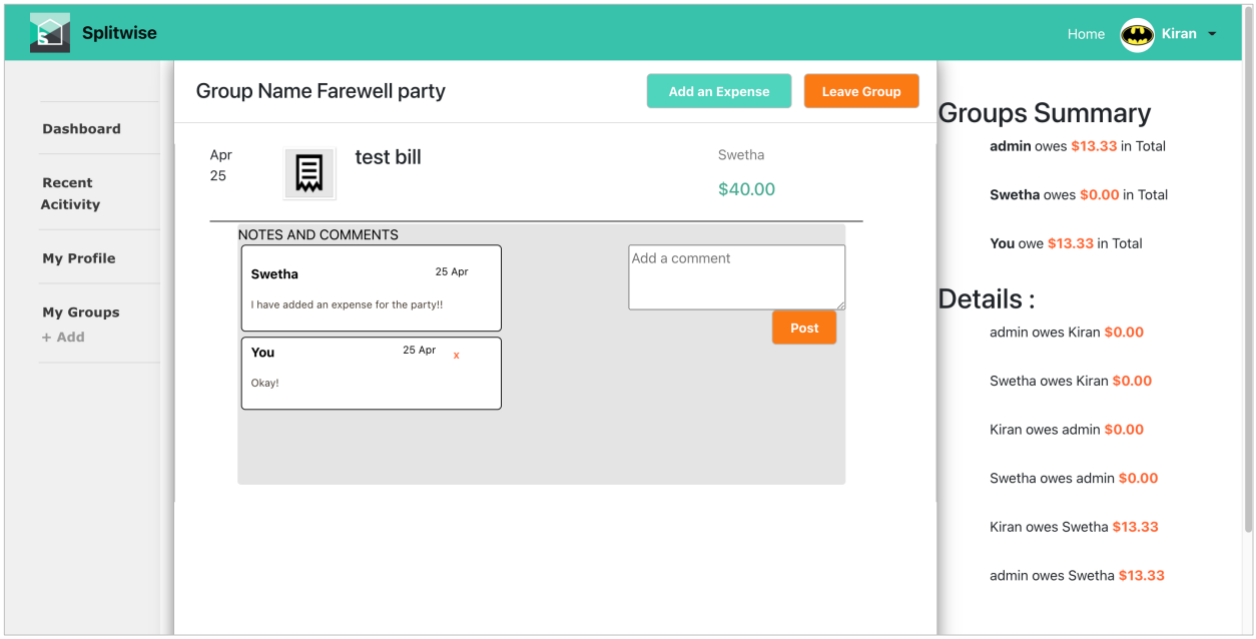
Create group



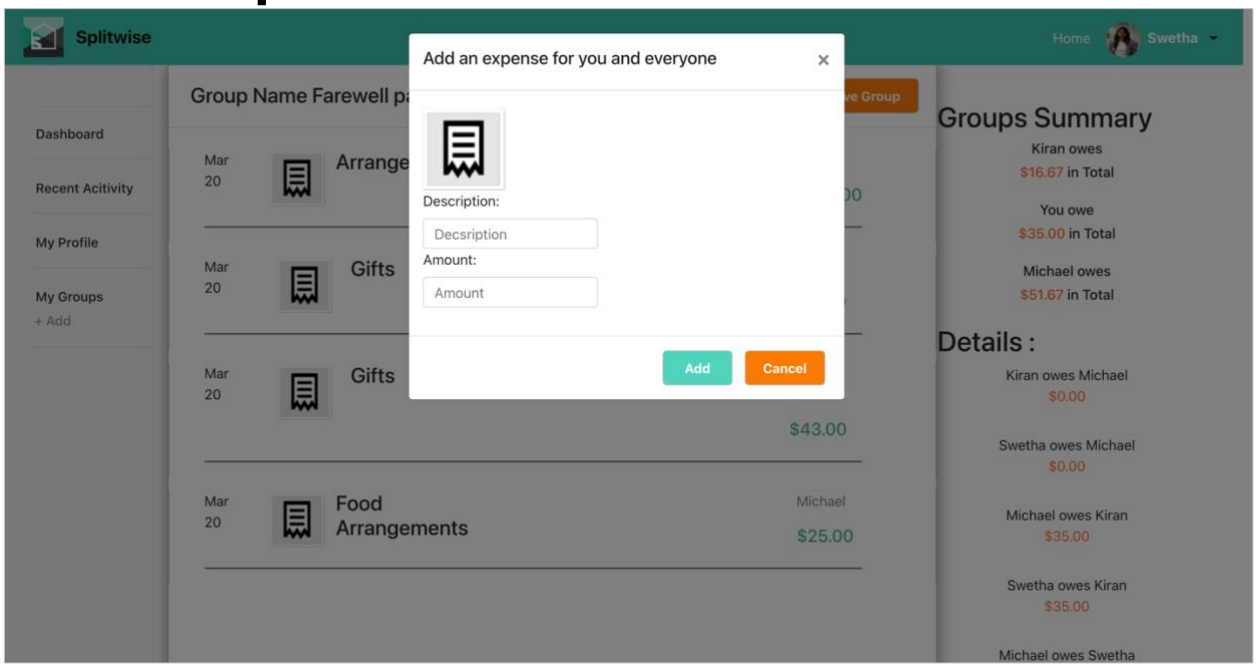
Update profile



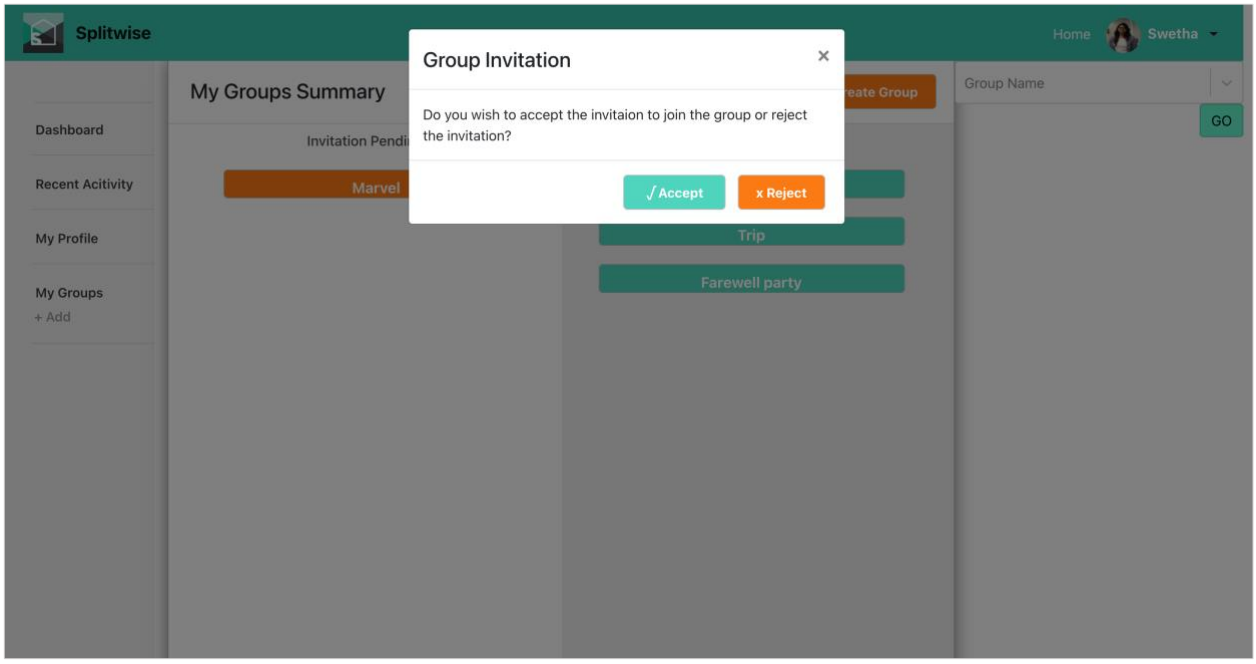
Group page



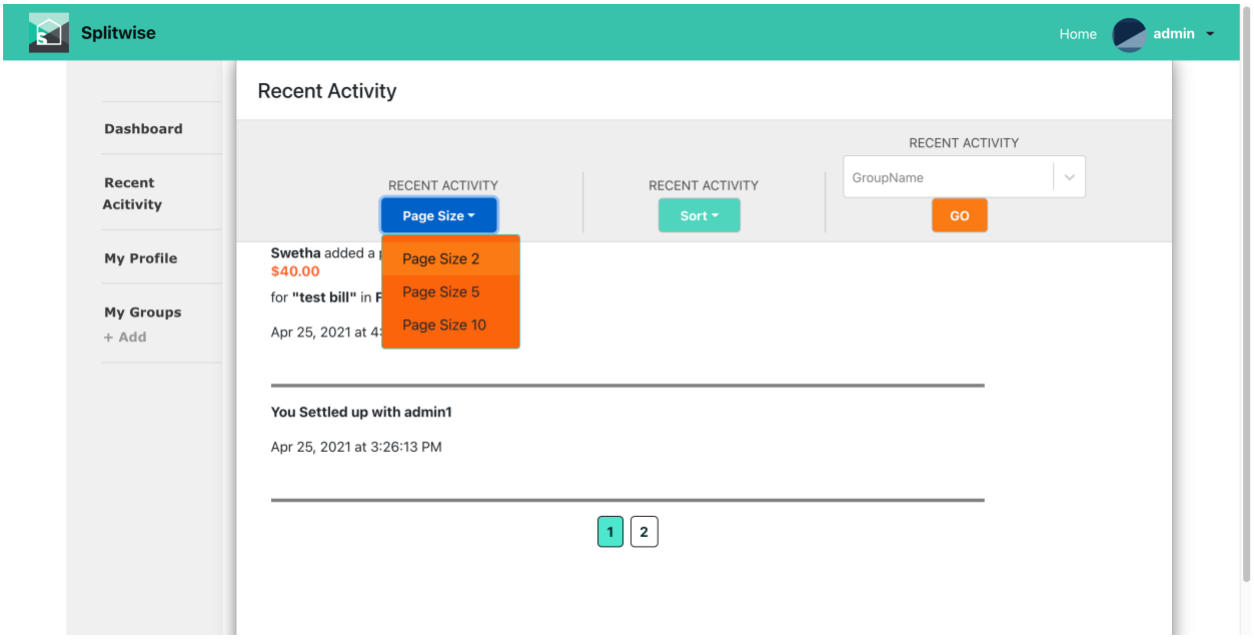
Add an expense



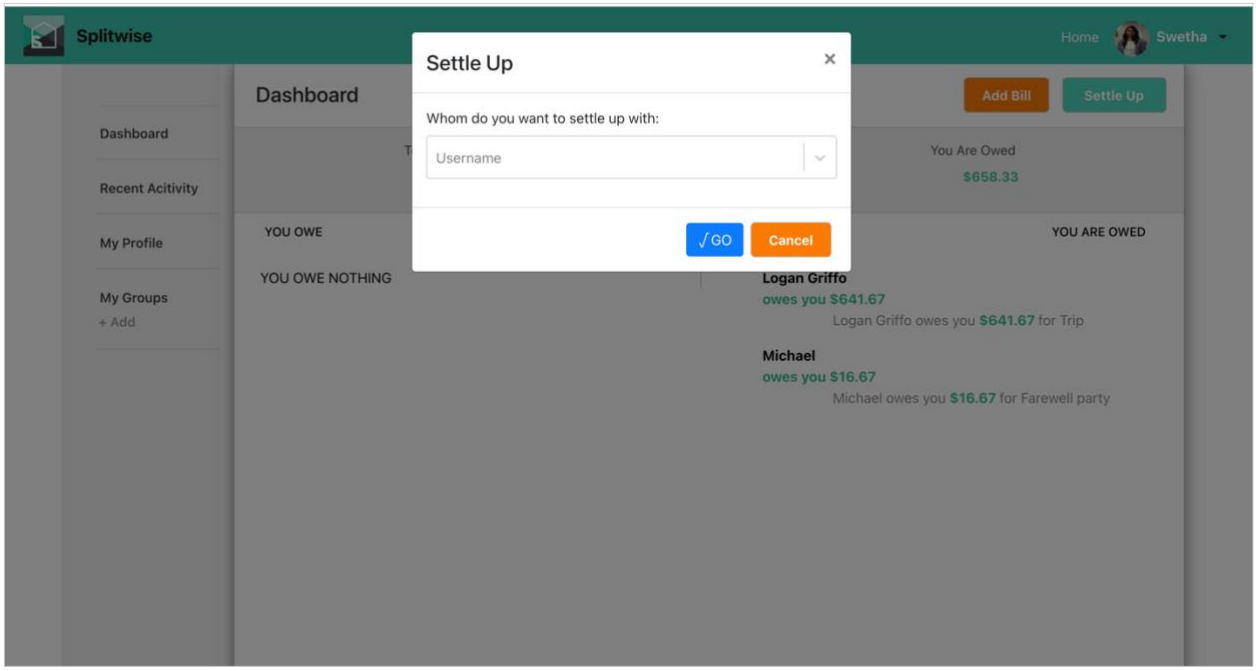
Accept/deny invitation



Recent activity



Settle up



Testing

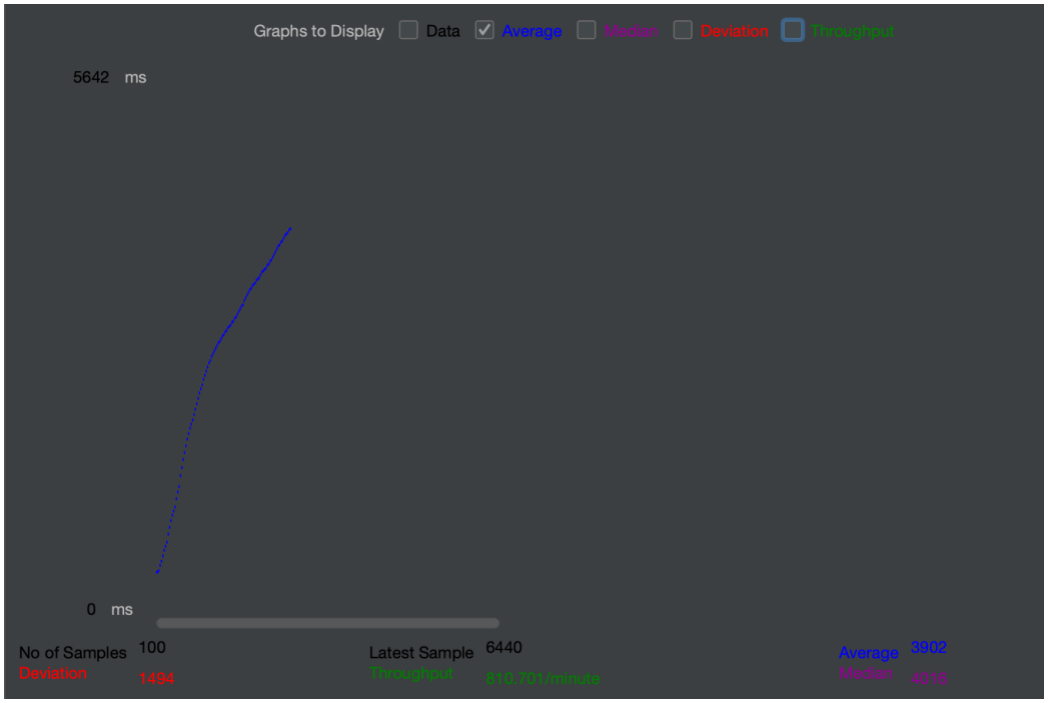
Backend services were tested using Jmeter and Mocha.
Below are the test results.

<u>Jmeter Testing:</u>

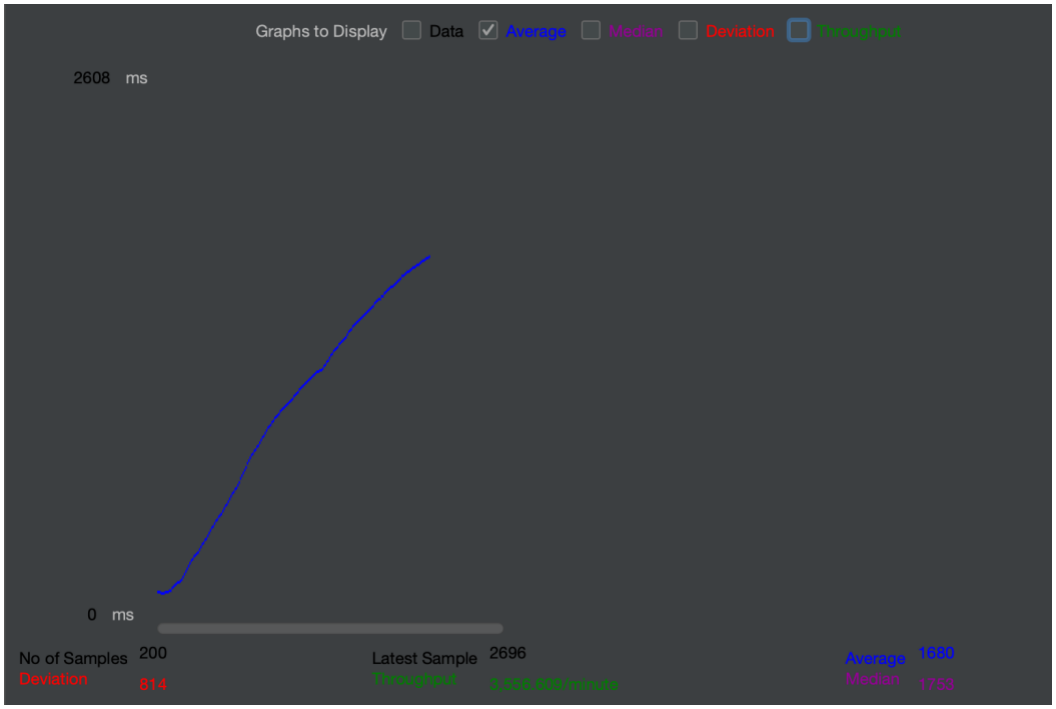
With connection pooling

GET request for /getuserdetails

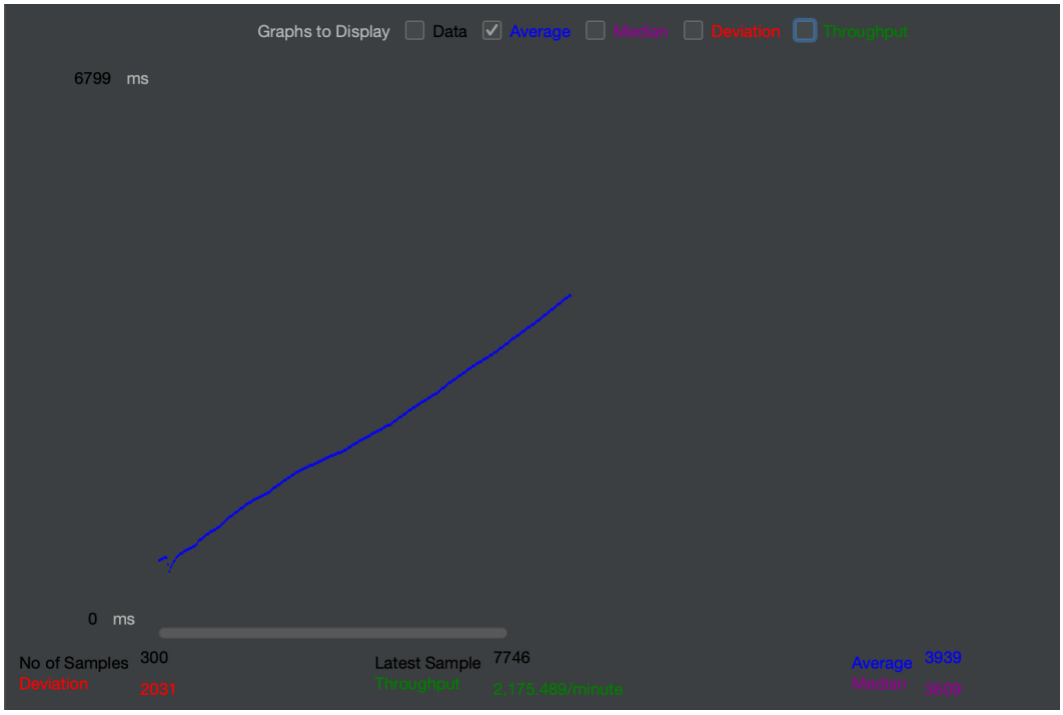
- 100 Concurrent Users:



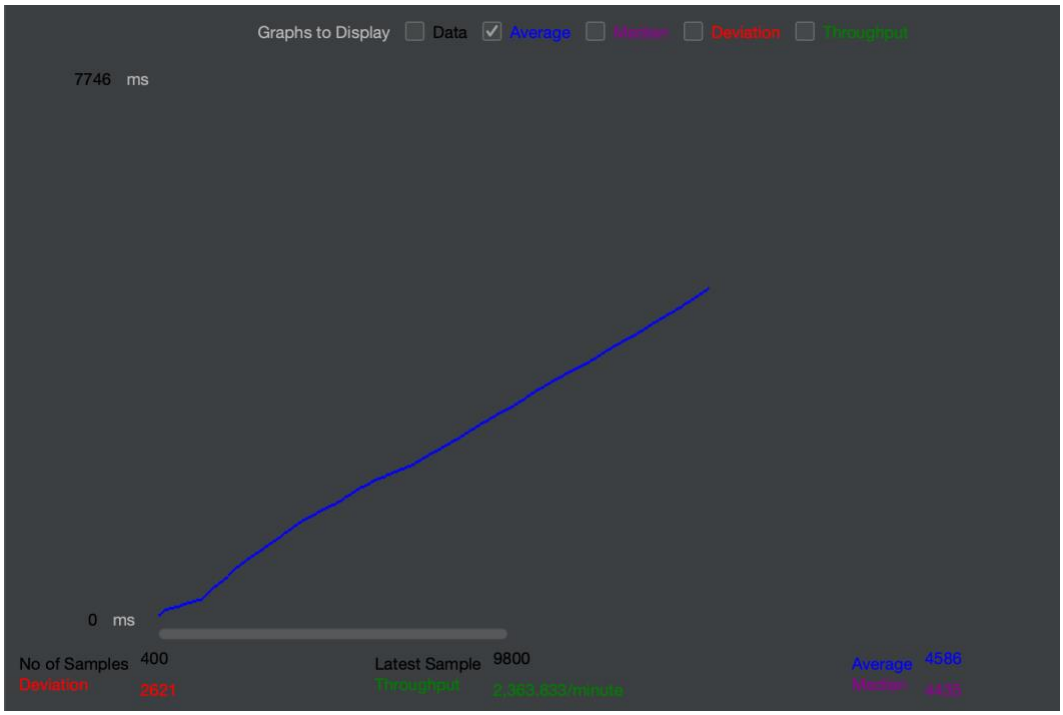
» 200 Concurrent Users:



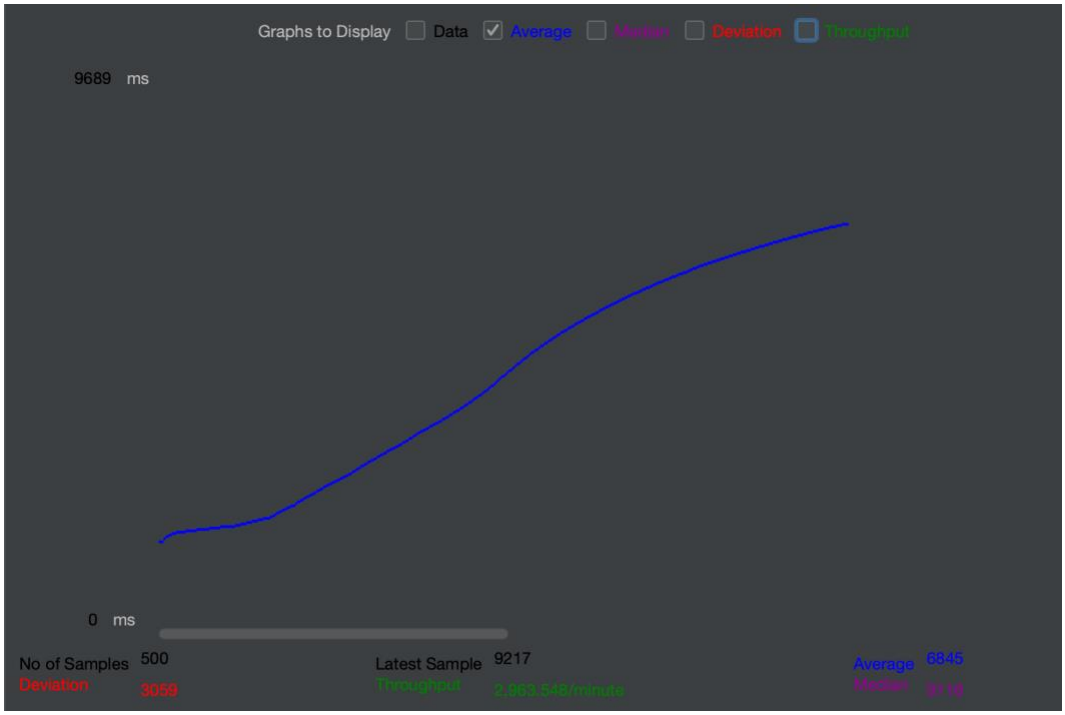
» 300 Concurrent Users:



» 400 Concurrent Users:



» 500 Concurrent Users:

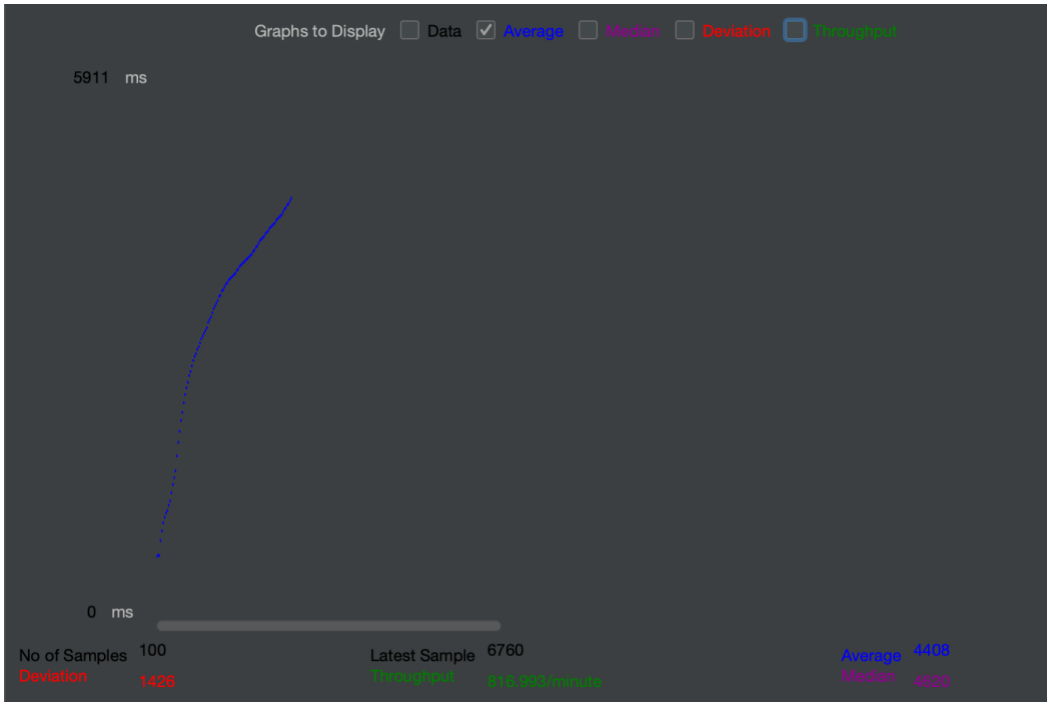


With connection pooling, we see that the average time taken by 100, 200, 300, 400, 500 concurrent users to execute the GET request on /getuserdetails API is 3902ms, 1608ms, 3939ms, 4586ms, 6845ms respectively. The average time increases with the increase in number of samples.

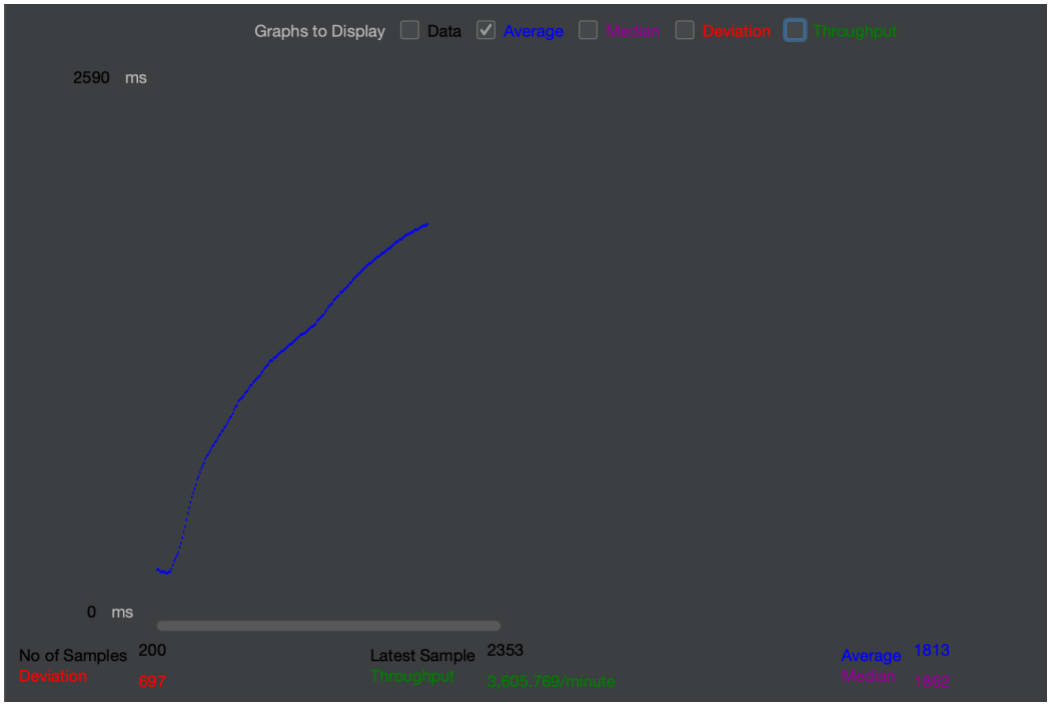
Without connection pooling

Get request for /getuserdetails

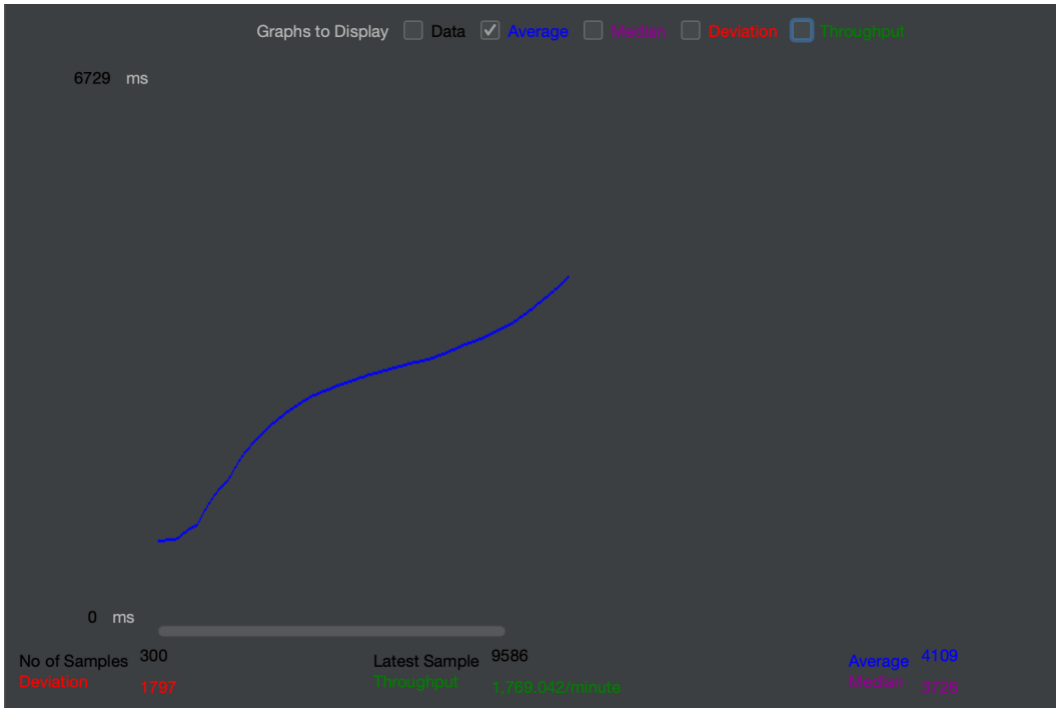
- 100 Concurrent Users:



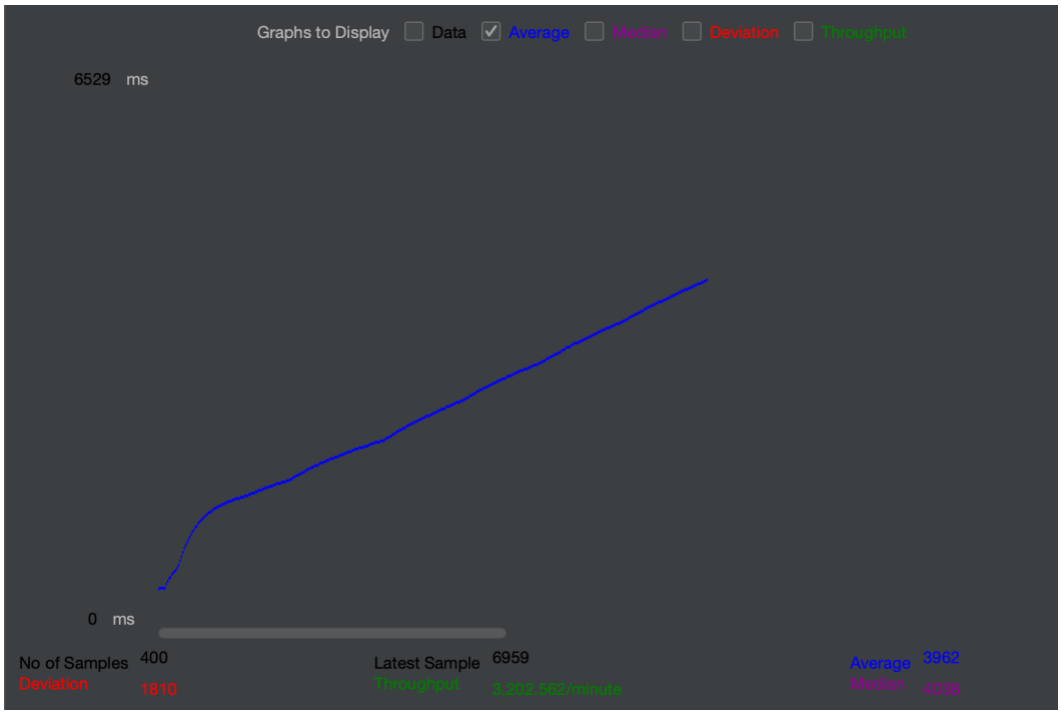
» 200 Concurrent Users:



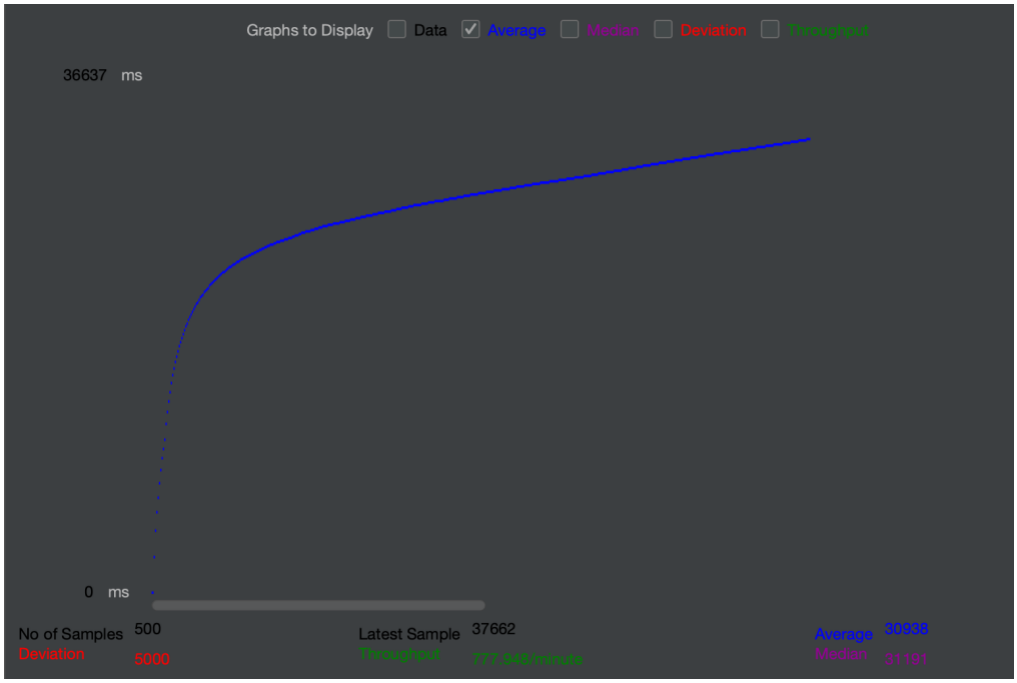
» 300 Concurrent Users:



» 400 Concurrent Users:



» 500 Concurrent Users:



Without connection pooling, we see that the average time taken by 100, 200, 300, 400, 500 concurrent requests to execute the GET request on /getuserdetails is 4408ms, 1818ms, 4109ms, 3942ms, 30938ms respectively. The average time increases with the increase in number of samples but do not drastically increase as in case of "without connection pooling". The connection pool gives a pool of connections to be reused thus reducing the overload of creating new connection for each and every request.

Mocha Testing:

Mocha testing is used to test the backend services.

```
Login Test
  ✓ Invalid Password
  ✓ Email not present
  ✓ succesfully logged in

Get User details
  ✓ should return the current user details

Update user profile
  ✓ should return username, email and profile photo

getting the groupinvites of the users
  ✓ should return the groups names of the invitations pending for tehcurrent user

leaving the group
  ✓ should return left group succesfully message


7 passing (96ms)
```

Backend services were tested for /POST login, /POST updating user details in MyProfile page, /POST leave group functionalities along with /GET user details and /GET group invites for a user. The functionalities were tested, and test cases passed for all 5 API calls.

- Login Test checked for invalid password, email not present and successful login for the /POST login API.
- Get User Details returned the details to be displayed on the profile page for the /GET API.
- Update User Profile updates the details provide by the user in MySQl database.
- To display the list of groups a user is part of /getgroupinvites API was tested.
- /POST leave group request was tested when users exit the group.

Frontend services were tested using React testing library.

React testing library is a light weight library for testing the react components and ensuring maintainable components throughout the life cycle of development.

```
Test Suites: 5 passed, 5 total
Tests:       10 passed, 10 total
Snapshots:   5 written, 5 total
Time:        15.13 s
Ran all test suites related to changed files.

Watch Usage
 › Press a to run all tests.
 › Press f to run only failed tests.
 › Press q to quit watch mode.
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.
```

Below are the test results for 5 components:

- Dashboard component: Checked if the dashboard can render and is rendered correctly.

```
PASS src/components/dashboard/dashboard.test.js (5.578 s)
  ● Console
```

- Login component: Checked if the login can render correctly.

```
PASS src/components/Login/login.test.js
  ● Console
```

- Group component:

```
PASS src/components/group/group.test.js (5.583 s)
  ● Console
```

- Profile_Page Component:

```
PASS src/components/profilepage/profilepage.test.js
  ● Console
```

» Create_new_group component:

```
PASS  src/components/create_new_group/create_new_group.test.js
  ● Console
```

Current Limitations of the Application

» Currently Add a bill in dashboard in not implemented.

» The bills are split equally among the users as of now.

Git commit history

```
commit 88163695f316bfa3691ca44e067f0d747bc28fcf (HEAD -> main, origin/main, origin/HEAD)
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Sun Apr 25 22:26:13 2021 -0700

    kafka updates and Readme

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 2bf683d249d25748dd6e30c4e31522467e8058fc0
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Sun Apr 25 03:27:55 2021 -0700

    jest

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 7aeaf82c5af4e4e122e49be6dfafcd562edb0d64 (15thApriltest)
Author: Ubuntu <ubuntu@ip-172-31-78-242.ec2.internal>
Date:   Sun Apr 25 08:22:22 2021 +0000

    Aws fixes

    Signed-off-by: Ubuntu <ubuntu@ip-172-31-78-242.ec2.internal>

commit cb6a382d80c08ab6cad5f4cc11a9a9eec78b861d
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Sat Apr 24 05:52:55 2021 -0700

    bug fixes

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit ad4621576542457e3d8e80f3117f0fd868ed0d0b
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Sat Apr 24 03:00:43 2021 -0700

    kafka changes and css changes

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 89a6149bd45635ef3b764301c6c1d34e4a3368a
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Thu Apr 22 23:28:29 2021 -0700

    kafka updates

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 1d9964931b44a9d4093b272a5c8c8226115b36b9
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Thu Apr 22 10:56:13 2021 -0700

    recentactivities

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 7f40ce30a43c40083f5f4752aee3953954bbdfbd
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Wed Apr 21 16:00:27 2021 -0700

    redux chnages in groups and recent acitvity pages

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit c2e6f0060df5e3283100920ed80eecbffa5d7446
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Wed Apr 21 03:43:00 2021 -0700

    Dashboard with redux changes

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit bfb6aceff4daac18a68cfeee06b375861746b8a6
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Mon Apr 19 06:12:34 2021 -0700

    my Groups REdux updates

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 84fb08aab60f9209acc4cd082af4533d3d2a99f8
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Mon Apr 19 03:57:58 2021 -0700

    create group redux updates

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit aab4edef4597692b92228d222967c50153bf8e2b
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Mon Apr 19 02:03:16 2021 -0700

    update Profile page redux changes

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 37f249e074e9cc7fba767ded8eaf7ac1c515f15b
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Sun Apr 18 18:52:22 2021 -0700

    bug fixes on login and signup pages with redux

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>
```

```
commit 23b527a0fc93801df28bdc6e9c1aa7f649b1abf8
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Sat Apr 17 23:55:24 2021 -0700

    eslint issues fixed on head

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 539dad3c3d10f5246177a005863d2b3eb36aa486
Merge: 90914c42 78f3df48
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Sat Apr 17 23:38:45 2021 -0700

    Merge branch 'main' of https://github.com/Swes-sjsu/273-Lab-1-Splitwise into main

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 90914c42f19b239f9e91a782a2de59e6b10324bb
Author: Swes-sjsu <78192205+Swes-sjsu@users.noreply.github.com>
Date:   Sat Apr 17 22:46:22 2021 -0700

    Initial commit

commit 7c04b206a27ce9c1b1f56c4becf600d1a7bcd198 (HEAD -> 29thMarch, 17thApril)
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Sat Apr 17 22:03:47 2021 -0700

    component folders added

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit b4efdb29fab7f6678ce2cf35e38f9383075b6936
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Thu Apr 15 16:23:35 2021 -0700

    Backend Apis for transactions

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 41048449ddc9dece3361d353952044ba42cb28aa
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Tue Apr 13 00:05:53 2021 -0700

    Backend Apis

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 8e628b0997112d9ce483e6103055c1f57c852c65
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Sat Apr 10 09:36:25 2021 -0700

    create-page api

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit b97567950c2c3532ae011fc8a91ece0a941fb61b
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Thu Apr 8 09:13:59 2021 -0700

    backend api for getuser details and user options along with signup and login

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit c3a06fd69965ae7e76f99462df60276447efd893
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Wed Apr 7 00:47:16 2021 -0700

    working passport jwt for login
    /node_modules

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 8d334e6f7dda9ff571963f658ad8d4ef5c3e54a2
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Mon Apr 5 21:44:35 2021 -0700

    login with jwt

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>
```

```
commit 1b76770fcf8ea18cc9259ba134cca89071008373
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Tue Mar 30 21:11:09 2021 -0600

    Signup and login pages with mongose

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit 0c72785c1d281d1b3f6c77e9b7bc2eb6da03a37e
Author: Swes-sjsu <swetha.singireddy@sjsu.edu>
Date:   Mon Mar 29 23:35:51 2021 -0600

    Initial Login and Signup pages

    Signed-off-by: Swes-sjsu <swetha.singireddy@sjsu.edu>

commit a93ef2818a9b21456471b273bedb38799941da11
Author: Swes-sjsu <78192205+Swes-sjsu@users.noreply.github.com>
Date:   Sun Mar 28 11:15:36 2021 -0700

    Initial commit
(END)
```

Questions

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

In Lab1 we were using cookie-based authentication. Cookies are stored on web browser when credentials are valid which includes a Session ID. Passport and JWT is token-based authentication where an access token is used to authenticate a user into the application. JWT token are self-contained, and they help to securely transmit information. JWT tokens are easier to maintain and does not pose a security risk of unauthorized individuals viewing

the information when they do not have the token. They do not maintain any state and are easily scalable as compared to cookies. JWT also allows you to store metadata like userid and email addresses but cookies will have just the Session ID. JWT tokens might have additional overhead as the data will be encoded and decoded for every request. But JWT can store metadata which can help reduce lookup in case of user level data to be retrieved reducing the overall time to get the requested data.

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

- Kafka is a messaging engine which is replicated and distributed to process large data.
- Producers send events to the Kafka broker which are consumed by the consumer. The data is sent from the producers as soon as possible and the producers need not worry about its processing as Kafka ensure that the request is processed once the message is sent to the broker. Kafka consumers processes the data and sends the data back to the producers.
- Producers need not wait until the data is available thus reducing the latency. When there is huge data that needs to be streamed/processed, asking the producers to wait makes the application slower and will not be able to process all the requests.
- Kafka can replicate providing fault tolerance. Kafka is a distributed system capable of scaling with lesser downtime.

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

- MySQL being a relational database. It can be used to store relational information like the users and the groups they are part of. Additionally, a table can be used to store the balances between the users of each group making it easier to display the total balances.
- MongoDb being a NoSQL solution, can be used to store the expenses and the comments related to the expenses as both expenses and comments can be retrieved at the same time without having to access two different tables.