Twitter Application Simulation

CMPE 273

Enterprise Distributed Systems

Project Team 1

Professor: Simon Shim

Github:
https://github.com/MaheshReddy108/CMPE-273-Twitter-Application-Simulation

Team Members:

1. Sruthi Duvvuri
2. Kanika Khanna
3. Mahesh Reddy Konatham
4. Manish Katturu
5. Shashank Reddy Gopireddy

Table of contents

Table of contents **2**

Team Individual contributions: **4**

Object management policy **6**

Chosen System Architecture design **7**

ExpressJs: 8

Reactjs: 8

Handling of heavyweight resources **9**

Load balancing: 9

Session Handling: 9

Tweet/List/people search: 9

Profile Images: 9

Pagination: 10

Policy to write data into the database **10**

A screen capture of client application **11**

Create a Twitter account: 11

Validations: 12

Email already exists: 12

Username already exists: 12

Profile with Image 16

Updating profile details 16

Tweets created by user 17

Retweets by user 17

Liked Tweets by user 18

Reply tweets 18

People followed by user 19

3
View search profile 20
Tweets by user searched 21
Lists by user searched 21
Follow people 22
Search for topic 22
pagination 23
List of followers in Message Timeline 23
Bookmarked tweets by user 24
Lists user member 26
Creating List 26
Adding member to List 27
View Tweets in List 27
Top 10 tweets with more likes 28
Top 10 tweets with more views 28
Top 5 Tweets with more Retweets 29
User Dashboard 30

A code listing of main server code 33

Server code listing for Activity Routes and Kafka-backend 34

Server code listing for Auth Routes and Kafka-backend 35

Server code listing for Message Routes and Kafka-backend 36

Server code listing for User Routes and Kafka-backend 37

A code listing of database access class 38

A code listing of test class 39

JMETER Testing: 47

Kafka and Backend Logs for one api 66

Lessons Learnt 68

Team Individual contributions:

1. Sruthi Duvvuri:
    - Design and implementation of Home Page and Navigation Component.
    - Implemented Tweet and User Feed modules
    - Worked on functionalities like fetching tweets, like, unlike, re-tweet, replies,bookmark and deleting tweets.
    - Backend APIs related to Tweets
    - Configured linter for standardizing code.
    - Implemented Graphs using MongoDB charts and integrated them to application.
    - Mocha Testing

2. Kanika Khanna:
    - Frontend for user authentication(login/signup)
    - Frontend for User Profile, Lists section on user dashboard
    - Frontend for followers, following people
    - Frontend for view profile of people searched
    - Backend APIs for user authentication(login/signup)
    - Backend APIs related to the users
    - Kafka Backend
    - JMeter Testing

3. Mahesh Reddy Konatham:
    - Messaging Frontend
    - Follower list Frontend
    - Search for people Backend Api
    - Search for Topic Backend Api
    - Search for people Frontend
    - Search for people Frontend
    - Redis for get_profile backend

- Pagination Frontend
- Kafka for Search people and search topic apis
- Deployment with auto scaling and load balancing
- Enzyme Testing

4. Manish Katturu:

- Backend API for Lists such as creating a list, adding members, get subscribed members etc.
- Backend API for messages.
- Backend API for Bookmarks.
- Few Backend API of tweets.
- Front end for uploading and display of images.
- Contributed for frontend for tweetbox and tweetitem display.
-

5. Shashank Reddy Gopireddy:
    - Worked on User Dashboard
    - Collaboratively worked on functionalities of Lists such as listCreation, viewMembers, subscribeList
    - Worked on validations part of editProfile
    - Worked on setup of services such as Kafka, Passport and Redis
    - Collaboratively worked on deployment part mainly load balancers and setup of multiple instances
    - Worked on end to end testing of functionalities of both frontend and backend

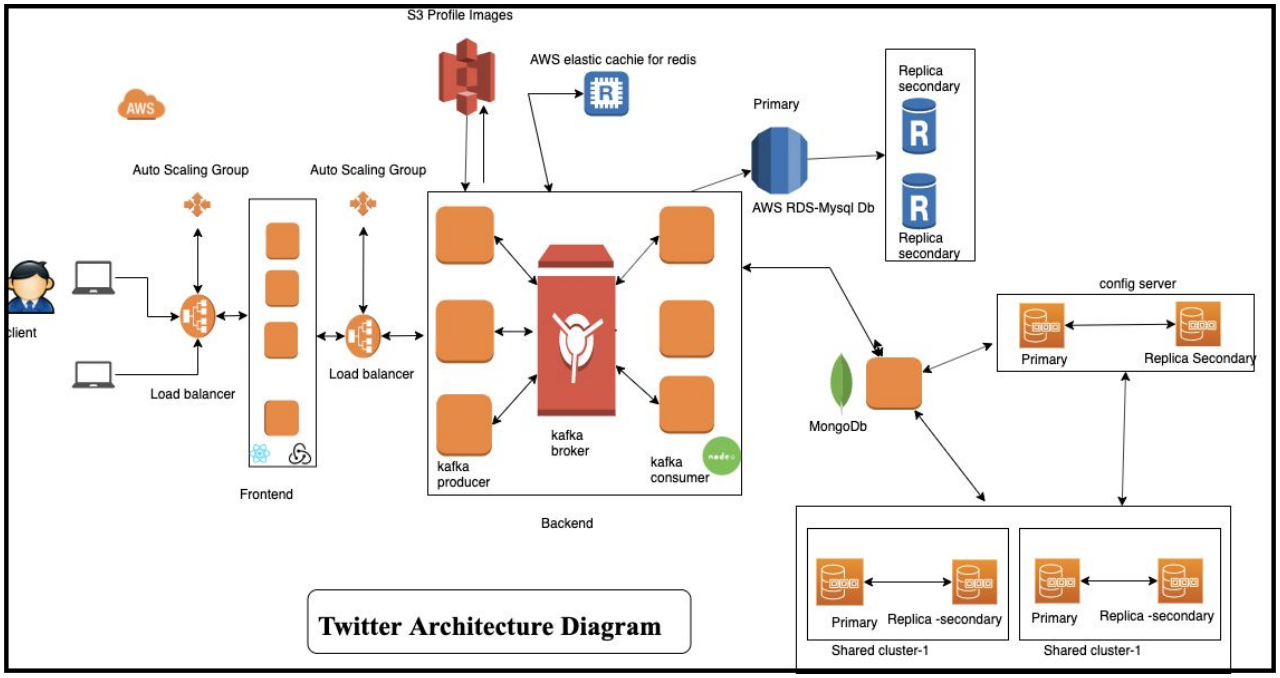
Object management policy

This project mainly has four objects which are important to consider for our design:

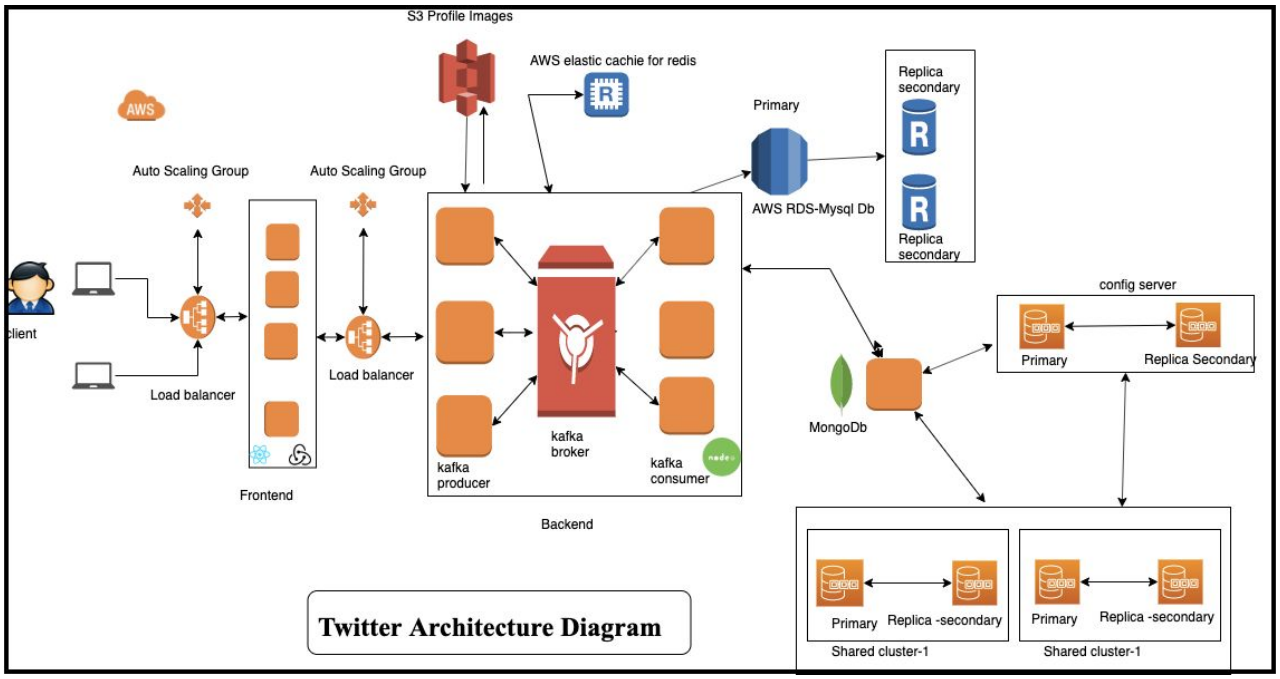
1. User
2. Tweet
3. List
4. User Feed

A user can log in to the application and post a tweet which can be either text or image or hyperlink or can be a combination of all of these things with the character limit of 280, once a user posts a tweet any other user can like or reply or bookmark to that tweet. User can search for other users and follow them. User can create a list with curated group of users accounts. A user can subscribe list, add members to the list. The user Feed is populated with the tweets of the users whom he is following and also from other important users.

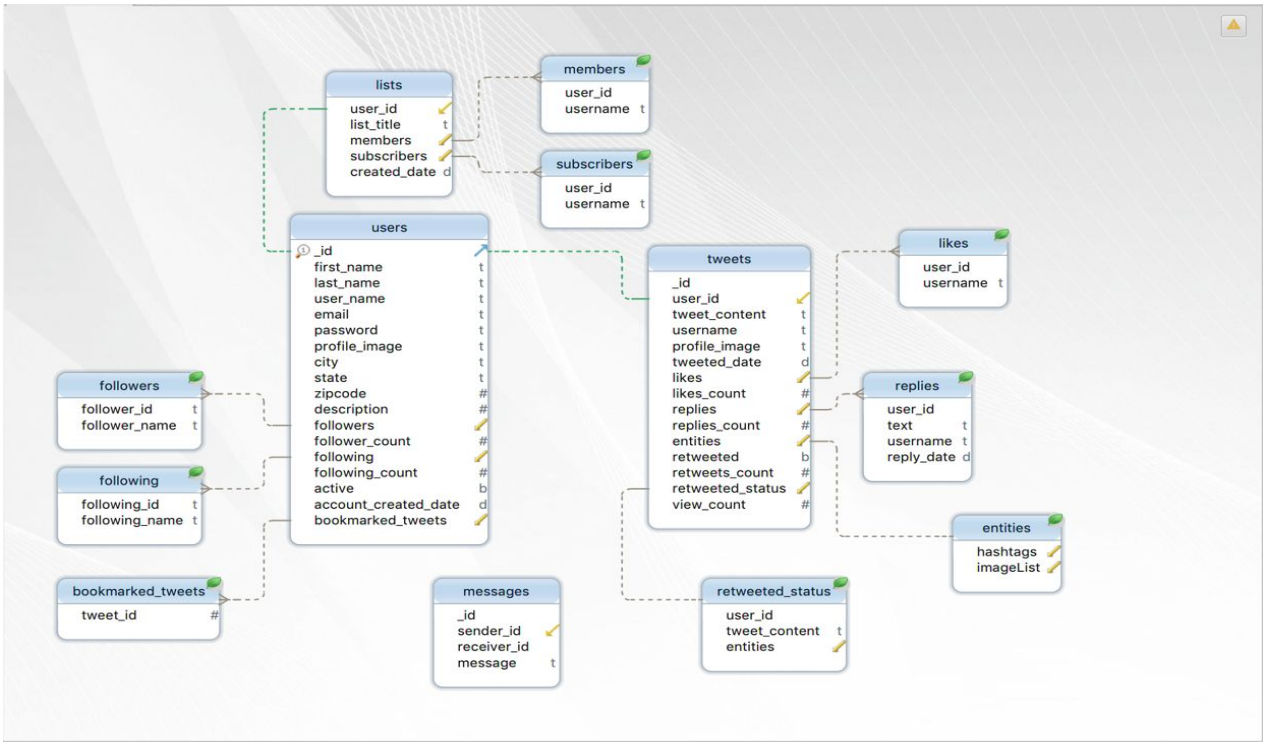
Chosen System Architecture design



e



Database design



NodeJs:

Nodejs is a powerful tool which is used to creation of REST application services which is based on javascript that handle heavy I/O website based operations. Nodejs works on single thread so that it can simultaneous concurrent operations without creating multiple threads for executing.

ExpressJs:

In this project we are using node js based express framework which is lightweight web based Application Framework which facilitates various Apis to build our Application.

Reactjs:

We followed the MVC model for our Application which is Model View and controller based. The views are based on Reactjs framework. Reactjs is a powerful and very useful javascript based website building framework using which we can build reusable components with very easy efforts.

Handling of heavyweight resources

Load balancing:

In this project our Application's each server has its own based software stack and the load is distributed through requests all the serves in a round robin fashion. Since the web-sessions are all RESTful, each of them are stateless and can be handled by anyone without depending on any one server.This also helps in preventing single point of failure and works as a good backup system.

Session Handling:

For this project we are using Jwt and json web-token for session handling the session management. The user when logged in will be given a unique web token which can be used for uniquely identifying and can also be used for load balancing when the request hits different servers also.

Tweet/List/people search:

In this project we are implementing redis cache for getting people results for search and getting Home Timeline and User Timeline..etc and for several other apis. The logic used for redis is hashmap for key names and getting the values in json format.

Profile Images:

In this project we are implementing multer using aws S3 bucket for handling and managing profile images and other media content like images from the tweets. In this way we are making it more scalable, fast and flexible.

Pagination:

We are retrieving only few people and topics when the user searches for the people or topics and when the next page is clicked we retrieve the next set of list also, in this way we are implementing lazy-fetching on request from user.

Policy to write data into the database

In the below situations we might go for NoSql:

- If the data is too big and consistency can be eventually achieved.
- When the data models have to be restructured based on the changing requirements.
- As No-SQL databases allow flexibility to scale easily, they can be used to build large scale applications.
- NoSQL performs well in the case where the read operations are much greater than write operations to a database.
- NoSQL is flexible than MySQL when we have an unstructured schema and it can be changed at any point of time.

In the below situations, we should prefer MySQL:

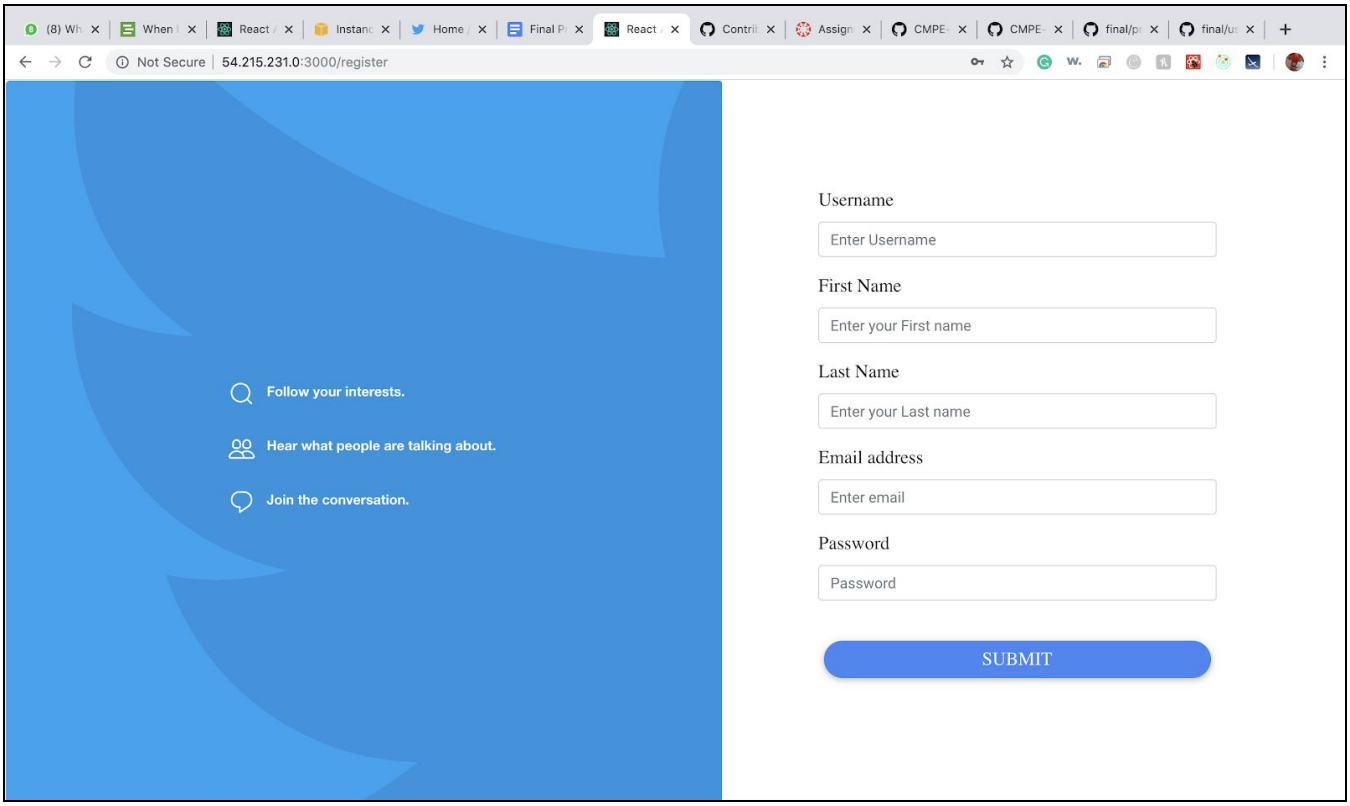
- When the atomicity, consistency, isolation, durability is required strictly, then we should prefer MySQL database.

- If we have a related and well-structured data, if there is no change of schema is required in the future, then preferring MySQL is better.
- If we have a confidential data i.e. security is the major concern, then MySQL is always suggested.

When we have both structured and unstructured data, MongoDB is less secure than MySQL, where MySQL facilitates better security for user confidential data. We had user confidential data like firstname, last name, email, password etc, so we preferred using MySQL which in turn also maintained a structured data for the user details. On the other hand, we had data like profile views, tweet views in MongoDB. Every time somebody views somebody else's profile or views a tweet, that information gets stored in the local storage. Once the threshold is reached, a batch script is called and updates the view profile and view tweets information in the MongoDB.

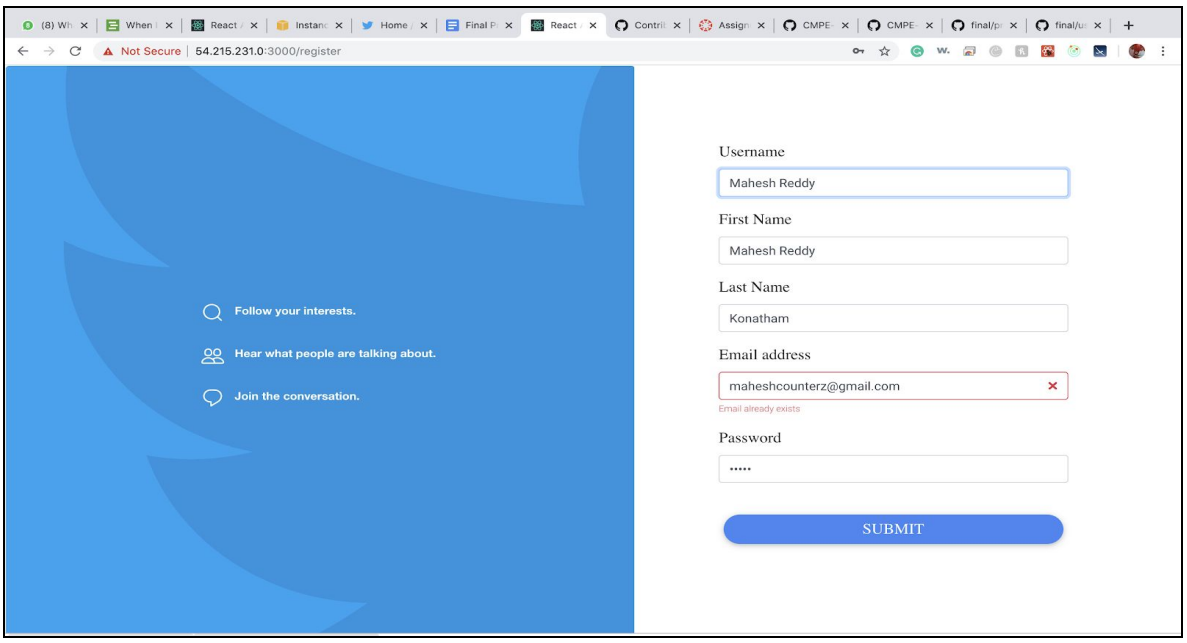
A screen capture of client application

Create a Twitter account:

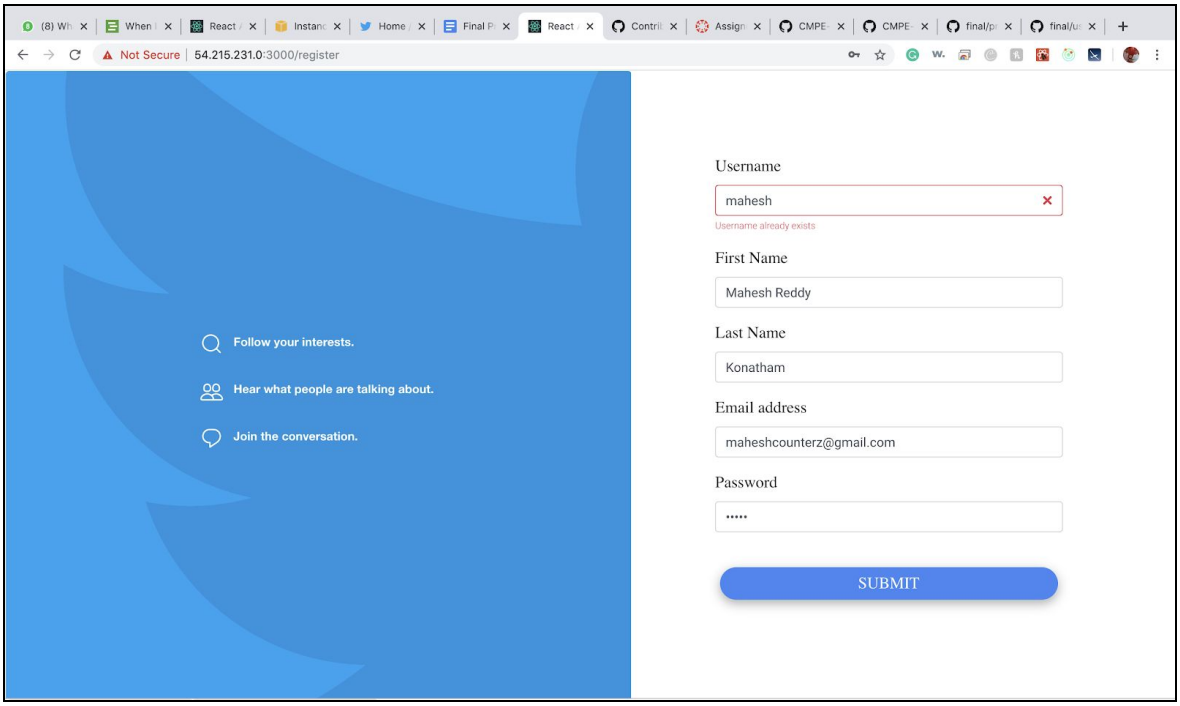


Validations:

Email already exists:

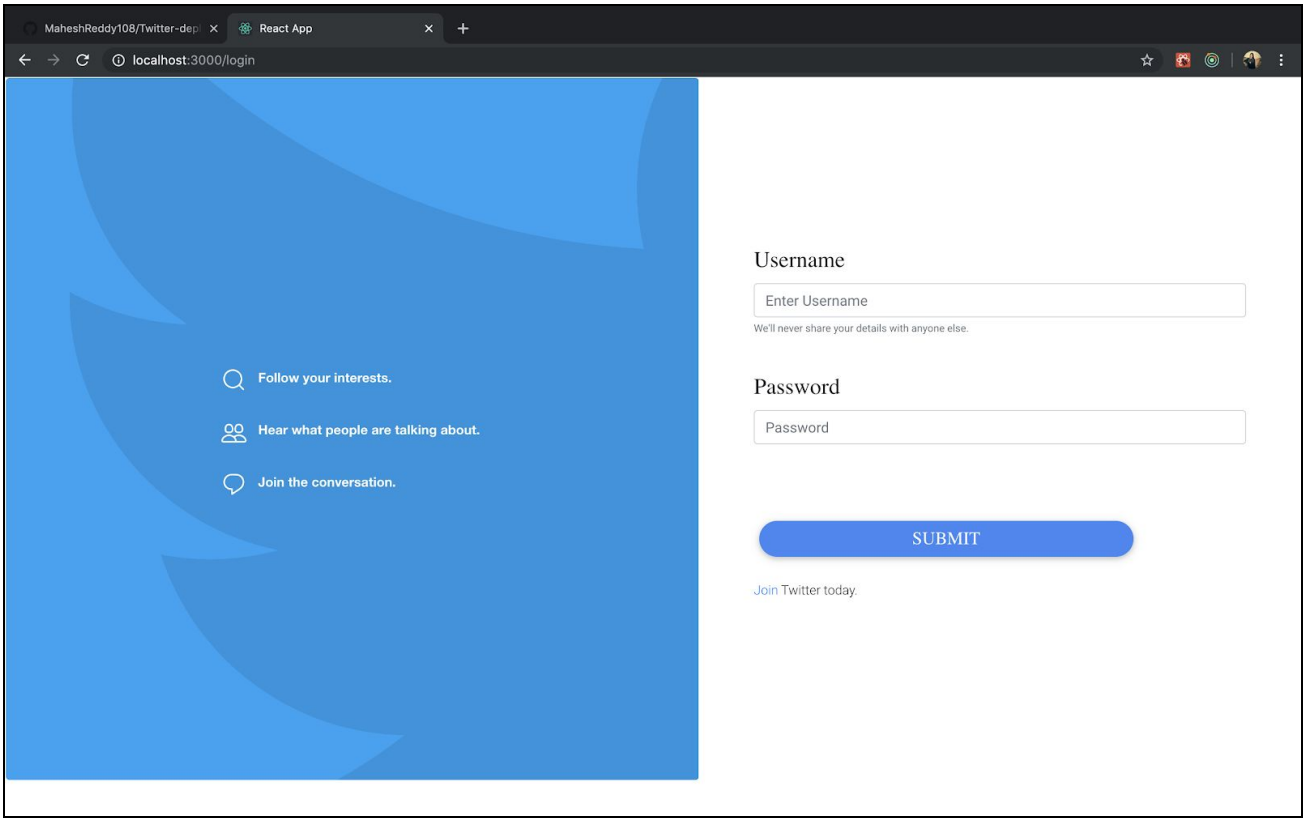


Username already exists:



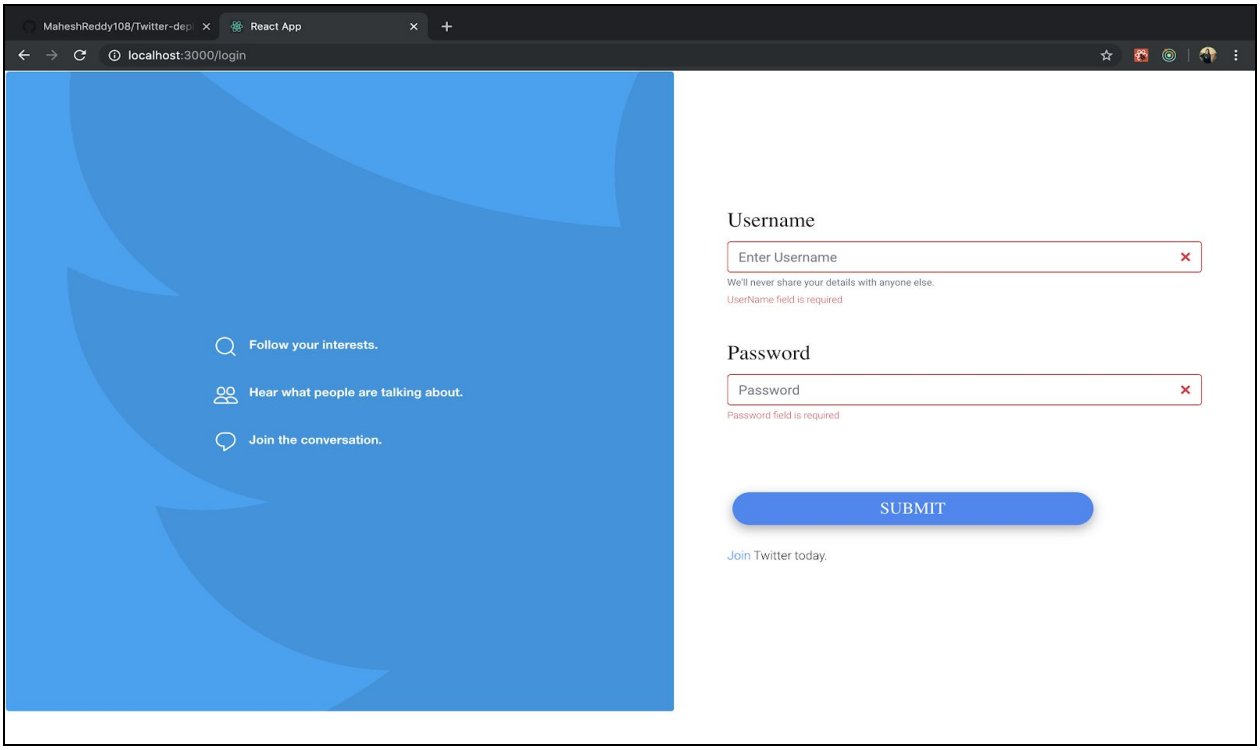
Tweet text length is limited to 280 Characters:

Login

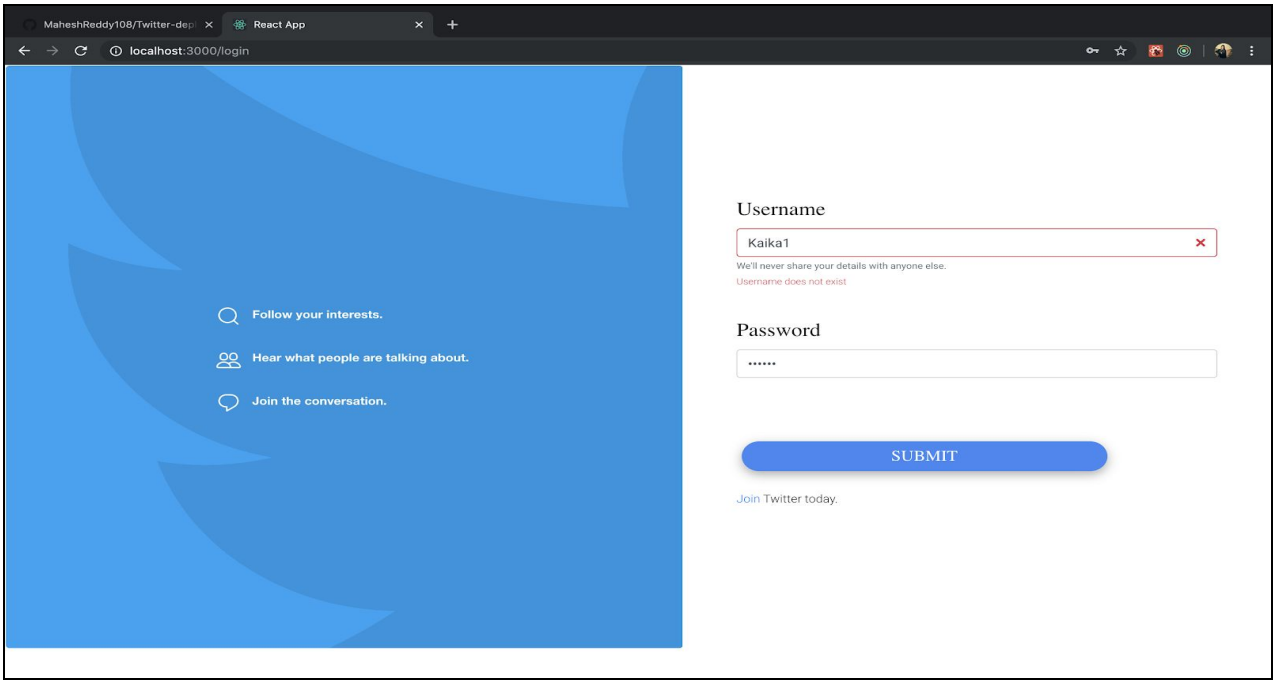


Login Validations

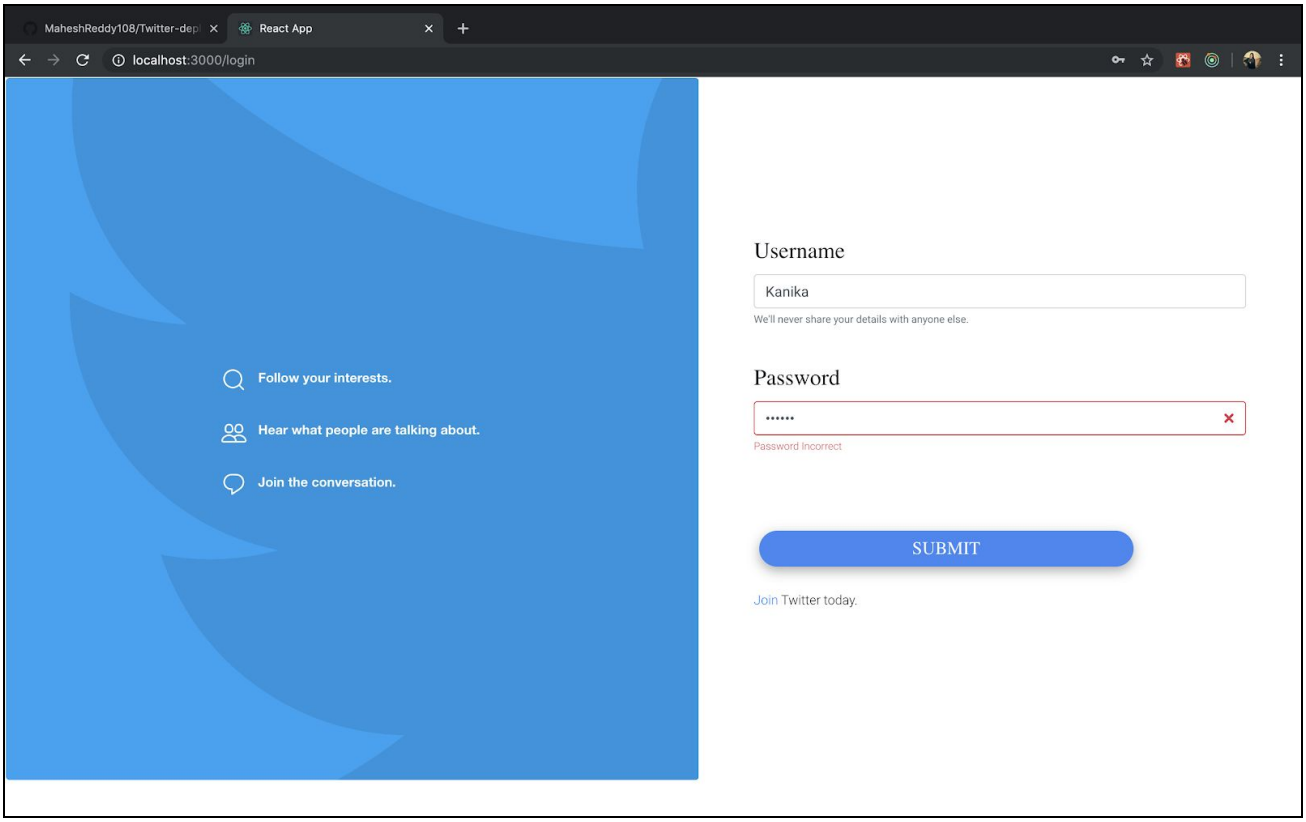
Input without any information



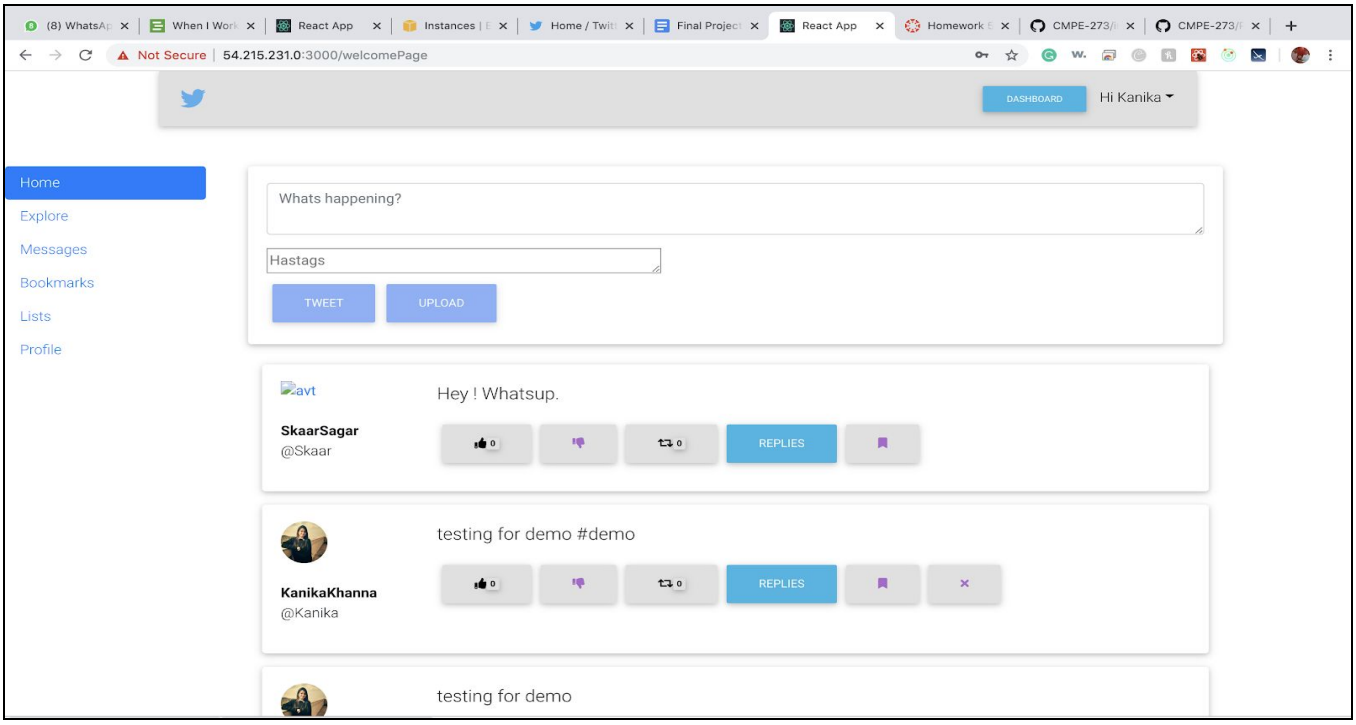
Invalid Username



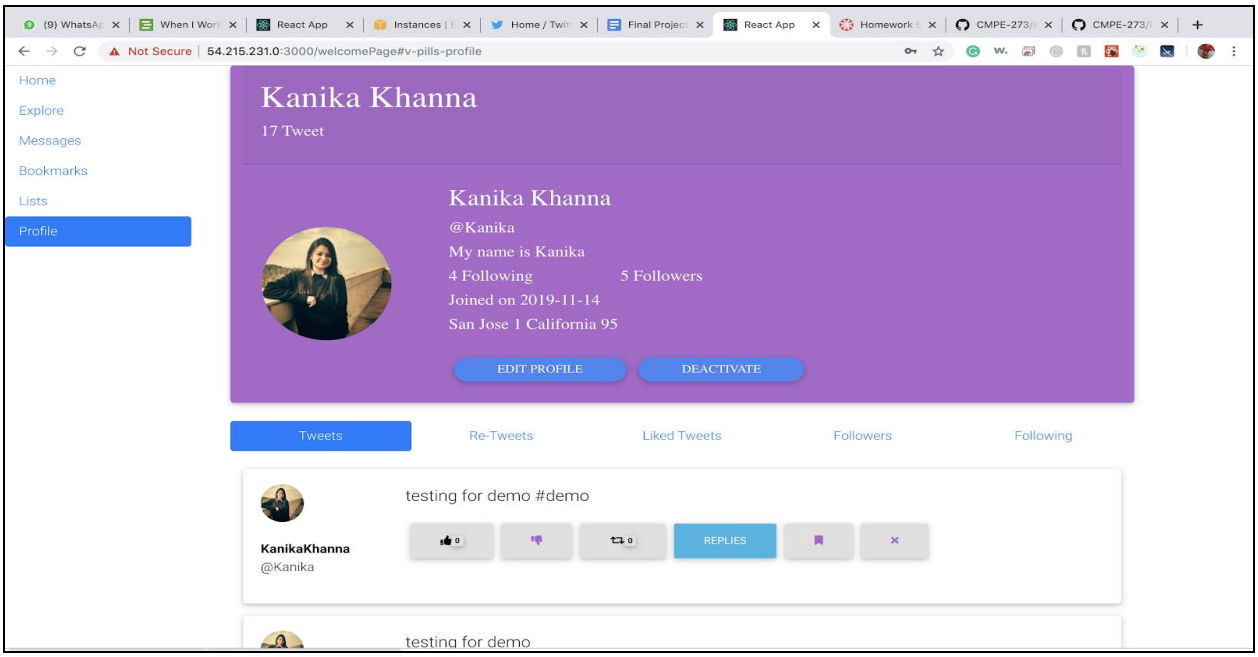
Invalid Password



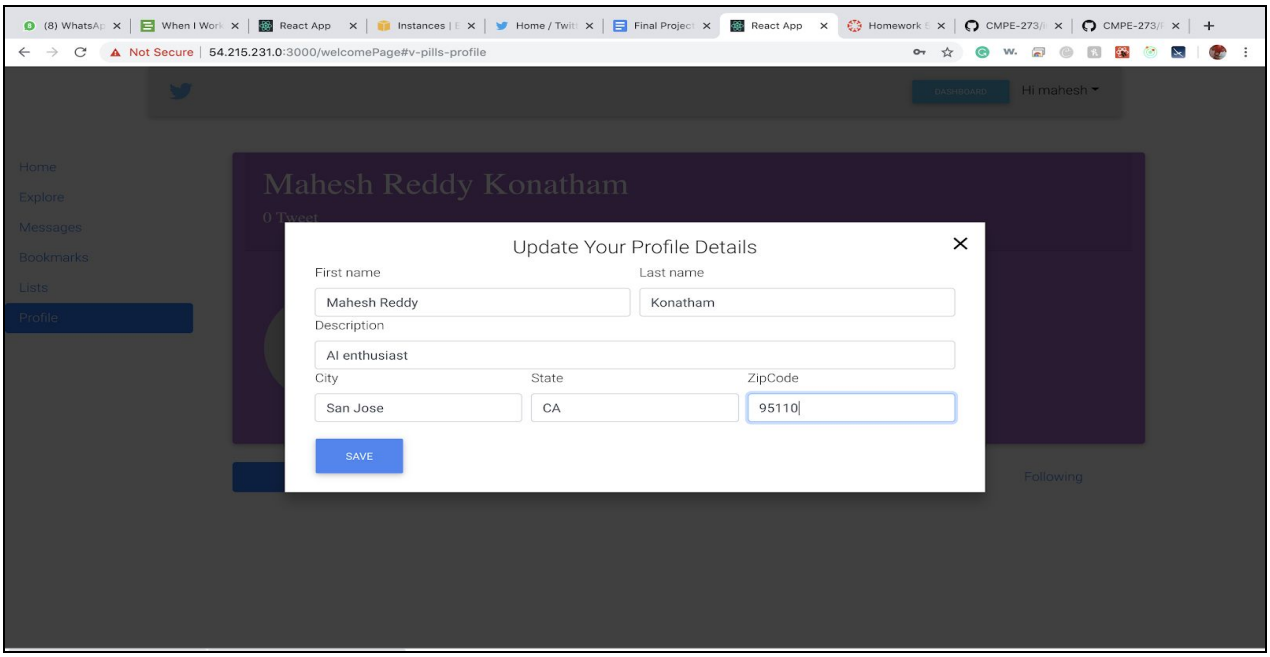
Home Page



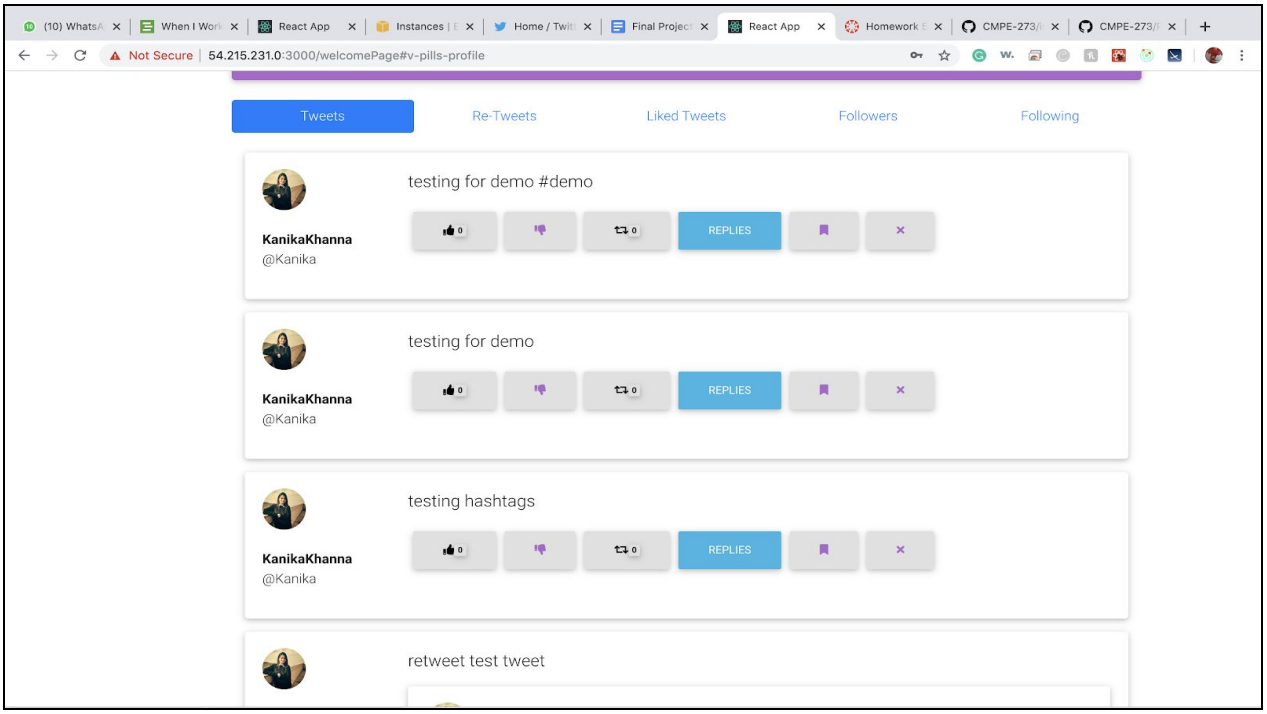
Profile with Image



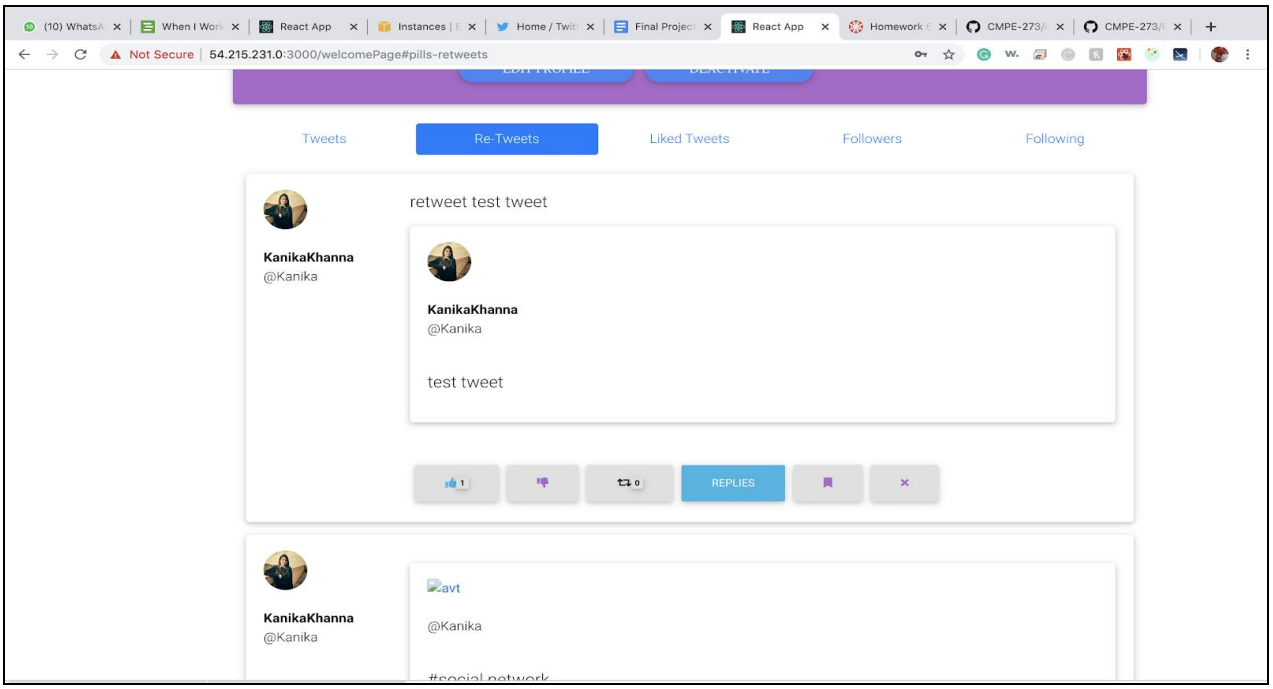
Updating profile details



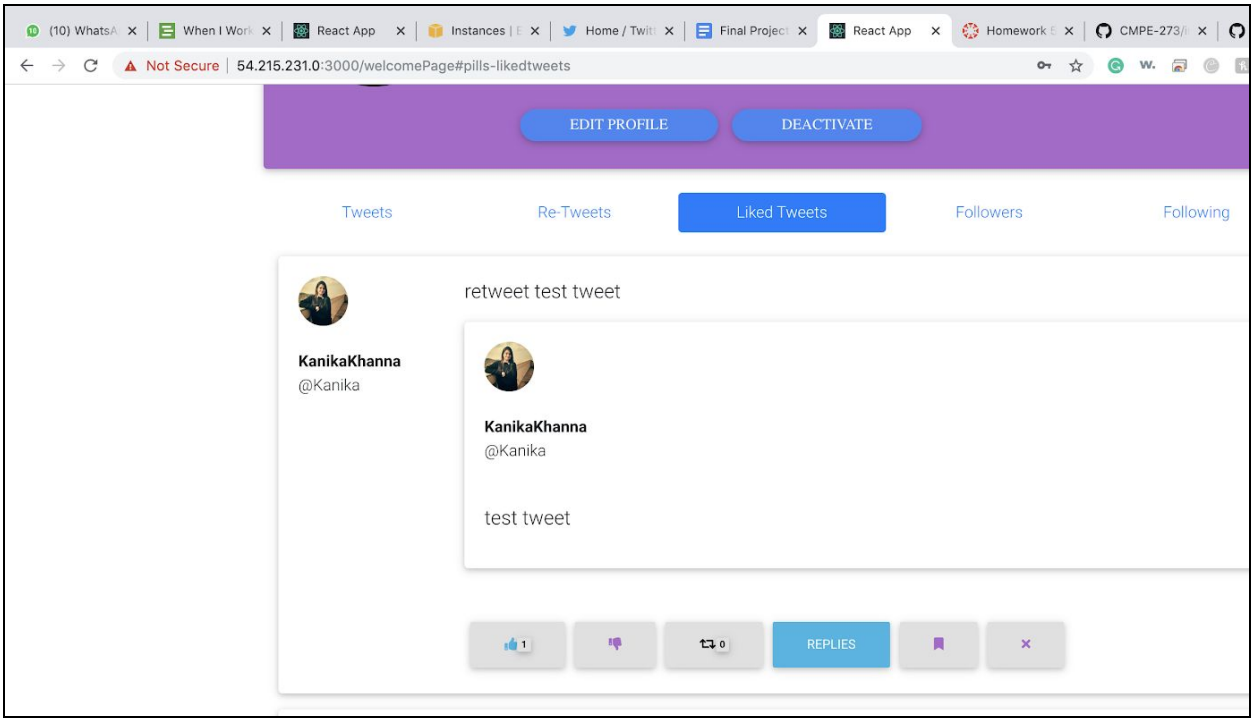
Tweets created by user



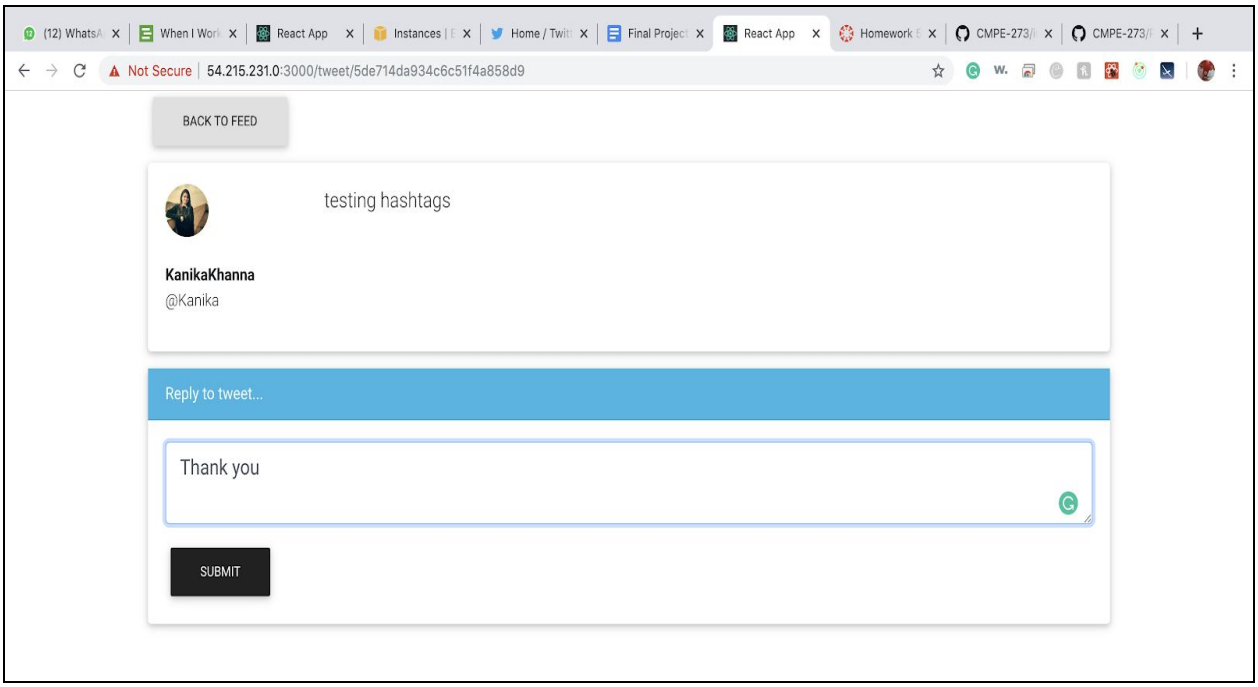
Retweets by user



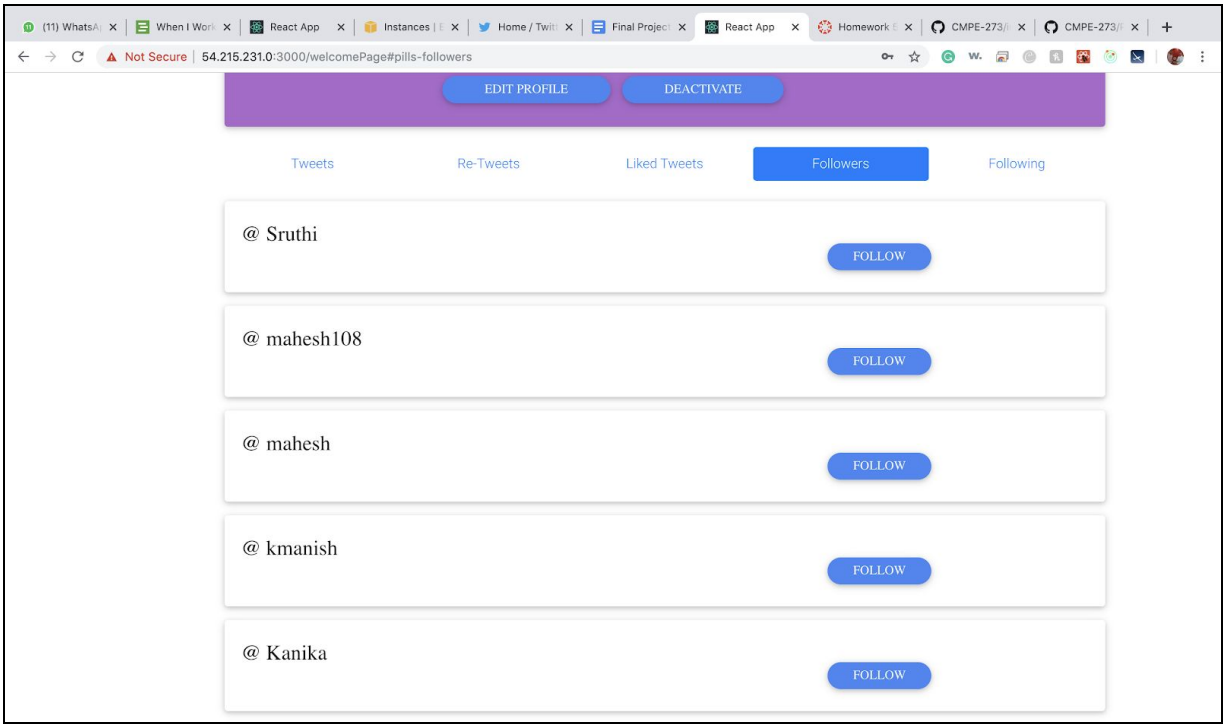
Liked Tweets by user



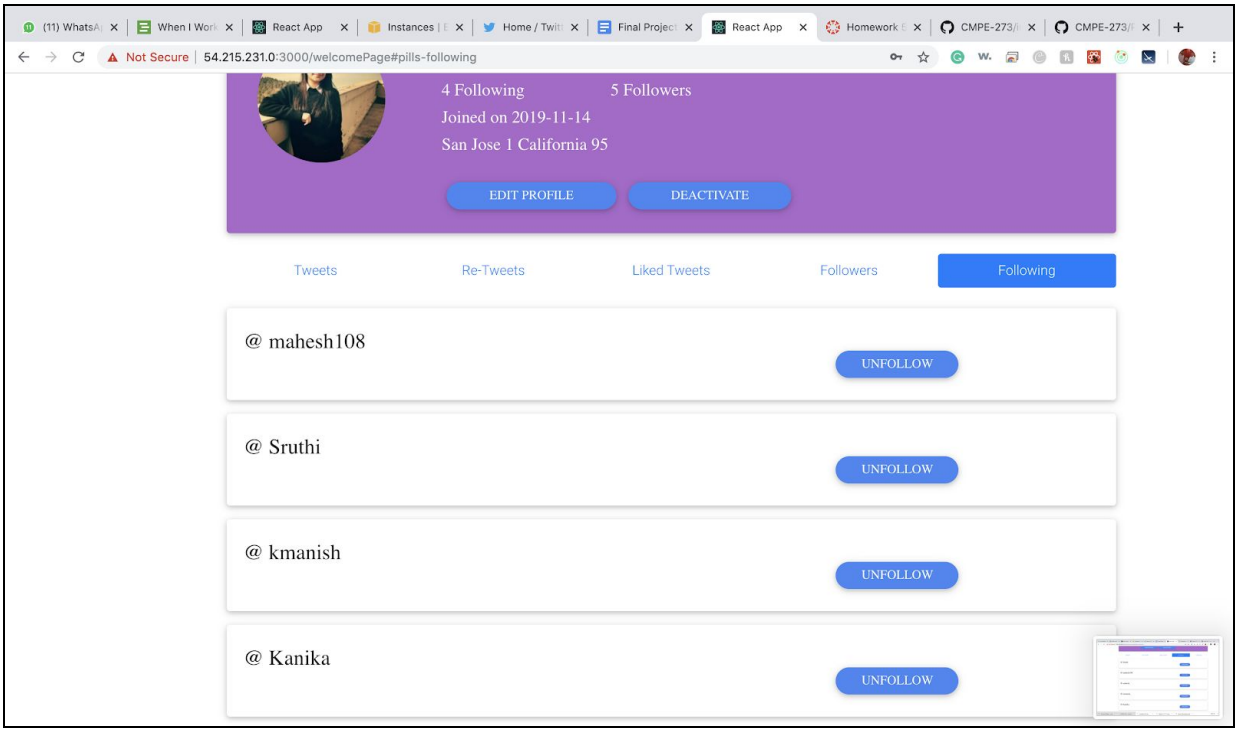
Reply tweets



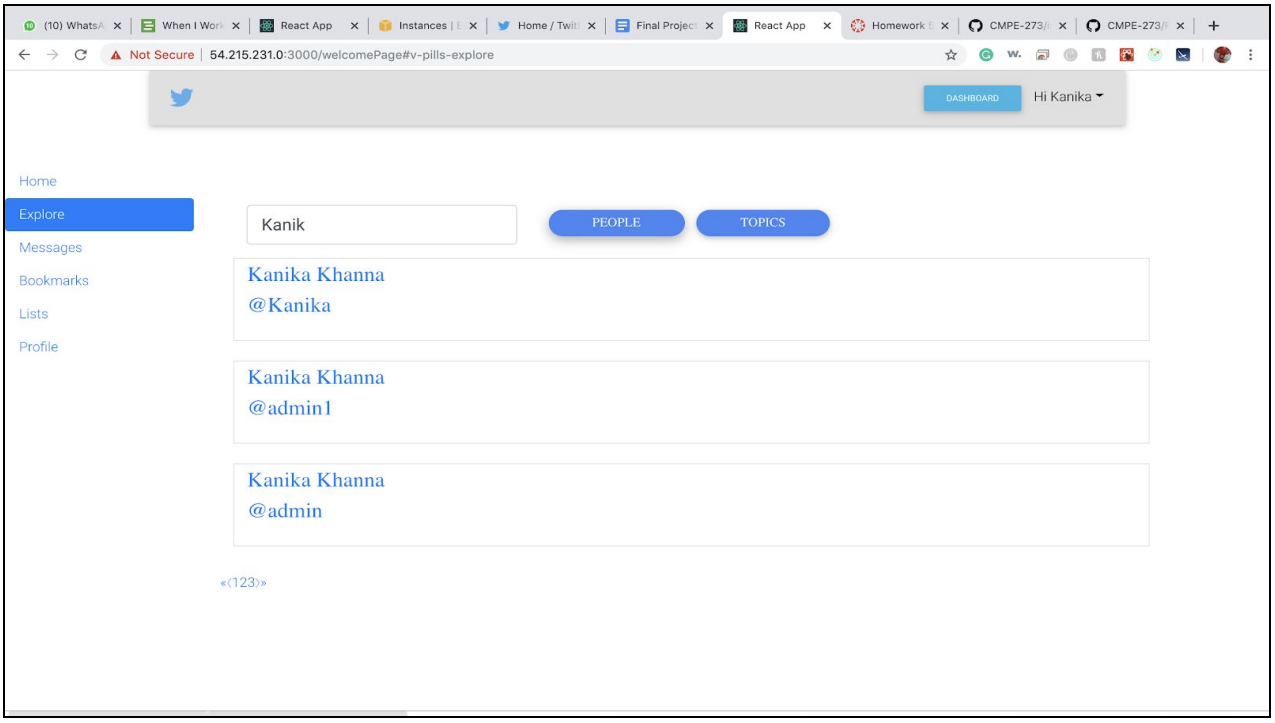
Followers of user



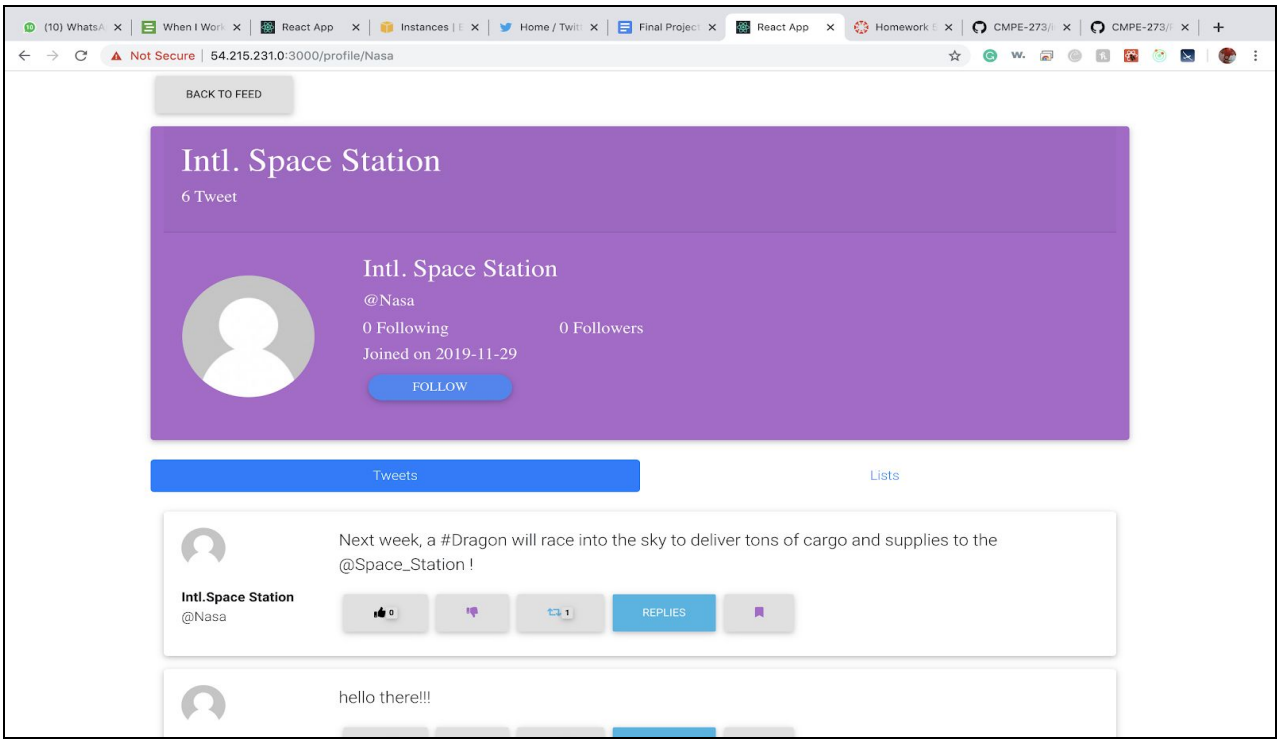
People followed by user



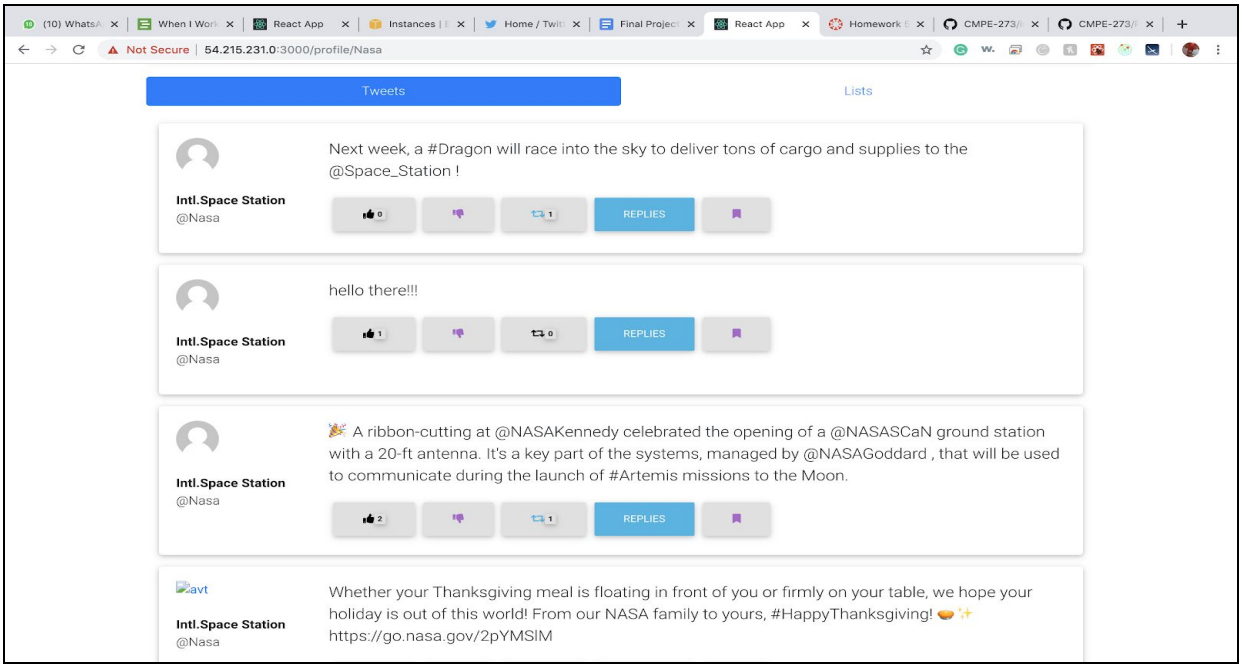
Search for people



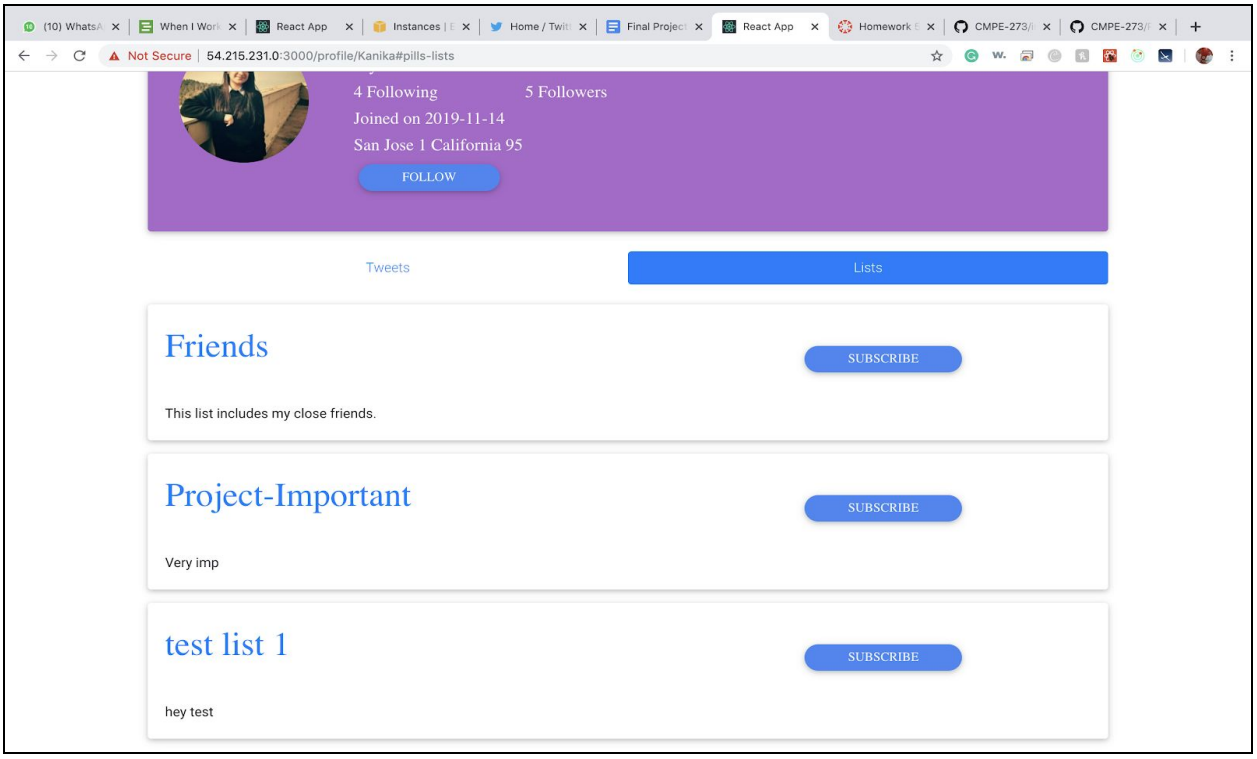
View search profile



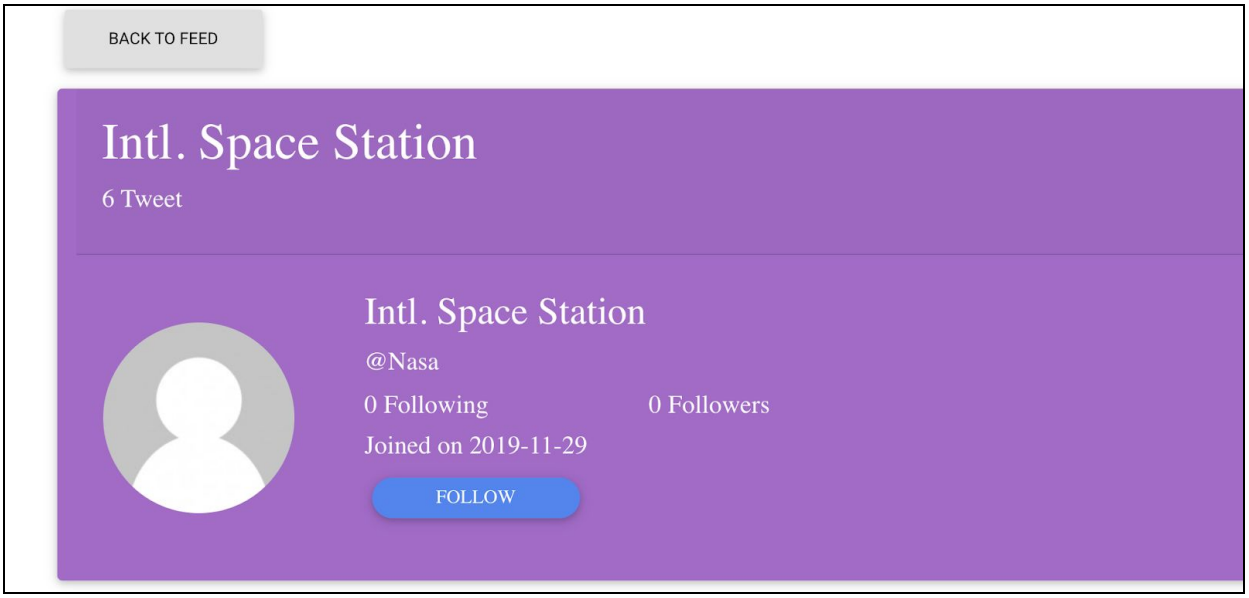
Tweets by user searched



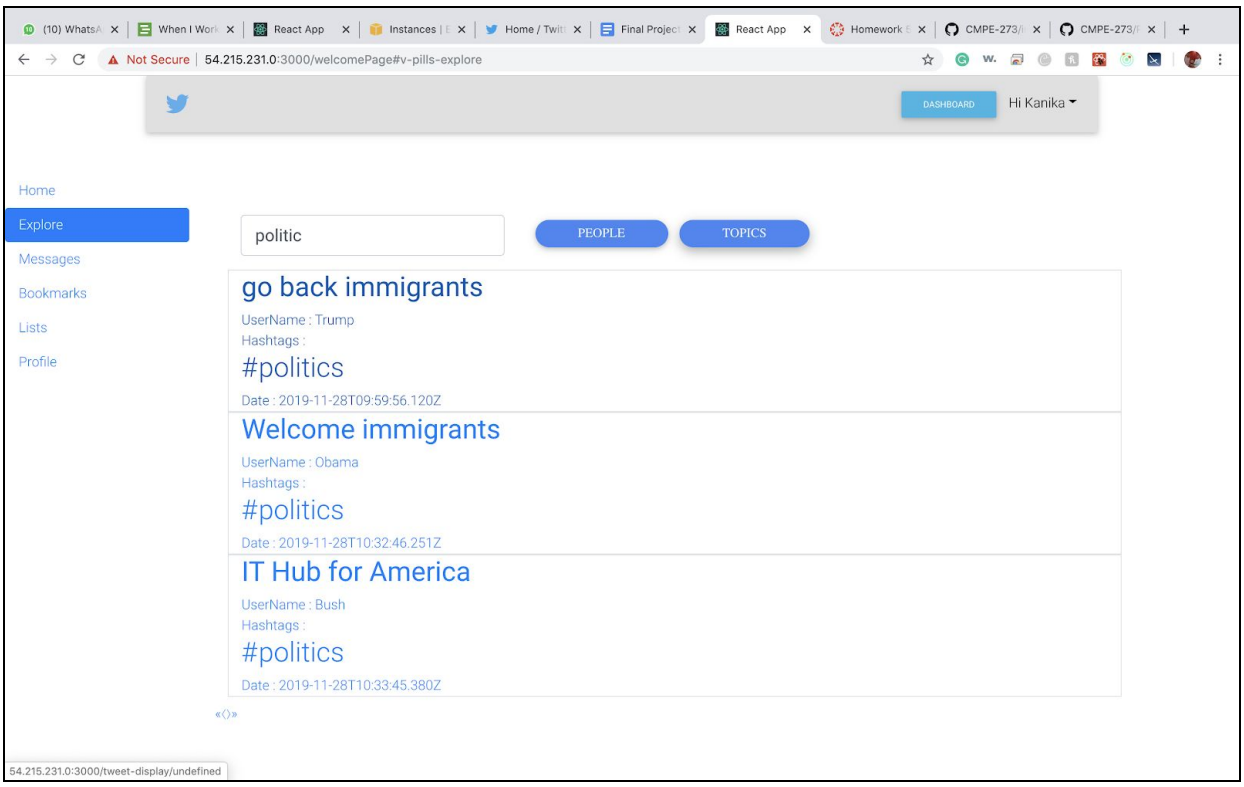
Lists by user searched



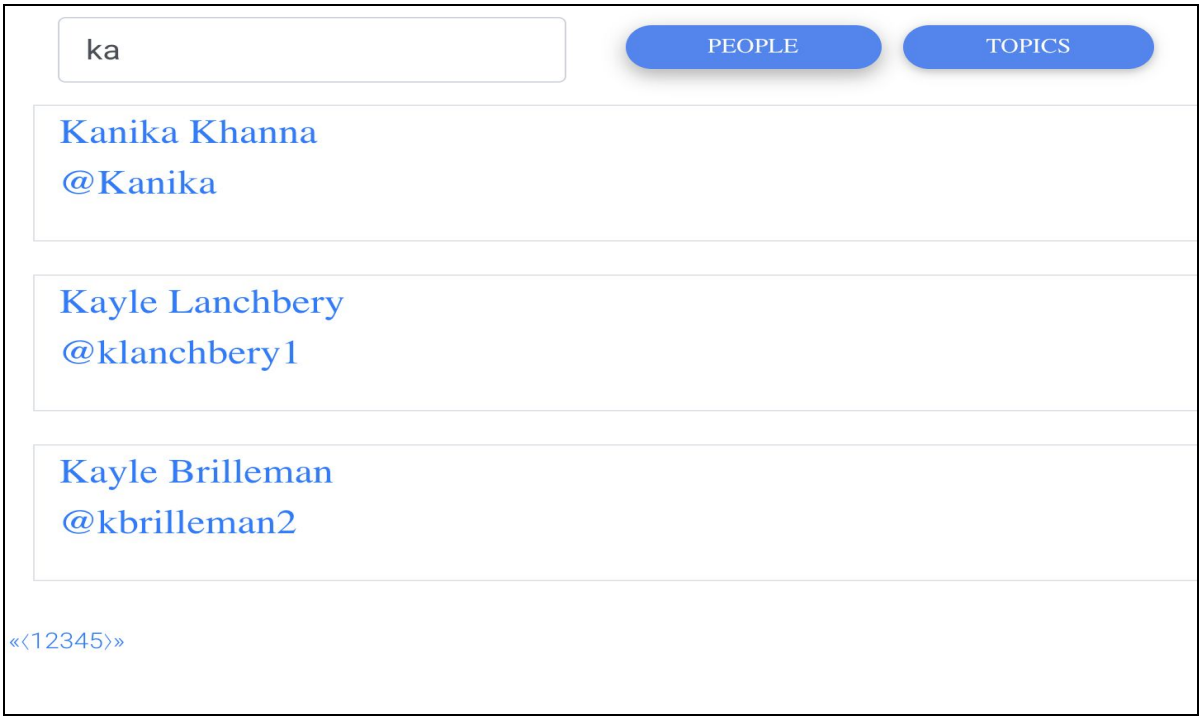
Follow people



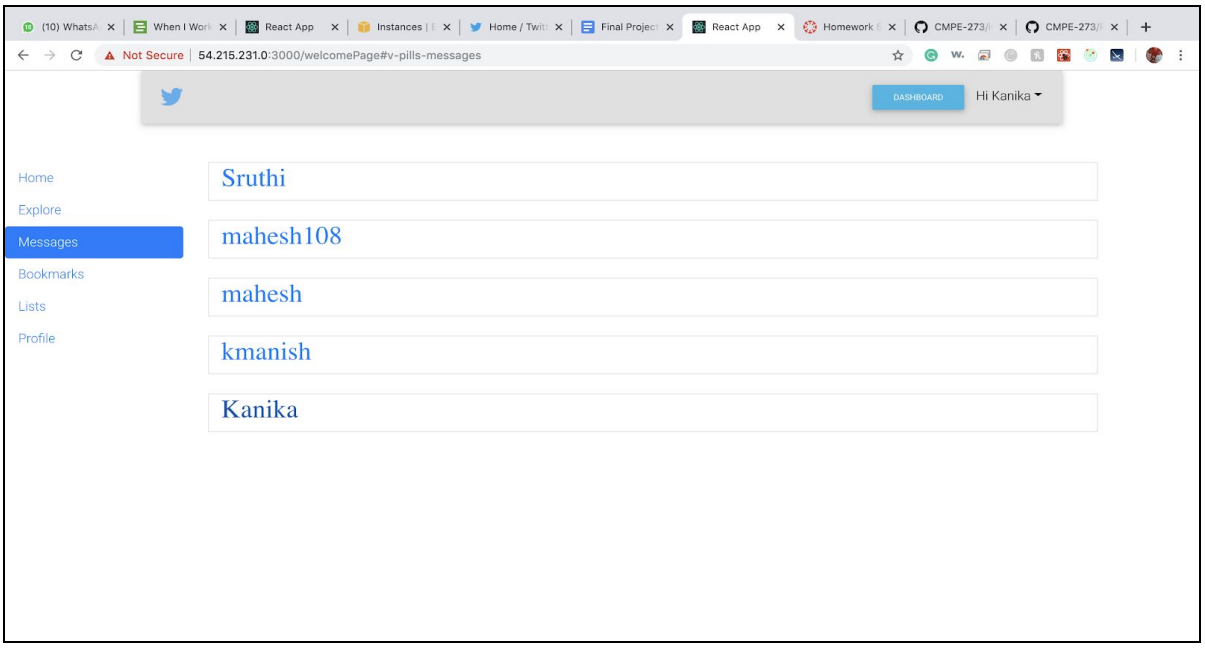
Search for topic



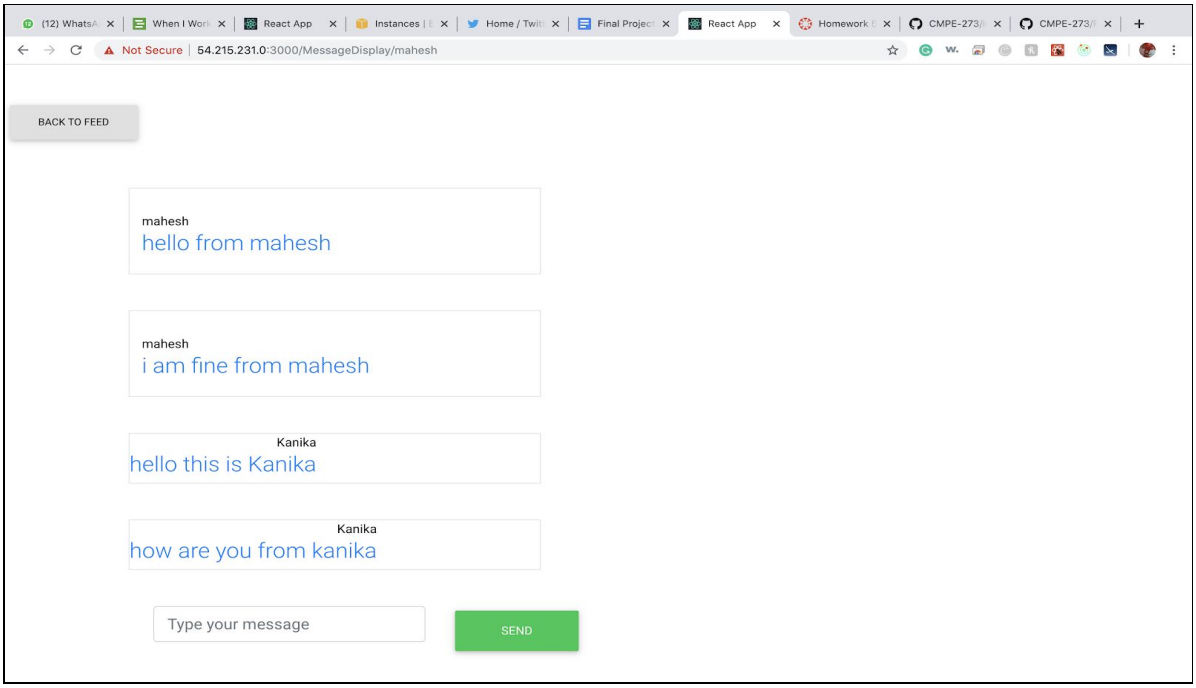
pagination



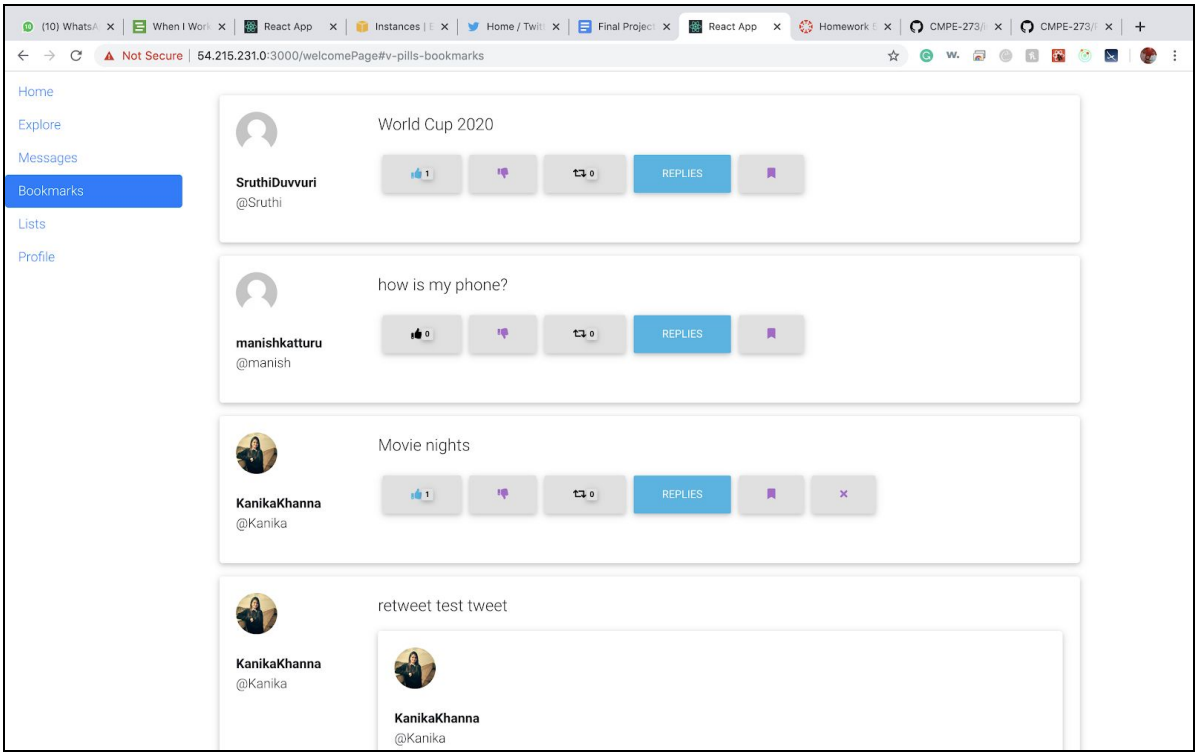
List of followers in Message Timeline



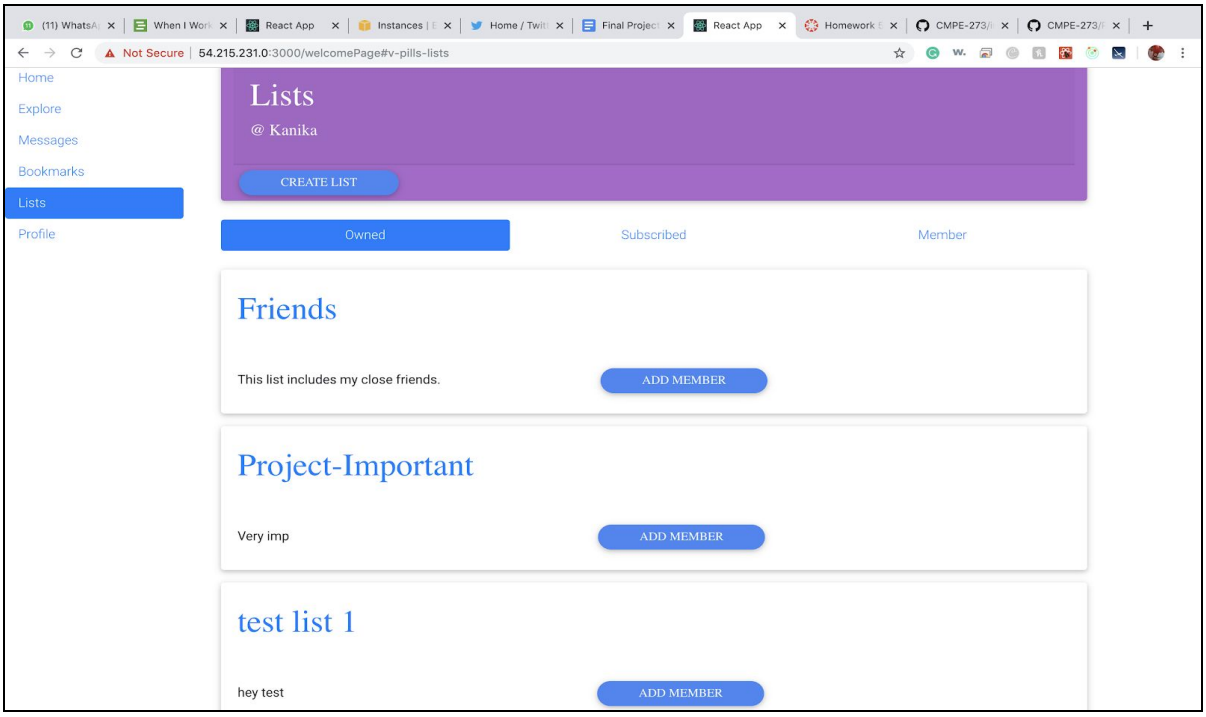
Messages Inbox



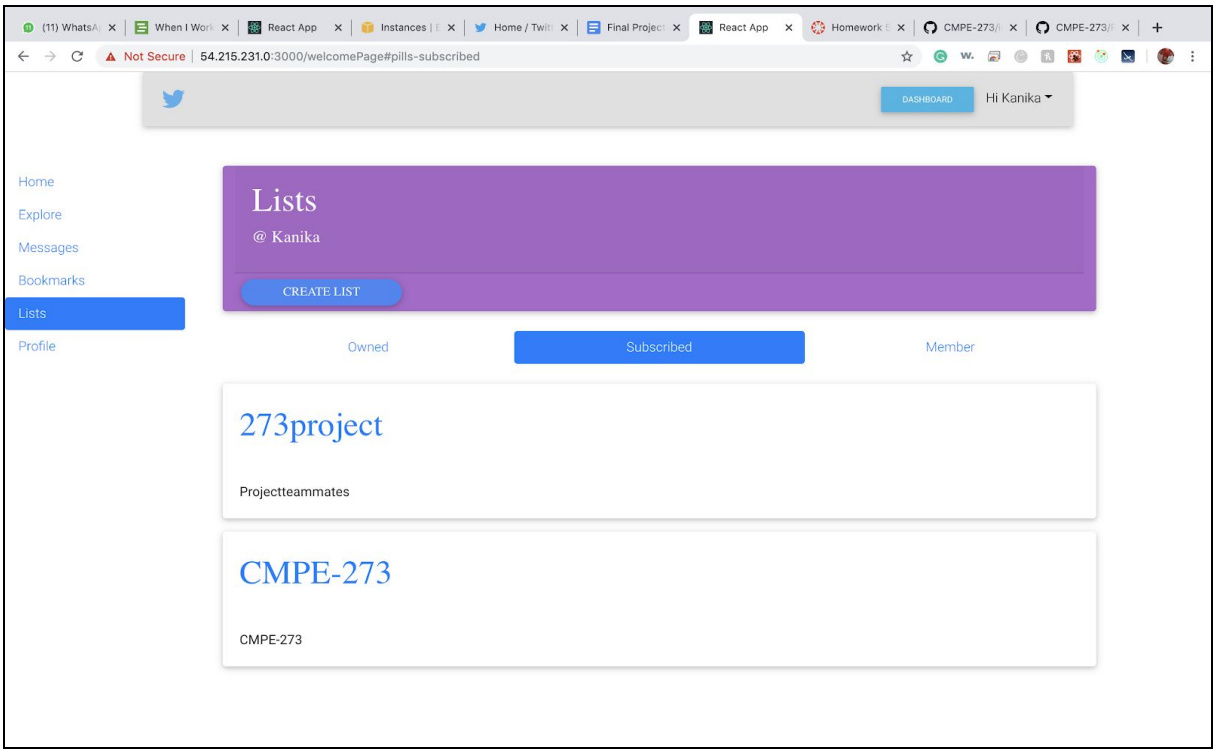
Bookmarked tweets by user



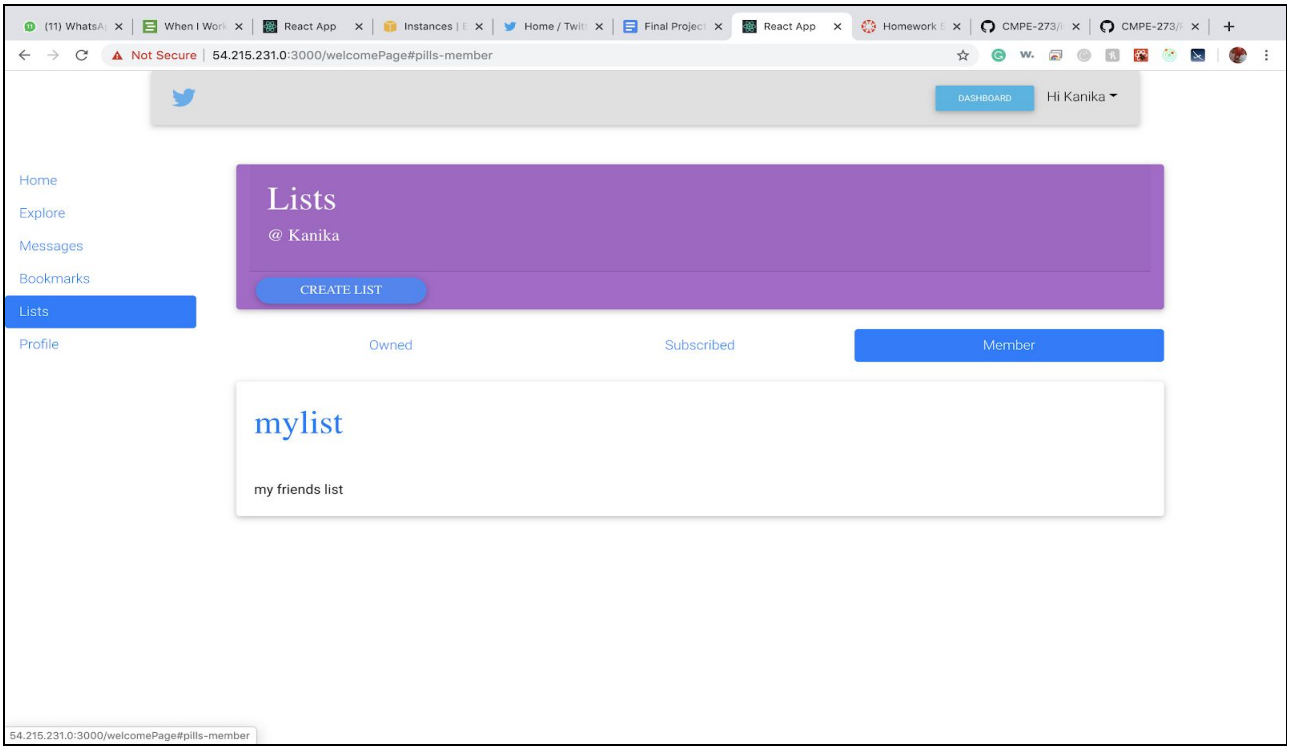
Lists owned by users



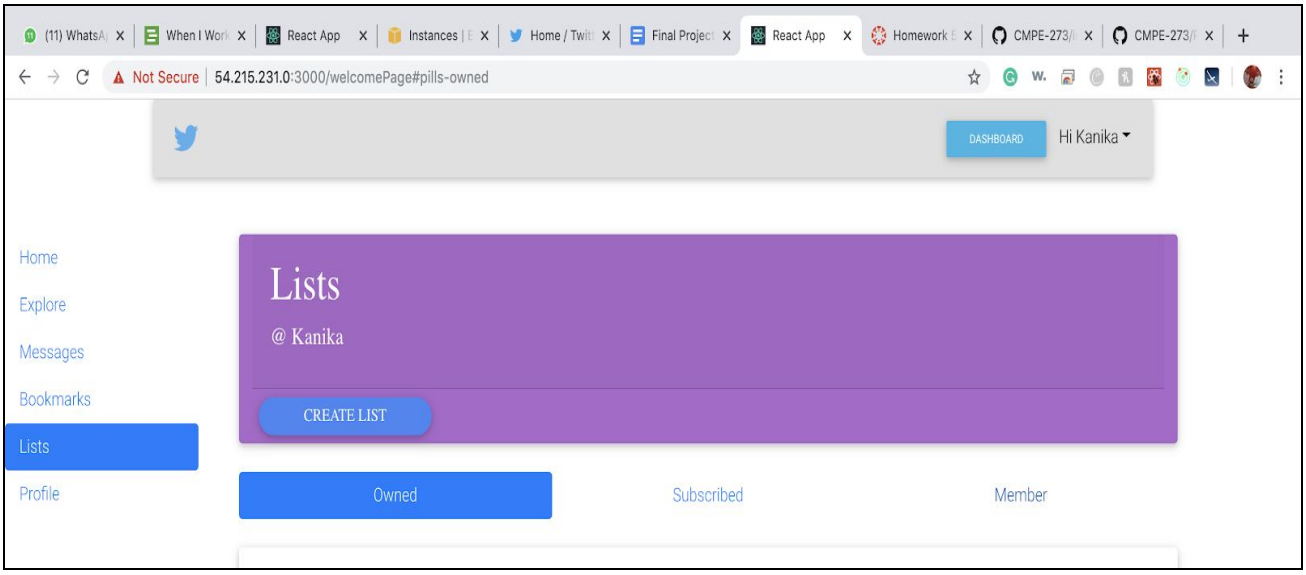
Lists subscribed by user



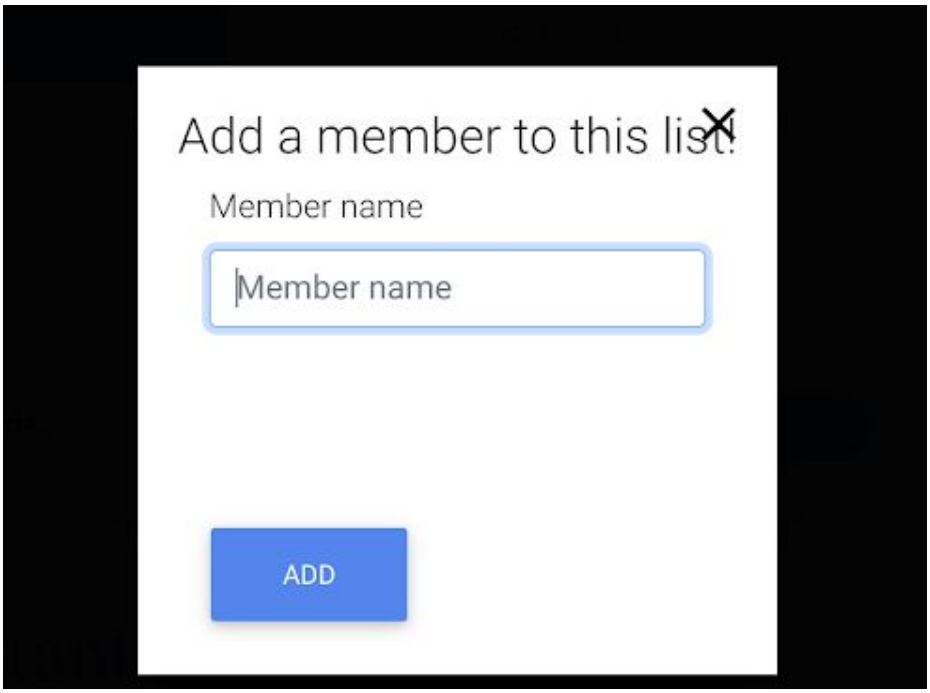
Lists user member



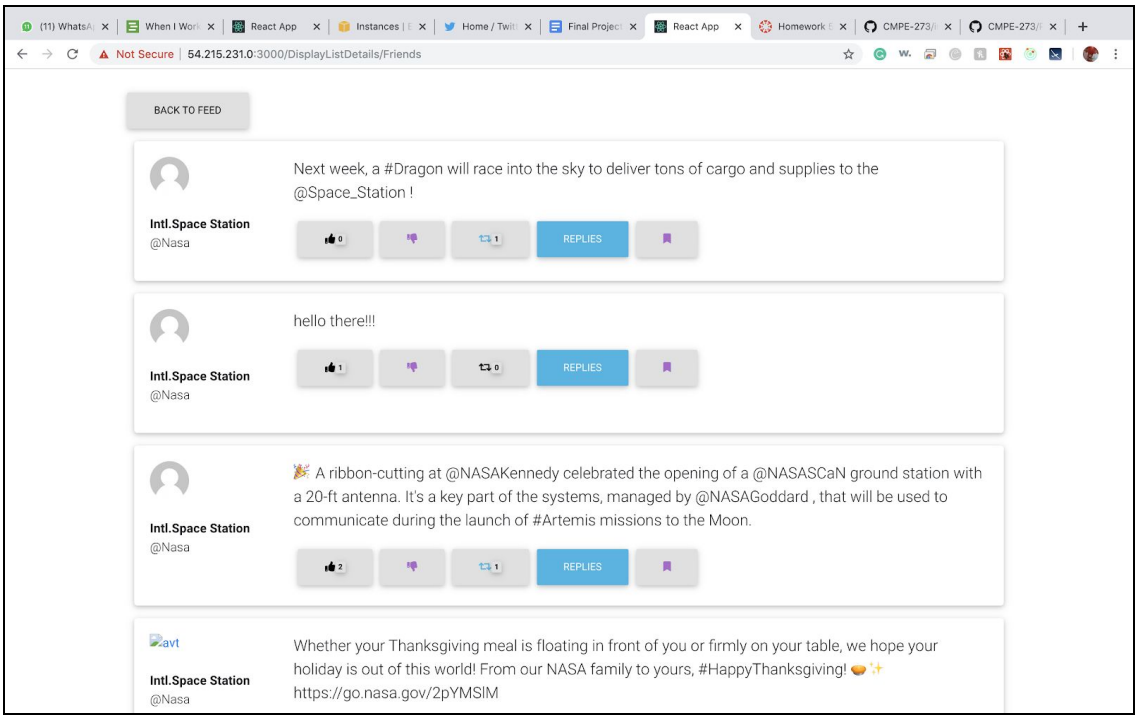
Creating List



Adding member to List

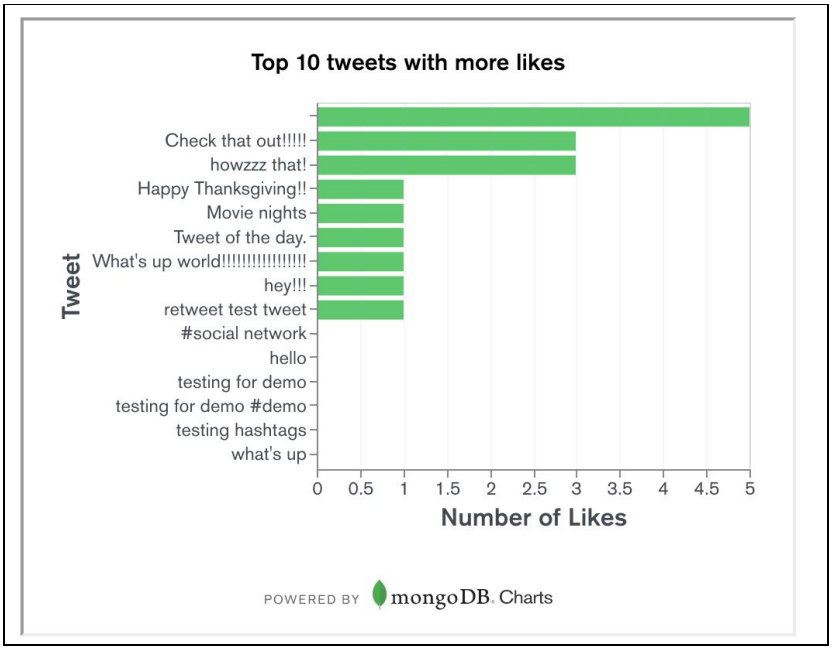


View Tweets in List

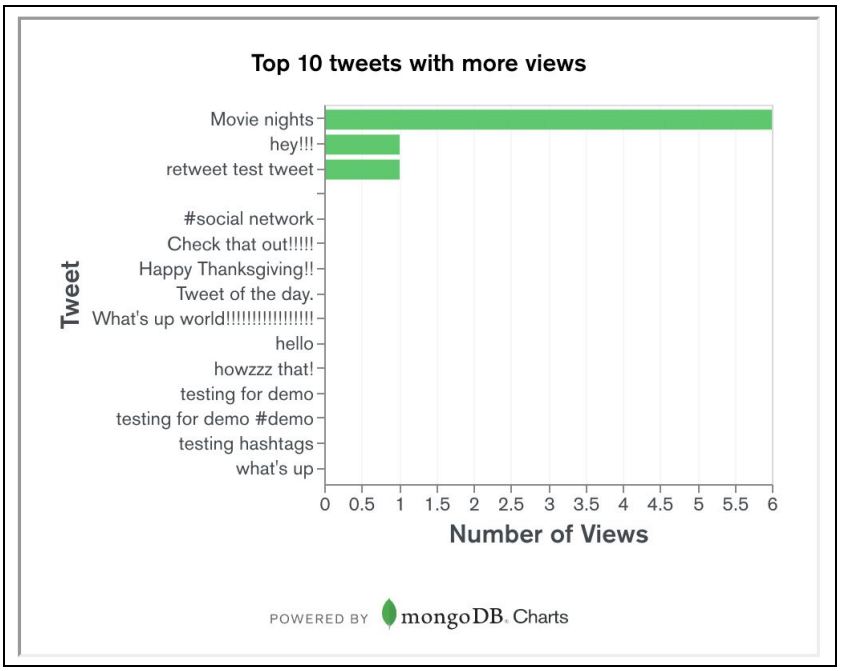


Graphs

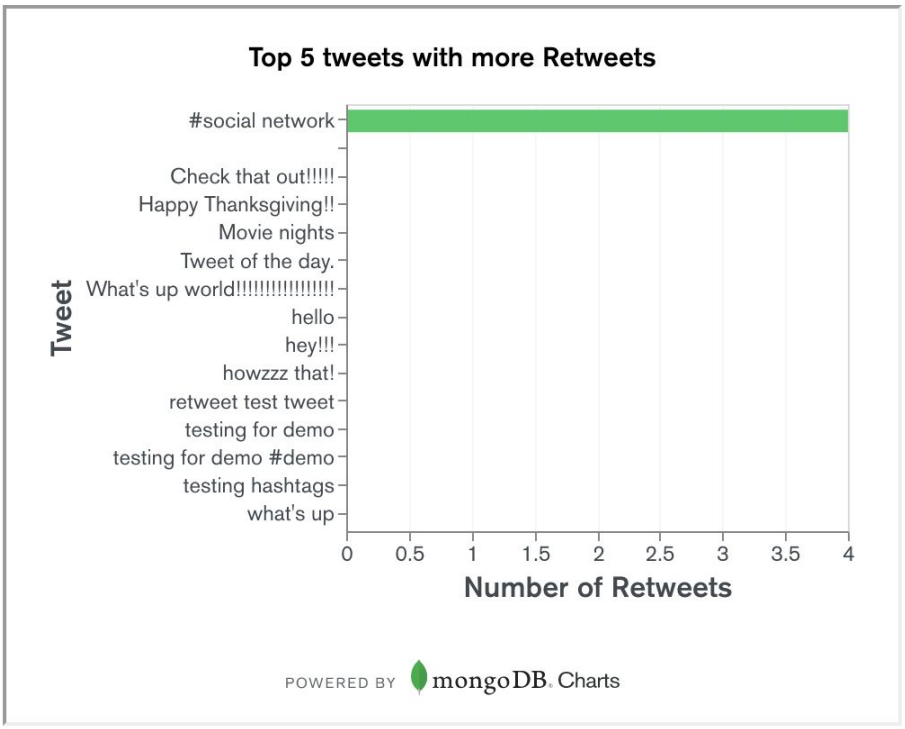
Top 10 tweets with more likes



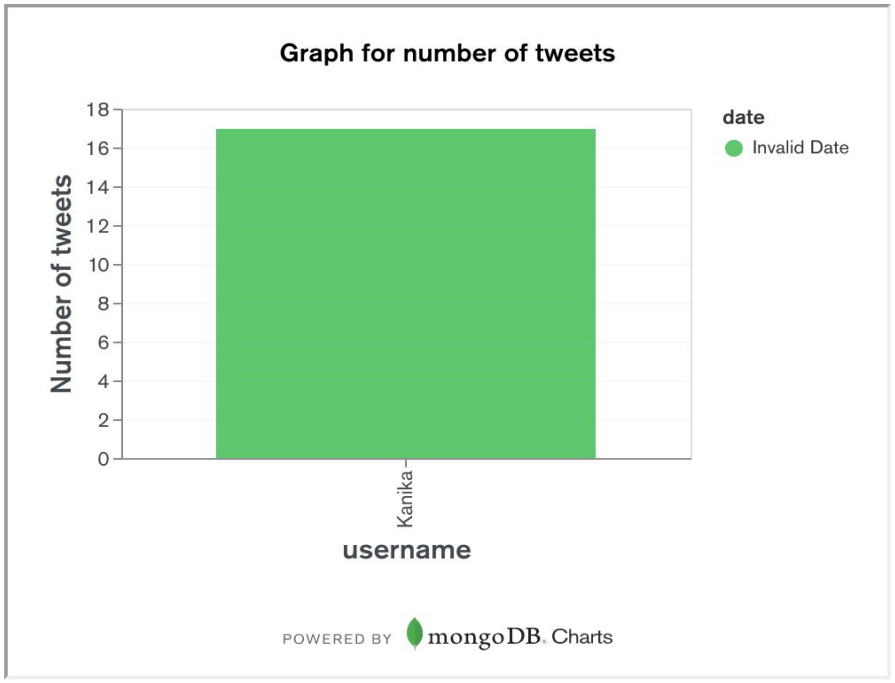
Top 10 tweets with more views



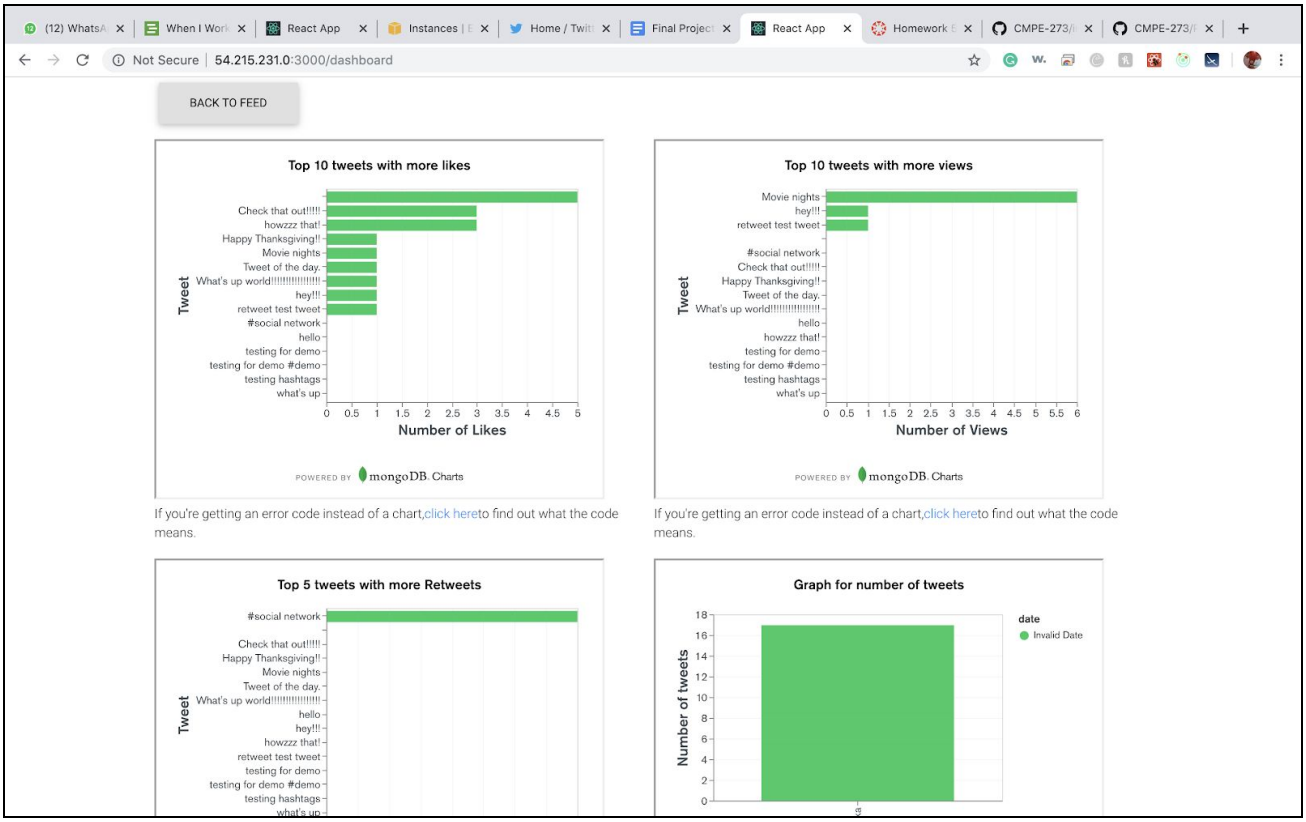
Top 5 Tweets with more Retweets



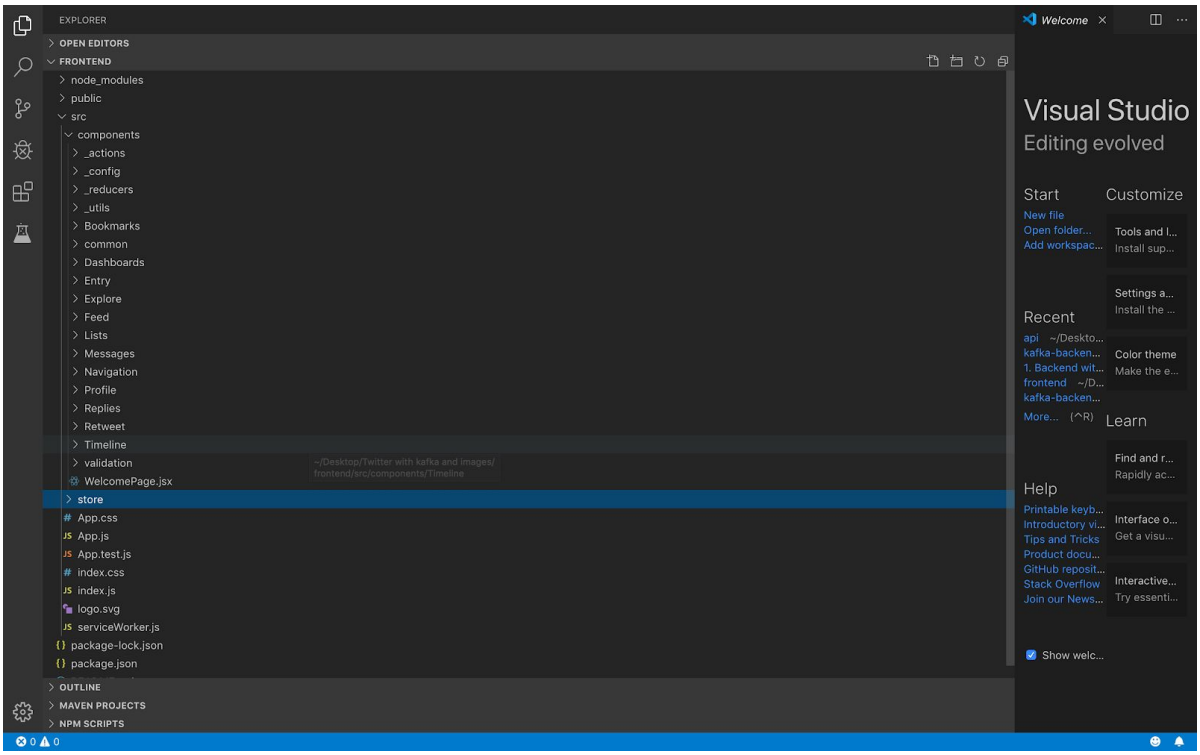
Number of tweets

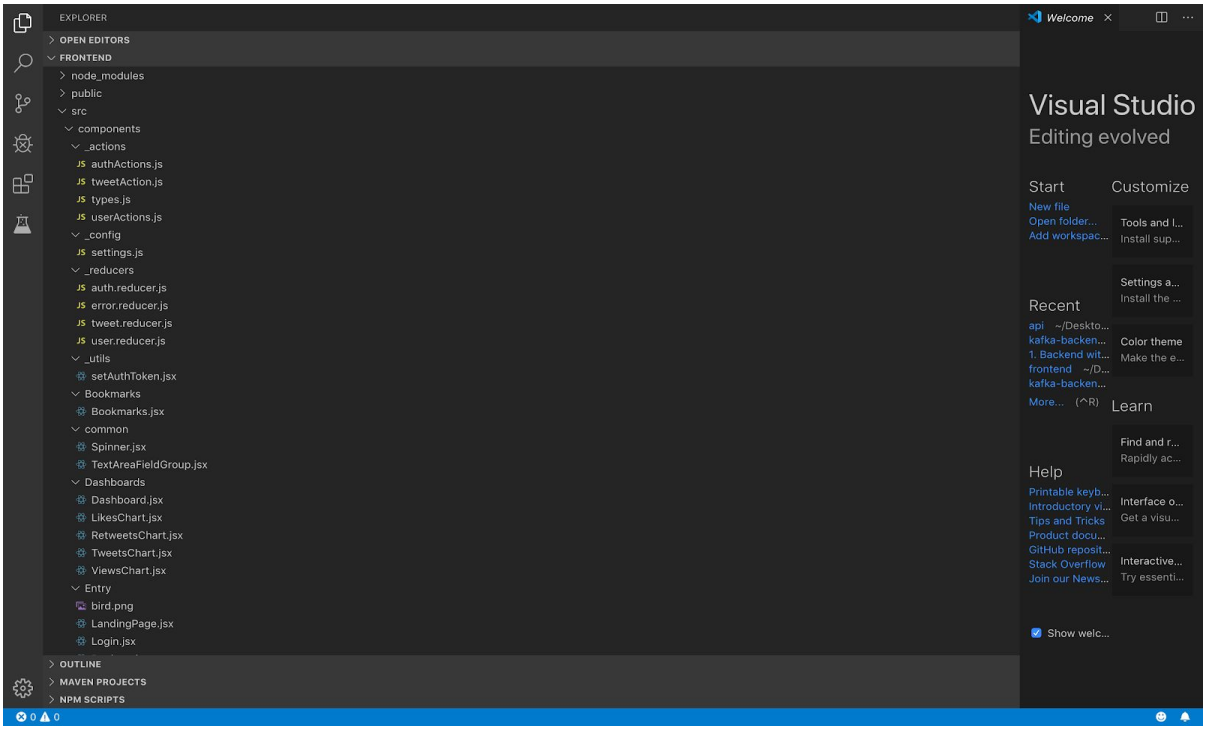


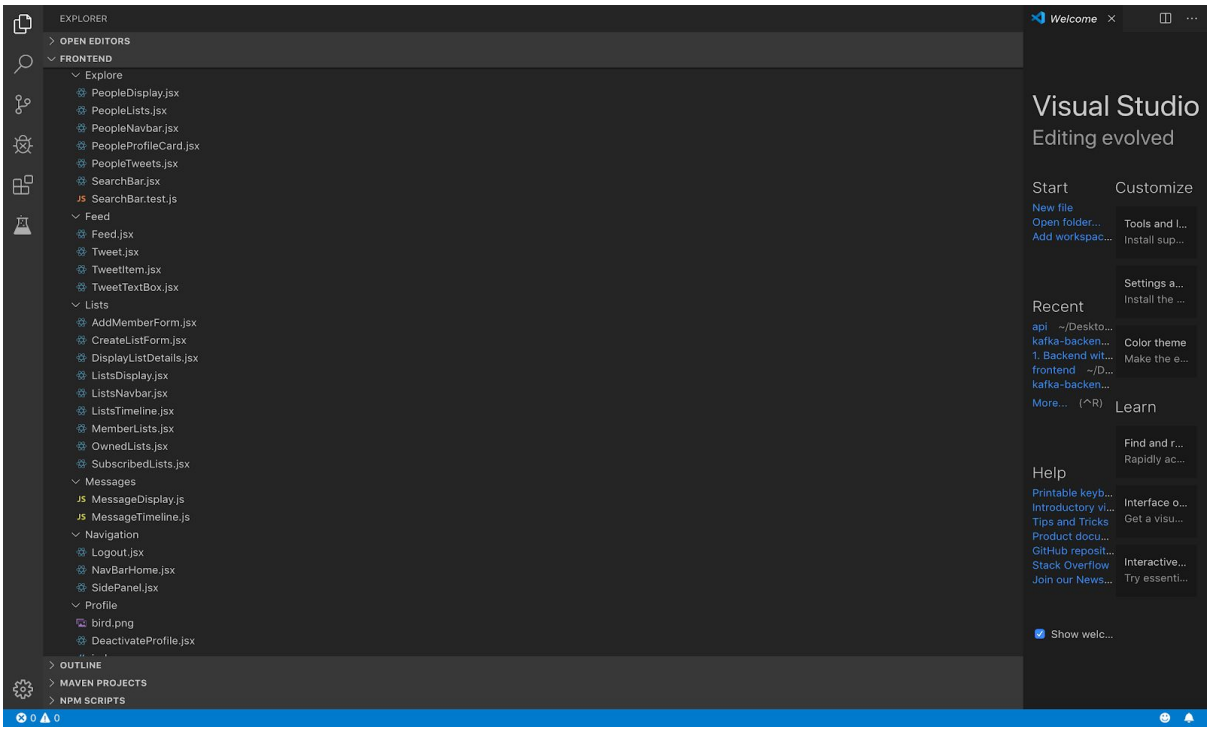
User Dashboard

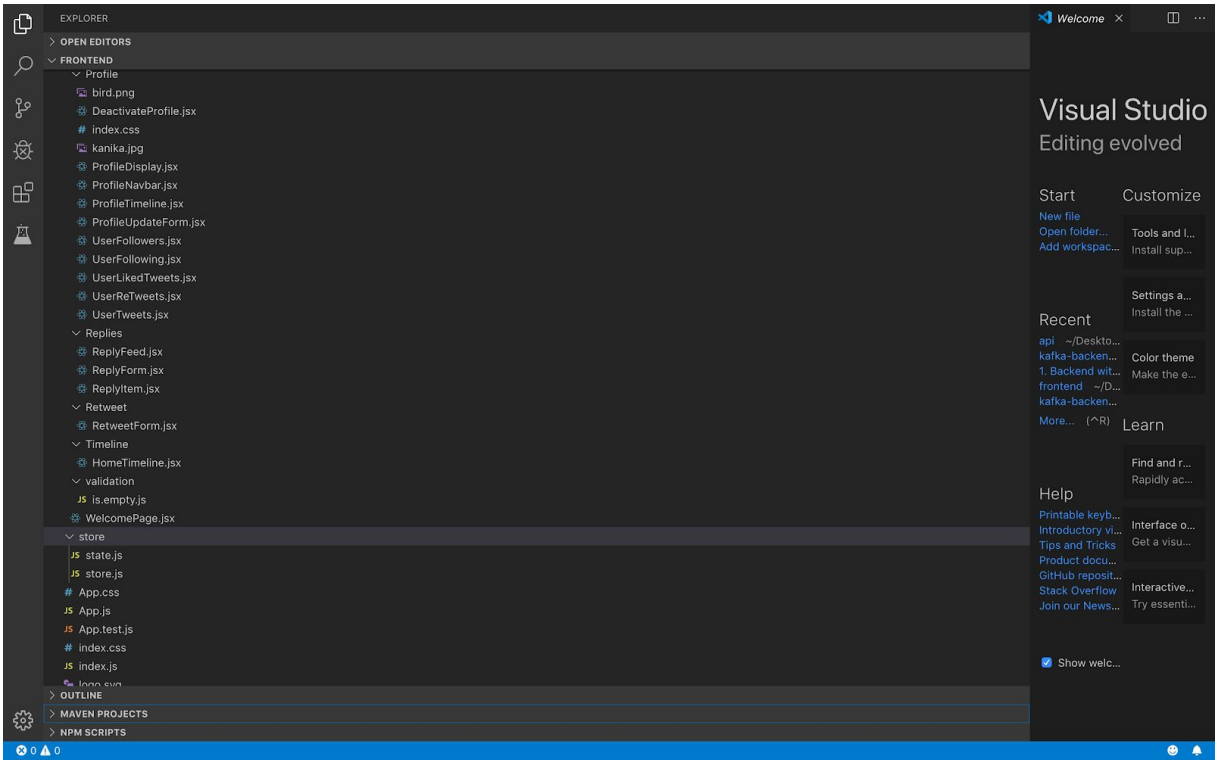


A code listing of client application:

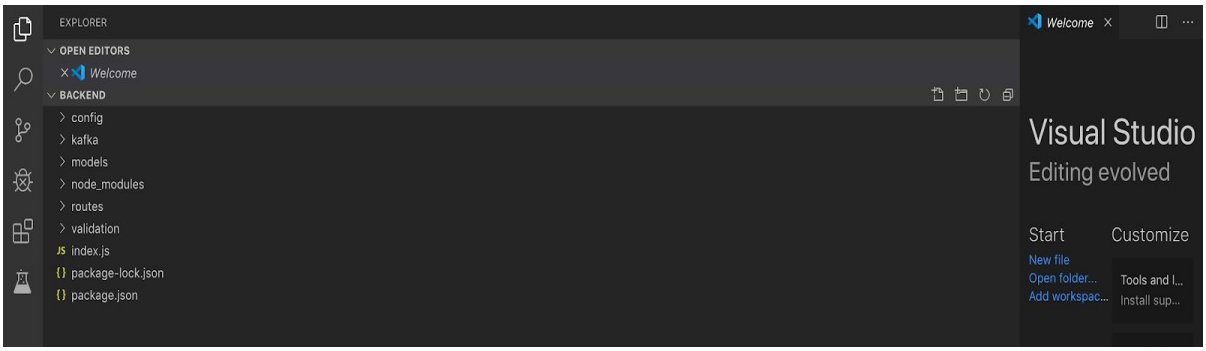


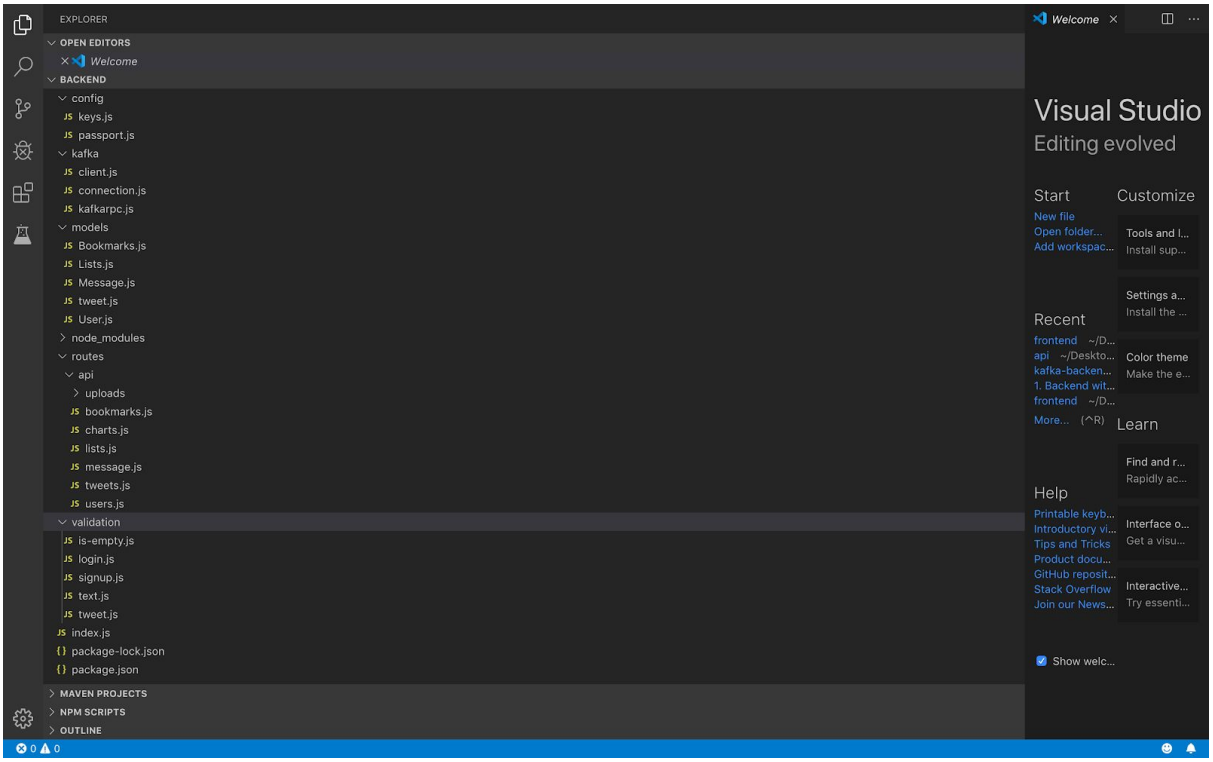



EXPLORER
OPEN EDITORS
FRONTEND
Explore
PeopleDisplay.jsx
PeopleLists.jsx
PeopleNavbar.jsx
PeopleProfileCard.jsx
PeopleTweets.jsx
SearchBar.jsx
SearchBar.test.js
Feed
Feed.jsx
Tweet.jsx
TweetItem.jsx
TweetTextBox.jsx
Lists
AddMemberForm.jsx
CreateListForm.jsx
DisplayListDetails.jsx
ListsDisplay.jsx
ListsNavbar.jsx
ListsTimeline.jsx
MemberLists.jsx
OwnedLists.jsx
SubscribedLists.jsx
Messages
MessageDisplay.js
MessageTimeline.js
Navigation
Logout.jsx
NavBarHome.jsx
SidePanel.jsx
Profile
bird.png
DeactivateProfile.jsx
OUTLINE
MAVEN PROJECTS
NPM SCRIPTS
Welcome
Visual Studio
Editing evolved
Start
New file
Open folder...
Add workspac...
Customize
Tools and l...
Install sup...
Settings a...
Install the ...
Recent
api ~/Deskto...
kafka-backen...
1. Backend wit...
frontend ~/D...
kafka-backen...
More... (^R)
Color theme
Make the e...
Learn
Find and r...
Rapidly ac...
Help
Printable keyb...
Introductory vi...
Tips and Tricks
Product docu...
GitHub reposit...
Stack Overflow
Join our News...
Interface o...
Get a visu...
Interactive...
Try essenti...
Show welc...


EXPLORER
OPEN EDITORS
FRONTEND
Profile
bird.png
DeactivateProfile.jsx
index.css
kanika.jpg
ProfileDisplay.jsx
ProfileNavbar.jsx
ProfileTimeline.jsx
ProfileUpdateForm.jsx
UserFollowers.jsx
UserFollowing.jsx
UserLikedTweets.jsx
UserReTweets.jsx
UserTweets.jsx
Replies
ReplyFeed.jsx
ReplyForm.jsx
ReplyItem.jsx
Retweet
RetweetForm.jsx
Timeline
HomeTimeline.jsx
validation
is.empty.js
WelcomePage.jsx
store
state.js
store.js
App.css
App.js
App.test.js
index.css
index.js
OUTLINE
MAVEN PROJECTS
NPM SCRIPTS
Welcome
Visual Studio
Editing evolved
Start
New file
Open folder...
Add workspac...
Customize
Tools and l...
Install sup...
Settings a...
Install the ...
Recent
api ~/Deskto...
kafka-backen...
1. Backend wit...
frontend ~/D...
kafka-backen...
More... (^R)
Color theme
Make the e...
Learn
Find and r...
Rapidly ac...
Help
Printable keyb...
Introductory vi...
Tips and Tricks
Product docu...
GitHub reposit...
Stack Overflow
Join our News...
Interface o...
Get a visu...
Interactive...
Try essenti...
Show welc...

A code listing of main server code





Server code listing for Activity Routes and Kafka-backend

```
...ain config/zookeeper.properties ... ...ka.Kafka config/server.properties ... ...nloads/kafka_2.11-1.1.0 — -bash ..._Terminal TERM=xterm-256color ..._Terminal TERM=xterm-256color ..._Terminal TERM=xterm-256color
    list_Name: 'neeeww',
    list_Desc: 'newwwww',
    subscribers: [],
    members: [ [Object] ],
    createdAt: 2019-12-04T08:18:27.312Z,
    updatedAt: 2019-12-04T08:18:41.499Z,
    __v: 1 } ]
after handle[object Object]
{ response_topic: { '0': 4 } }
{ response_topic: { '0': 5 } }
{ response_topic: { '0': 6 } }
profile is.... { follower_count: 6,
  following_count: 5,
  active: 'Active',
  views: 422,
  _id: 5dcde47b914b4b8ea83a0072,
  username: 'Kanika',
  first_name: 'Kanika',
  last_name: 'Khanna',
  password:
   '$2a$10$R6S6yI7rm/6VV3N1Us0JBOn5BpjUfrZBnTK7DsANPmd9zazNEGe2m',
  email: 'kaniks.k247@gmail.com',
  avatar:
   '//www.gravatar.com/avatar/68479a1e0445e3d6583b55741660d330?s=200&r=pg&d=mm',
  followers:
   [ { _id: 5de74d30e926db492edbfc64,
       follower_id: 5de6f223ce75533b93a0efdb',
       follower_name: 'madhu' },
     { _id: 5de565223d53b00eae1929f4,
       follower_id: 5dce038639b452aa3fe6fa9b',
       follower_name: 'Sruthi' },
     { _id: 5de4870b32fe71105c7f813b,
       follower_id: 5de486f932fe71105c7f813B',
       follower_name: 'mahesh108' },
     { _id: 5de47df2cc5218fea7118e47,
       follower_id: 5de2d0e654b572058ca81bca',
       follower_name: 'mahesh' },
     { _id: 5dd958544666dd15b4d01cf4,
       follower_id: 5dd8b437c7cb132514cc4eda',
       follower_name: 'kmanish' },
     { _id: 5dd957af27cc4522Ace583af,
       follower_id: 5dd8b437c7cb132514cc4eda',
       follower_name: 'Kanika' } ],
  following:
   [ { _id: 5de770462df41a1bddca2c6f, following_name: 'Divya' },
     { _id: 5de487352b384001c35956ee, following_name: 'mahesh108' },
     { _id: 5de479c2cc5218fea7118e43, following_name: 'Sruthi' },
     { _id: 5de2e466d59c59e7312bbd0b,
       following_id: 5dd8b437c7cb132514cc4eda',
       following_name: 'kmanish' },
     { _id: 5ddf90e52598dc4af233f07c,
       following_id: 5dd8b437c7cb132514cc4eda',
       following_name: 'Kanika' } ],
  date: 2019-11-14T23:34:19.888Z,
  __v: 38,
  city: 'San Jose 1',
  description: 'My name is Kanika test',
  state: 'California',
  zipcode: 95 }
after handle[object Object]
{ response_topic: { '0': 7 } }
```

```
...ain config/zookeeper.properties ... ...ka.Kafka config/server.properties ... ...nloads/kafka_2.11-1.1.0 — -bash ..._Terminal TERM=xterm-256color ..._Terminal TERM=xterm-256color ... ..._Terminal TERM=xterm-256color
  _id: 5dce038639b452aa3fe6fa9b,
  username: 'Sruthi',
  first_name: 'Sruthi',
  last_name: 'Duvvuri',
  password:
   '$2a$10$1/U6/HCH7hgpEeysnTiTQ.j5pyBZR6TM3ANulKK0dNmdscoMdEpum',
  email: 'sruthi.duvvuri1@gmail.com',
  avatar:
   '//www.gravatar.com/avatar/58f05caa3c763a6770030da9bbb2e303?s=200&r=pg&d=mm',
  followers:
   [ { _id: 5de74f3ae926db492edbfc74,
       follower_id: 5de6f223ce75533b93a0efdb',
       follower_name: 'madhu' },
     { _id: 5de479c2cc5218fea7118e44,
       follower_id: 5dcde47b914b4b8ea83a0072',
       follower_name: 'Kanika' } ],
  following:
   [ { _id: 5de565223d53b00eae1929f3,
       following_id: 5dcde47b914b4b8ea83a0072',
       following_name: 'Kanika' } ],
  date: 2019-11-15T01:46:46.629Z,
  __v: 11,
  city: 'San Jose',
  description: 'Persuing MSSE at SJSU',
  state: 'California',
  zipcode: 95112 }
after handle[object Object]
{ response_topic: { '0': 8 } }
{ response_topic: { '0': 9 } }
message received for get_user_owned_lists  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"7ff14c56d1a037b8d8978f1425bB59e0\",\"replyTo\":\"response_topic\",\"data\":{\"ownerID\":\"5dce038639b452aa3fe6fa9b\"}}"
inside get_user_owned_lists api of kafka backend
user owned lists are are.... [ { _id: 5de4704212f0a0fa23a08031,
    ownerID: 5dce038639b452aa3fe6fa9b ',
    list_Name: 'CMPE-273',
    list_Desc: 'CMPE-273',
    subscribers: [ [Object] ],
    members: [ [Object] ],
    createdAt: 2019-12-02T02:00:34.543Z,
    updatedAt: 2019-12-02T04:46:16.650Z,
    __v: 2 },
  { _id: 5de5a81c330b9e1b6f692829,
    ownerID: 5dce038639b452aa3fe6fa9b',
    list_Name: 'New test list',
    list_Desc: 'New test list',
    subscribers: [],
    members: [ [Object] ],
    createdAt: 2019-12-03T00:11:08.783Z,
    updatedAt: 2019-12-03T00:11:31.201Z,
    __v: 1 },
  { _id: 5de744feb3a28128f13b9355,
    ownerID:  5dce038639b452aa3fe6fa9b',
    list_Name: 'To-Do list',
    list_Desc: 'To-Do list',
    subscribers: [],
    members: [ [Object] ],
    createdAt: 2019-12-04T05:32:46.962Z,
    updatedAt: 2019-12-04T05:33:02.047Z,
    __v: 1 } ]
after handle[object Object]
{ response_topic: { '0': 10 } }
```

Server code listing for Auth Routes and Kafka-backend

```
...ain config/zookeeper.properties   ...ka.Kafka config/server.properties   ...nloads/kafka_2.11-1.1.0 — -bash   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color
client ready!
(node:7442) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
mongodb connected
message received for login  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"c4edccbe6c4627a964b10e560e055d5c\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\",\"password\":\"12345\"}}"
inside login of kafka backend
match found
payload is  { id: '5dcde47b914b4b8ea83a0072',
  first_name: 'Kanika',
  last_name: 'Khanna',
  username: 'Kanika',
  avatar:
   '//www.gravatar.com/avatar/68479a1e0445e3d6583b55741660d330?s=200&r=pg&d=mm' }
after handle[object Object]
{ response_topic: { '0': 0 } }
message received for get_followers  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"c15740589e28f808225d96dfecbb74ca\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_followers api of kafka backend. username is.. Kanika
message received for get_following  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"e5d6212af622f3f852c7f354830219e1\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_following api of kafka backend. username is.. Kanika
message received for get_followers  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"b647d1f1d95d2a16d831f001211a4025a\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_followers api of kafka backend. username is.. Kanika
message received for get_user_owned_lists  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"3994cce57507492d8ba102dcee7e139b\",\"replyTo\":\"response_topic\",\"data\":{\"ownerID\":\"5dcde47b914b4b8ea83a0072\"}}"
inside get_user_owned_lists api of kafka backend
message received for get_subscribed_lists  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"d75f6448014f8cbbe371a5621e78f9617\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_subscribed_lists api of kafka backend
followers are .... [ { _id: 5de74d30e926db492edbfc64,
    follower_id: '5de6f223ce75533b93a0efdb',
    follower_name: 'madhu' },
  { _id: 5de565223d53b00ea1929f4,
    follower_id: '5dce038639b452aa3fe6fa9b',
    follower_name: 'Sruthi' },
  { _id: 5de4870b32fe71105c7f813b,
    follower_id: '5de486f932fe71105c7f8138',
    follower_name: 'mahesh108' },
  { _id: 5de47df2cc5218fea7118e47,
    follower_id: '5de2d0e654b5f2058ca81bca',
    follower_name: 'mahesh' },
  { _id: 5dd958544666dd15b4d01cf4,
    follower_id: '5dd8b437c7cb132514cc4eda',
    follower_name: 'kmanish' },
  { _id: 5dd957ef27cc45224ce683af,
    follower_id: '5dd8b437c7cb132514cc4eda',
    follower_name: 'Kanika' } ]
after handle[object Object]
{ response_topic: { '0': 1 } }
message received for get_member_lists_of_user  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"7cc879201a81d3c225a894dd6f67eb4a\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_member_lists_of_user api of kafka backend
following are.... [ { _id: 5de770462df41a1bddca2c6f, following_name: 'Divya' },
  { _id: 5de487352b3840016c3595ee, following_name: 'mahesh108' },
  { _id: 5de479c2cc5218fea7118e43, following_name: 'Sruthi' },
  { _id: 5de2e466d59c59e7312bbd0b,
    following_id: '5dd8b437c7cb132514cc4eda',
    following_name: 'kmanish' },
  { _id: 5ddf90e52598dc4af233f07c,
    following_id: '5dd8b437c7cb132514cc4eda',
    following_name: 'Kanika' } ]
```

```
...ain config/zookeeper.properties   ...ka.Kafka config/server.properties   ...nloads/kafka_2.11-1.1.0 — -bash   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color
{ response_topic: { '0': 3 } }
message received for get_profile  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"9d4fd1c0d0196a58e58cba63dbb1867f\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_profile of kafka backend
user owned lists are are.... [ { _id: 5dda282decd14b0c1cf81966,
    ownerID: '5dd8b437c7cb132514cc4eda',
    list_Name: 'mylist',
    list_Desc: 'my friends list',
    make_Private: 'Yes',
    subscribers: [ [Object], [Object], [Object], [Object] ],
    members: [ [Object] ],
    createdAt: 2019-11-24T06:50:21.843Z,
    updatedAt: 2019-11-24T20:46:26.894Z,
    __v: 4 } ]
after handle[object Object]
user owned lists are are.... [ { _id: 5dd9f54fe413fd310839190e,
    ownerID: '48935348509435',
    list_Name: '273project',
    list_Desc: 'Projectteammates',
    make_Private: 'Yes',
    subscribers: [ [Object] ],
    members: [],
    createdAt: 2019-11-24T03:13:19.500Z,
    updatedAt: 2019-11-24T03:13:19.500Z,
    __v: 0 },
  { _id: 5de4704212f0a0fa23a00031,
    ownerID: '5dce038639b452aa3fe6fa9b',
    list_Name: 'CMPE-273',
    list_Desc: 'CMPE-273',
    subscribers: [ [Object] ],
    members: [ [Object] ],
    createdAt: 2019-12-02T02:00:34.543Z,
    updatedAt: 2019-12-02T04:46:16.650Z,
    __v: 2 } ]
after handle[object Object]
user owned lists are are.... [ { _id: 5de37553a083fff6e8ad2b2b,
    ownerID: 5dcde47b914b4b8ea83a0072',
    list_Name: 'Friends',
    list_Desc: 'This list includes my close friends.',
    subscribers: [ [Object], [Object] ],
    members: [ [Object], [Object] ],
    createdAt: 2019-12-01T08:09:55.471Z,
    updatedAt: 2019-12-03T00:12:31.535Z,
    __v: 4 },
  { _id: 5de4983c638f930287a0f60c,
    ownerID: '5dcde47b914b4b8ea83a0072',
    list_Name: 'Project-Important',
    list_Desc: 'Very imp',
    subscribers: [],
    members: [ [Object] ],
    createdAt: 2019-12-02T04:51:08.762Z,
    updatedAt: 2019-12-02T22:28:12.714Z,
    __v: 1 },
  { _id: 5de6f408934c6c51f4a858d2,
    ownerID: '5dcde47b914b4b8ea83a0072',
    list_Name: 'test list 1',
    list_Desc: 'hey test',
    subscribers: [],
    members: [ [Object], [Object] ],
    createdAt: 2019-12-03T23:47:20.122Z,
    updatedAt: 2019-12-03T23:47:48.531Z,
    __v: 2 },
```

Server code listing for Message Routes and Kafka-backend

```
...ain config/zookeeper.properties ...  ...ka.Kafka config/server.properties ...  ...nloads/kafka_2.11-1.1.0 — -bash  ..._Terminal TERM=xterm-256color  ..._Terminal TERM=xterm-256color ...  ..._Terminal TERM=xterm-256color
    { _id: 5dd958544666dd15b4d01cf4,
      follower_id: '5dd8b437c7cb132514cc4eda',
      follower_name: 'kmanish' },
    { _id: 5dd957af27cc45224ce583af,
      follower_id: '5dd8b437c7cb132514cc4eda',
      follower_name: 'Kanika' } ],
  following:
   [ { _id: 5de770462df41a1bddca2c6f, following_name: 'Divya' },
     { _id: 5de487352b3840016c3595ee, following_name: 'mahesh108' },
     { _id: 5de479c2cc5218fea7118e43, following_name: 'Sruthi' },
     { _id: 5de2e466d59c59e7312bbd0b,
       following_id: '5dd8b437c7cb132514cc4eda',
       following_name: 'kmanish' },
     { _id: 5ddf90e52598dc4af233f07c,
       following_id: '5dd8b437c7cb132514cc4eda',
       following_name: 'Kanika' } ],
  date: 2019-11-14T23:34:19.888Z,
  __v: 38,
  city: 'San Jose 1',
  description: 'My name is Kanika test',
  state: 'California',
  zipcode: 95 }
after handle[object Object]
{ response_topic: { '0': 15 } }
message received for get_followers  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"6f22dc94e36d46e93a012532e4c3daa0\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_followers api of kafka backend. username is.. Kanika
followers are .... [ { _id: 5de74d30e926db492edbfc64,
    follower_id: '5de6f223ce75533b93a0efdb',
    follower_name: 'madhu' },
  { _id: 5de565223d53b00eae1929f4,
    follower_id: '5dce038639b452aa3fe6fa9b',
    follower_name: 'Sruthi' },
  { _id: 5de4870b32fe71105c7f813b,
    follower_id: '5de486f932fe71105c7f8138',
    follower_name: 'mahesh108' },
  { _id: 5de47df2cc5218fea7118e47,
    follower_id: '5de2d0e654b5f2058ca81bca',
    follower_name: 'mahesh' },
  { _id: 5dd958544666dd15b4d01cf4,
    follower_id: '5dd8b437c7cb132514cc4eda',
    follower_name: 'kmanish' },
  { _id: 5dd957af27cc45224ce583af,
    follower_id: '5dd8b437c7cb132514cc4eda',
    follower_name: 'Kanika' } ]
after handle[object Object]
{ response_topic: { '0': 16 } }
message received for get_following  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"c8264ece5338880e1efcbe8a1c61c62\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_following api of kafka backend. username is.. Kanika
following are.... [ { _id: 5de770462df41a1bddca2c6f, following_name: 'Divya' },
  { _id: 5de487352b3840016c3595ee, following_name: 'mahesh108' },
  { _id: 5de479c2cc5218fea7118e43, following_name: 'Sruthi' },
  { _id: 5de2e466d59c59e7312bbd0b,
    following_id: '5dd8b437c7cb132514cc4eda',
    following_name: 'kmanish' },
  { _id: 5ddf90e52598dc4af233f07c,
    following_id: '5dd8b437c7cb132514cc4eda',
    following_name: 'Kanika' } ]
after handle[object Object]
{ response_topic: { '0': 17 } }
```

```
...ain config/zookeeper.properties ...  ...ka.Kafka config/server.properties ...  ...nloads/kafka_2.11-1.1.0 — -bash  ..._Terminal TERM=xterm-256color  ..._Terminal TERM=xterm-256color ...  ..._Terminal TERM=xterm-256color
    following_name: 'kmanish' },
  { _id: 5ddf90e52598dc4af233f07c,
    following_id: '5dd8b437c7cb132514cc4eda',
    following_name: 'Kanika' } ]
after handle[object Object]
{ response_topic: { '0': 24 } }
{ response_topic: { '0': 25 } }
message received for update_profile  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"7bb73d43d846ae1ae5a1836aa99b4a71\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\",\"first_name\":\"Kanika\",\"last_name\":\"Khanna\",\"description\":\"My name is Ka
nika \",\"city\":\"San Jose\",\"state\":\"CA\",\"zipcode\":\"95112\"}}"
inside update profile of kafka backend
new profile is.... { follower_count: 6,
  following_count: 5,
  active: 'Active',
  views: 422,
  _id: 5dcde47b914b4b8ea83a0072,
  username: 'Kanika',
  first_name: 'Kanika',
  last_name: 'Khanna',
  password:
   '$2a$10$R6S6yI7rm/6VV3N1Us0JBOn5BpjUfrZBnTK7DsANPmd9zazNEGe2m',
  email: 'kaniks.k247@gmail.com',
  avatar:
   '//www.gravatar.com/avatar/68479a1e0445e3d6583b55741660d330?s=200&r=pg&d=mm',
  followers:
   [ { _id: 5de74d30e926db492edbfc64,
       follower_id: '5de6f223ce75533b93a0efdb',
       follower_name: 'madhu' },
     { _id: 5de565223d53b00eae1929f4,
       follower_id: '5dce038639b452aa3fe6fa9b',
       follower_name: 'Sruthi' },
     { _id: 5de4870b32fe71105c7f813b,
       follower_id: '5de486f932fe71105c7f8138',
       follower_name: 'mahesh108' },
     { _id: 5de47df2cc5218fea7118e47,
       follower_id: '5de2d0e654b5f2058ca81bca',
       follower_name: 'mahesh' },
     { _id: 5dd958544666dd15b4d01cf4,
       follower_id: '5dd8b437c7cb132514cc4eda',
       follower_name: 'kmanish' },
     { _id: 5dd957af27cc45224ce583af,
       follower_id: '5dd8b437c7cb132514cc4eda',
       follower_name: 'Kanika' } ],
  following:
   [ { _id: 5de770462df41a1bddca2c6f, following_name: 'Divya' },
     { _id: 5de487352b3840016c3595ee, following_name: 'mahesh108' },
     { _id: 5de479c2cc5218fea7118e43, following_name: 'Sruthi' },
     { _id: 5de2e466d59c59e7312bbd0b,
       following_id: '5dd8b437c7cb132514cc4eda',
       following_name: 'kmanish' },
     { _id: 5ddf90e52598dc4af233f07c,
       following_id: '5dd8b437c7cb132514cc4eda',
       following_name: 'Kanika' } ],
  date: 2019-11-14T23:34:19.888Z,
  __v: 38,
  city: 'San Jose',
  description: 'My name is Kanika ',
  state: 'CA',
  zipcode: 95112 }
after handle[object Object]
{ response_topic: { '0': 26 } }
```

```
...ain config/zookeeper.properties   ....Kafka config/server.properties   ...nloads/kafka_2.11-1.1.0 — -bash   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color
{ response_topic: { '0': 33 } }
message received for login  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"07e7c720f7545ac73496ff16ff1713a5\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Sruthi\",\"password\":\"12345\"}}"
inside login of kafka backend
match found
payload is  { id: '5dce038639b452aa3fe6fa9b',
  first_name: 'Sruthi',
  last_name: 'Duvvuri',
  username: 'Sruthi',
  avatar:
   '//www.gravatar.com/avatar/58f05caa3c763a6770030da9bbb2e303?s=200&r=pg&d=mm' }
after handle[object Object]
{ response_topic: { '0': 34 } }
message received for get_followers  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"e58cf009864ef46cc7917ce6371ec466\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Sruthi\"}}"
inside get_followers api of kafka backend. username is.. Sruthi
message received for get_user_owned_lists  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"f5aab8dd5f05a81aea8d86d739913707\",\"replyTo\":\"response_topic\",\"data\":{\"ownerID\":\"5dce038639b452aa3fe6fa9b\"}}"
inside get_user_owned_lists api of kafka backend
message received for get_subscribed_lists  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"243dc66c0a23d3a3ad6a53dbc594df66\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Sruthi\"}}"
inside get_subscribed_lists api of kafka backend
message received for get_member_lists_of_user  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"09e38ea310b6c906adcd0ba417be90d6\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Sruthi\"}}"
inside get_member_lists_of_user api of kafka backend
message received for get_profile  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"b92533a55a5b0ae88fec1b226300000a\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Sruthi\"}}"
inside get_profile of kafka backend
followers are .... [ { _id: 5de74f3ae926db492edbfc74,
    follower_id: '5de6f223ce75533b93a0efdb',
    follower_name: 'madhu' },
  { _id: 5de479c2cc5218fea7118e44,
    follower_id: '5dcde47b914b4b8ea83a0072',
    follower_name: 'Kanika' } ]
after handle[object Object]
user owned lists are are.... [ { _id: 5de37553a083fff6e8ad2b2b,
    ownerID: '5dcde47b914b4b8ea83a0072',
    list_Name: 'Friends',
    list_Desc: 'This list includes my close friends.',
    subscribers: [ [Object], [Object] ],
    members: [ [Object], [Object] ],
    createdAt: 2019-12-01T08:09:55.471Z,
    updatedAt: 2019-12-03T00:12:31.535Z,
    __v: 4 } ]
after handle[object Object]
user owned lists are are.... [ { _id: 5de4704212f0a0fa23a08031,
    ownerID: '5dce038639b452aa3fe6fa9b',
    list_Name: 'CMPE-273',
    list_Desc: 'CMPE-273',
    subscribers: [ [Object] ],
    members: [ [Object] ],
    createdAt: 2019-12-02T02:00:34.543Z,
    updatedAt: 2019-12-02T04:46:16.650Z,
    __v: 2 },
  { _id: 5de5a81c330b9e1b6f692829,
    ownerID: '5dce038639b452aa3fe6fa9b',
    list_Name: 'New test list',
    list_Desc: 'New test list',
    subscribers: [],
    members: [ [Object] ],
    createdAt: 2019-12-03T00:11:08.783Z,
    updatedAt: 2019-12-03T00:11:31.201Z,
```

Server code listing for User Routes and Kafka-backend

```
...ain config/zookeeper.properties   ...ka.Kafka config/server.properties   ...nloads/kafka_2.11-1.1.0 — -bash   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color
{ response_topic: { '0': 3 } }
message received for get_profile  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"9d4fd1c00019ba58e58cba63dbb1867f\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_profile of kafka backend
user owned lists are are.... [ { _id: 5dda282decd14b0c1cf81966,
    ownerID: '5dd8b437c7cb132514cc4eda',
    list_Name: 'mylist',
    list_Desc: 'my friends list',
    make_Private: 'Yes',
    subscribers: [ [Object], [Object], [Object], [Object] ],
    members: [ [Object] ],
    createdAt: 2019-11-24T06:50:21.843Z,
    updatedAt: 2019-11-24T20:46:26.894Z,
    __v: 4 } ]
after handle[object Object]
user owned lists are are.... [ { _id: 5dd9f54fe413fd310839190e,
    ownerID: '4893534850943 5',
    list_Name: '273project',
    list_Desc: 'Projectteammates',
    make_Private: 'Yes',
    subscribers: [ [Object] ],
    members: [],
    createdAt: 2019-11-24T03:13:19.500Z,
    updatedAt: 2019-11-24T03:13:19.500Z,
    __v: 0 },
  { _id: 5de4704212f0a0fa23a08031,
    ownerID: '5dce038639b452aa3fe6fa9b',
    list_Name: 'CMPE-273',
    list_Desc: 'CMPE-273',
    subscribers: [ [Object] ],
    members: [ [Object] ],
    createdAt: 2019-12-02T02:00:34.543Z,
    updatedAt: 2019-12-02T04:46:16.650Z,
    __v: 2 } ]
after handle[object Object]
user owned lists are are.... [ { _id: 5de37553a083fff6e8ad2b2b,
    ownerID: '5dcde47b914b4b8ea83a0072',
    list_Name: 'Friends',
    list_Desc: 'This list includes my close friends.',
    subscribers: [ [Object], [Object] ],
    members: [ [Object], [Object] ],
    createdAt: 2019-12-01T08:09:55.471Z,
    updatedAt: 2019-12-03T00:12:31.535Z,
    __v: 4 },
  { _id: 5de4983c638f930287a0f60c,
    ownerID: '5dcde47b914b4b8ea83a0072',
    list_Name: 'Project-Important',
    list_Desc: 'Very imp',
    subscribers: [],
    members: [ [Object] ],
    createdAt: 2019-12-02T04:51:08.762Z,
    updatedAt: 2019-12-02T22:28:12.714Z,
    __v: 1 },
  { _id: 5de6f408934c6c51f4a858d2,
    ownerID: '5dcde47b914b4b8ea83a0072',
    list_Name: 'test list 1',
    list_Desc: 'hey test',
    subscribers: [],
    members: [ [Object], [Object] ],
    createdAt: 2019-12-03T23:47:20.122Z,
    updatedAt: 2019-12-03T23:47:48.531Z,
    __v: 2 },
```

A code listing of database access class

server.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > server.js > ...

```javascript
var connection = new require("./kafka/Connection");
var express = require("express");
var mongoose = require("mongoose");
var passport = require("passport");
var schema = mongoose.Schema;
var app = express();
const bodyParser = require("body-parser");
const db = require("./config/keys").mongoURI;
mongoose
  .connect(db)
  .then(() => console.log("mongodb connected"))
  .catch(err => console.log(err));
app.use(passport.initialize());
require("./config/passport")(passport);

function handleTopicRequest(topic_name, fname) {
  //var topic_name = 'root_topic';
  var consumer = connection.getConsumer(topic_name);
  var producer = connection.getProducer();
  console.log("server is running ");
  consumer.on("message", function(message) {
    console.log("message received for " + topic_name + " ", fname);
    console.log(JSON.stringify(message.value));
    var data = JSON.parse(message.value);

    fname.handle_request(data.data, function(err, res) {
      console.log("after handle" + res);
      var payloads = [
        {
          topic: data.replyTo,
          messages: JSON.stringify({
            correlationId: data.correlationId,
            data: res
          }),
          partition: 0
        }
      ];
      producer.send(payloads, function(err, data) {
        console.log(data);
      });
      return;
    });
  });
}
var register = require("./services/register.js");
var login = require("./services/login.js");
```

0 0 Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier

keys.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > config > keys.js > {e}<unknown>

```javascript
module.exports = {
  mongoURI:
    "mongodb+srv://admin:admin@cluster0-aloks.mongodb.net/twitterdb?retryWrites=true&w=majority",
  secretOrKey: "secret"
};
```

A code listing of test class

```
import { configure } from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';
configure({ adapter: new Adapter() });

const setUp = (props={})=>{
    const component = shallow(<SearchBar {...props}/>);
    return component;
}

describe('Search Bar component',()=>{

    let component;

    beforeEach(()=>{
        component = setUp();
    });

      it('should render without errors',()=>{
          const wrapper = component.find('.property-listing-content');
          expect(wrapper.length).toBe(2);
      });

      it('Should Render  buttons for people and topics',()=>{
        const button = component.find('.btn btn-primary btn-lg');
        expect(wrapper.length).toBe(2);
      })
})
```

A code listing of database creation class

Bookmarks.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > models > Bookmarks.js > ...

```js
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const BookmarkSchema = new Schema({
  user_id: {
    type: String
  },
  tweet_id: {
    type: String
  },
  tweet_content: {
    type: String
  }
});

module.exports = bookmarks = mongoose.model("bookmarks", BookmarkSchema);
```

Lists.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > models > Lists.js > ...

```js
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const ListSchema = new Schema(
  {
   listID:String,
   list_Name :String,
   ownerID:String,
   subscribers: [
    {
      user_id: {
        type: String,
      },
      username: {
        type: String
      }
    }
  ],
   members: [
    {
      user_id: {
        type: String
      },
      username: {
        type: String
      }
    }
  ],
   make_Private:String,
   list_Desc:String
  },
  { timestamps: true }
);

module.exports = lists = mongoose.model("lists", ListSchema);
```

0 0 Fetching data for better TypeScript IntelliSense Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier

User.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > models > User.js > ...

```javascript
const mongoose = require("mongoose");

const { Schema } = mongoose;
const UserSchema = new Schema({
  username: {
    type: String,
    required: true,
    unique: true
  },
  first_name: {
    type: String,
    required: true
  },
  last_name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  },
  city: {
    type: String,
    required: false
  },
  state: {
    type: String,
    required: false
  },
  zipcode: {
    type: Number,
    required: false
  },
  password: {
    type: String,
    required: true
  },
  description: {
    type: String,
    required: false
  },
  avatar: {
    type: String
  },
  followers: [
```

0 0 Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier

User.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > models > User.js > ...

```javascript
  followers: [
    {
      follower_id: {
        type: String,
        required: false
      },
      follower_name: {
        type: String,
        required: false
      }
    }
  ],
  follower_count: {
    type: Number,
    default: 0
  },
  following: [
    {
      following_id: {
        type: String,
        required: false
      },
      following_name: {
        type: String,
        required: false
      }
    }
  ],
  following_count: {
    type: Number,
    default: 0
  },
  active: {
    type: String,
    default: "Active"
  },
  date: {
    type: Date,
    default: Date.now
  },
  bookmarked_tweets: {
    type: Schema.Types.ObjectId,
    ref: "tweets"
  },
  views: {
    type: Number,
```

0 0 Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier

Message.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > models > Message.js > MessageSchema

```javascript
const mongoose = require("mongoose");

const { Schema } = mongoose;

// Create Schema

const MessageSchema = new Schema({
  sender_name: {
    type: String,
  },
  receiver_name: {
    type: String,
  },
  message: {
    type: String
  },
  date: {
    type: Date,
    default: Date.now
  },
});

module.exports = Message = mongoose.model("messages", MessageSchema);
```

0 0 Ln 17, Col 10 Spaces: 2 UTF-8 LF JavaScript Prettier

tweet.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > models > tweet.js > ...

```javascript
const mongoose = require("mongoose");

const { Schema } = mongoose;

// Create Schema

const TweetSchema = new Schema({
  user: {
    type: Schema.Types.ObjectId,
    ref: "users"
  },
  tweet_content: {
    type: String
  },
  username: {
    type: String
  },
  firstname: {
    type: String
  },
  lastname: {
    type: String
  },
  avatar: {
    type: String
  },
  replies: [
    {
      user: {
        type: Schema.Types.ObjectId,
        ref: "users"
      },
      text: {
        type: String
      },
      firstname: {
        type: String
      },
      lastname: {
        type: String
      },
      username: {
        type: String
      },
      avatar: {
        type: String
```

0 0 Fetching data for better TypeScript IntelliSense Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier

tweet.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > models > tweet.js > ...

```
      },
      avatar: {
        type: String
      },
      date: {
        type: Date,
        default: Date.now
      }
    }
  ],
  replies_count: {
    type: Number,
    default: 0
  },
  likes: [
    {
      user: {
        type: Schema.Types.ObjectId,
        ref: "users"
      },
      username: {
        type: String
      }
    }
  ],
  likes_count: {
    type: Number,
    default: 0
  },
  tweeted_date: {
    type: Date,
    default: Date.now
  },
  hashtags: {
    type: Array
  },
  imageList: {
    type: Array
  },
  retweeted_status: {
    user: {
      type: Schema.Types.ObjectId,
      ref: "users"
    },
    username: {
      type: String
```

0 0 Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier

tweet.js

Users > kanikakhanna > Desktop > Twitter with kafka and images > kafka-backend > models > tweet.js > ...

```
  },
  retweeted_status: {
    user: {
      type: Schema.Types.ObjectId,
      ref: "users"
    },
    username: {
      type: String
    },
    firstname: {
      type: String
    },
    lastname: {
      type: String
    },
    avatar: {
      type: String
    },
    tweet_content: {
      type: String
    },
    hashtags: {
      type: Array
    },
    imageList: {
      type: Array
    }
  },
  retweets_count: {
    type: Number,
    default: 0
  },
  retweeted: {
    type: Boolean
  },
  view_count: {
    type: Number,
    default: 0
  }
});

// eslint-disable-next-line no-multi-assign
module.exports = Tweet = mongoose.model("tweets", TweetSchema);
```

0 0 Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier

Mocha Testing:

1) Login API:

test > test.js > describe("Login API") callback > it("Should be successful if credential is valid") callback > expect() callback

```
var assert = require("chai").assert;
const request = require("supertest");
const app = require("../index");
const expect = require("chai").expect;

describe("Login API", function() {
  it("Should be successful if credential is valid", function(done) {
    request(app)
      .post("/api/users/login")
      .set("Accept", "application/json")
      .set("Content-Type", "application/json")
      .send({ username: "Nasa", password: "12345" })
      .expect(200)
      .expect(function(response) {
        expect(response).not.to.be.empty;
        expect(response.body).to.be.an("object");
      })
      .end(done);
  });
});
```

OUTPUT DEBUG CONSOLE TERMINAL 1: node

```
Sruthis-MacBook-Pro:1. Backend without Kafka sruthiduvvuri$ npm test

> backend@1.0.0 test /Users/sruthiduvvuri/Documents/GitHub/CMPE-273-Twitter-Application-Simulation/1. Backend without Kafka
> mocha test


(node:8658) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser:
true } to MongoClient.connect.
Server running on port 4500
  Login API
inside login of backend
(node:8658) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover an
d Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
(node:8658) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
MongoDB Connected from Mongoose(pooling)
match found
    ✓ Should be successful if credential is valid (1282ms)


  1 passing (1s)
```

2) Register API:

```
describe("Register API", function() {
  it("Creation of Account should be successful", function(done) {
    request(app)
      .post("/api/users/register")
      .set("Accept", "application/json")
      .set("Content-Type", "application/json")
      .send({
        first_name: "Skaar",
        last_name: "Sagar",
        username: "Skaar",
        email: "skaar@gmail.com",
        password: "12345"
      })
      .expect(200)
      .expect(function(response) {
        expect(response).not.to.be.empty;
        expect(response.status).to.equal(200);
      })
      .end(done);
  });
});
```

OUTPUT DEBUG CONSOLE TERMINAL 1: node

```
(node:8700) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser:
true } to MongoClient.connect.
Server running on port 4500
  Register API
Inside register of  backend
data..... {
  first_name: 'Skaar',
  last_name: 'Sagar',
  username: 'Skaar',
  email: 'skaar@gmail.com',
  password: '12345'
}
(node:8700) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover an
d Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
(node:8700) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
MongoDB Connected from Mongoose(pooling)
    ✓ Creation of Account should be successful (1553ms)


  1 passing (2s)
```

3) Update Profile API:

```
43
44  describe("Update Profile API", function() {
45    it("Updation of profile should be successful", function(done) {
46      request(app)
47        .post("/api/users/update_profile")
48        .set("Accept", "application/json")
49        .set("Content-Type", "application/json")
50        .send({
51          username: "Skaar",
52          first_name: "Skaar",
53          last_name: "Sagar",
54          description: "Hey, I am using twitter",
55          city: "San Jose",
56          state: "CA",
57          zipcode: "95110"
58        })
59        .expect(200)
60        .expect(function(response) {
61          expect(response).not.to.be.empty;
62          expect(response.status).to.equal(200);
63        })
64        .end(done);
65    });
66  });
67
```

OUTPUT DEBUG CONSOLE TERMINAL 1: node

```
(node:8838) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser:
true } to MongoClient.connect.
Server running on port 4500
  Update Profile API
inside update profile of backend
(node:8838) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover an
d Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
(node:8838) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
MongoDB Connected from Mongoose(pooling)
    ✓ Updation of profile should be successful (1227ms)


  1 passing (1s)
```

4) Create Tweet API:

```
69  describe("Create Tweet", function() {
70      it("A tweet has to be created sucessful", function(done) {
71        request(app)
72          .post("/api/tweets/create_tweet")
73          .set("Accept", "application/json")
74          .set("Content-Type", "application/json")
75          .send({
76              user: "5de7151504586321fc5bdc9e",
77              tweet_content: "Hey ! Whatsup.",
78              username: "Skaar",
79              firstname: "Skaar",
80              lastname:"Sagar",
81          })
82          .expect(200)
83          .expect(function(response) {
84            expect(response).not.to.be.empty;
85            expect(response.status).to.equal(200);
86          })
87          .end(done);
88      });
89  });
```

OUTPUT DEBUG CONSOLE TERMINAL 1: node

```
^C
Sruthis-MacBook-Pro:1. Backend without Kafka sruthiduvvuri$ npm test

> backend@1.0.0 test /Users/sruthiduvvuri/Documents/GitHub/CMPE-273-Twitter-Application-Simulation/1. Backend without Kafka
> mocha test


(node:8889) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser:
true } to MongoClient.connect.
Server running on port 4500
  Create Tweet
(node:8889) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover an
d Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
(node:8889) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
MongoDB Connected from Mongoose(pooling)
    ✓ A tweet has to be created sucessful (1281ms)


  1 passing (1s)
```

5) Search Hashtags API:

```
90  describe("Search Hashtags", function() {
91    it("To search for hashtags", function(done) {
92      request(app)
93        .post("/api/tweets/search_topic")
94        .set("Accept", "application/json")
95        .set("Content-Type", "application/json")
96        .send({
97          searchText: "Sports"
98        })
99        .expect(200)
100       .expect(function(response) {
101         expect(response).not.to.be.empty;
102         expect(response.status).to.equal(200);
103       })
104       .end(done);
105   });
106 });
107
```

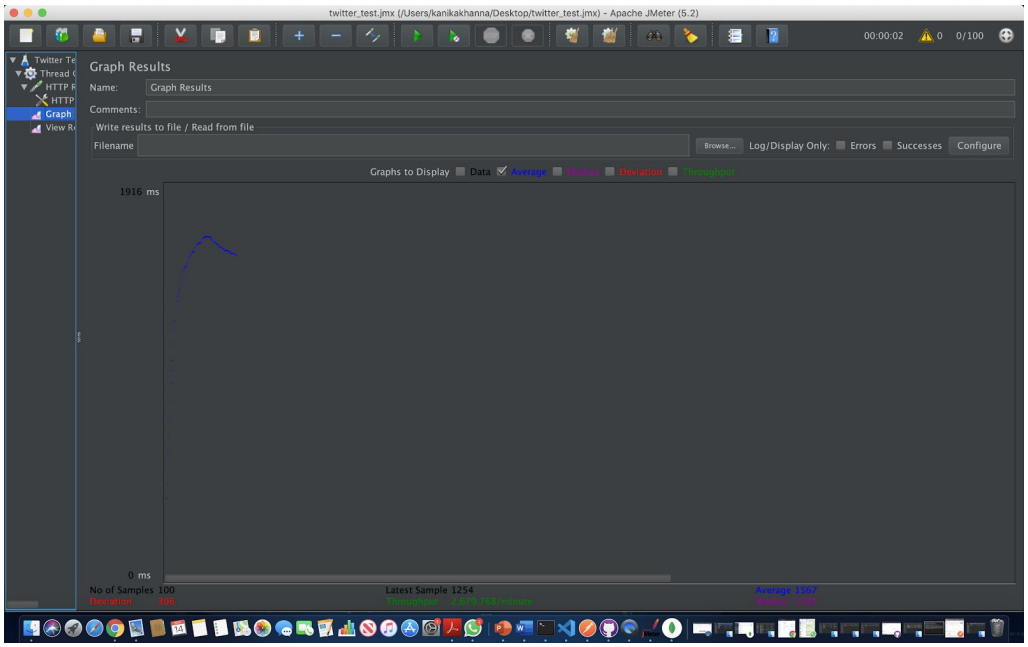
```
OUTPUT    DEBUG CONSOLE    TERMINAL                                   1: node

MongoDB Connected from Mongoose(pooling)
[
  {
    retweeted_status: { hashtags: [], imageList: [] },
    replies_count: 0,
    likes_count: 1,
    hashtags: [ 'Sports' ],
    imageList: [],
    retweets_count: 0,
    view_count: 0,
    _id: 5de56f0a68edef10deaa9625,
    user: 5dce038639b452aa3fe6fa9b,
    tweet_content: 'World Cup 2020',
    username: 'Sruthi',
    firstname: 'Sruthi',
    lastname: 'Duvvuri',
    avatar: '//www.gravatar.com/avatar/58f05caa3c763a6770030da9bbb2e303?s=200&r=pg&d=mm',
    replies: [],
    likes: [ [Object] ],
    tweeted_date: 2019-12-02T20:07:38.006Z,
    __v: 1
  }
]
    ✓ To search for hashtags (1314ms)
```

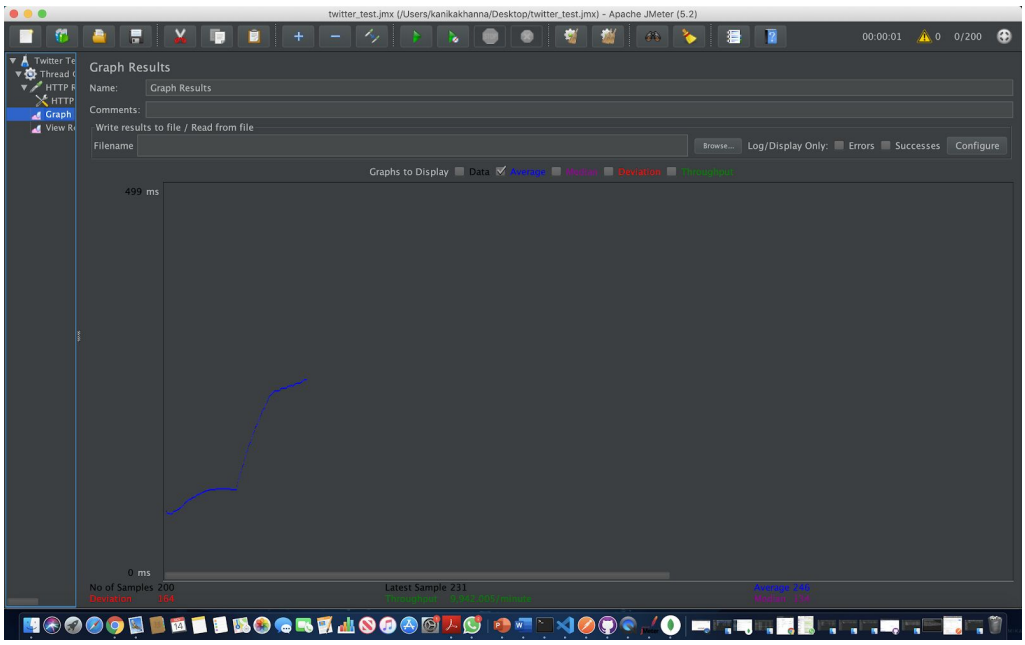
JMETER Testing:

Performance Graphs With Base:

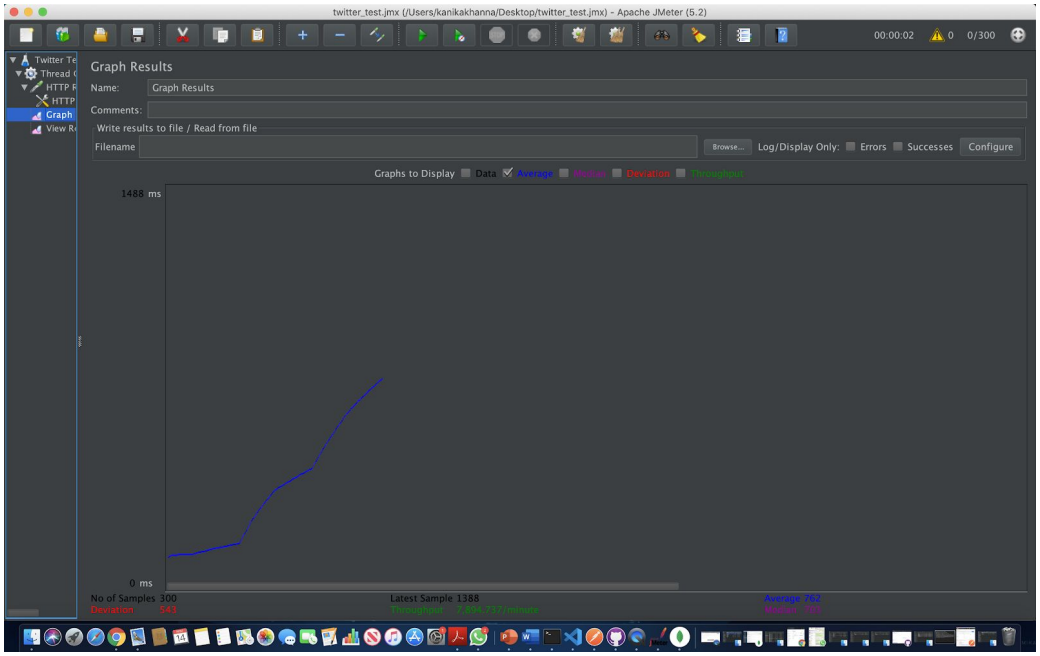
- With 100 concurrent users:



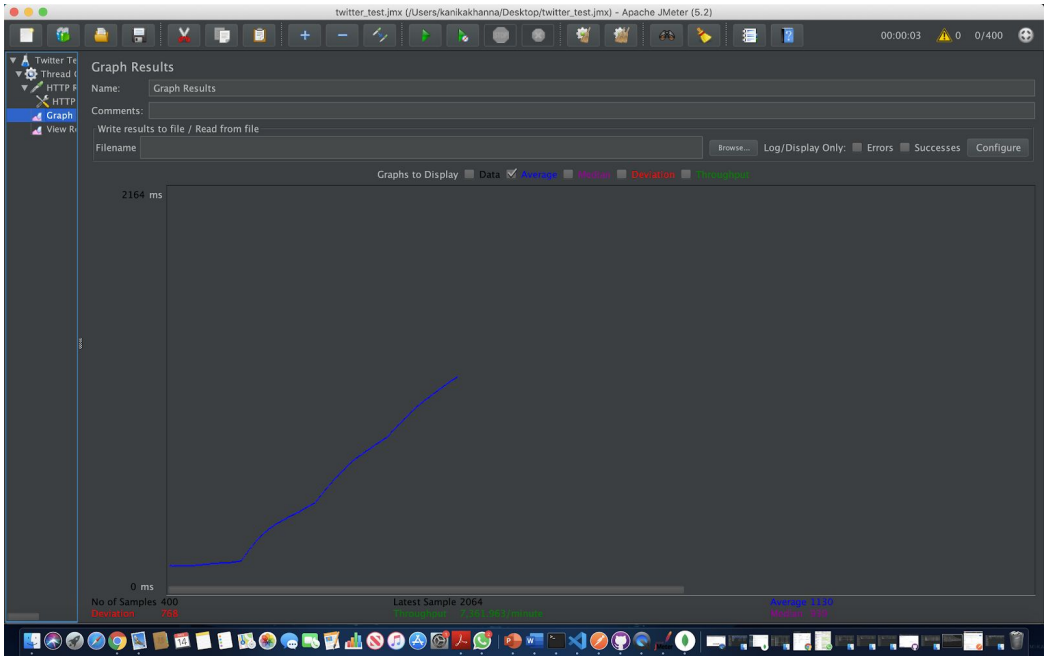
- With 200 concurrent users:



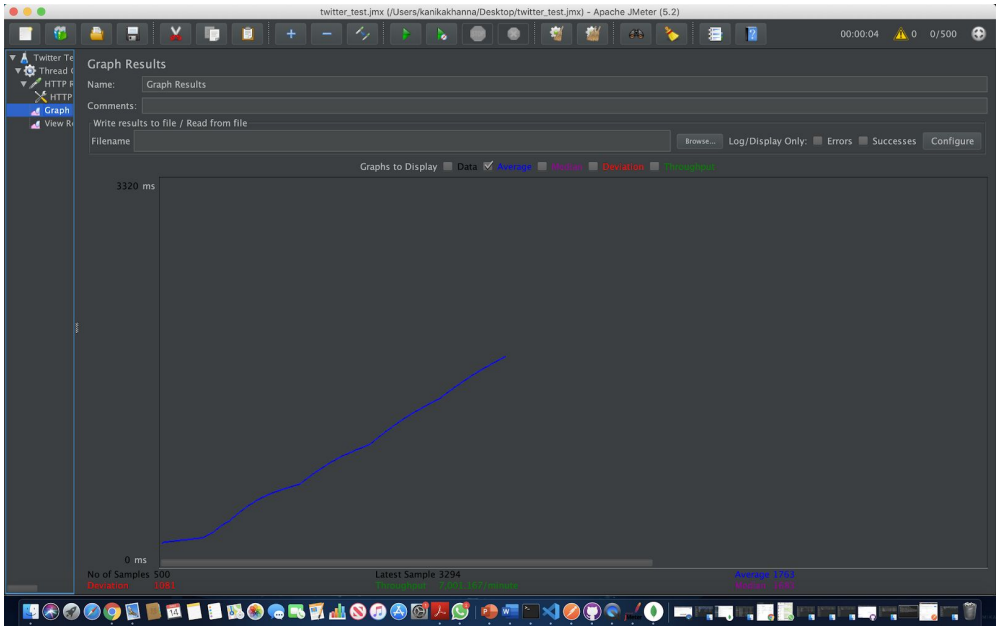
- With 300 concurrent users:



- With 400 concurrent users:

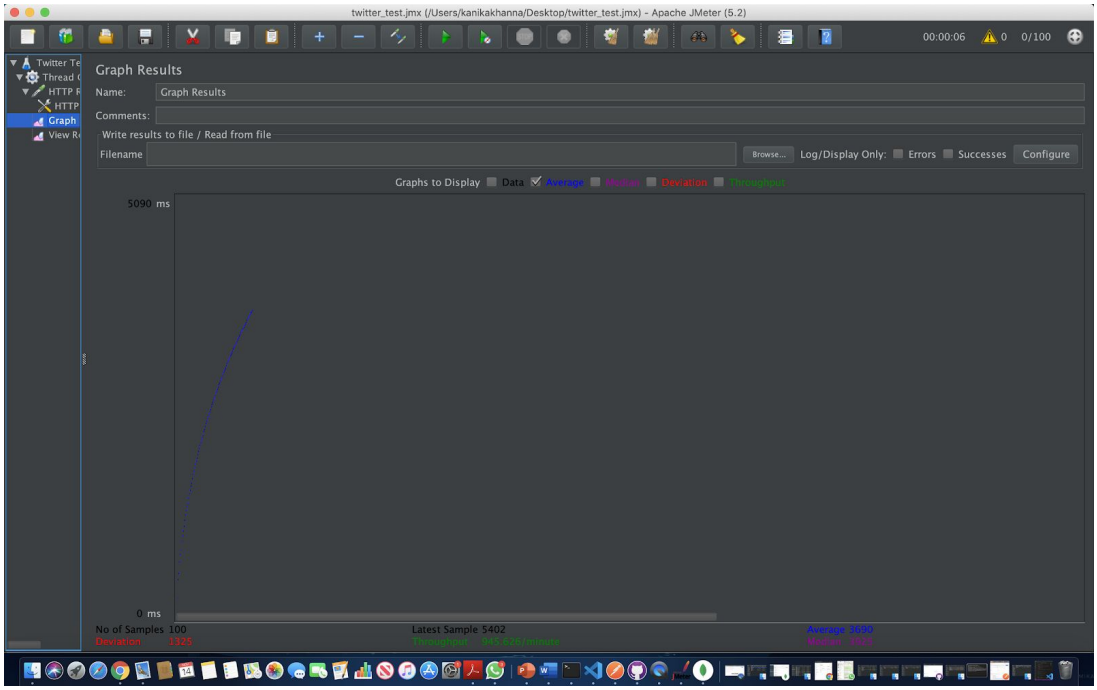


- With 500 concurrent users:

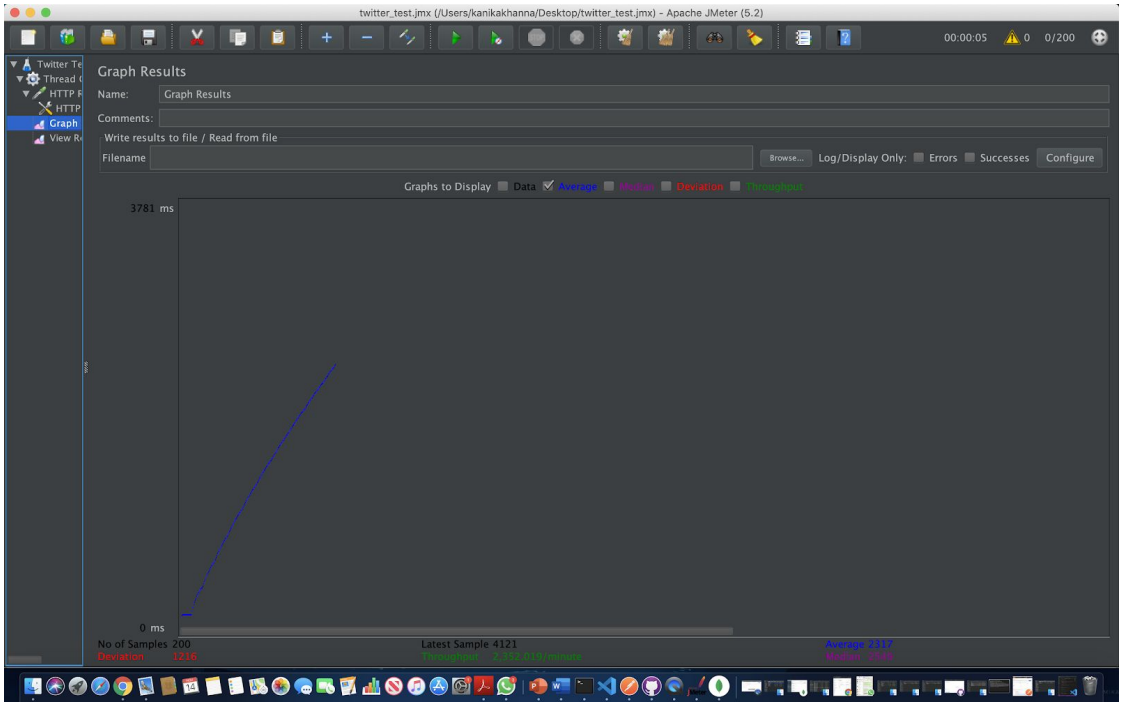


1. **Performance Graphs With Base and Connection Pooling:**

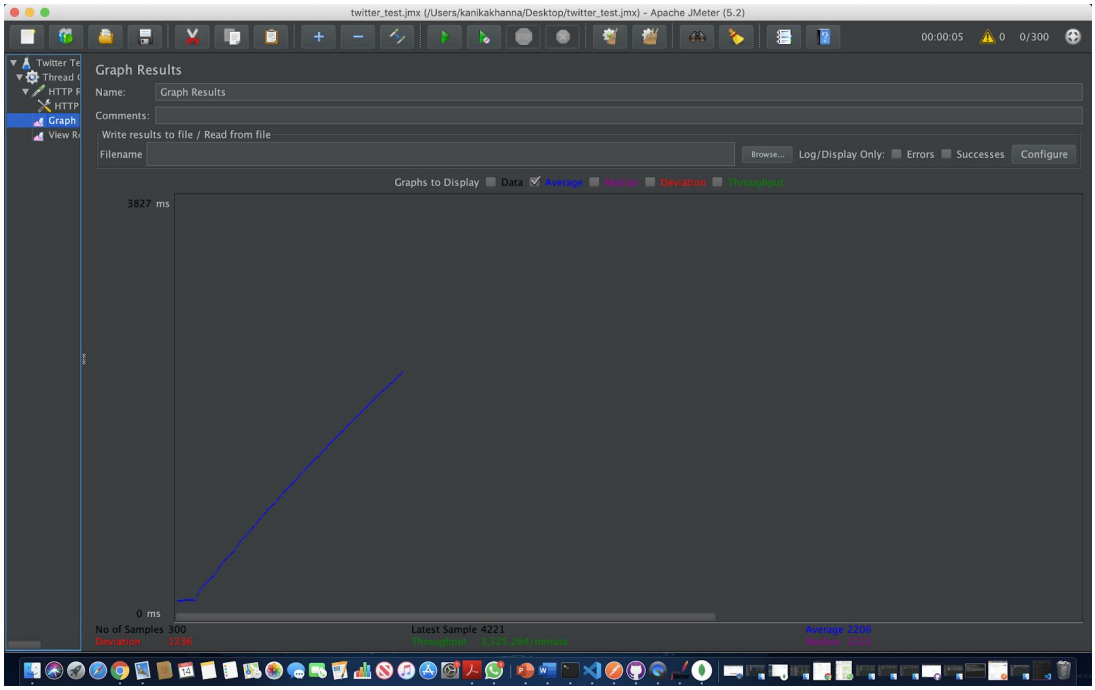
- With 100 concurrent users:



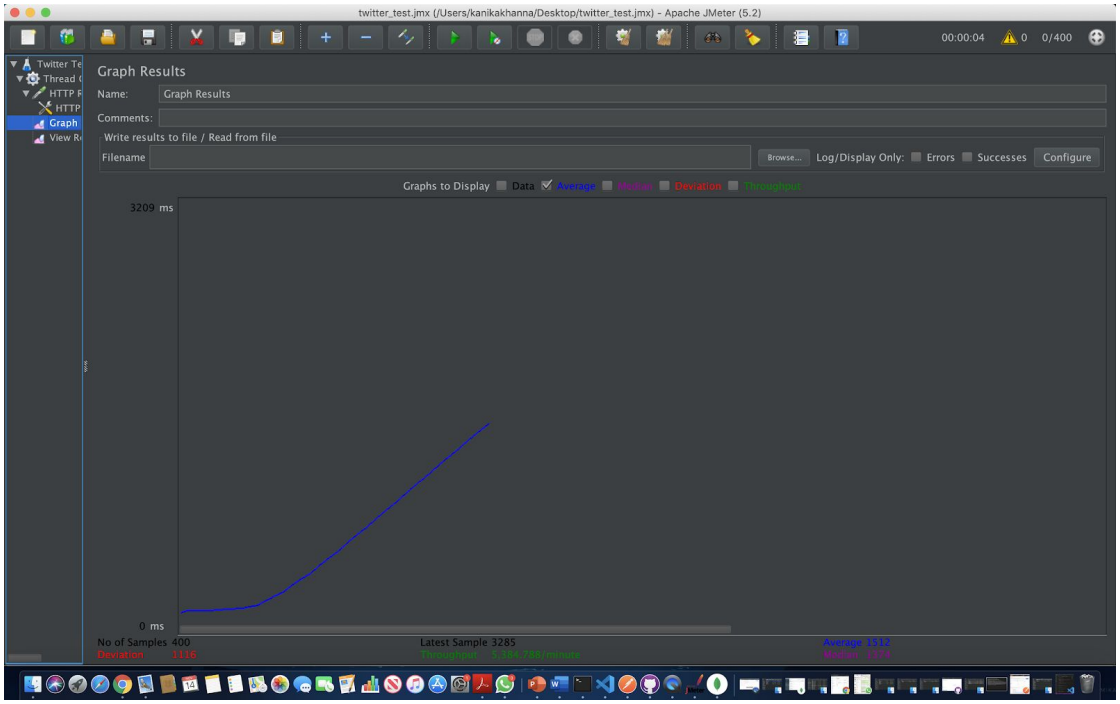
- With 200 concurrent users:



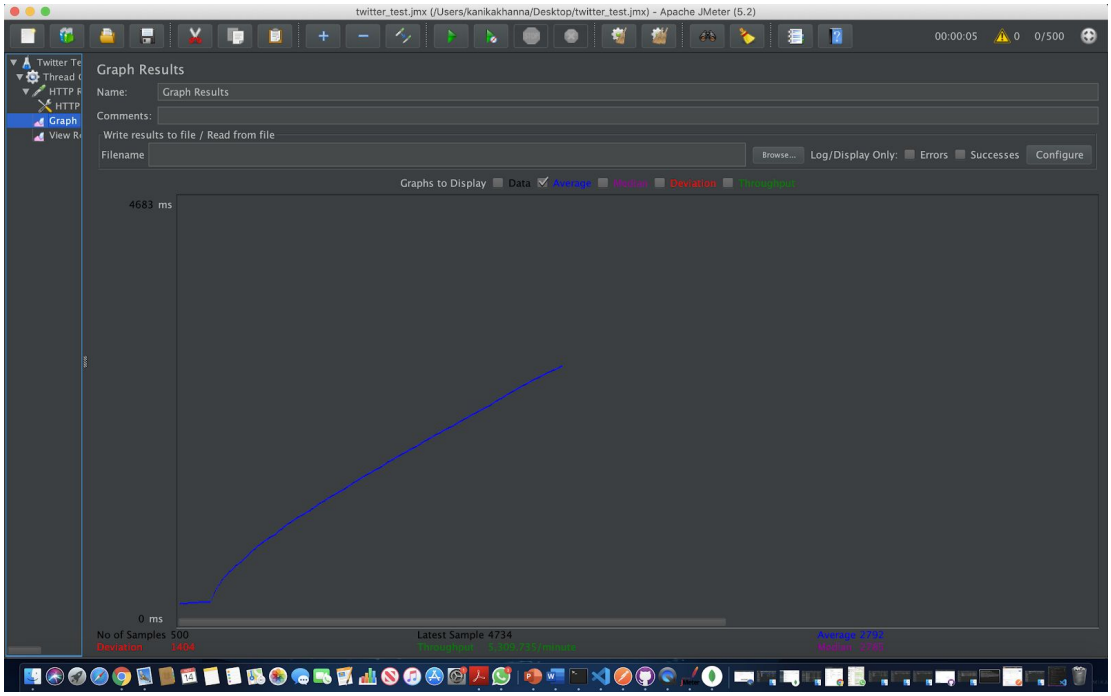
- With 300 concurrent users:



- With 400 concurrent users:

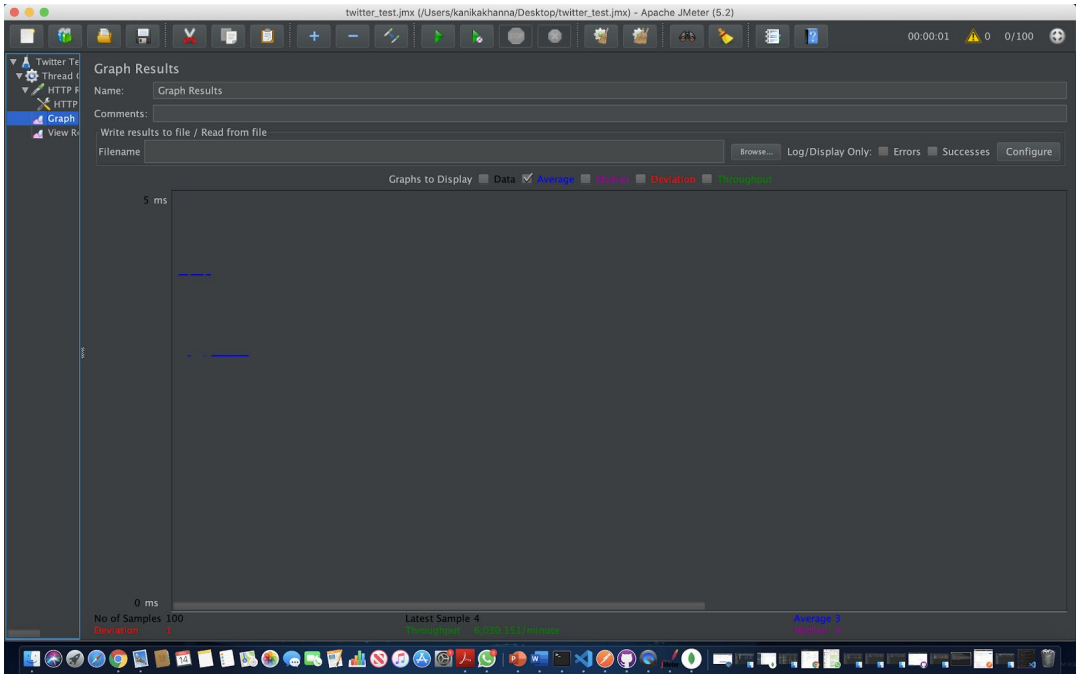


- With 500 concurrent users:

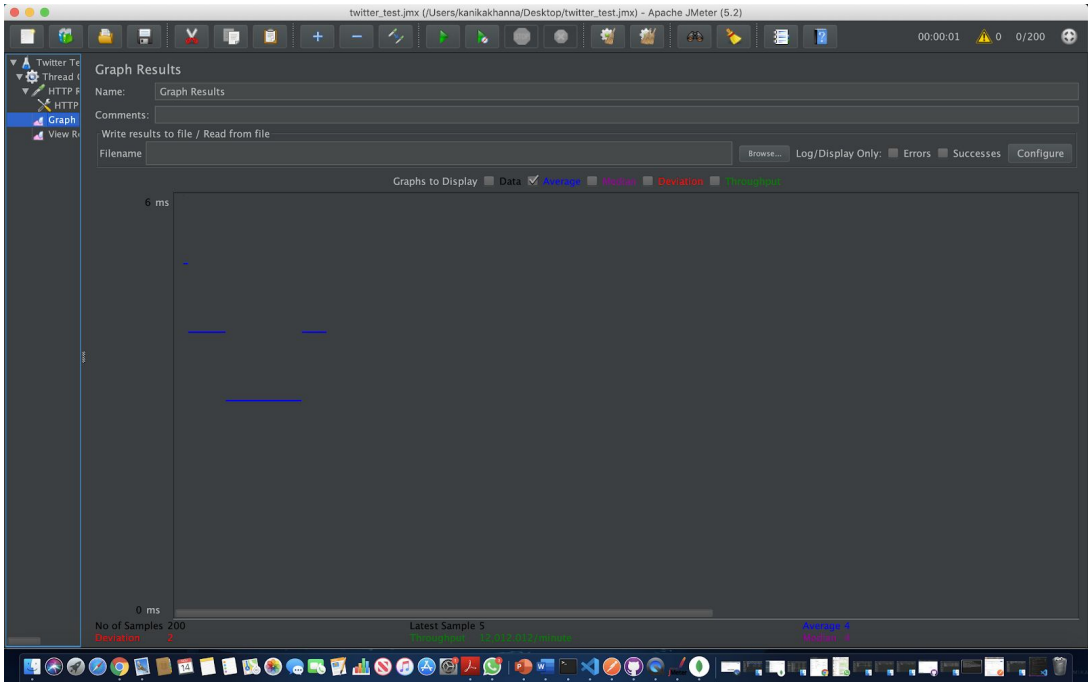


2. Performance Graph with Base, Connection Pooling and SQL Caching implemented:

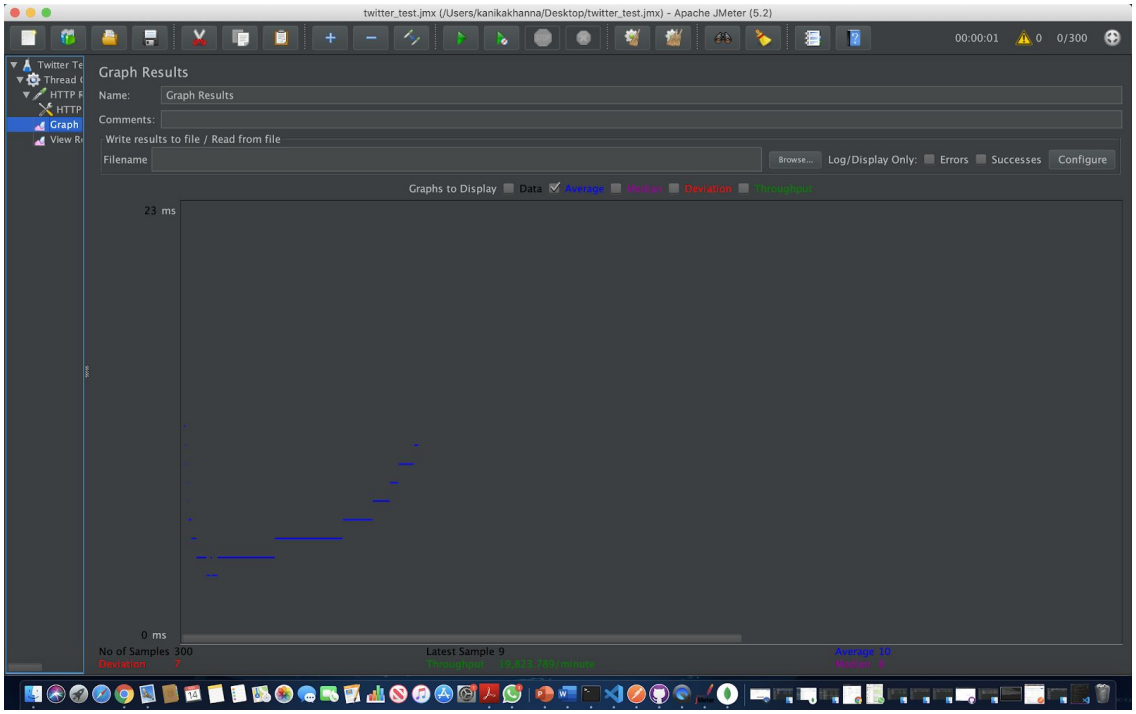
- With 100 concurrent Users:



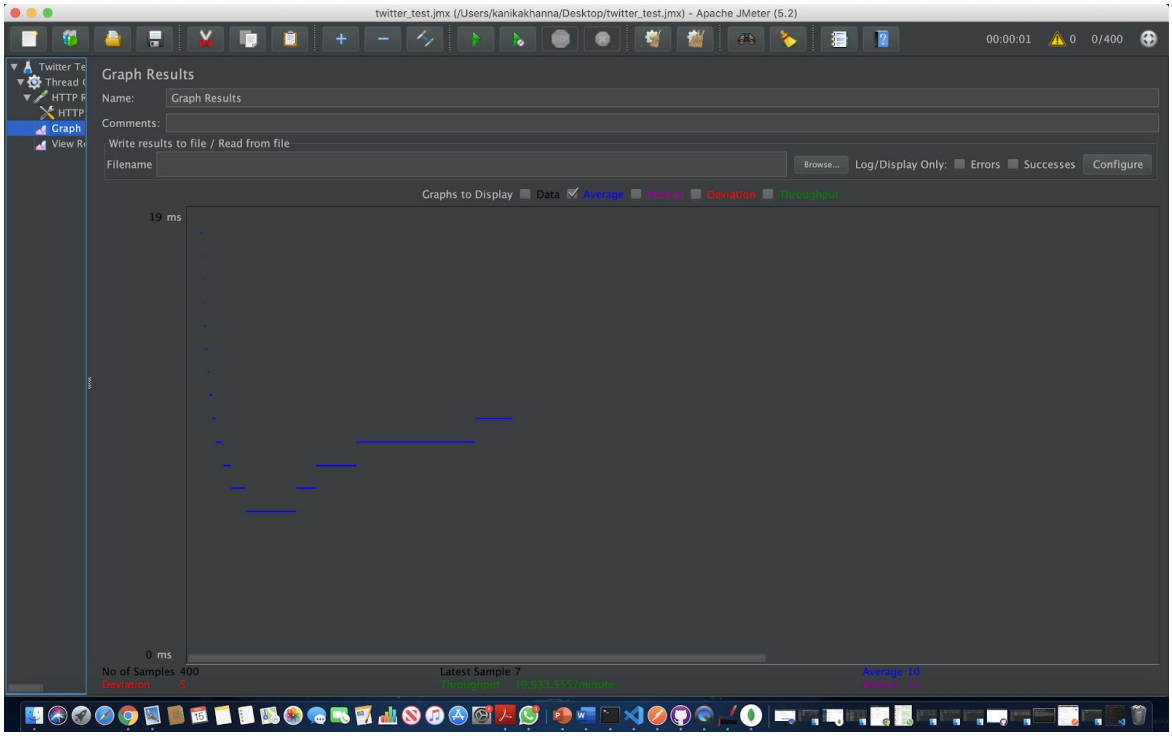
- With 200 concurrent Users:



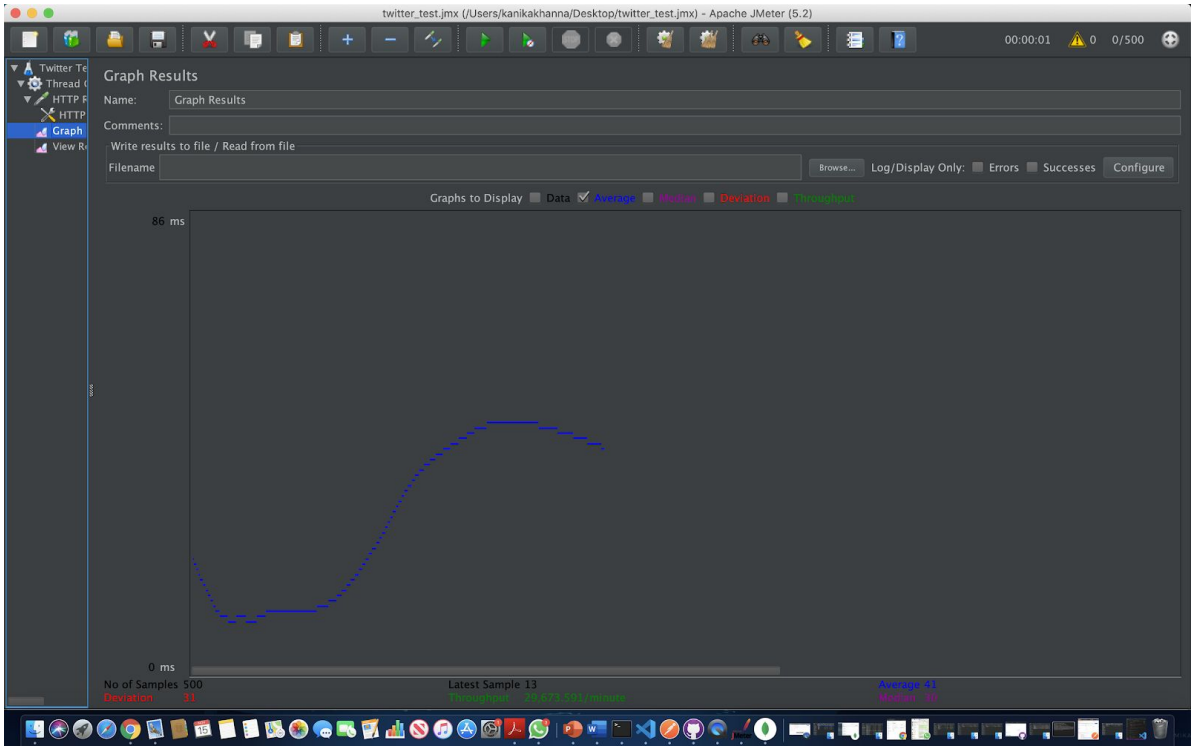
- With 300 concurrent Users:



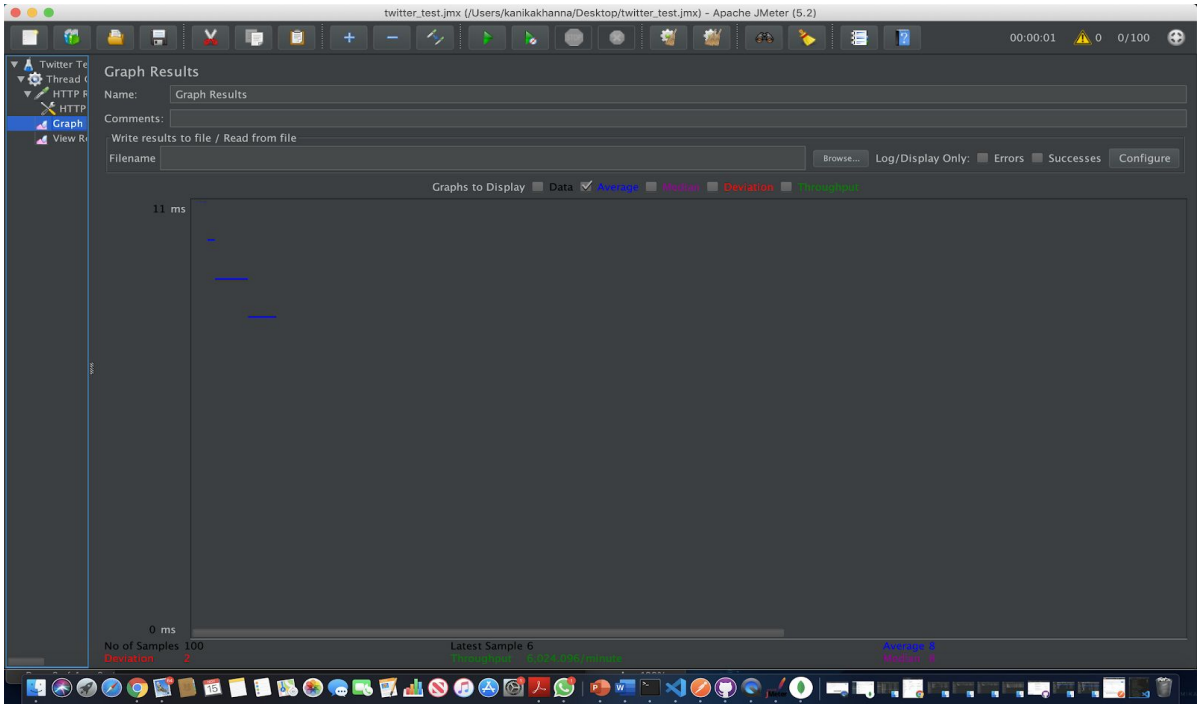
- With 400 concurrent Users:



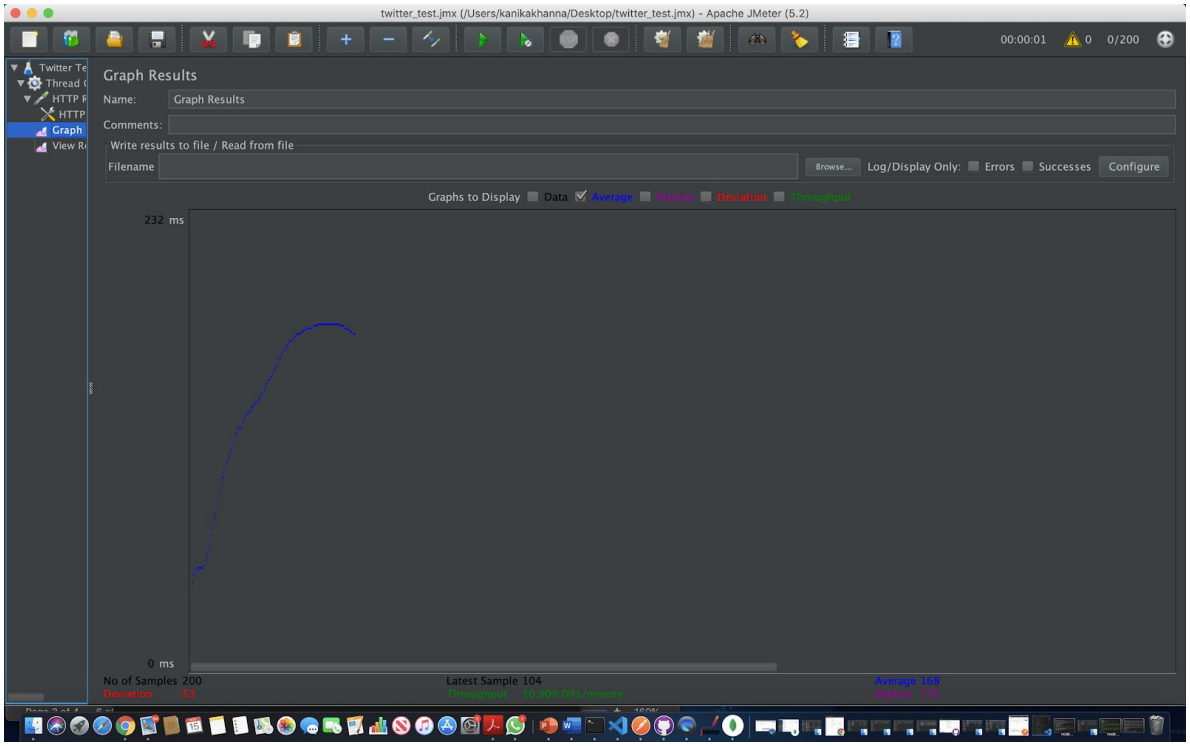
- With 500 concurrent Users:



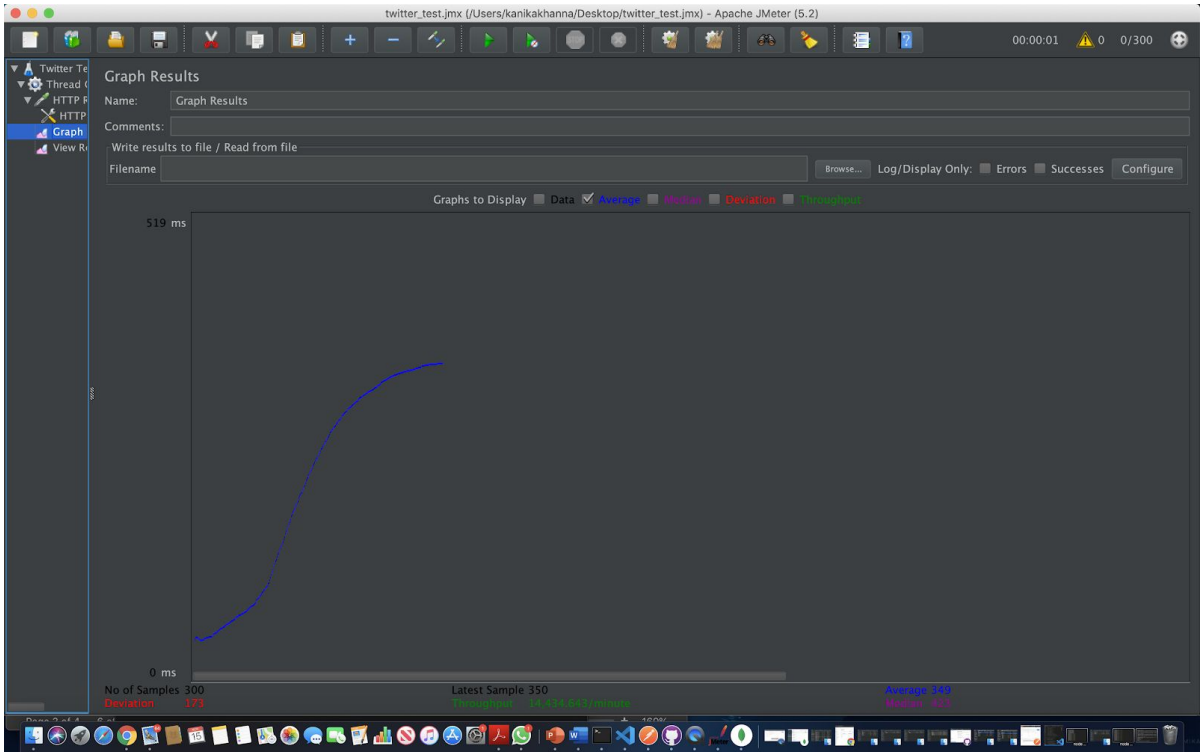
3. **Performance Graph with Base, Connection pooling, SQL caching and Kafka Implementation:**
   - With 100 Concurrent Users:



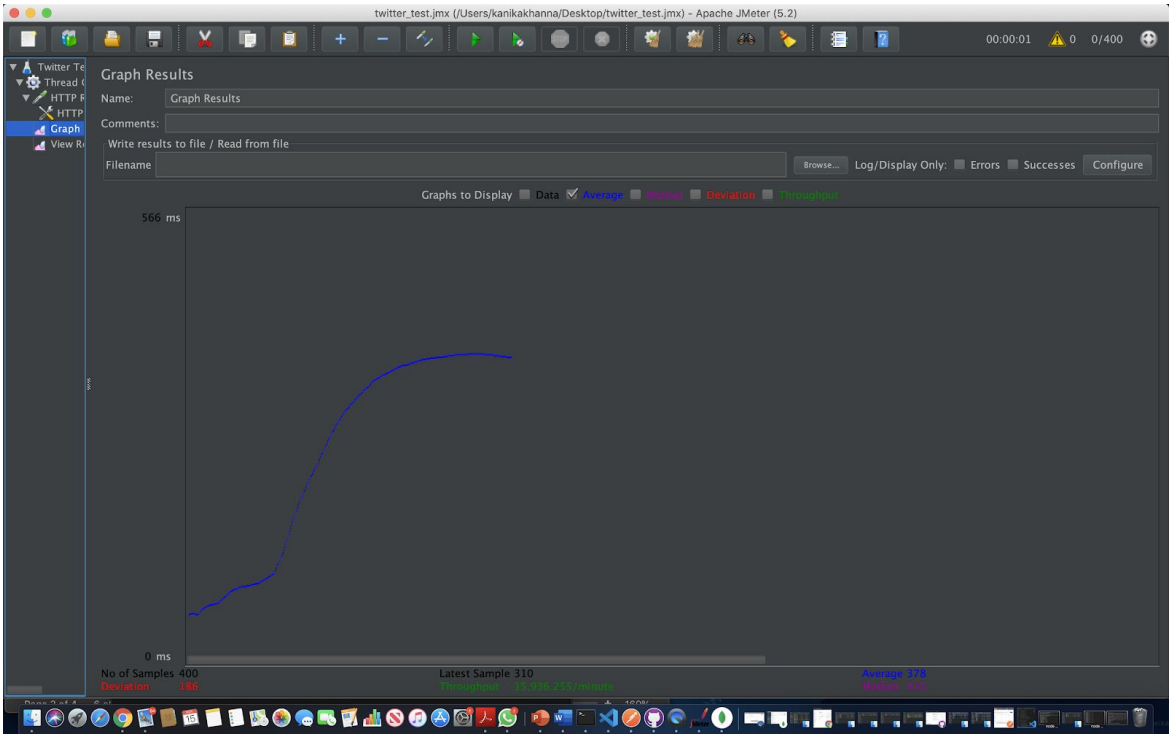
- With 200 Concurrent Users:



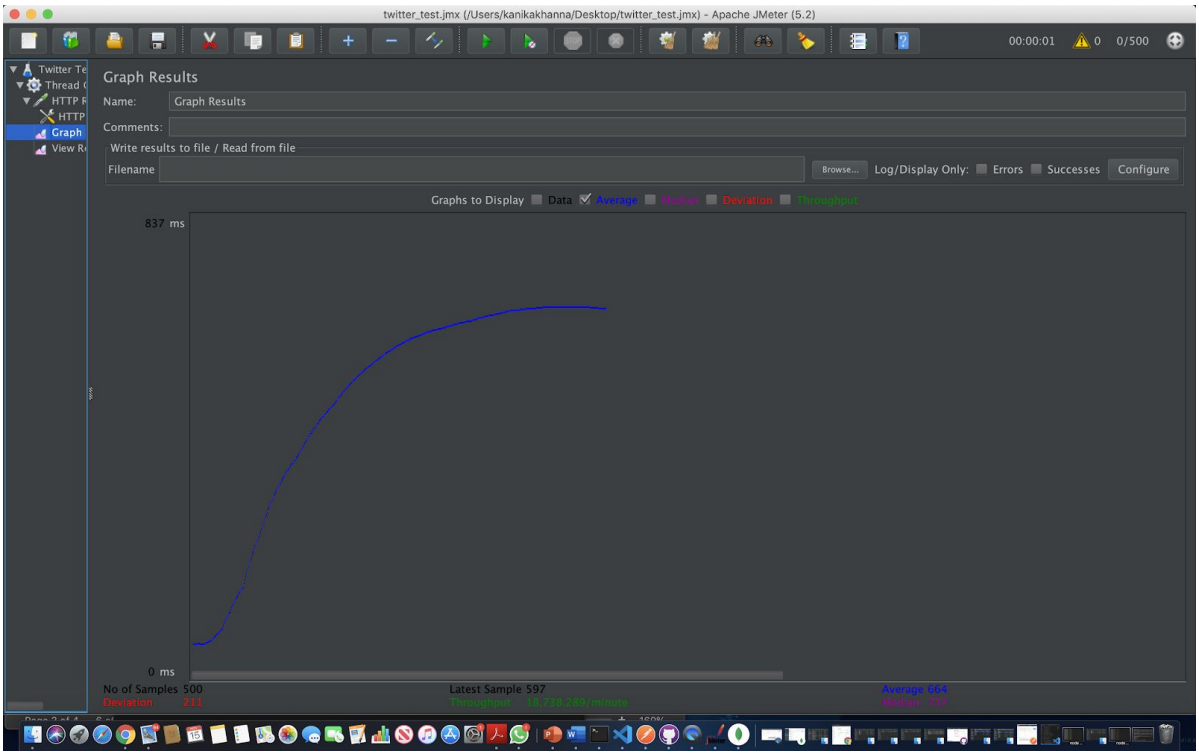
- With 300 Concurrent Users:



- With 400 Concurrent Users:

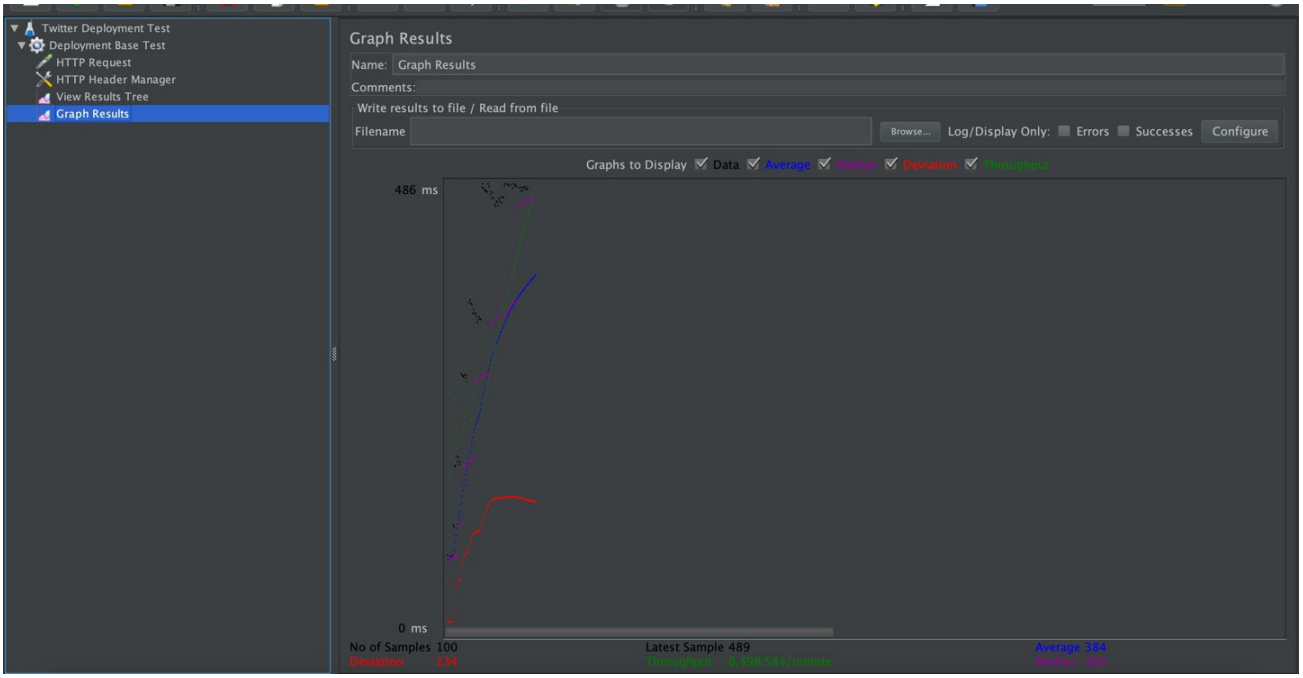


- With 500 Concurrent Users:



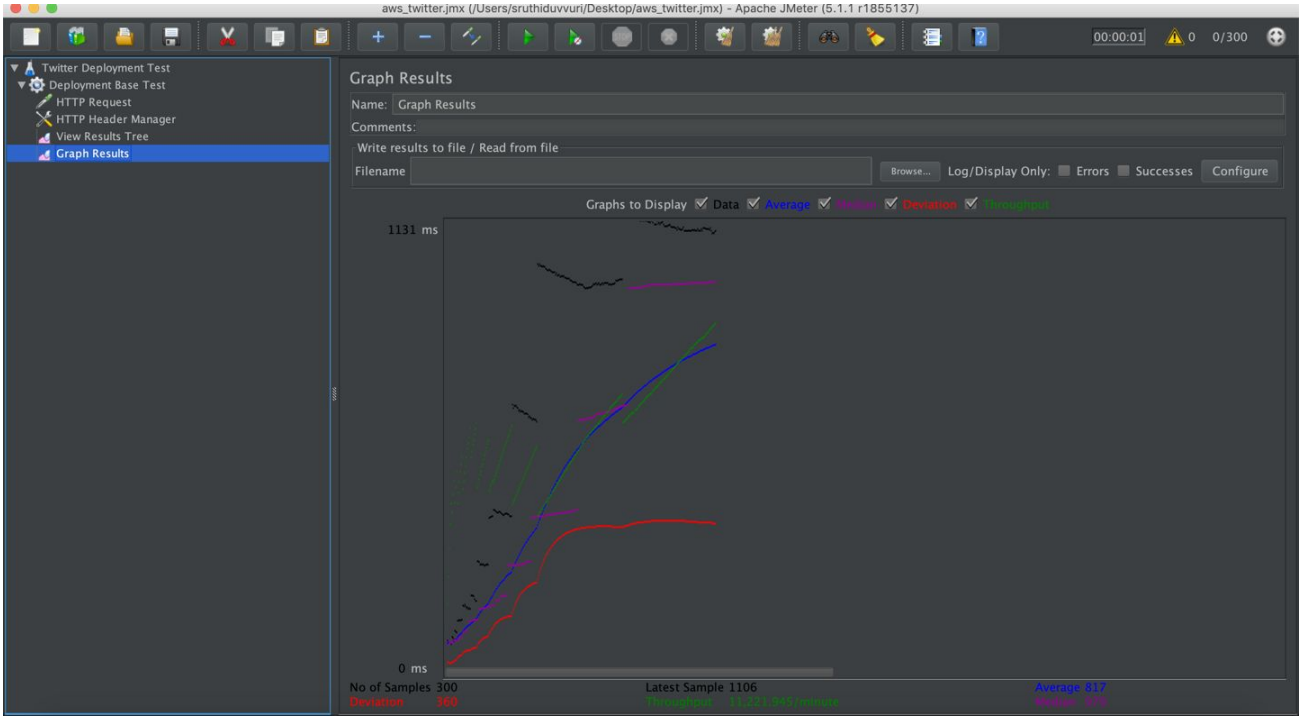
4. After Deployment Case 1: With 1-UI instance, 1-DB instance and 1-Backend

- With 100 concurrent users:

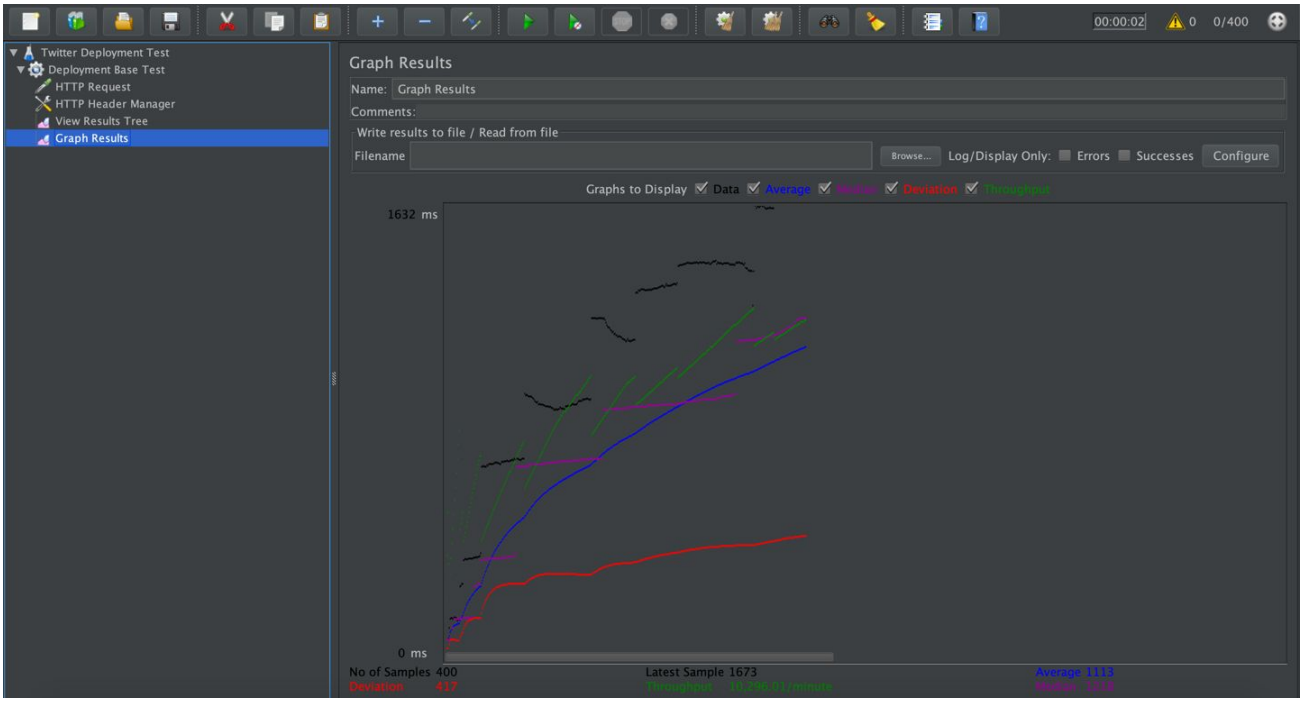


- With 200 concurrent users:

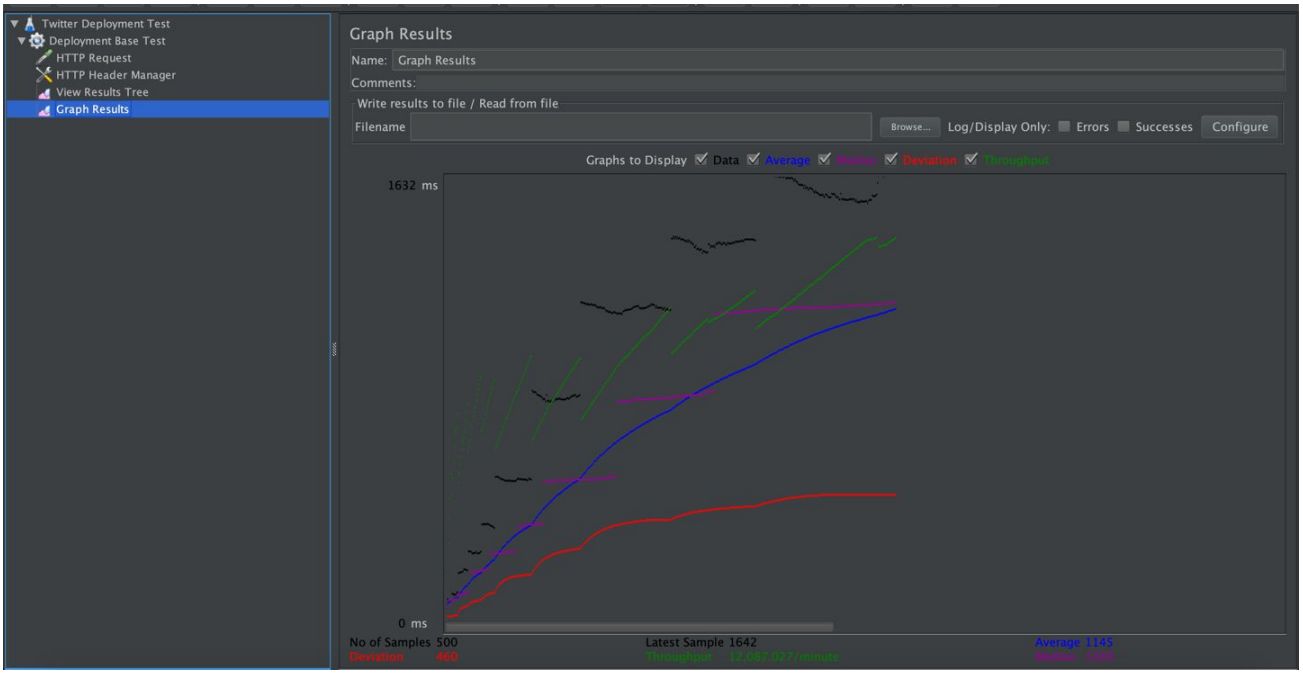
- With 300 concurrent users:



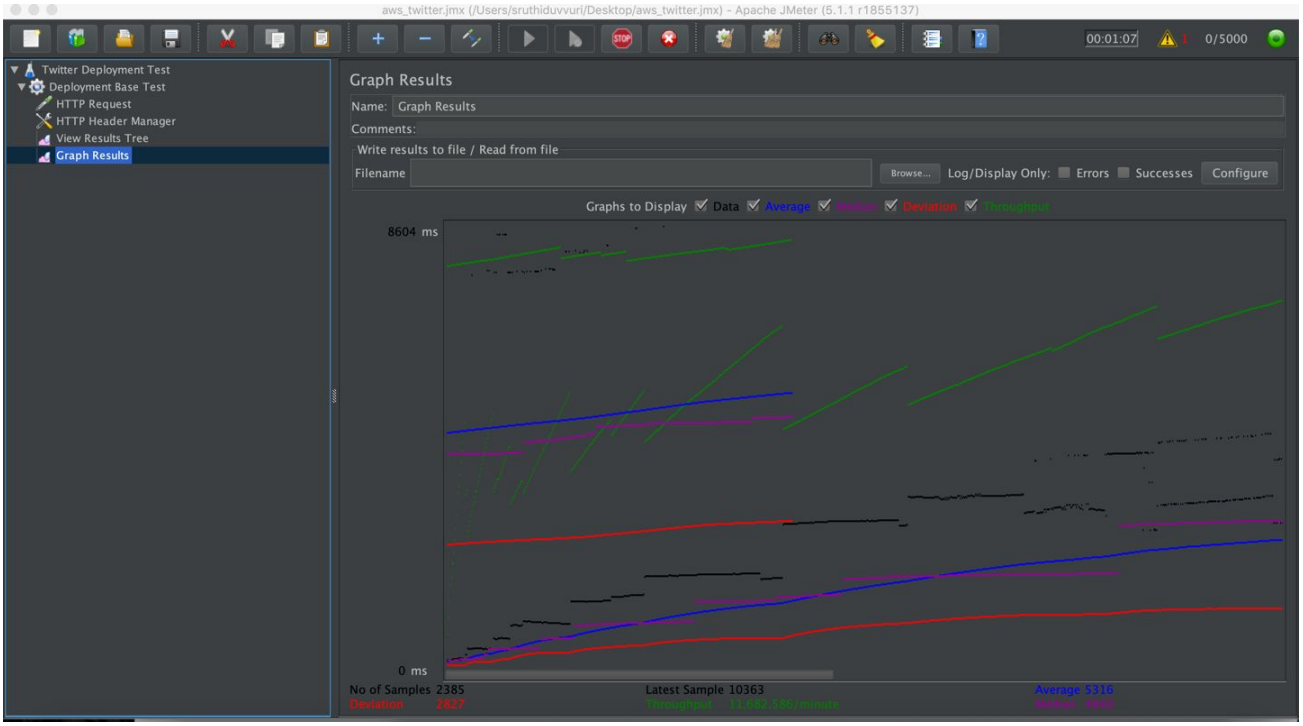
- With 400 concurrent users:



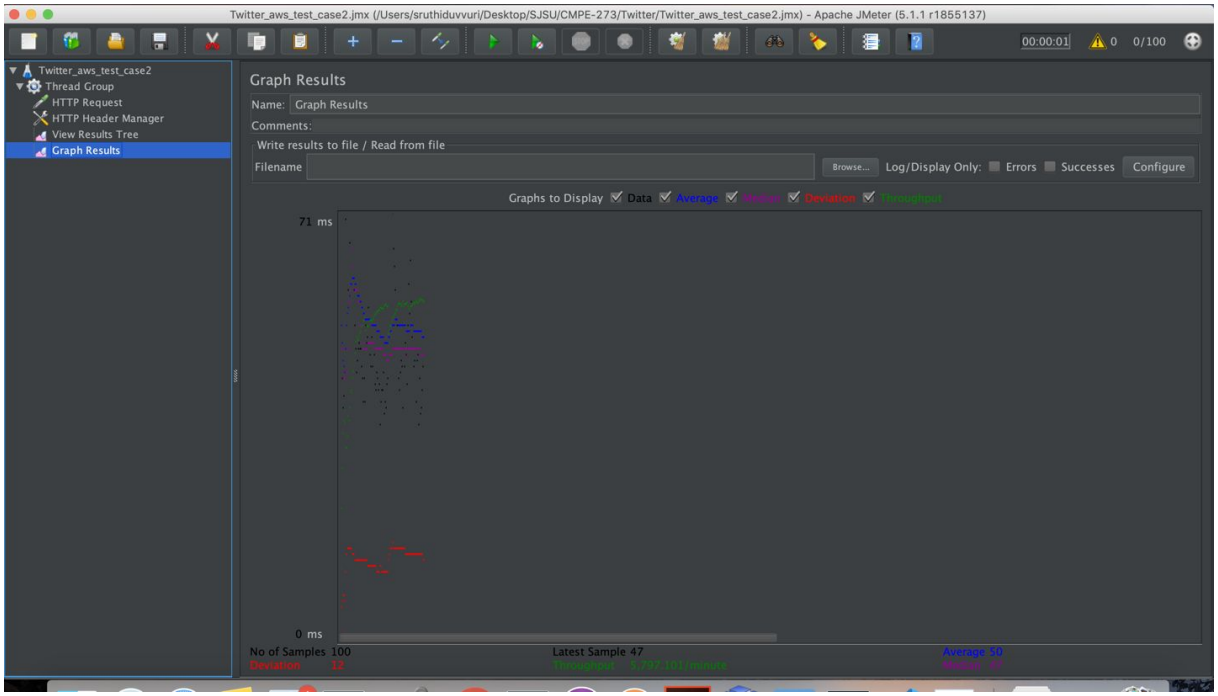
- With 500 concurrent users:



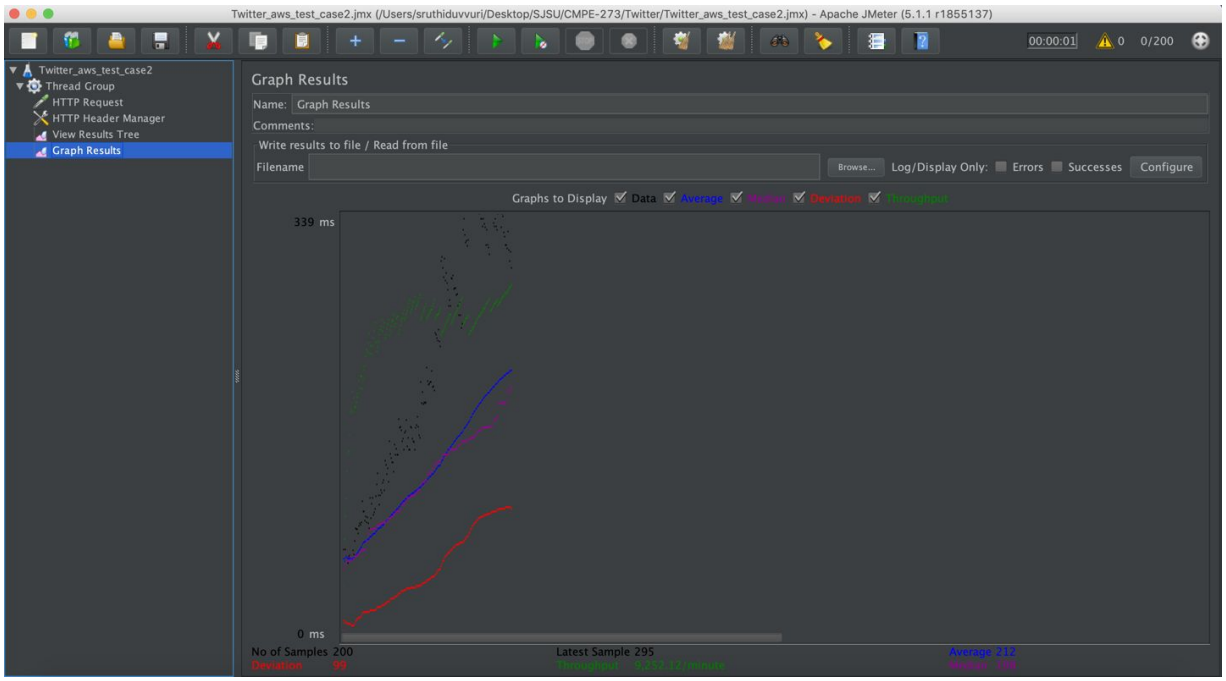
- With 2385 Concurrent users:



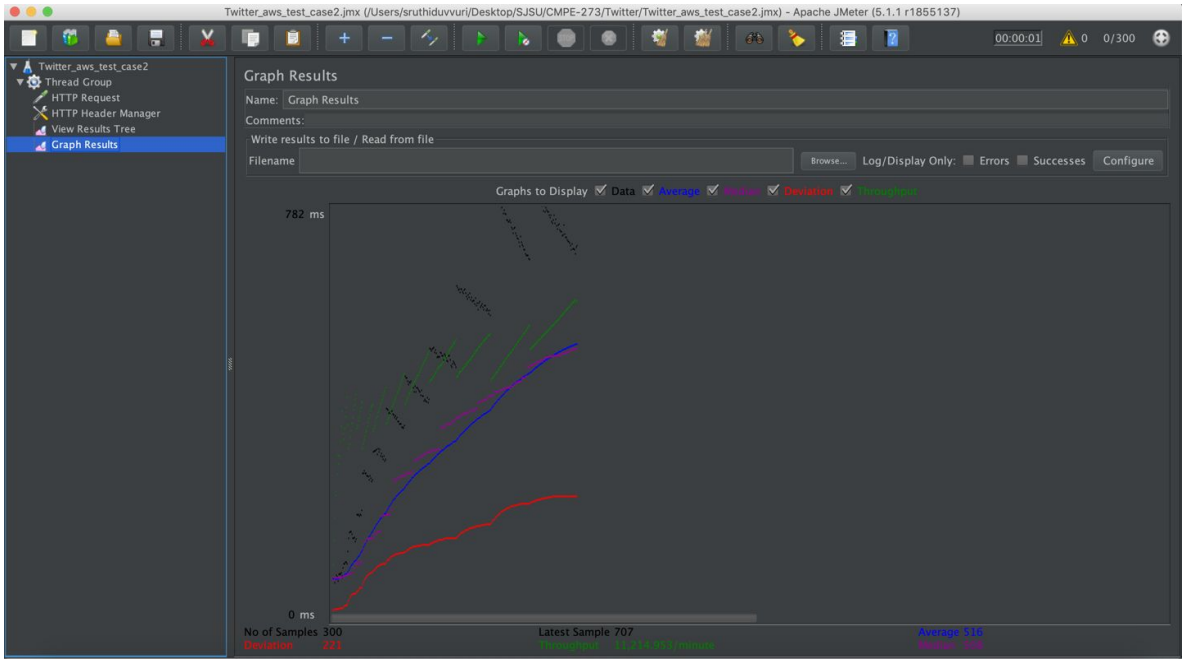
4. After Deployment Case 2: With 1-UI instance, 1-DB instance and 3-Backend
    - With 100 Concurrent users:



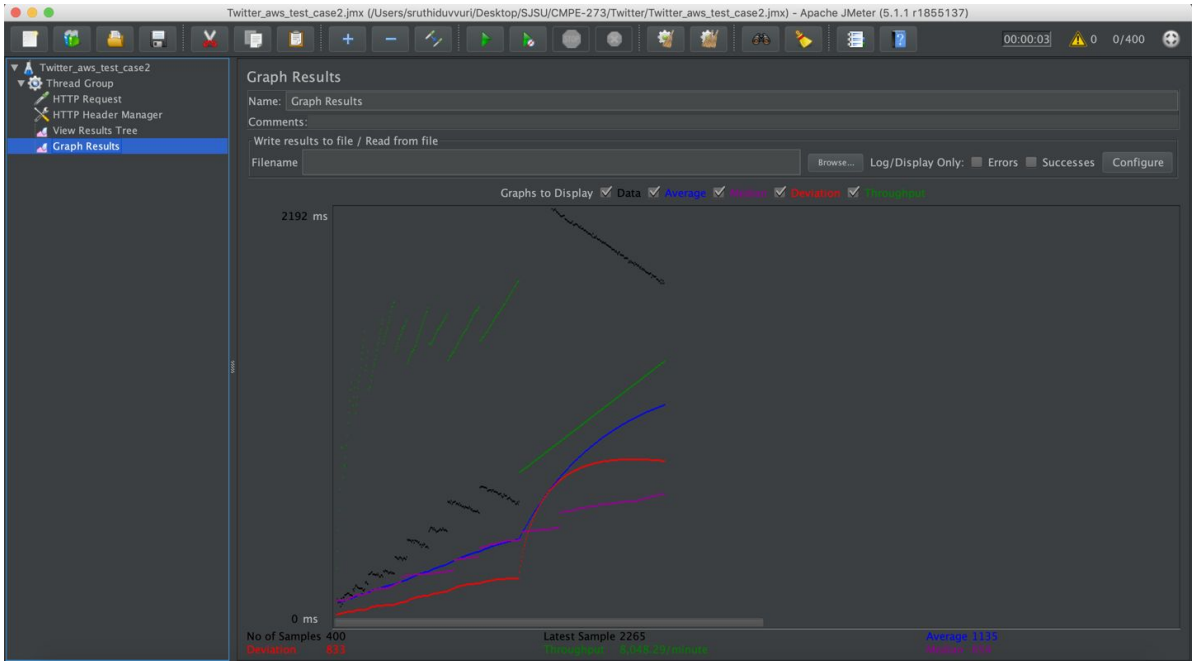
    - With 200 Concurrent users:



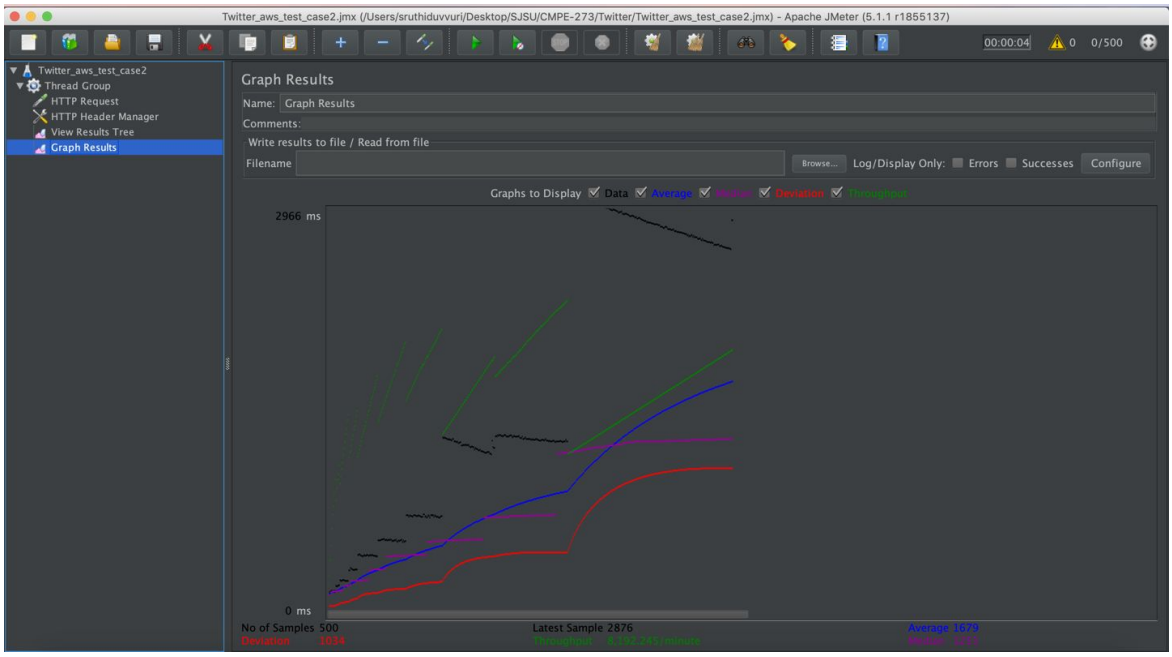
- With 300 Concurrent users:



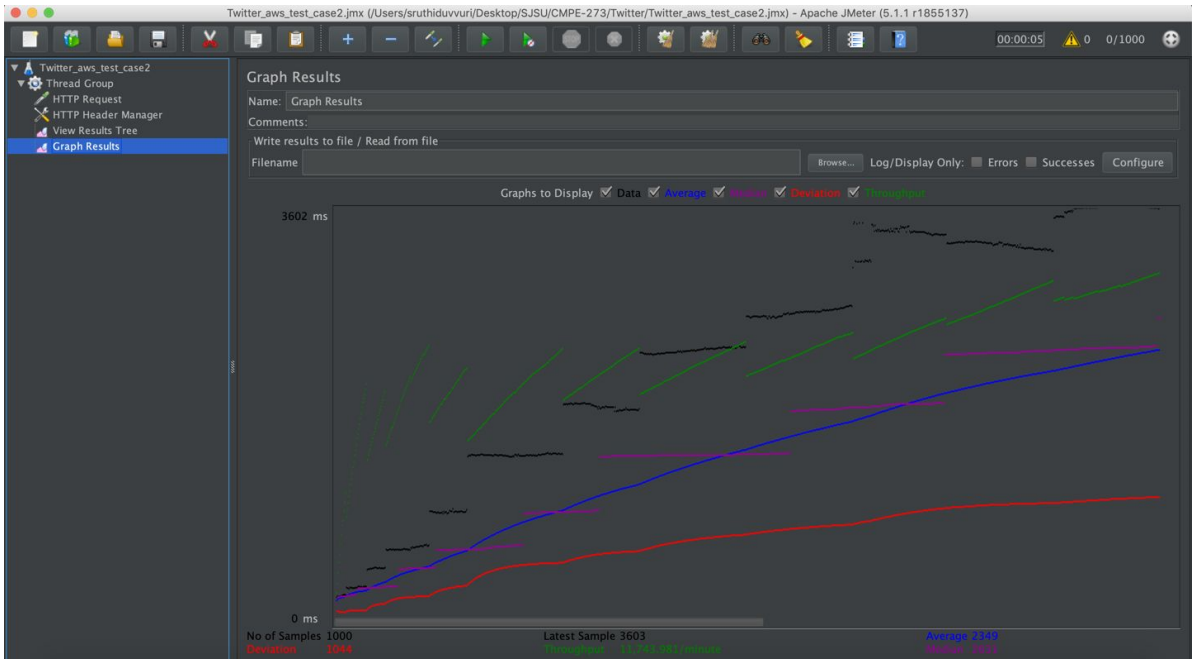
- With 400 Concurrent users:



- With 500 Concurrent users:

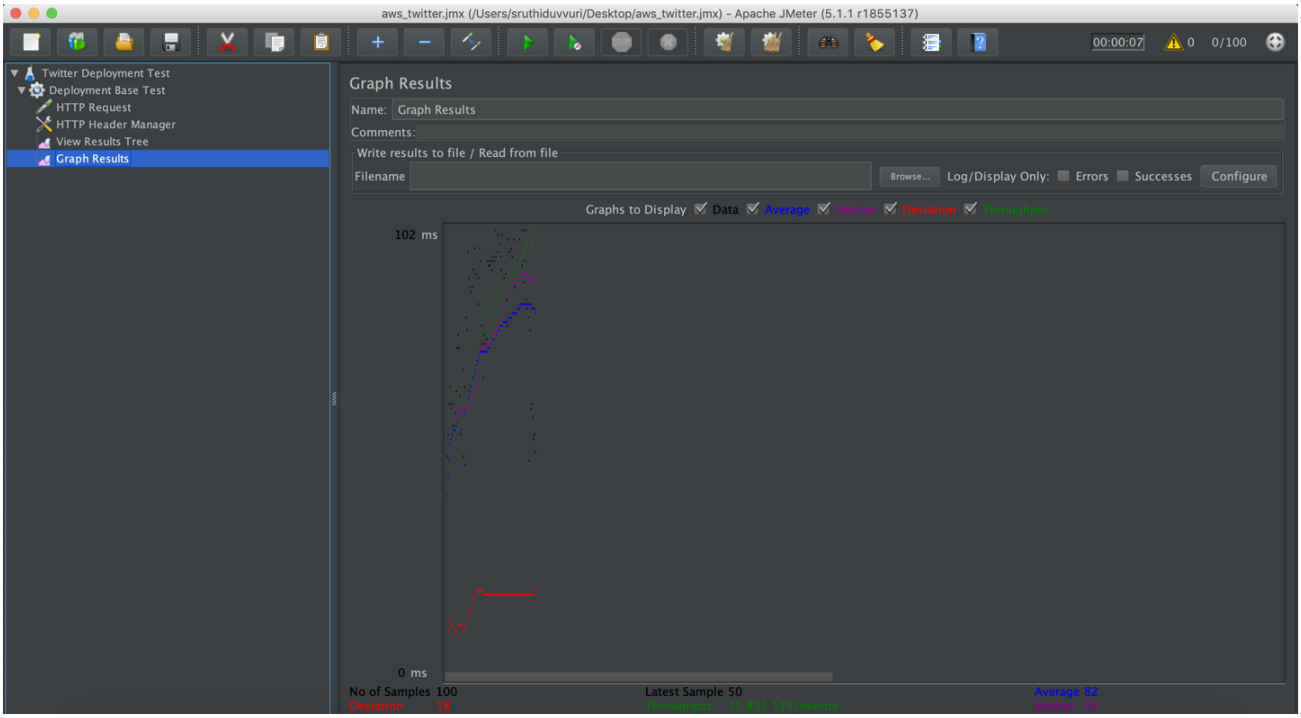


- With 1000 Concurrent users

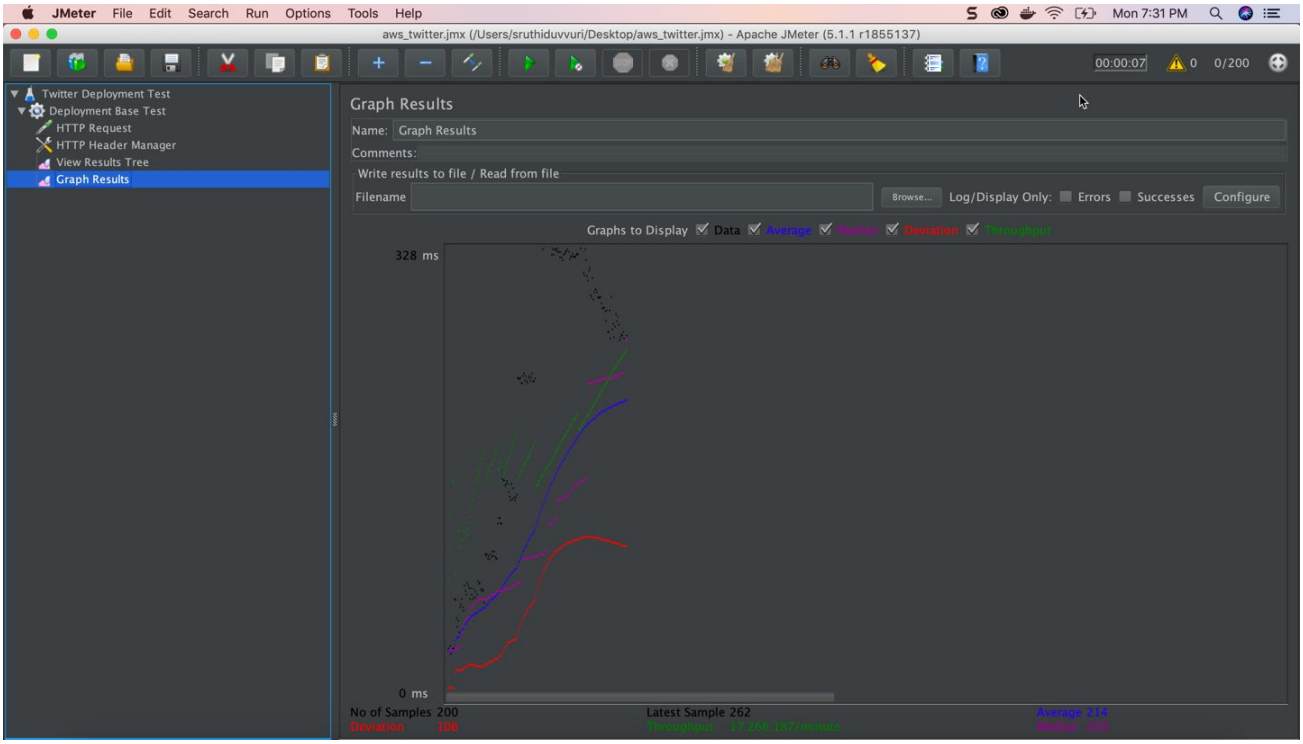


6. After Deployment Case 3: With 1-UI instance, 2-DB instance and 4-Backend:

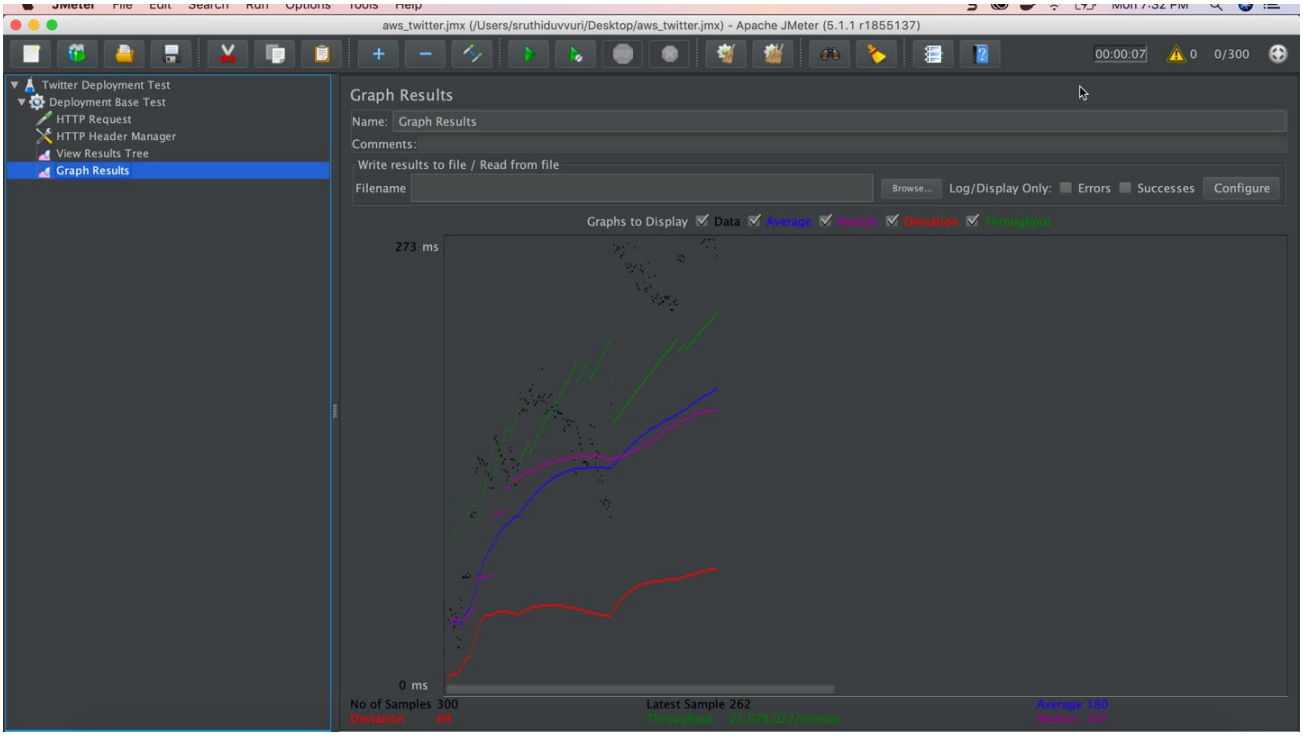
- With 100 concurrent users:



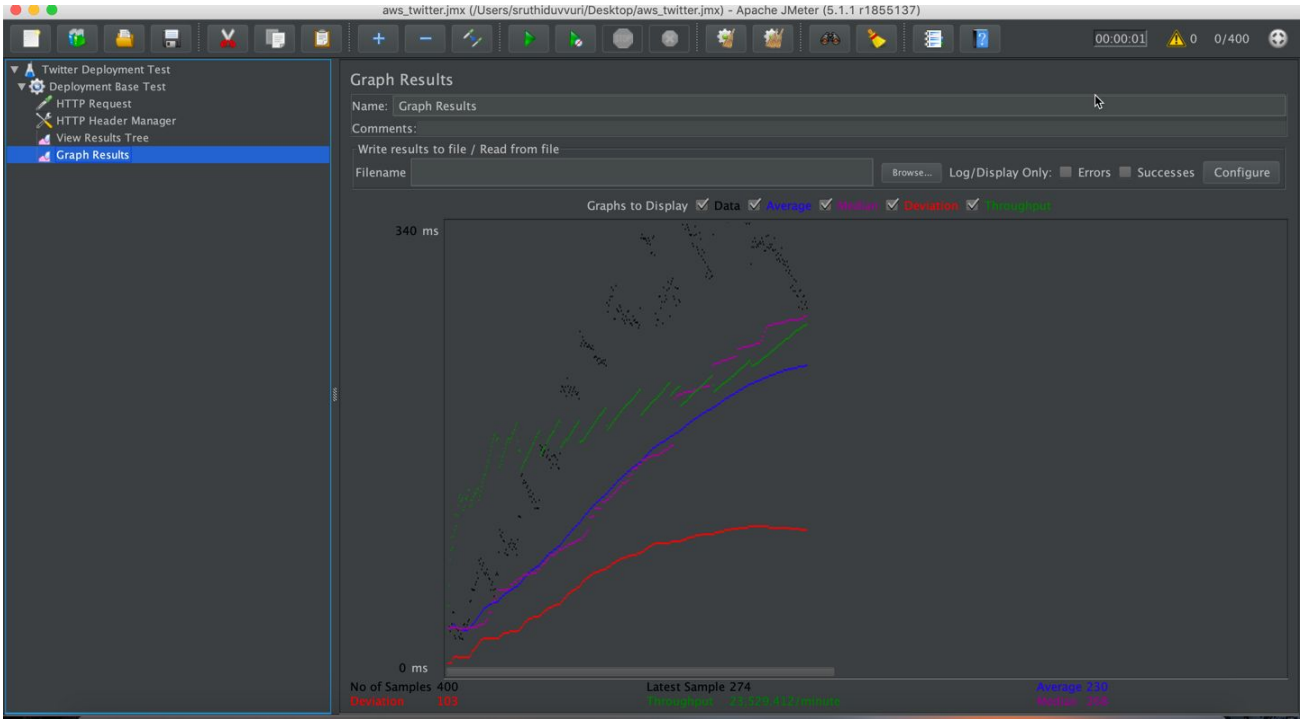
- With 200 concurrent users:



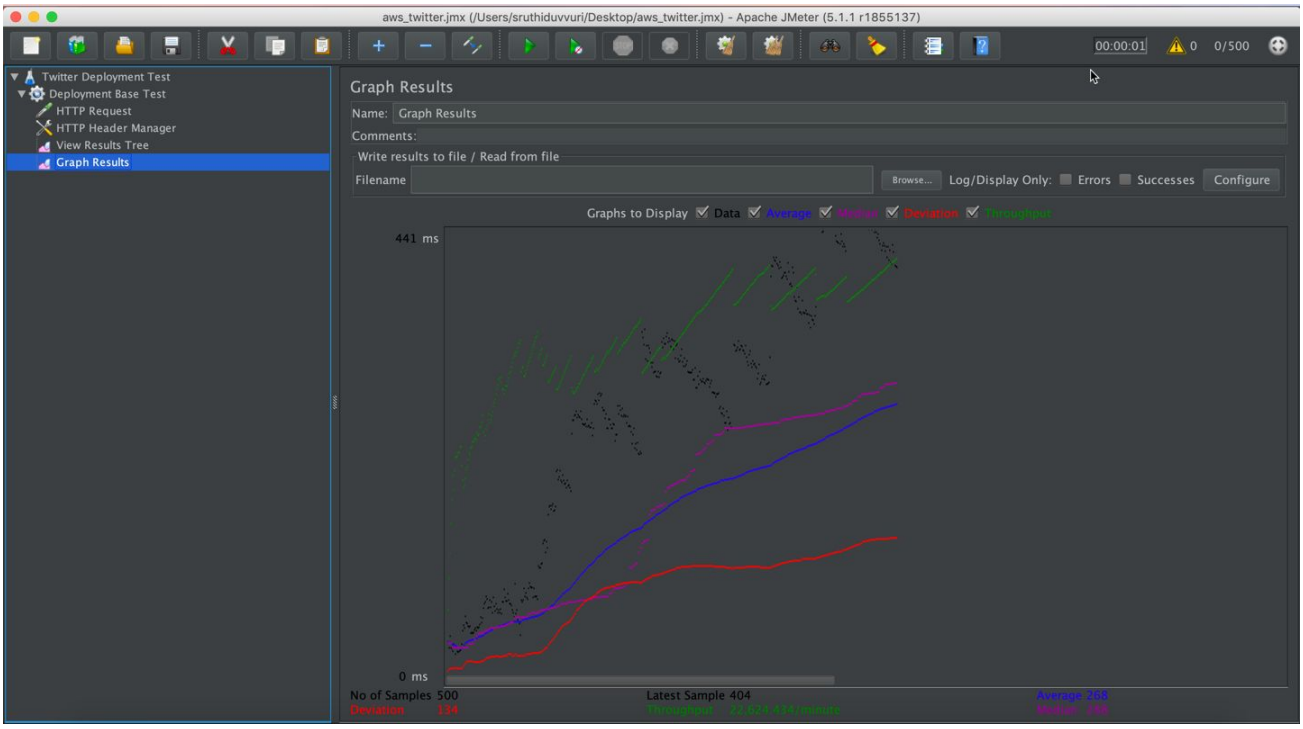
- With 300 concurrent users:



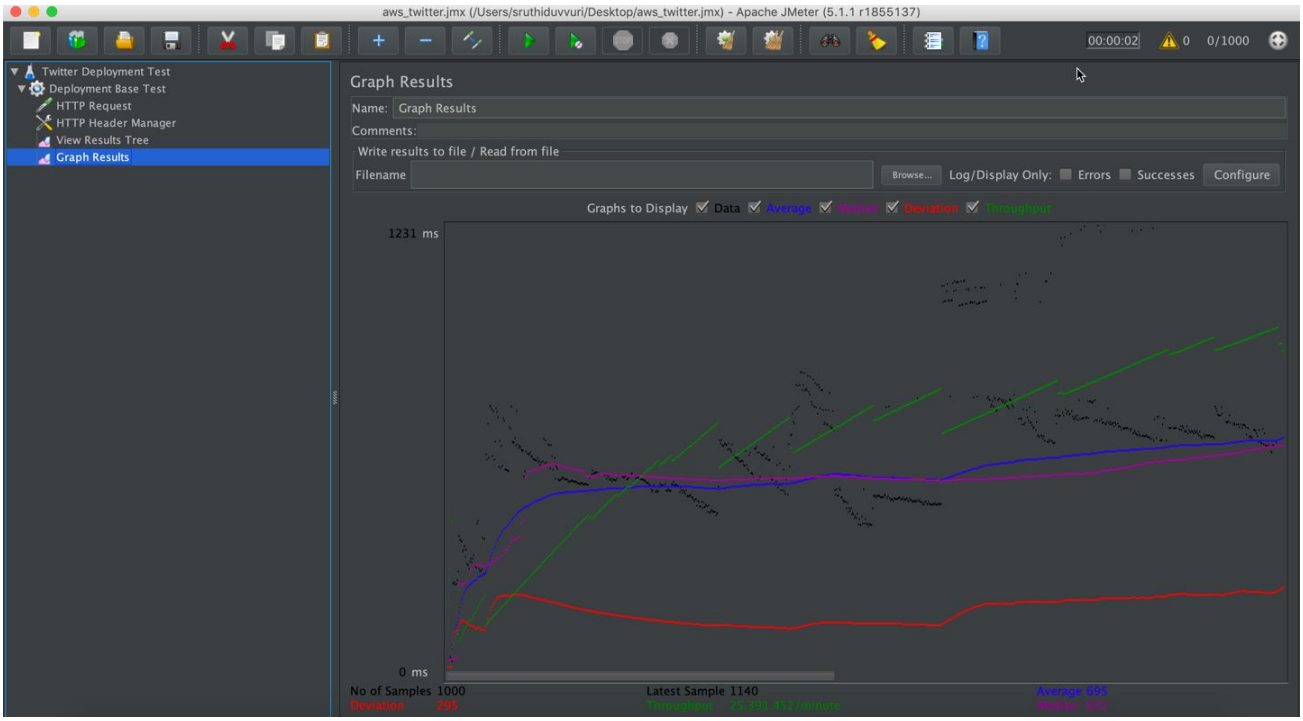
- With 400 concurrent users:



- With 500 concurrent users:



- With 1000 concurrent users:



Kafka and Backend Logs for one api

Kafka Start

```
...ain config/zookeeper.properties ...  ...ka.Kafka config/server.properties ...  ...nloads/kafka_2.11-1.1.0 — -bash  ..._Terminal TERM=xterm-256color  ..._Terminal TERM=xterm-256color ...  ..._Terminal TERM=xterm-256color
Last login: Wed Dec  4 12:10:16 on ttys003
KANIKAs-MacBook-Pro:kafka-backend kanikakhanna$ npm start

> backend@1.0.0 start /Users/kanikakhanna/Desktop/Twitter with kafka and images/kafka-backend
> nodemon server.js

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
producer ready
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
server is running
(node:7442) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
(node:7442) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { u
seUnifiedTopology: true } to the MongoClient constructor.
client ready!
(node:7442) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
mongodb connected
message received for login  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"c4edccbe6c4627a964b10e560e055d5c\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\",\"password\":\"12345\"}}"
inside login of kafka backend
match found
payload is  { id: '5dcde47b914b4b8ea83a0072',
  first_name: 'Kanika',
  last_name: 'Khanna',
```

Kafka giving Response

```
...ain config/zookeeper.properties   ...ka.Kafka config/server.properties   ...nloads/kafka_2.11-1.1.0 — -bash   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
client ready!
(node:7442) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { u
seUnifiedTopology: true } to the MongoClient constructor.
client ready!
(node:7442) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
mongodb connected
message received for login  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"c4edccbe6c4627a964b10e560e055d5c\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\",\"password\":\"12345\"}}"
inside login of kafka backend
match found
payload is  { id: '5dcde47b914b4b8ea83a0072',
  first_name: 'Kanika',
  last_name: 'Khanna',
  username: 'Kanika',
  avatar:
   '//www.gravatar.com/avatar/68479a1e0445e3d6583b55741660d330?s=200&r=pg&d=mm' }
after handle[object Object]
{ response_topic: { '0': 0 } }
message received for get_followers  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"c1574b589e28f80225d96dfecbb74ca\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_followers api of kafka backend. username is.. Kanika
message received for get_following  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"e5d6212af622f3f852c7f3548302192e1\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_following api of kafka backend. username is.. Kanika
message received for get_followers  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"b647d1f1d95d2a16d831f00121b4025a\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_followers api of kafka backend. username is.. Kanika
message received for get_user_owned_lists  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"3994cce57507492d8ba102dcee7e139b\",\"replyTo\":\"response_topic\",\"data\":{\"ownerID\":\"5dcde47b914b4b8ea83a0072\"}}"
inside get_user_owned_lists api of kafka backend
message received for get_subscribed_lists  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"d75f644814f8cbbe371a5621e78f9617\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
inside get_subscribed_lists api of kafka backend
followers are .... [ { _id: 5de74d30e926db492edbfc64,
    follower_id: '5de6f223ce75533b93a0efdb',
    follower_name: 'madhu' },
  { _id: 5de565223d53b00eae1929f4,
    follower_id: '5dce038639b452aa3fe6fa9b',
    follower_name: 'Sruthi' },
  { _id: 5de4870b32fe71105c7f813b,
    follower_id: '5de486f932fe71105c7f8138',
    follower_name: 'mahesh108' },
  { _id: 5de47df2cc5218fea7118e47,
    follower_id: '5de2d0e654b5f2058ca81bca',
    follower_name: 'mahesh' },
  { _id: 5dd958544666dd15b4d01cf4,
    follower_id: '5dd8b437c7cb132514cc4eda',
    follower_name: 'kmanish' },
  { _id: 5dd957af27cc45224ce583af,
    follower_id: '5dd8b437c7cb132514cc4eda',
    follower_name: 'Kanika' } ]
after handle[object Object]
{ response_topic: { '0': 1 } }
message received for get_member_lists_of_user  { handle_request: [Function: handle_request] }
"{\"correlationId\":\"7cc879201a481d3c225a096d6f67eb4a\",\"replyTo\":\"response_topic\",\"data\":{\"username\":\"Kanika\"}}"
```

Backend Response

```
...ain config/zookeeper.properties   ...ka.Kafka config/server.properties   ...nloads/kafka_2.11-1.1.0 — -bash   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color   ..._Terminal TERM=xterm-256color
Last login: Wed Dec  4 12:19:33 on ttys004
KANIKAs-MacBook-Pro:backend kanikakhanna$ npm start

> backend@1.0.0 start /Users/kanikakhanna/Desktop/Twitter with kafka and images/backend
> nodemon index.js

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
producer ready
(node:7524) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
Server running on port 4500
(node:7524) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { u
seUnifiedTopology: true } to the MongoClient constructor.
(node:7524) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
MongoDB Connected from Mongoose(pooling)
inside login of backend
Req Body :  { username: 'Kanika', password: '12345' }
in make request
{ username: 'Kanika', password: '12345' }
1
returning next
in response
in response1
true
client ready!
in response2
{ login: { '0': 0 } }
msg received
response { success: true,
  token:
   'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVkY2RlNDdiOTE0YjRiOGVhODNhMDA3MiIsImZpcnN0X25hbWUiOiJLYW5pa2EiLCJsYXN0X25hbWUiOiJLaGFubmEiLCJ1c2VybmFtZSI6IkthbmlrYSIsImF2YXRhciI6Ii8vd3d3LmdyYXZhdGFyLmNvbS9hdmF0YXIvNjg0NzlhMWUwNDQ1ZTNkNjU4M2I1NTc0MTY2MGQzMzA_cz0yMDAmcj1wZyZkPW1tIiwiaWF0IjoxNTc1NDkwODMxLCJleHAiOjE1NzU3MDY4MzF9.tcoHGcv4GFP155mvcLK5kTig2iNTXg-Q0cvUxG7q9Kc' }
in result
{ success: true,
  token:
   'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVkY2RlNDdiOTE0YjRiOGVhODNhMDA3MiIsImZpcnN0X25hbWUiOiJLYW5pa2EiLCJsYXN0X25hbWUiOiJLaGFubmEiLCJ1c2VybmFtZSI6IkthbmlrYSIsImF2YXRhciI6Ii8vd3d3LmdyYXZhdGFyLmNvbS9hdmF0YXIvNjg0NzlhMWUwNDQ1ZTNkNjU4M2I1NTc0MTY2MGQzMzA_cz0yMDAmcj1wZyZkPW1tIiwiaWF0IjoxNTc1NDkwODMxLCJleHAiOjE1NzU3MDY4MzF9.tcoHGcv4GFP155mvcLK5kTig2iNTXg-Q0cvUxG7q9Kc' }
Inside else
inside get_followers of backend
Req Body :  { username: 'Kanika' }
in make request
{ username: 'Kanika' }
in response
in response1
true
inside tweet
inside get_following of backend
Req Body :  { username: 'Kanika' }
in make request
{ username: 'Kanika' }
in response
in response1
true
in response2
{ get_followers: { '0': 0 } }
inside get_followers of backend
Req Body :  { username: 'Kanika' }
in make request
{ username: 'Kanika' }
```

Lessons Learnt

- Inorder to achieve the Higher availability and prevent single failure, replicating the DB instance into multiple instances and thereby forming a master-slave configuration.
- Horizontal Scaling rather than vertical scaling.
- Always using the cache storage and caching everything which helps in storing the results which helps in avoiding the same computations again and again, which inturn improves the performance. It also helps in reducing the latency.
- Providing the users the needed data. As consistency, partitioning and availability cannot be achieved together
- Maintaining modularized and clean code helps in speedy integrations.