

---

# CMPE 273 - Lab 2 Report

## Twitter Clone Implementation including mongodb and rabbitmq

Madhuri Motoori - April 16, 2016

---

### Introduction:

This web application is designed to serve as a clone of twitter application. In this application I have implemented various modules that support basic user functionalities (signin, signup, logout ), updating profile , retrieving following list , followers list , ability to follow users, ability to tweet and retweet , implementing hashtag functionality. To improve performance and scalability I used rabbitmq and connection pooling with mongodb.

### Goals:

The goal of this project is to develop Twitter application including rabbitmq for asynchronous message transfer , connection pooling handling proper validations and exceptions. Another goal is to store sessions in mongodb.

### Purpose of this project :

- To understand how introduction of rabbitmq using queues increased performance and scalability.
- To gain mongodb database knowledge and understand where to use sql and nosql databases.
- To understand how connection pooling helps to improve performance.

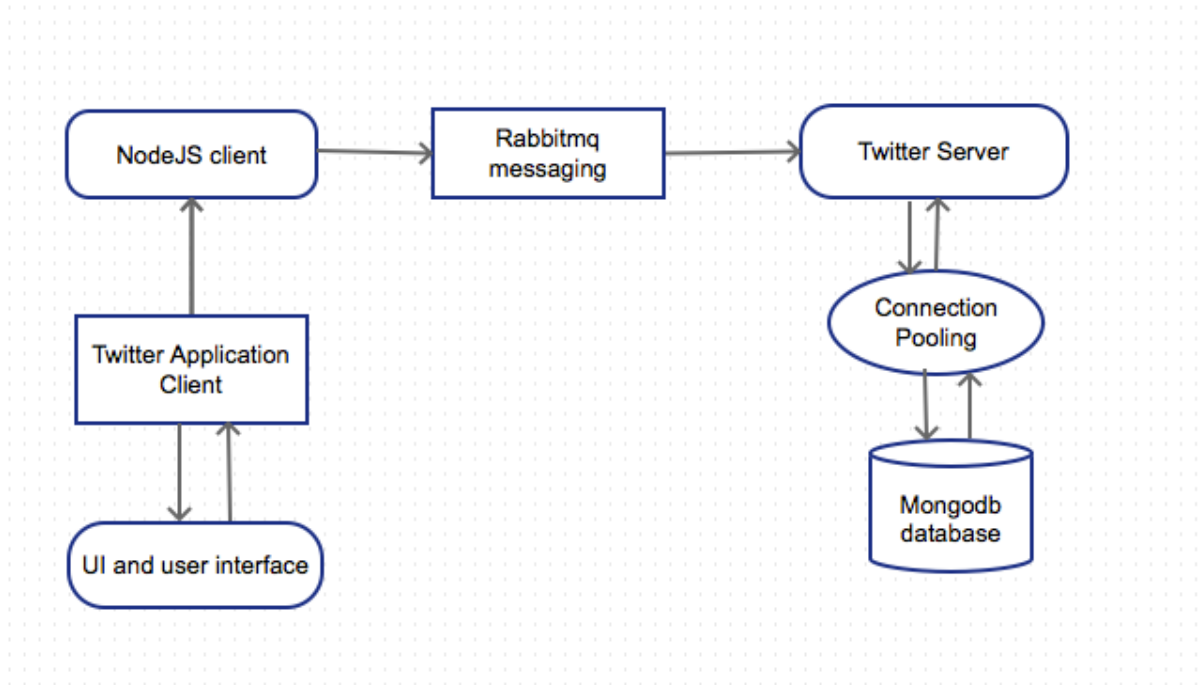
### System design:

This application is a prototype of twitter application. The NodeJS client gets the GET or POST requests and based on the given resource on the url appropriate methods are being called. When the user logs in successfully user session is created and the sessions are stored in mongodb. For access of another user related pages the server proceeds only if there is a valid session.

The appropriate methods in the nodeJS client uses rabbitmq for asynchronous

---

message transfer which are in turn handled by appropriate methods in the Twitter server which communicates with the mongodb to get required results as shown in the project architecture image.



Similar to Lab1, Model view controller approach is being followed here where Nodejs is used as a model , ejs are used as views and angular acts as controller to take the requests from user , bring required information from the model and send it to views. In lab2 along with the above functionalities AMQP is used as messaging protocol , rabbitmq as middleware and mongodb as database .

Twitter server interacts with the database in asynchronous manner to get the data required based on the request. System design of twitter application is explained below in detail using the screenshots.

## Exception Handling:

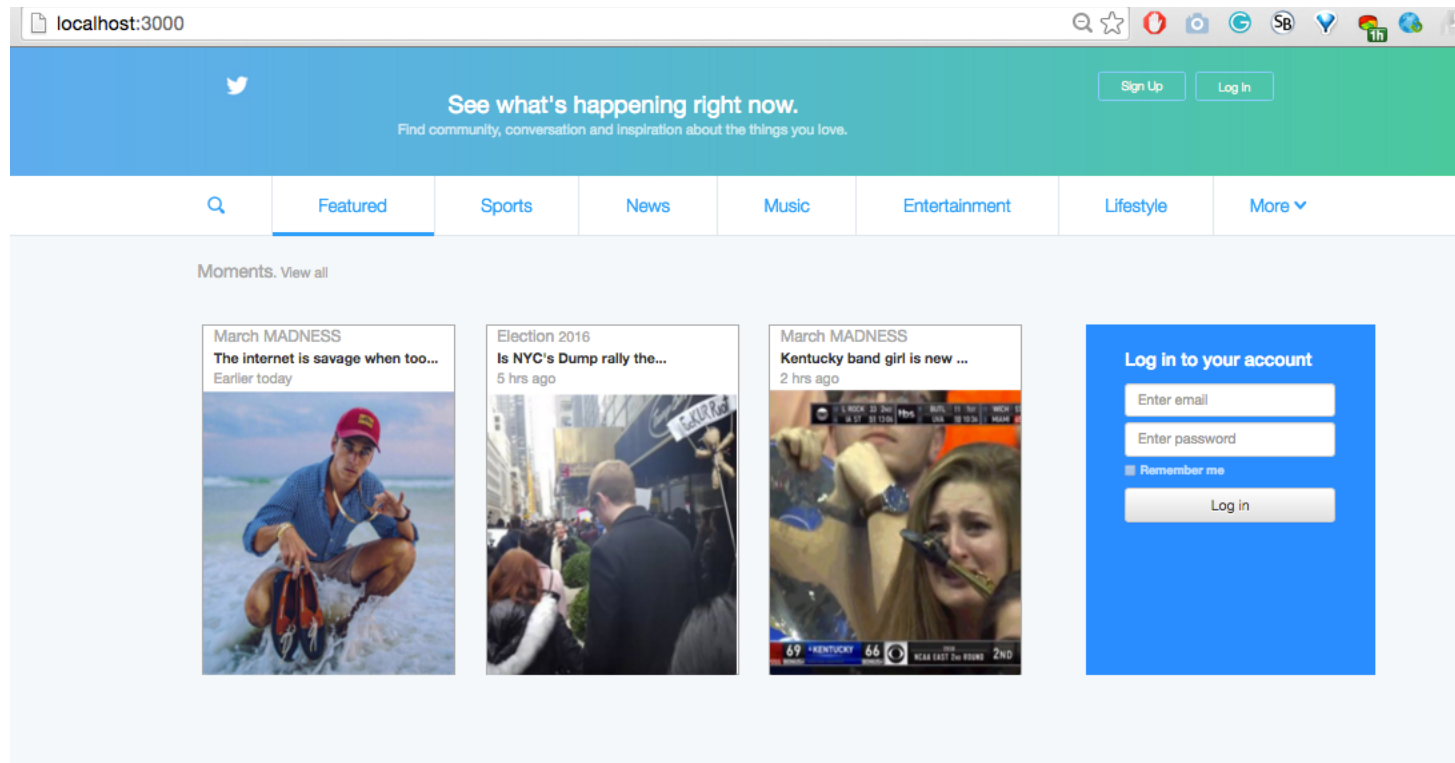
All exceptions are being handled and results are shown to the user. The error results are displayed to user using angularJs ng-show/ng-hide attribute of angular JS wherever required. Form validation feature of html5 is also being used. Exception cases related to sessions are also handled like if user session is invalid then he is redirected to loginPage , similarly if user tries to hit server to another page it is redirected to

loginPage. Nodejs code is also taken care to handle exceptions if the input data is undefined or empty. Exception handling cases are shown in screenshots in detail.

## Screenshots:

### Basic User functionalities:

Main site page from where user can signup and login



---

## SignUp:

If we click the Signup button , it will be redirected to signup page. The user has to signup by providing required information. The system takes care of all exceptions and proper validations. The password provided by user is encrypted providing proper security.



### Join Twitter today.

☐ Male ☐ Female

☐ Remember me

By signing up, you agree to the [Terms of Service](#) and [Privacy Policy](#), including [Cookie Use](#). Others will be able to find you by email or phone number when provided.

---

## Exception handling at signup page:

If username already exists, it throws error message like below,

---

### Join Twitter today.

Username exists, Please choose a different username

test1

test1@gmail.com

test1firstname

test1lastname

☒ Male ☐ Female

11/11/1991

\*\*\*\*\*

☐ Remember me

Submit

By signing up, you agree to the Terms of Service and Privacy Policy, including Cookie Use. Others will be able to find you by email or phone number when provided.

---

If email already exists, it throws appropriate error message

## Join Twitter today.

Email already exists

test9

test1@gmail.com

test1firstname

test1lastname

☒ Male ☐ Female

11/11/1991

.....

☐ Remember me

Submit

By signing up, you agree to the [Terms of Service](#) and [Privacy Policy](#), including [Cookie Use](#). Others will be able to find you by email or phone number when provided.


---

The form is not submitted if all the required information is not provided like below

---

## Join Twitter today.

test1@gmail.com

 Please fill out this field.

☒ Male ☐ Female

☐ Remember me

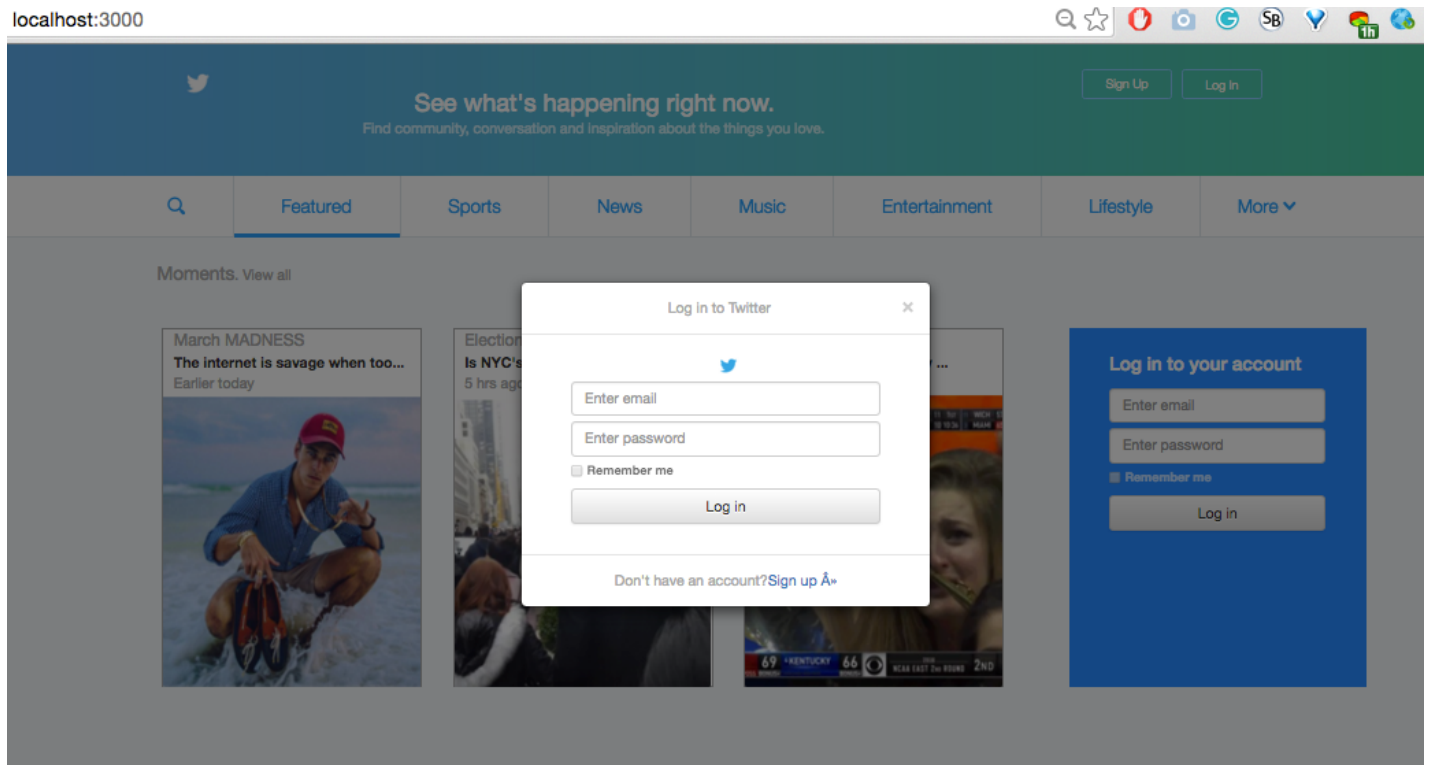
By signing up, you agree to the Terms of Service and Privacy Policy, including Cookie Use. Others will be able to find you by email or phone number when provided.

## LogIn:

User can login into the system by providing correct username and password that which he provided while he signed up. The information provided by user and validated using the data from the database. The password is decrypted and checked with user entered password. The system handles proper validation and exception handling. If the information provided by the user is correct then he will be redirected to the homepage.

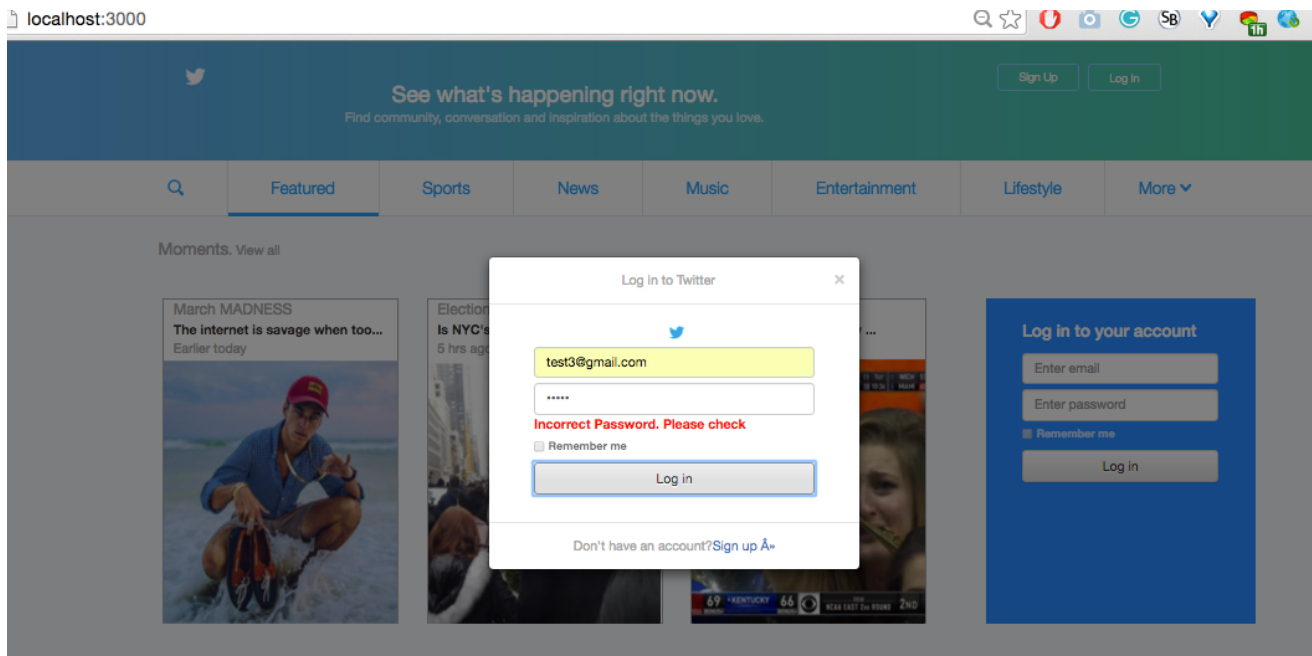
Login modal from main site page:

If we click the login button from main page , the following modal appears

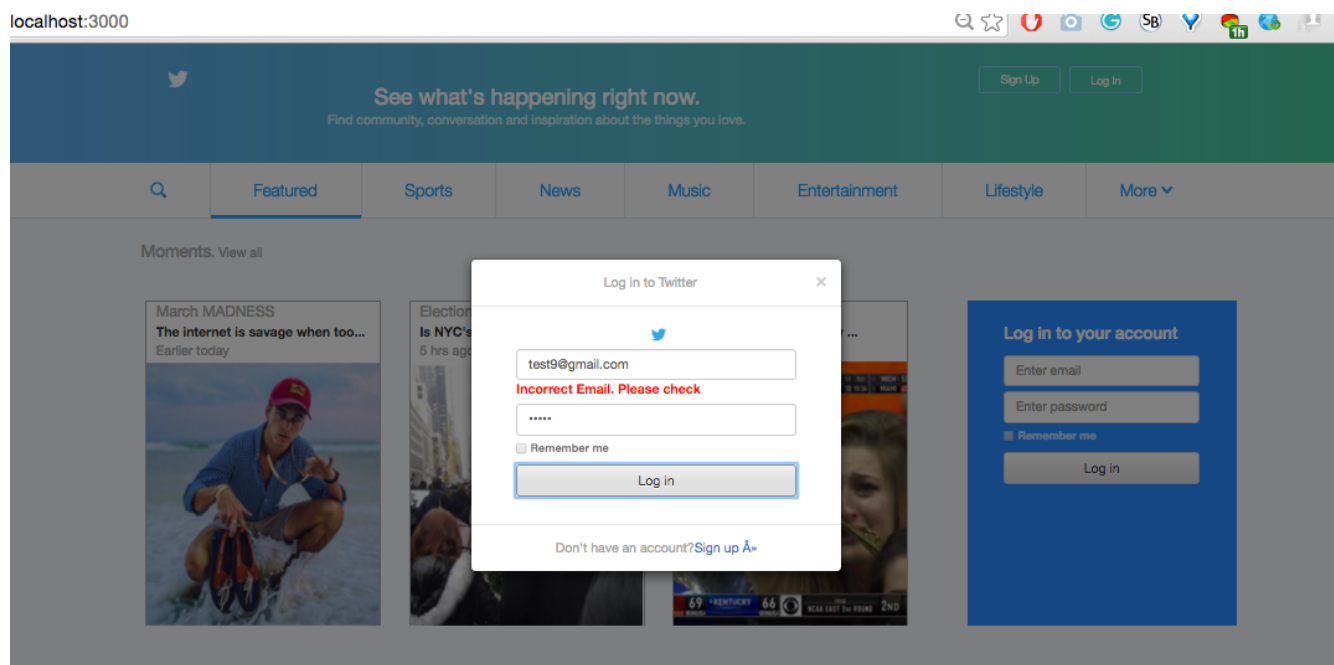




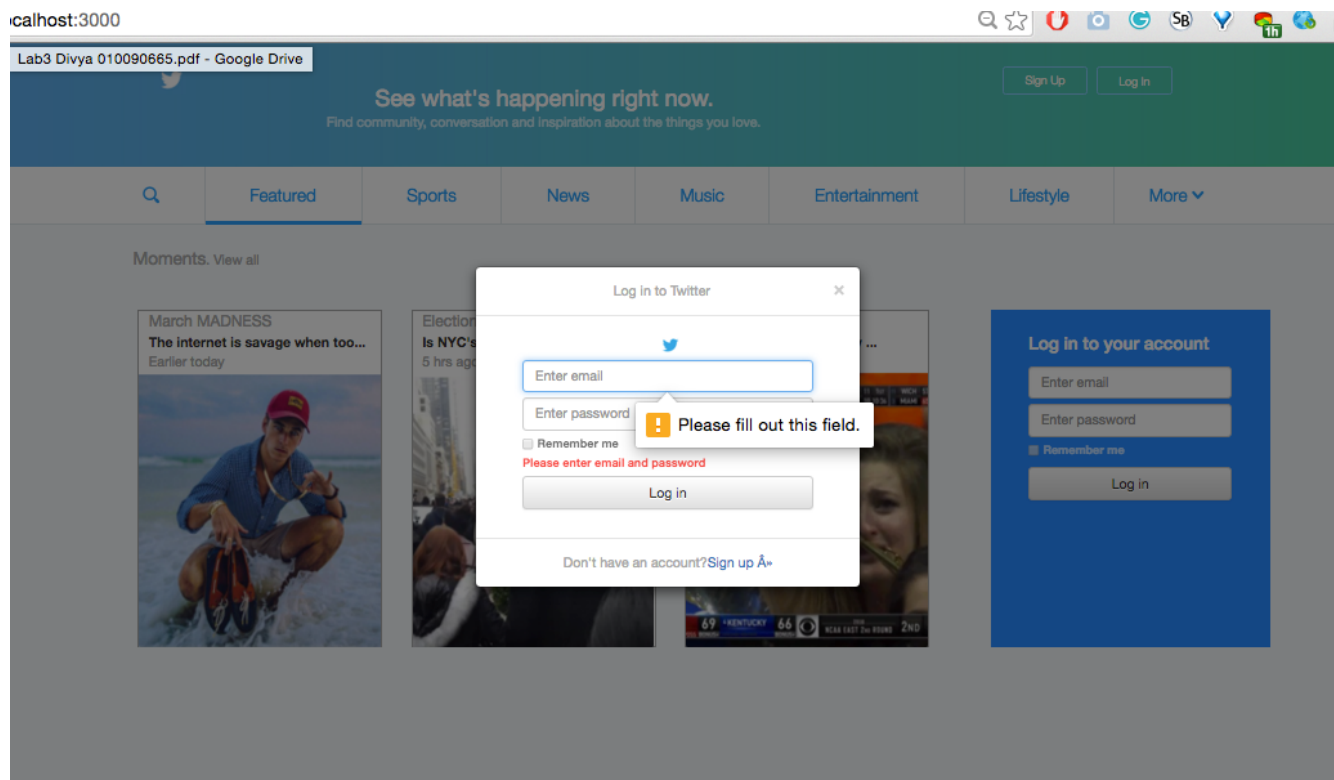
Error cases: If user provides incorrect password, appropriate error message is shown



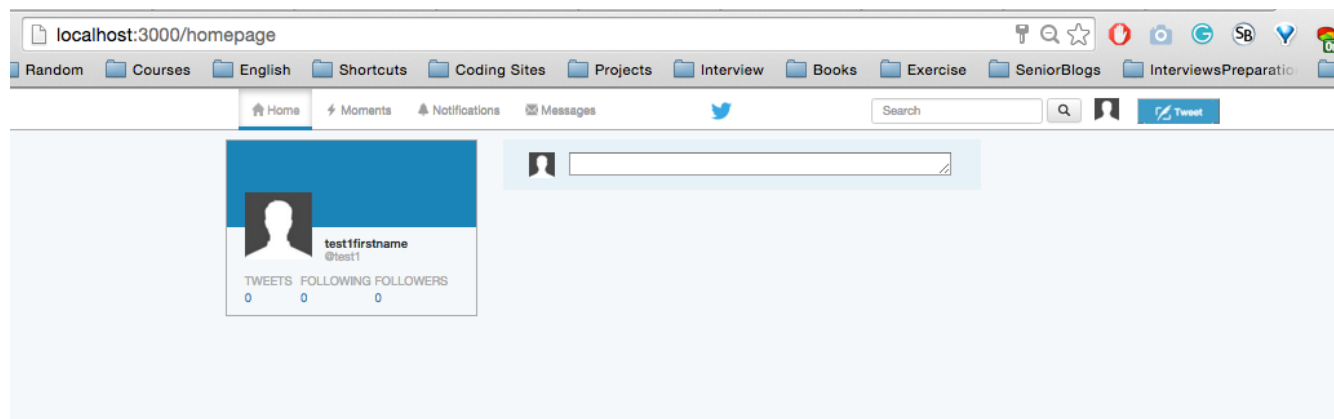
If user provides incorrect email, the following error message is shown



If user does not provide all the information, appropriate error message is shown

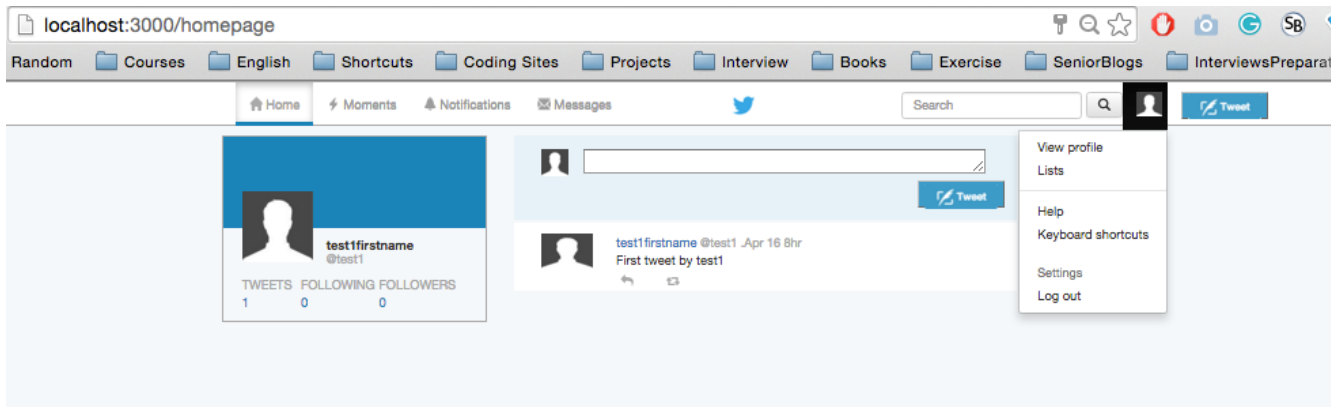


The user will be redirected to the homepage if all the provided information is correct,

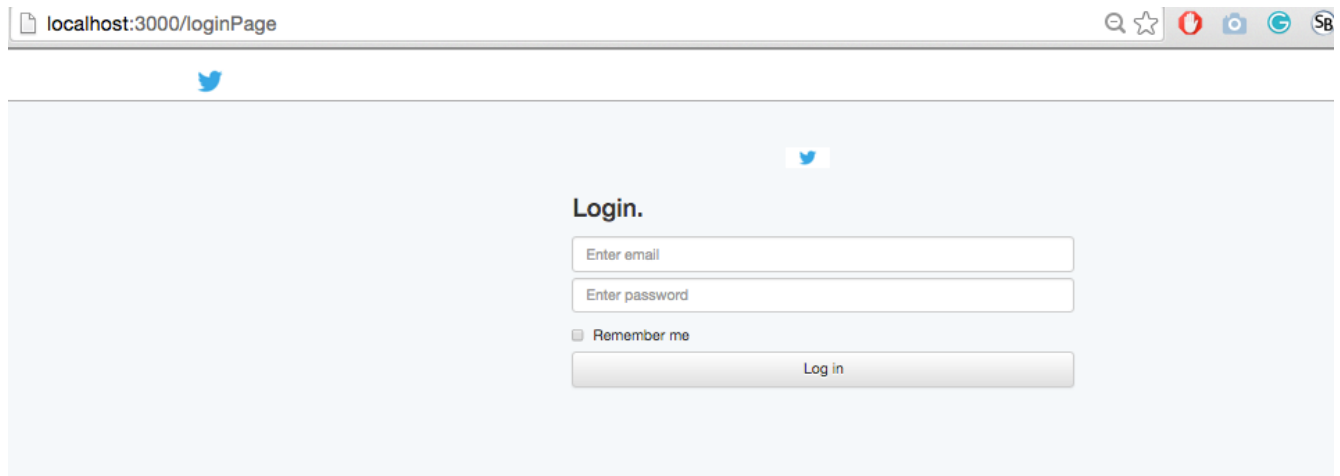


## Signup:

The user will be able to logout from the application when he clicks logout which appears in the dropdown menu of the picture which is available in the navigation bar.



When user clicks logout , angularJS sends http request to get /logout page. In this module nodeJS destroys the user session and user is redirected to loginPage like below.

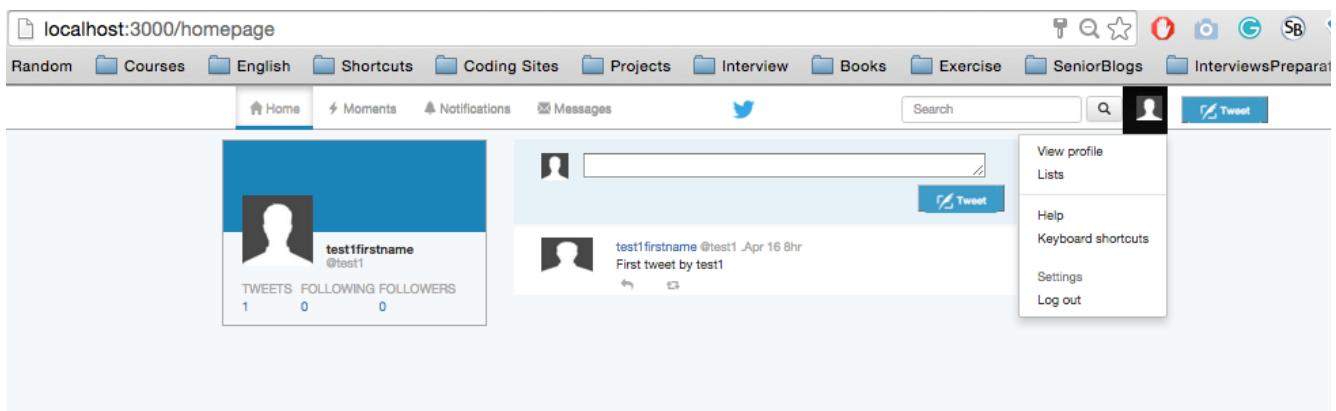


## Profile functionalities:

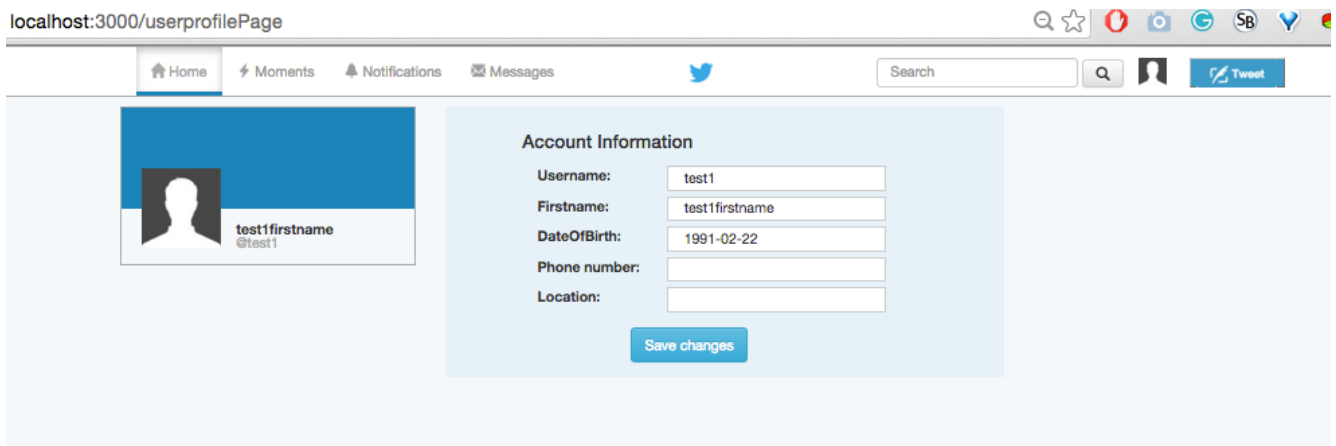
### a. About:

The about page contains details of the user like birthday, twitter handle (username), contact information and location .

User is redirected to profile about page when he clicks view profile in the dropdown menu of image displayed on the navigation bar like below,



User redirected to the page only after checking valid user session else will be redirected to loginpage.



---

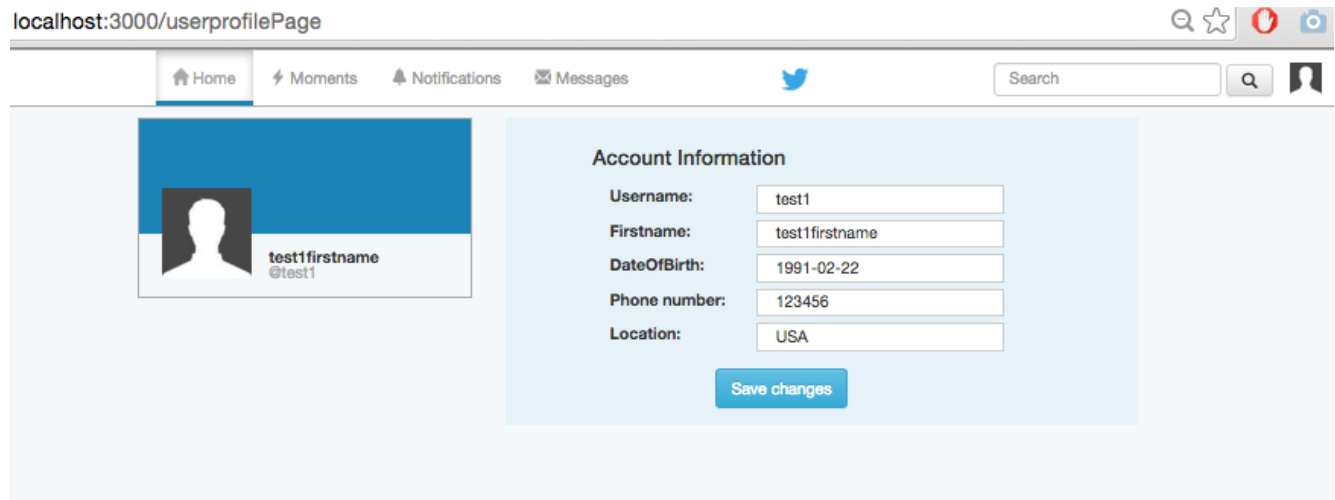
Here only the values like twitter handle , firstname, dob which are received from user during signup are displayed. But user can edit the fields and update them using save changes button.

Lets say test1 user above, inputs phone number as 123456 and location as USA as below and clicks the save changes button.

The changes are saved into database from backend like below.

```
{
  "_id" : 1,
  "username" : "test1",
  "email" : "test1@gmail.com",
  "password" : "$2a$10$gopNq/j5fvu7xaJkmTmxD.GeM/ISGJNV4dq1AGP70nDgu1IgxPg
7i",
  "firstname" : "test1firstname",
  "lastname" : "test1lastname",
  "gender" : "male",
  "dob" : "1991-02-22",
  "tweets" : [
    1
  ],
  "following" : [ ],
  "followers" : [ ],
  "phonenumber" : "123456",
  "location" : "USA"
}
```

If user goes to his profile page he will be able to see those updated changes,

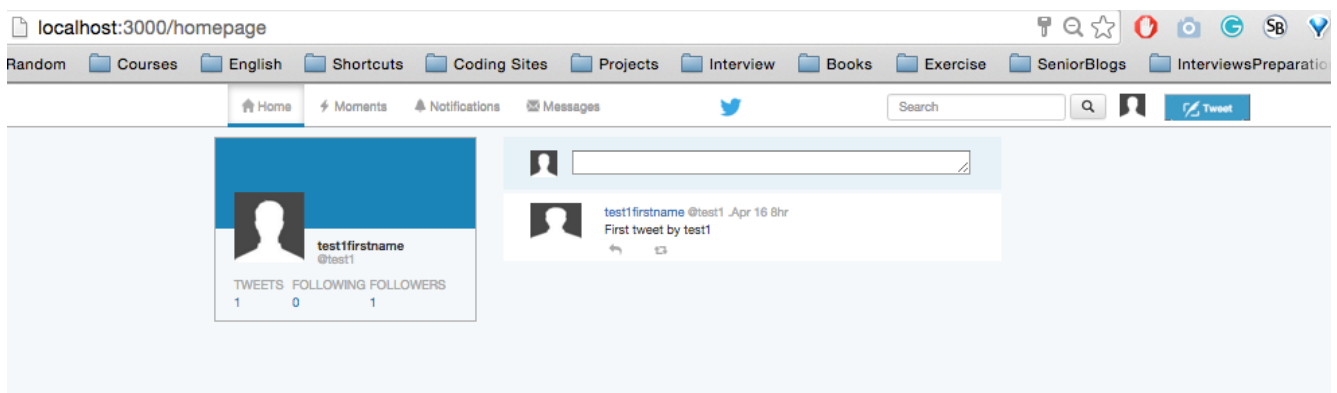


## b. Followers and Following list and able to follow people.

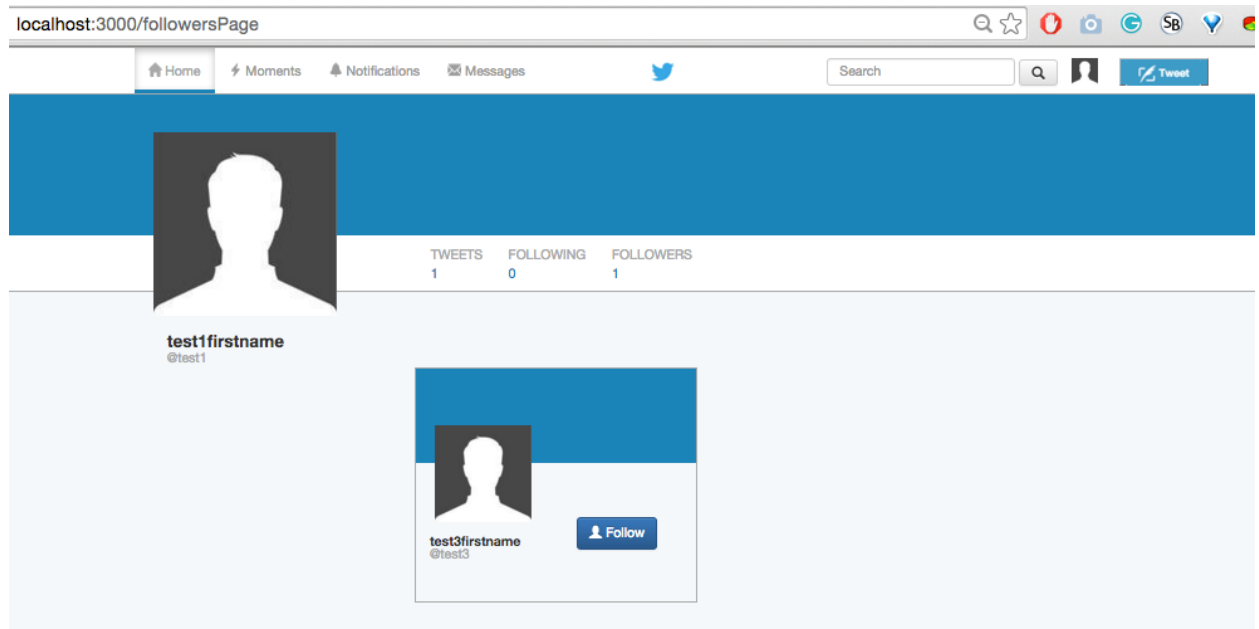
### Followers list:

When user clicks the Followers hyperlink which will turn to blue color only after hovering, it directs the user to followers page like below. This page provides information about the users who are following this user.

Incase of test1 user , at present situation the number of his followers are only one as below.

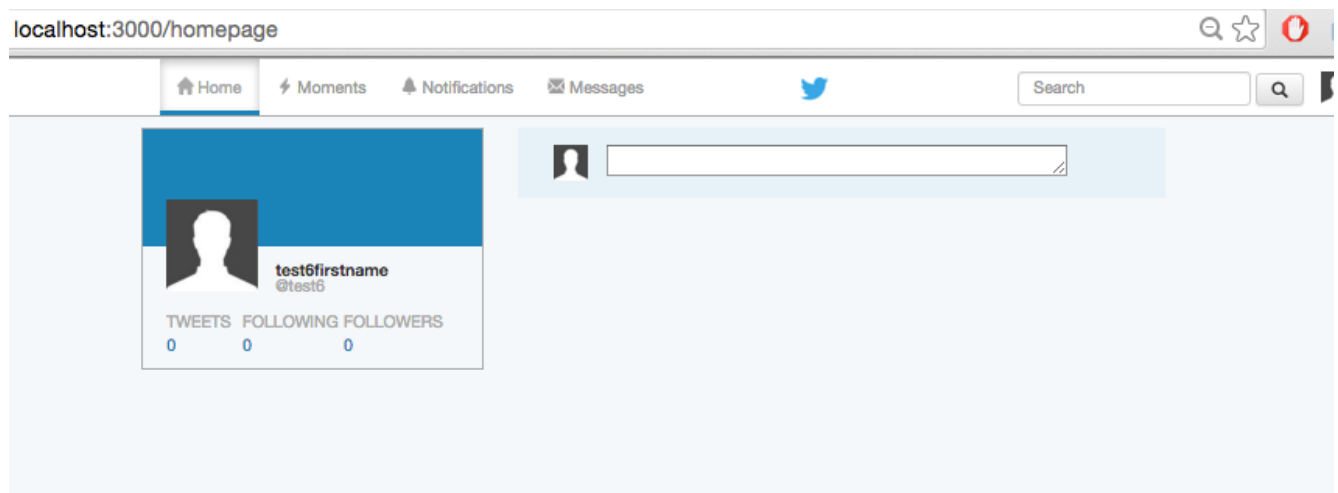


When user clicks followers it redirects the user to /followerspage which provides information about the followers like below.

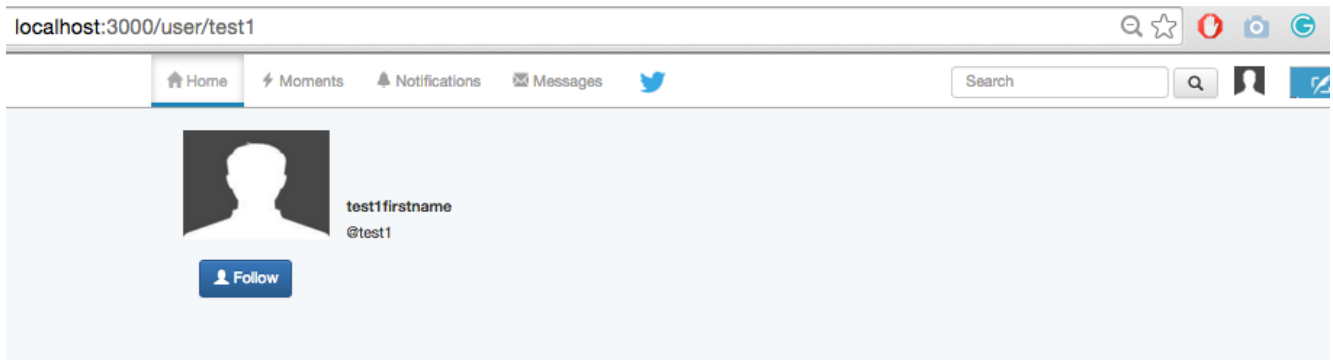


Lets add test6 and test7 users to follow test1 like below. For this scenario am logging out of test1 account and logging in from test6 account.

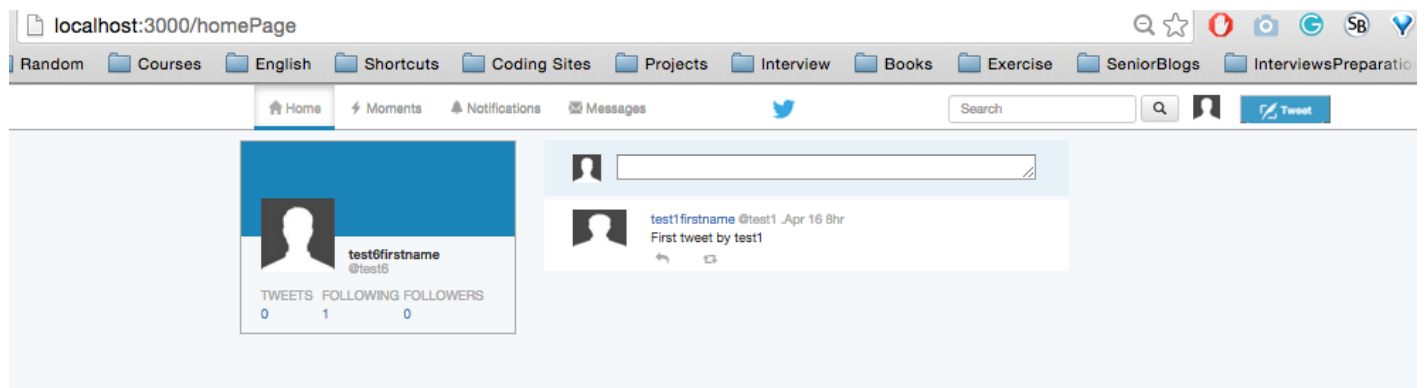
The homepage of test6 is like below he is not following anyone for now.



test6 follows test1 like below, by clicking Follow button. (Follow functionality implementation is explained later in this report in detail).



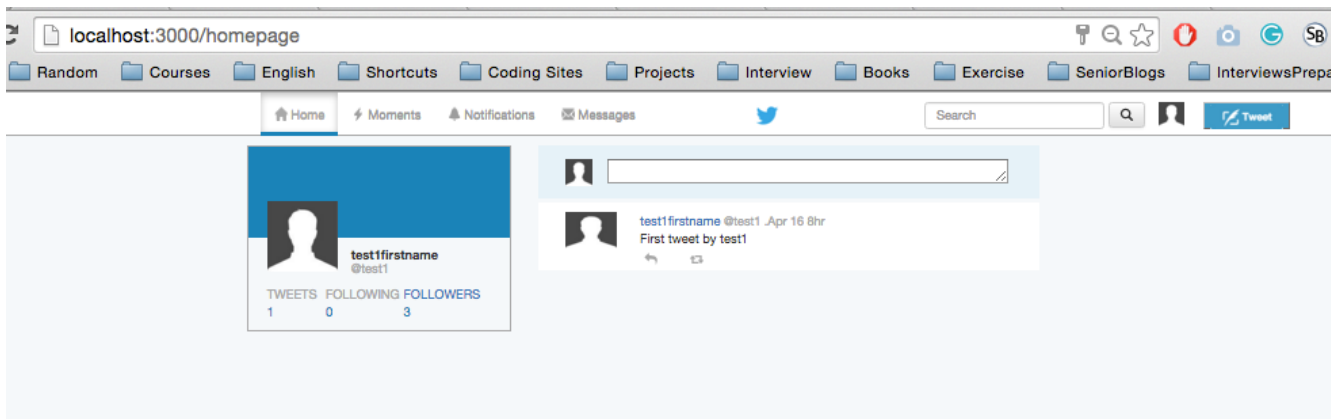
When follow button is clicked , the number of following list is increased by one in the user page like below.



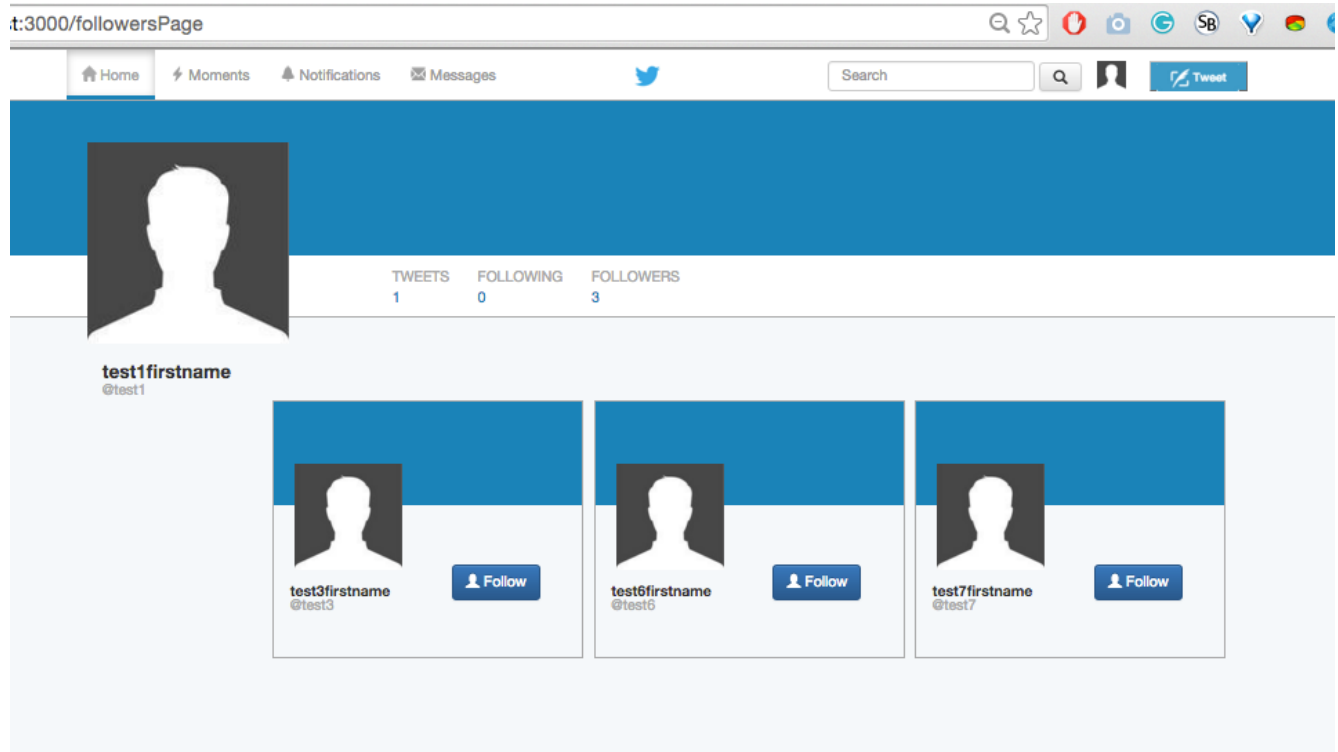
Similar process is carried out in case of test7 user to follow test1.

Now if we go to test1 user account and check the following list , the number of followers count is increased to three from one and the details are updated in the list.





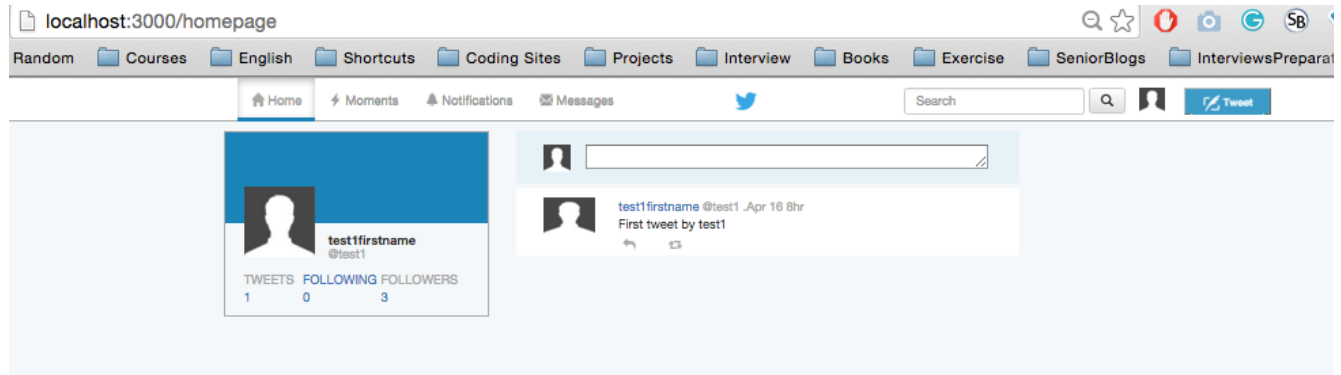
The followers list contains details of test6 and test7 too like below.



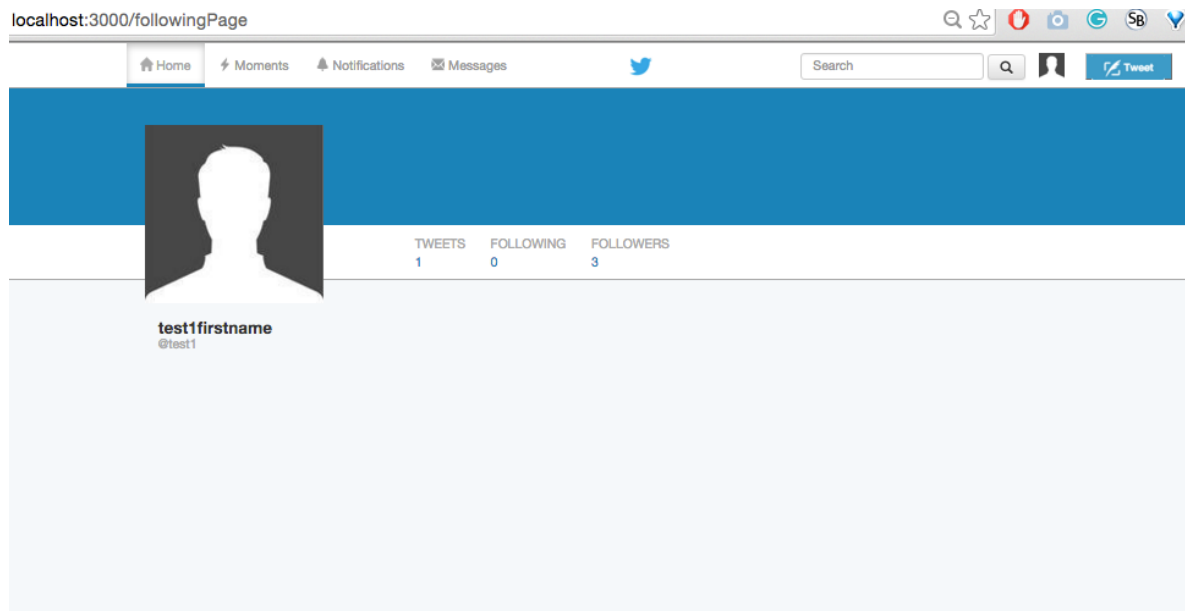
## FollowingList:

When user clicks the Following hyperlink in the user home page which turns to blue color only after hover, directs the user to following page like below. This page provides information about the users whom user is following.

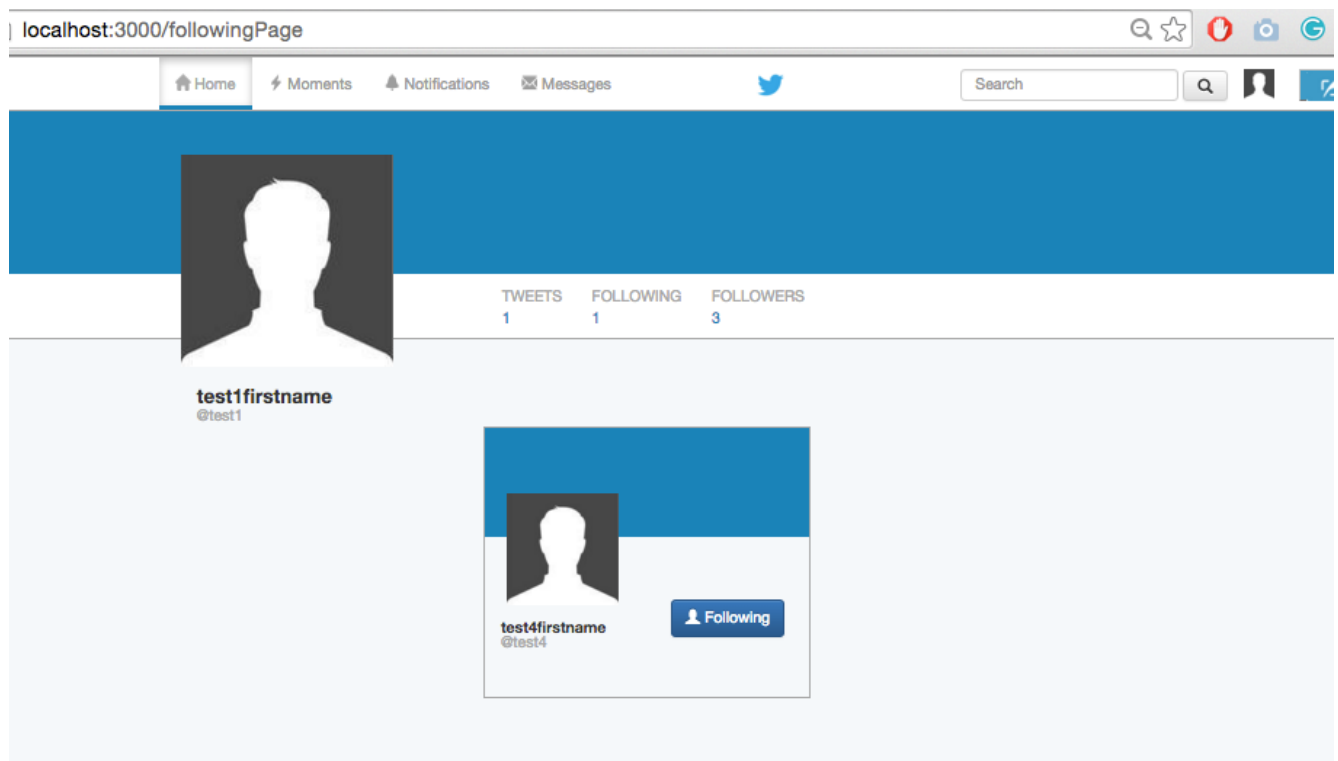
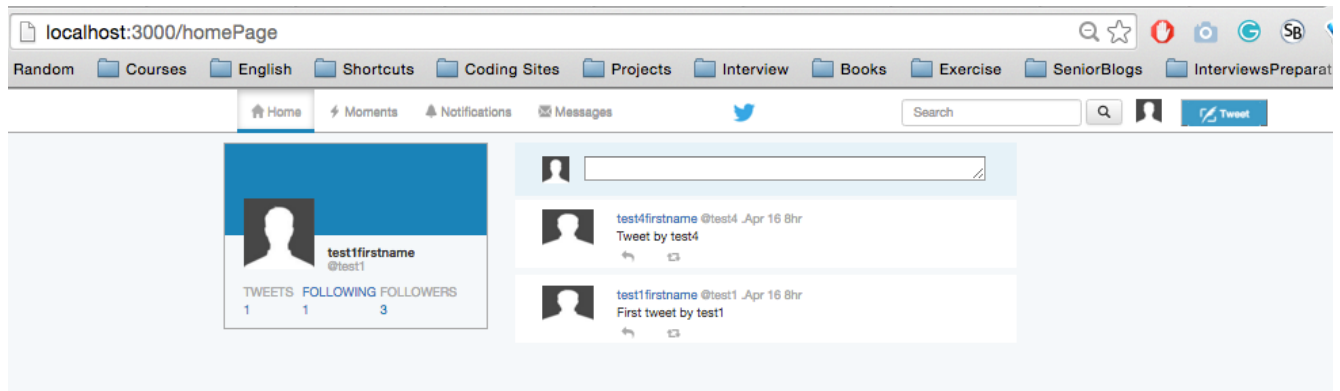
Incase of test1 user , at present situation the number of users he is following are 0 like below.



Clicking the following link , it redirects to followingPage which contains information about the users whom the user(test1 in this example) is following . As there are no followers nothing is displayed in the page.



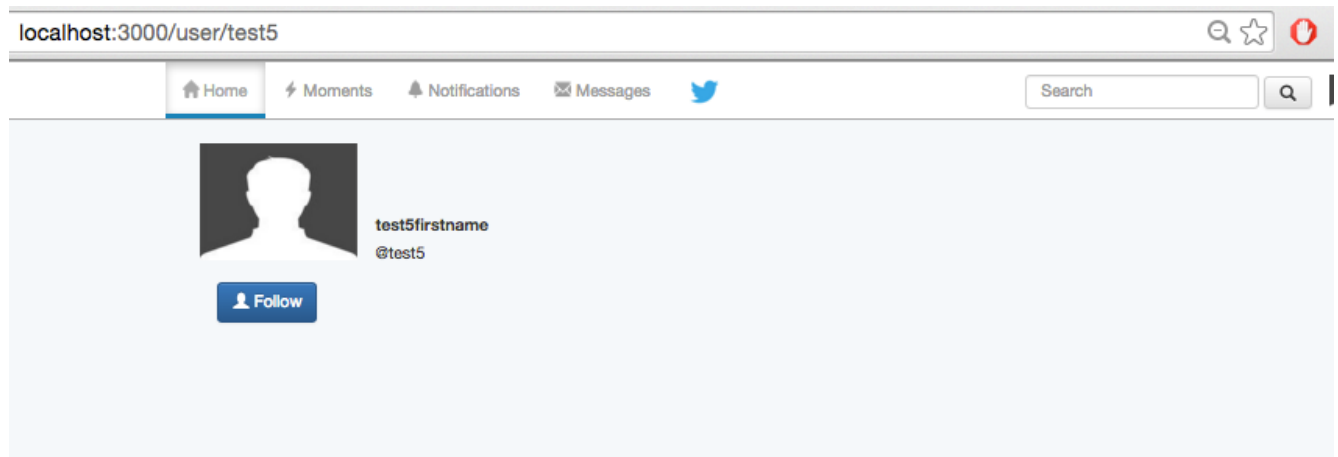
Let test1 follow test4 , then following count will be increased to one and followingList page contains information about test4.



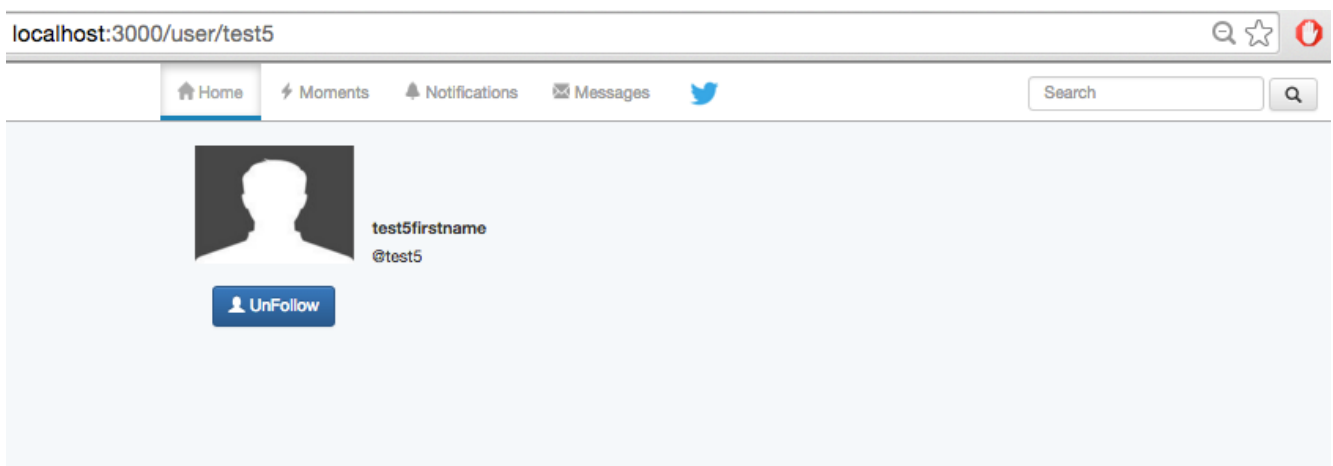
## Follow:

In this application user will be able to follow other users using dynamic url .  
In dynamic urls the username is passed. Using this information twitter server will check if the user is already available in the system or not. If not available the user will be redirected to the homepage else will be directed to the user page which contains follow button .

In this scenario, lets say test1 wants to follow test5 then a request is sent using dynamic url passing test5 in the url which opens test5 page.



By clicking follow button the test1 following list will be increased by one  
and if he goes the test5 user page again then unfollow button is shown like below.



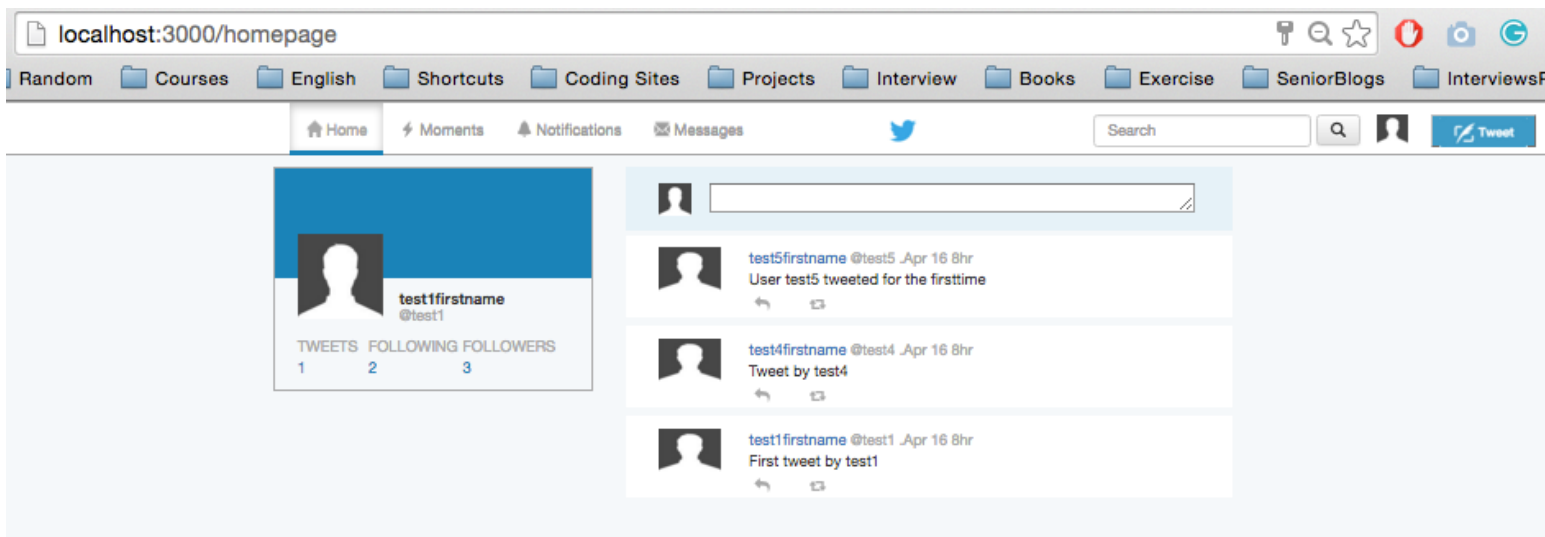
---

If test1 tries to follow user test9 who is not existing , then he will be redirected to homepage.

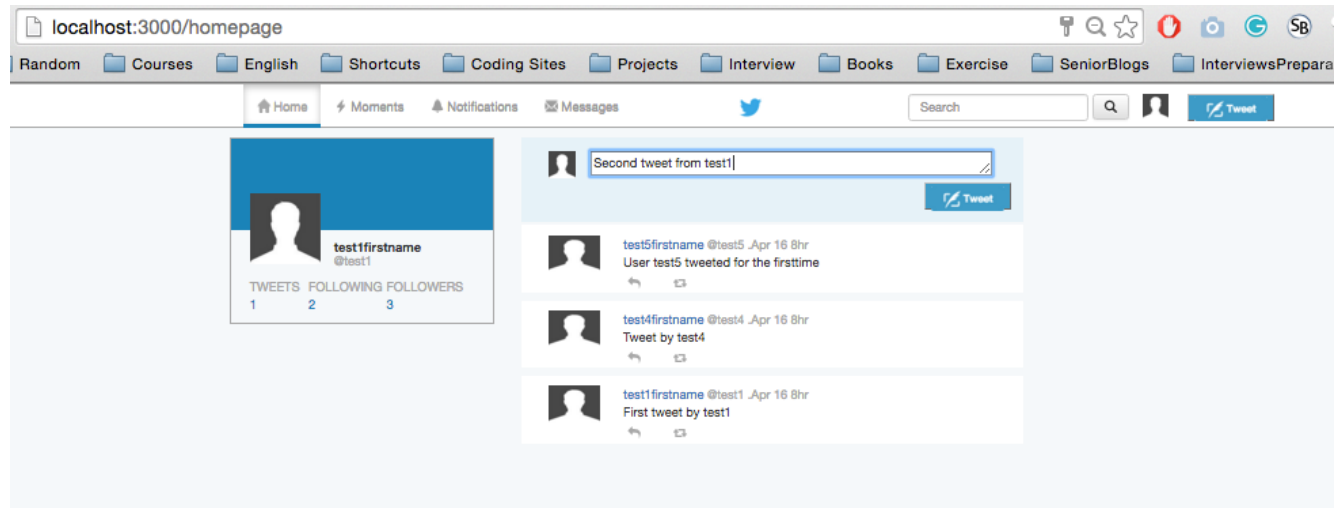
## Show user tweets:

If user clicks the textbox in the user main page a tweet button is shown. When user enters the tweet in textarea and clicks the tweet button. The tweet will be displayed in the user main page according to the time and number of tweets count will be increased by one. Incase if tweet value is empty , server will not insert the tweet into the database.

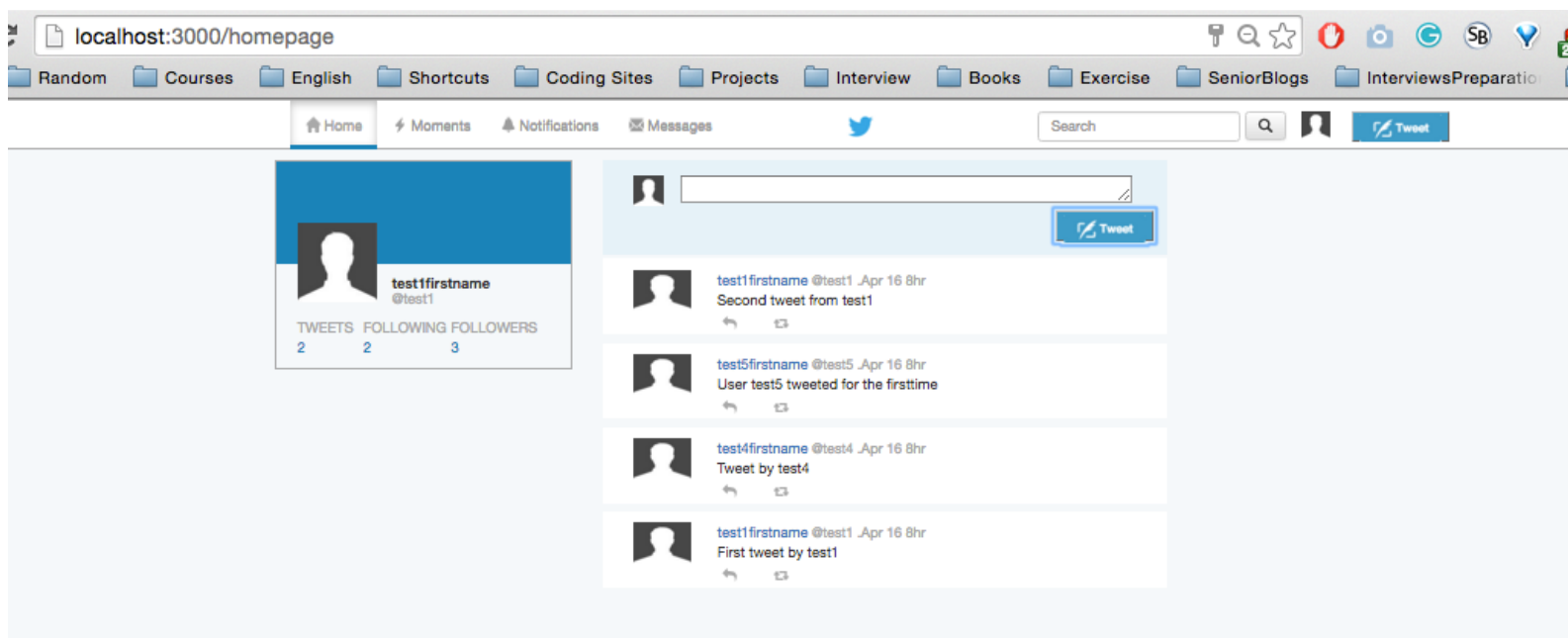
Example of test1, in present state the user tweets and followers tweets are displayed like below.



When test1 clicks the text box, tweet button is shown like below ,



When user clicks the tweet button, the required information is stored and tweets of the user are shown in the homepage like below and tweets count is increased by one.

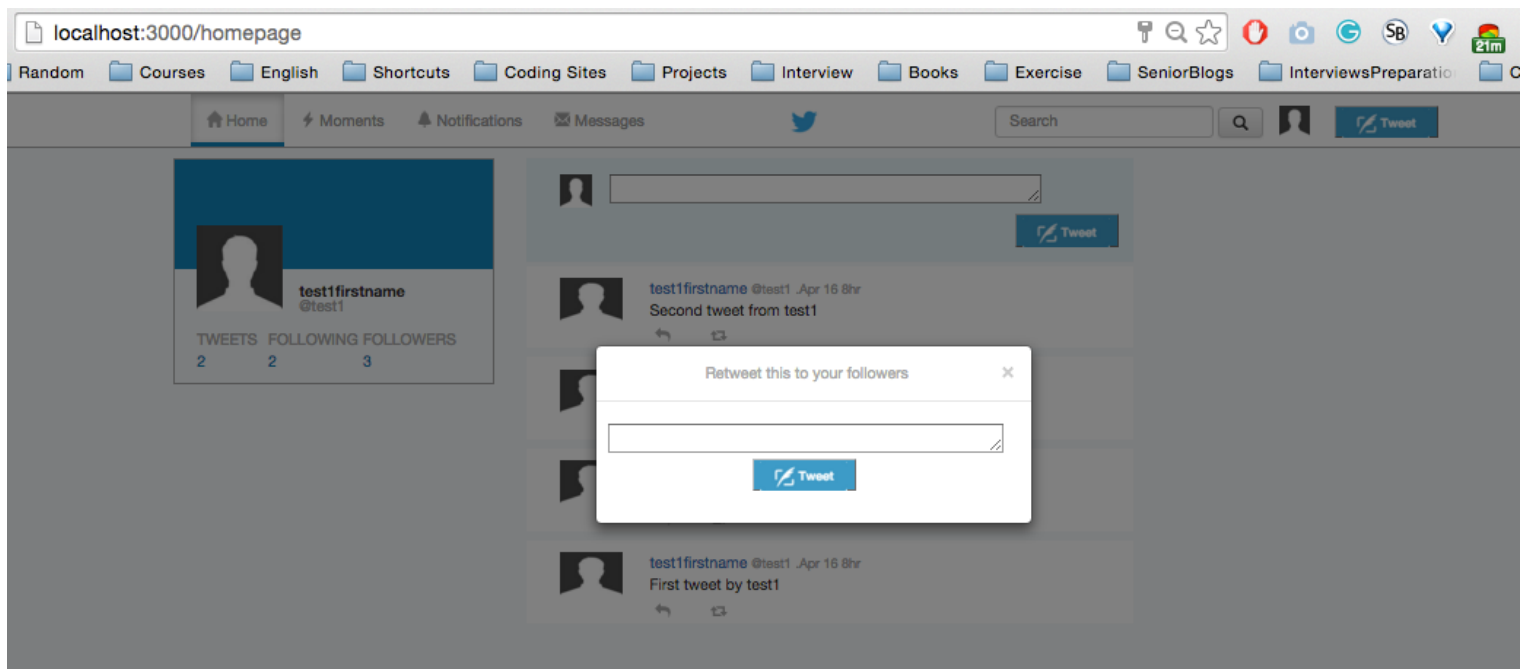


---

## Show user's retweets:

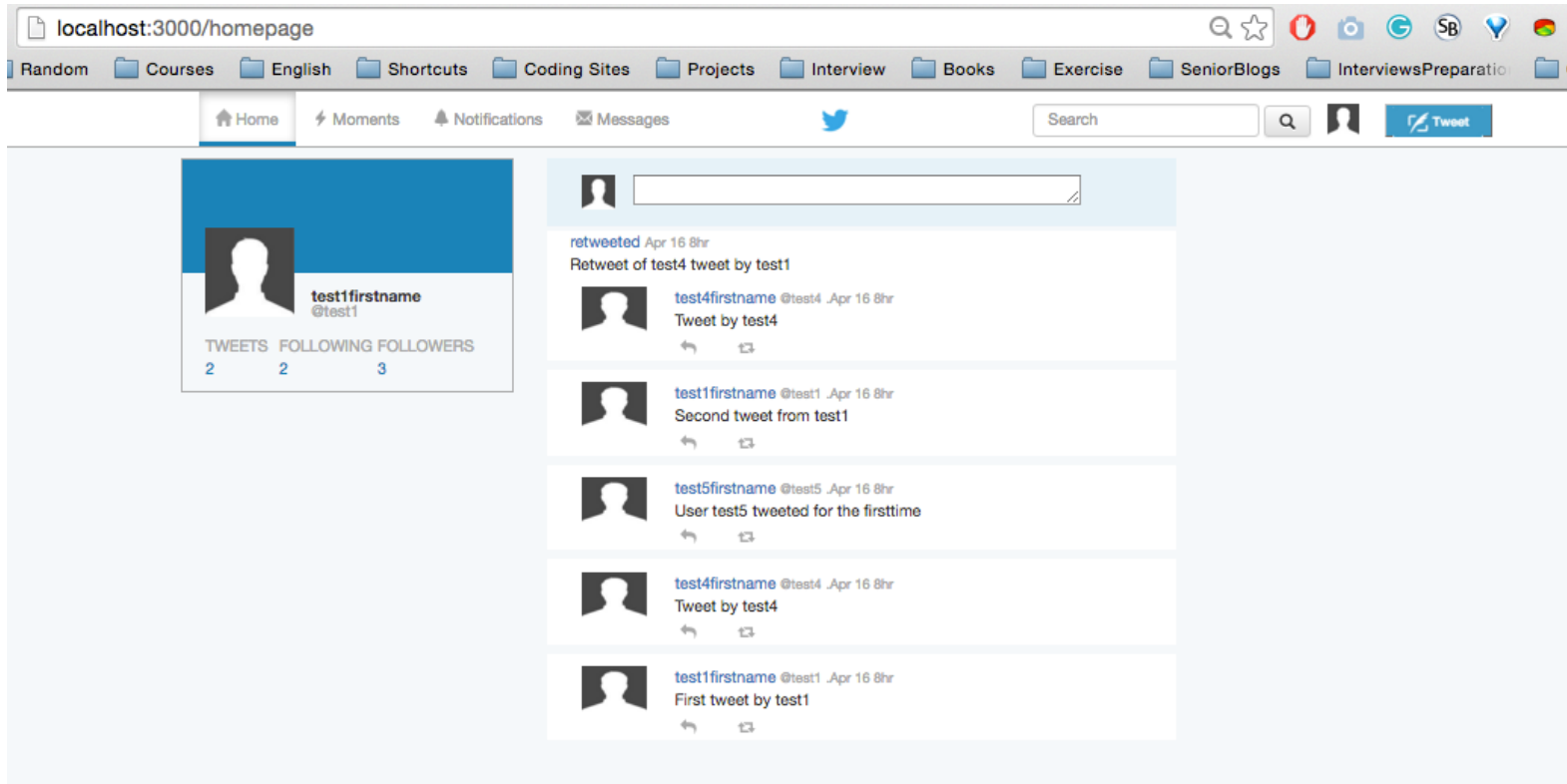
User is able to retweet the tweets using the retweet font icon displayed in the bottom of each tweet. When it is clicked , a modal is shown where the user can give retweet comment and retweet the tweet like below.

Lets say test1 want to retweet test4 tweet "Tweet from test4" , dialogue box appears like below.



test1 can provide retweet comment and click tweet button like below. User can also not provide retweet comment. When it is retweeted the retweet will appear in user home page and his followers like below according to the time.

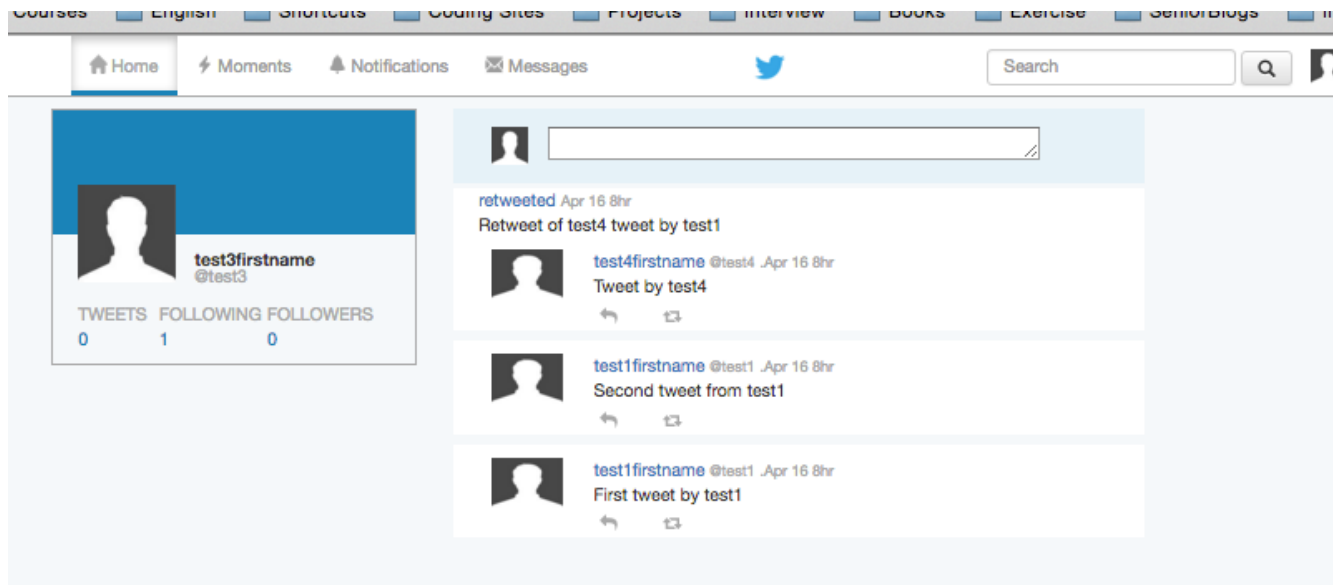
test1mainpage:



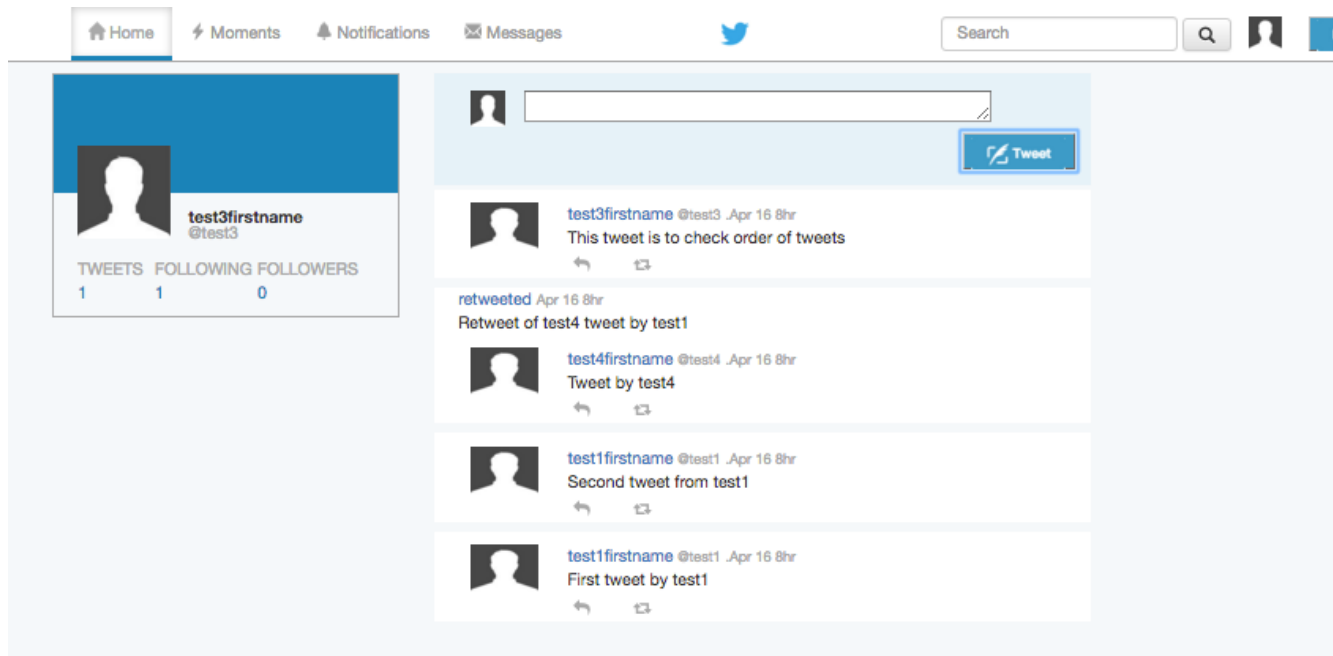
Retweets also appear in the followers page of this user like below,  
From the followers list test1 follows test3, test6 and test7.

test3mainpage: It is updated with retweet of test1 along with the tweet info which was retweeted.





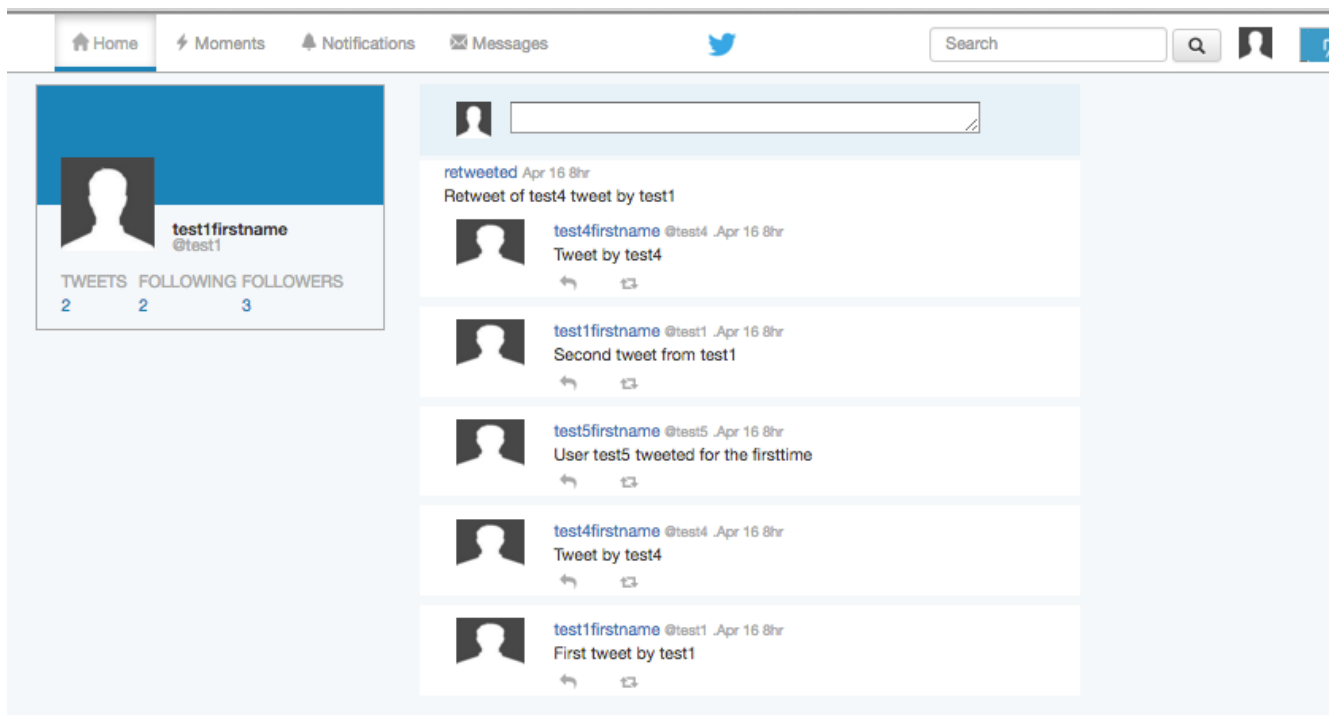
The tweet is displayed in order according to time like below, when test3 tweets another tweet, the retweet by test1 goes below the new tweet.



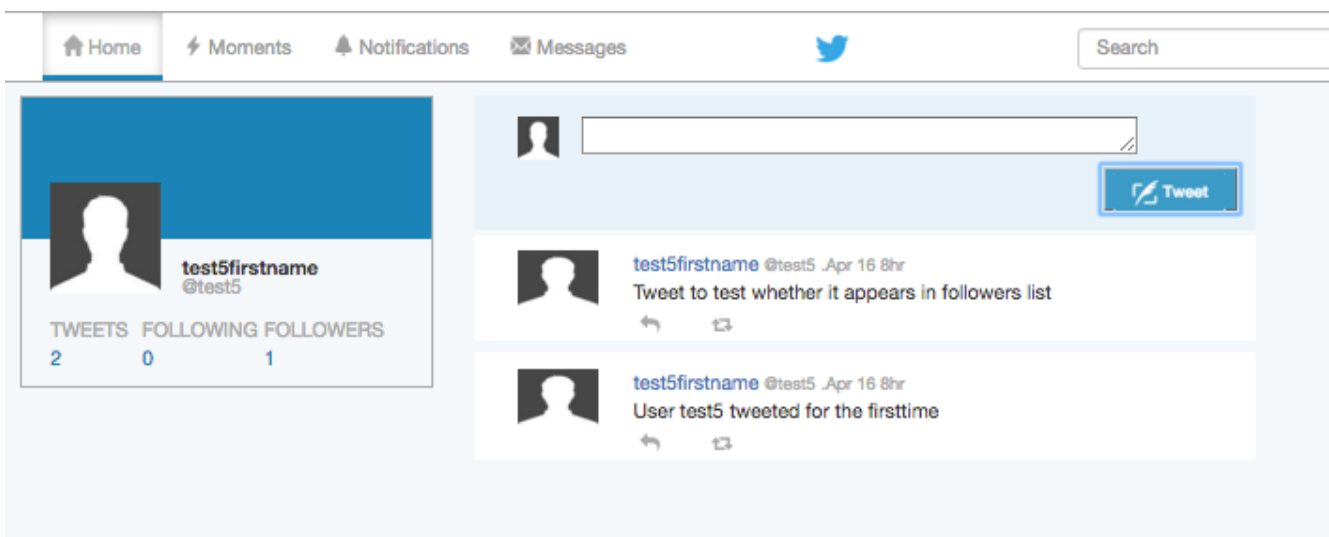
### 3. Twitter feed functionality:

In this application user is shown the tweets of all the users he is following with an option to retweet like below. The tweets are shown in the order of time .

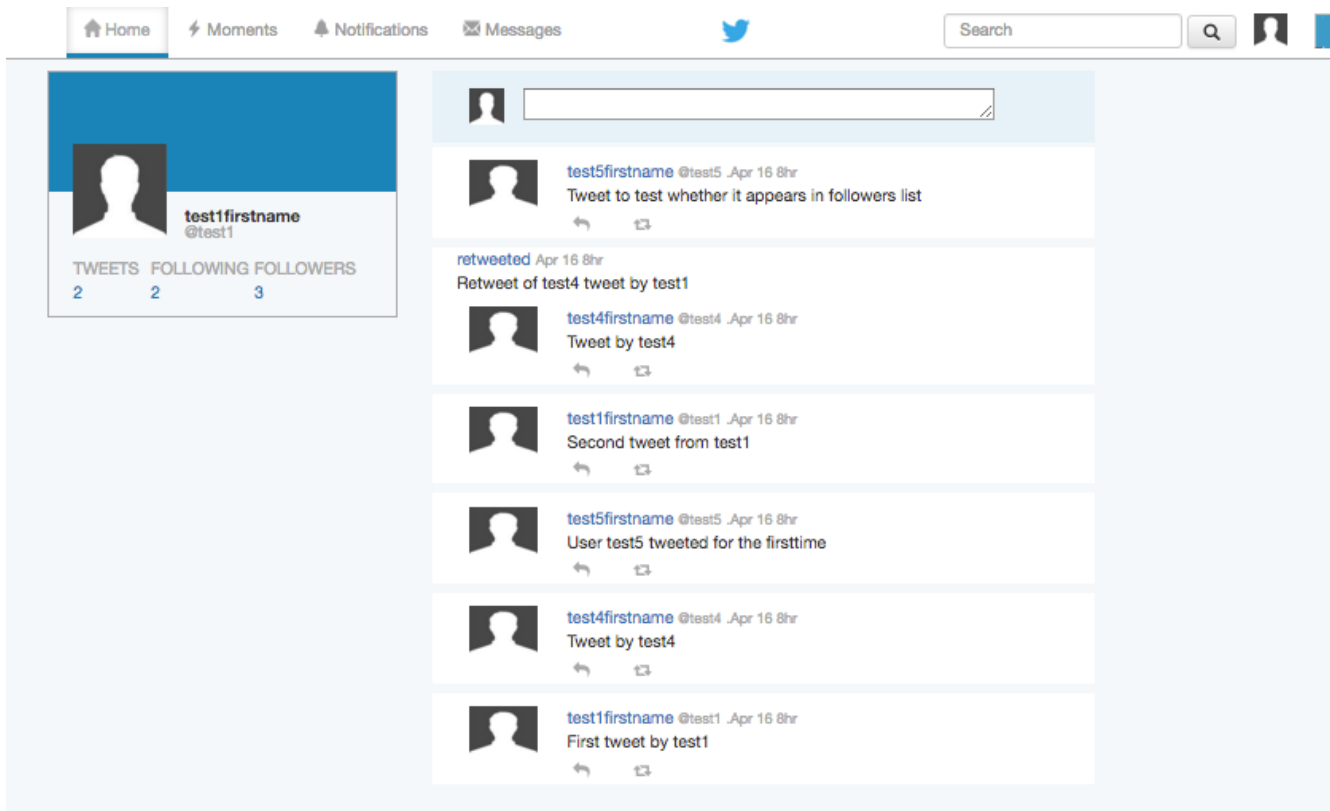
test1 is following test4, test5. The tweets related to them will be displayed in test1 page like below.



Let test5 tweet new tweet "Tweet to test whether it appears in followers list" like below,



This new tweet appears in test1 tweets feed like below, along with option to retweet

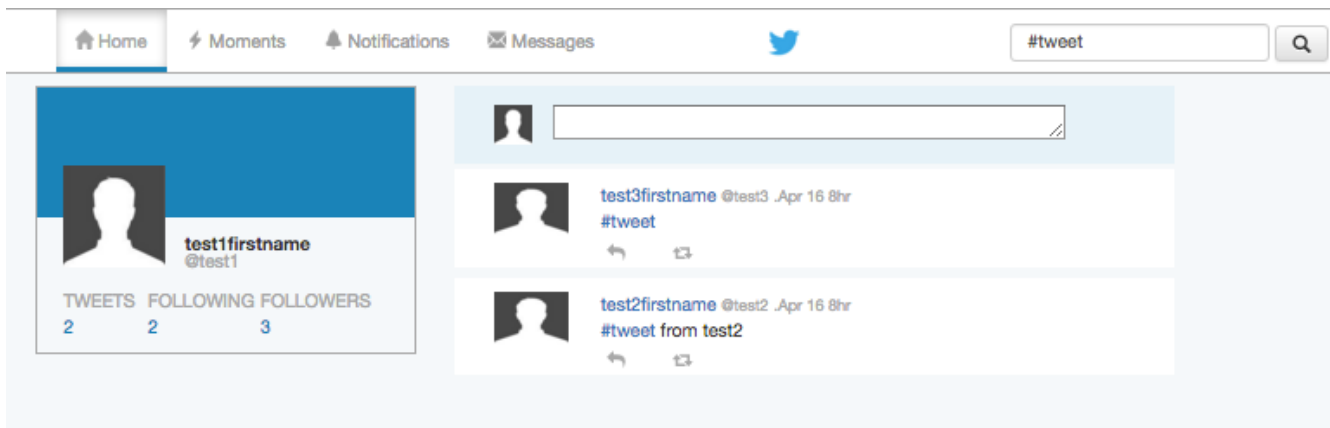


## 4. Hashtag functionality

### In search:

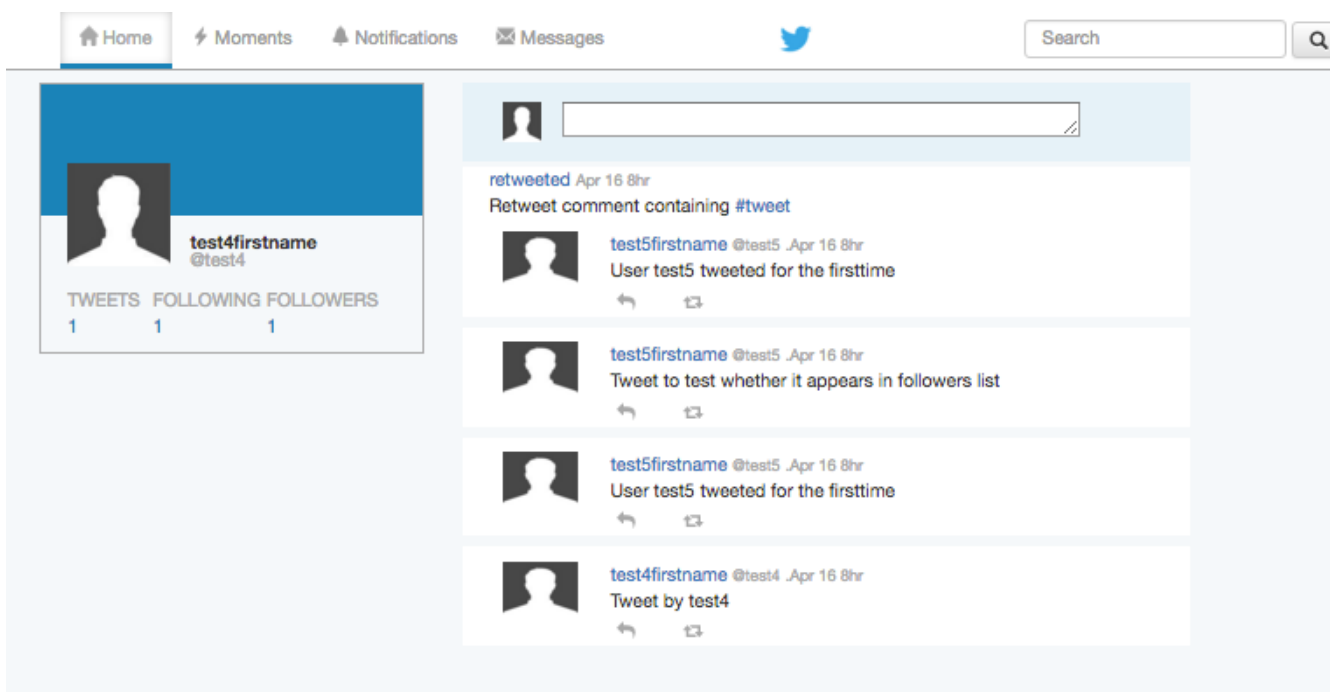
If user inputs the “#<text>” in the search box displayed in the navigation bar , all the tweets and retweets which contains the given keyword will be displayed on the homepage of user.

From test1 account lets search for “#tweet”, all the tweets related to it will be displayed like below.

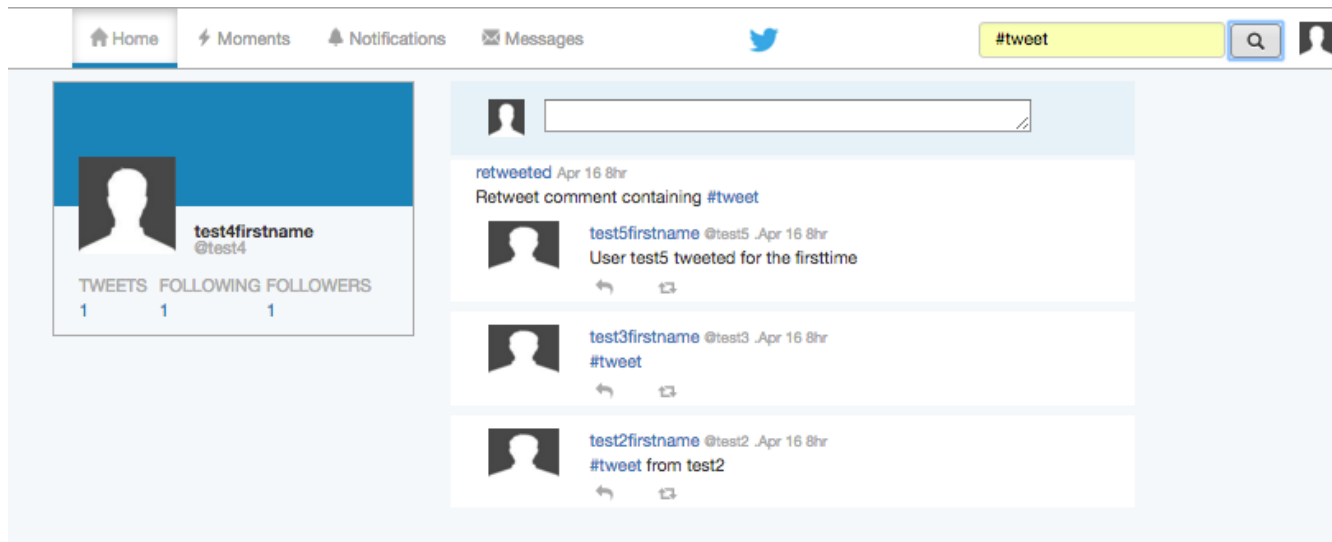


Though the test1 does not follow test2 but as it contains the #tweet in one of its tweet it is displayed in test1 tweet feed.

Lets make test4 retweet test5 tweet like below where test4 retweet comment contains "#tweet" like below

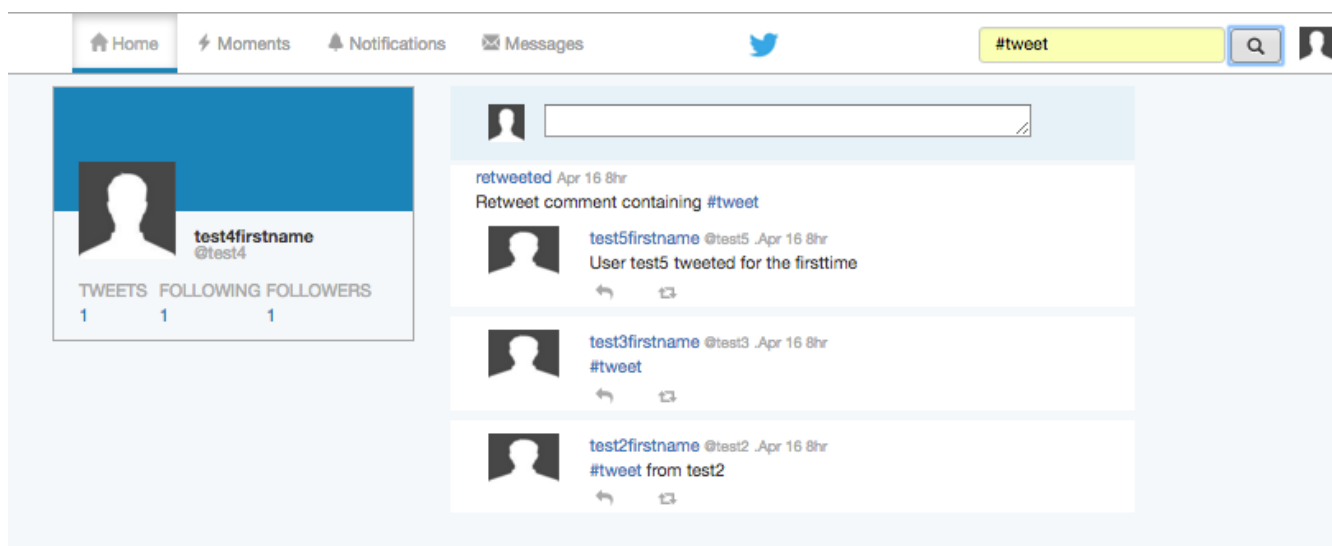


If we search for #tweet , it displays all the tweets including retweets containing hashkeyword like below.



### Hashtag functionality from tweets:

hashtagkeywords will be displayed in the tweet feed as a link , when it is clicked they are displayed in the search box and related hashtagtweets are displayed on user tweet feed area like below and they are shown based on the order of time the tweets created.



## 5. ConnectionPooling:

Connection pooling is implemented inside mongo.js file of routes folder.

With the help of connection pooling the system was able to handle multiple requests at the same time.

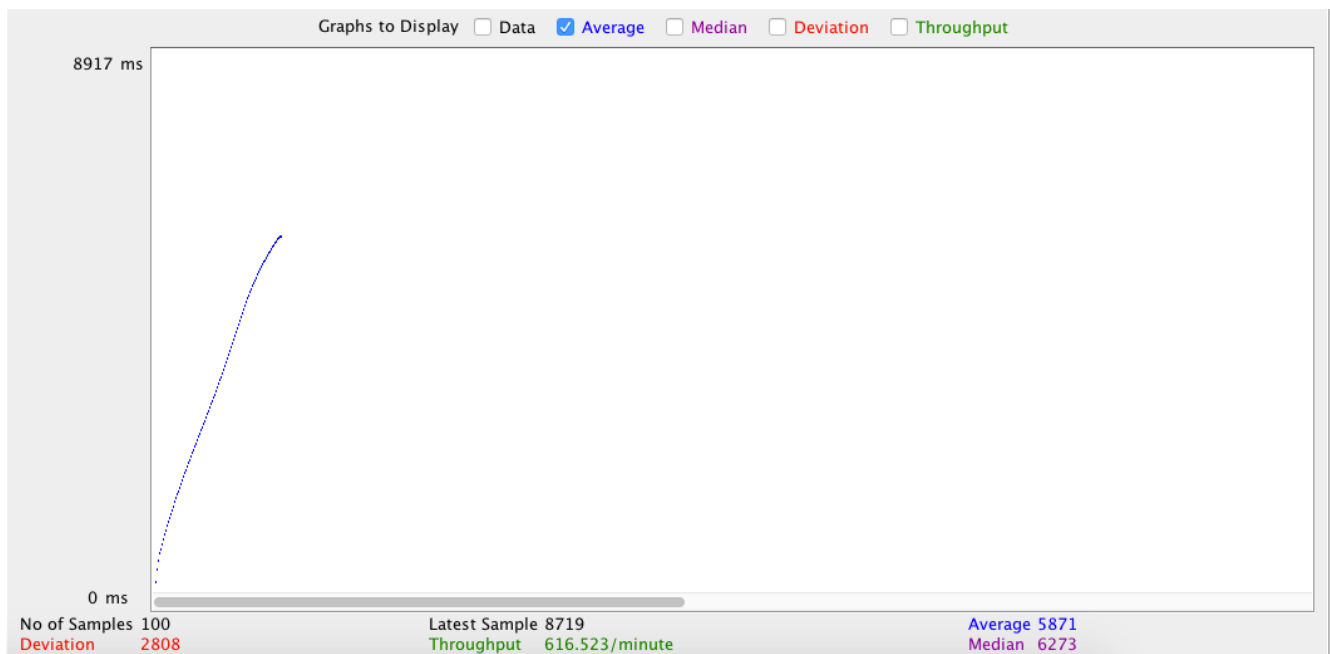
With connection pooling as the connections are not created each and every time whenever connection object is required so time taken is less compared to other. The pool is initiated once and the db object which was created is used and sent back to the pool after usage.

Initially without connection pooling the request were not able to handled as so many connection pools are opened using mongo.client. The maximum number of users the application was able to handle was 204 without connection pooling .

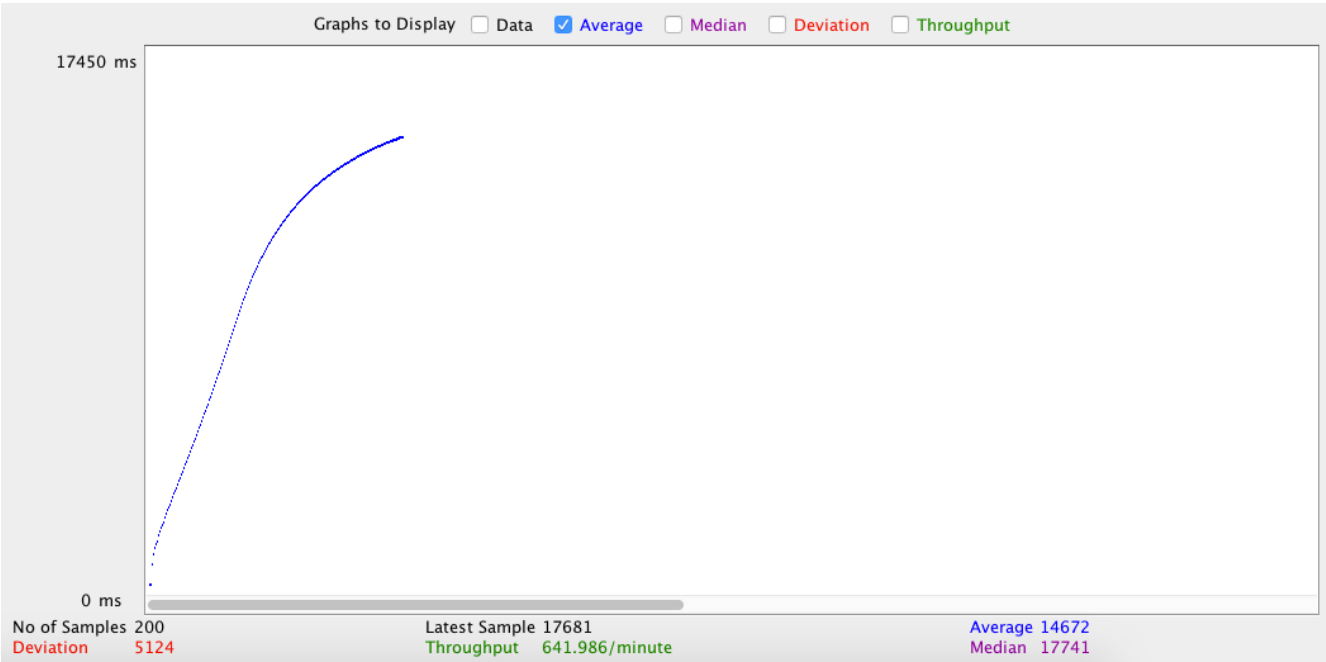
The performance is increased using connection pooling.

### Performance tests without Rabbitmq:

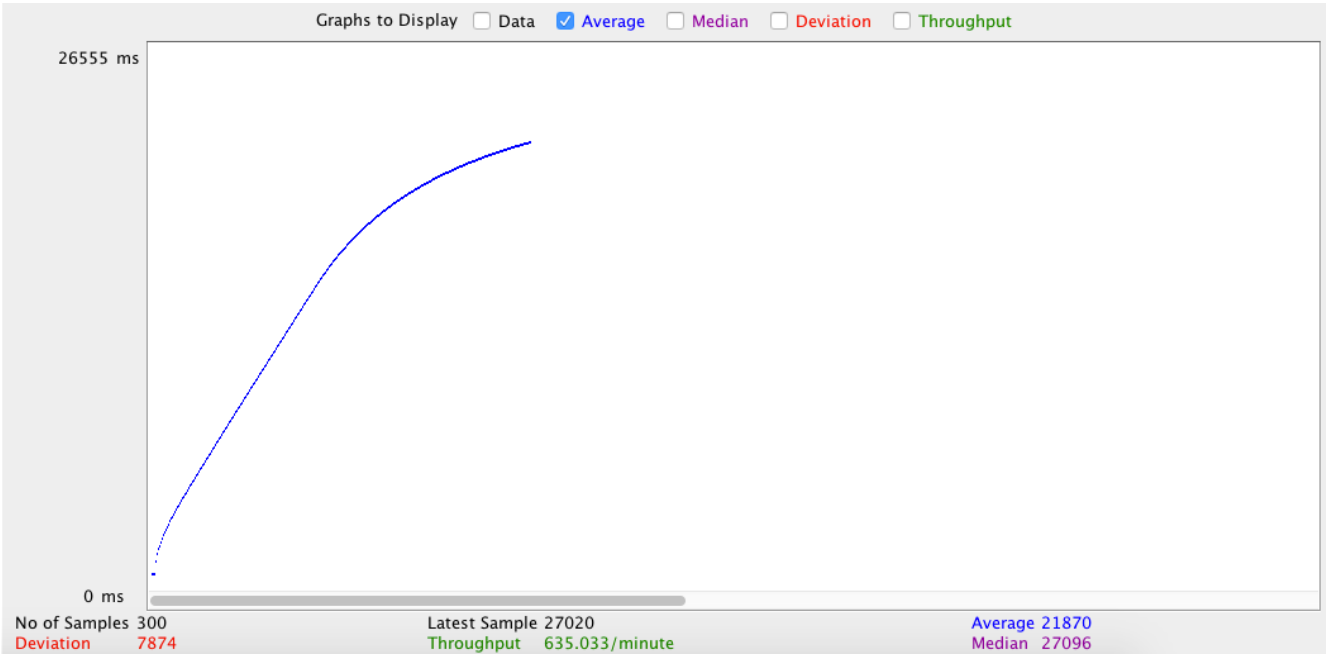
100 users:



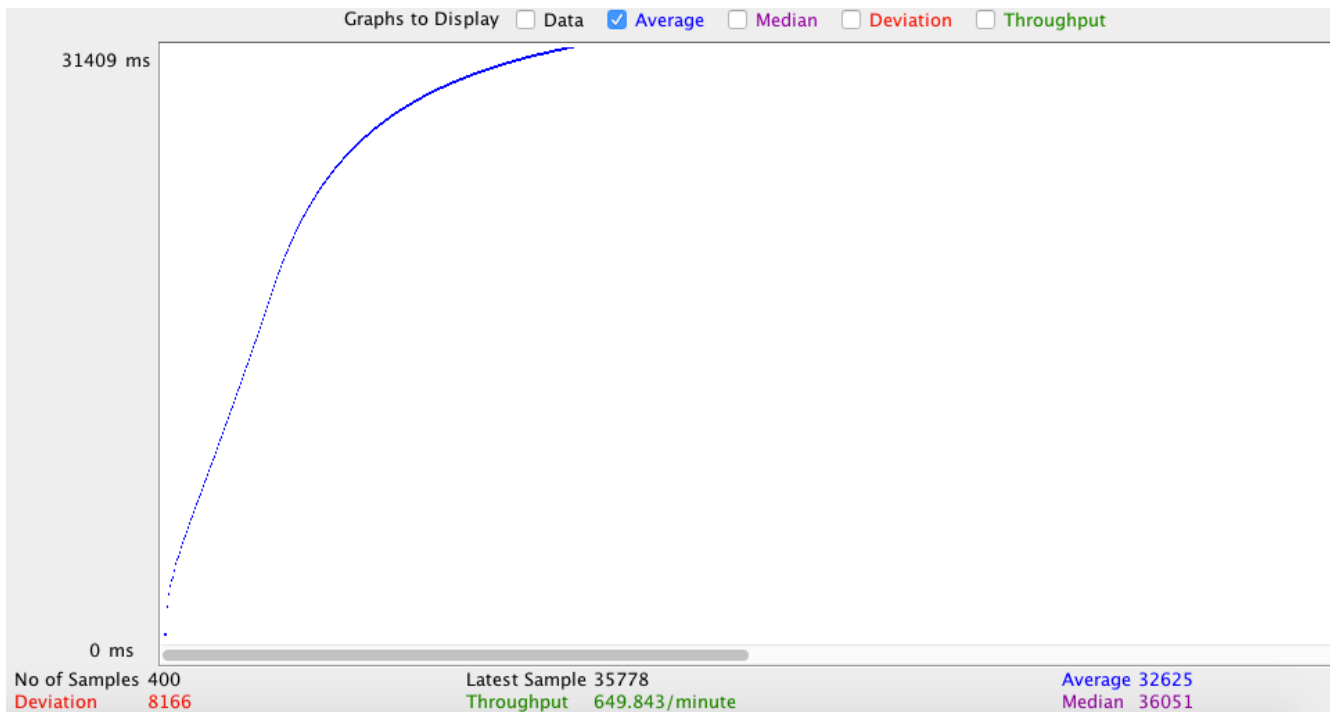
200 users:



300 users:



### 400 users:



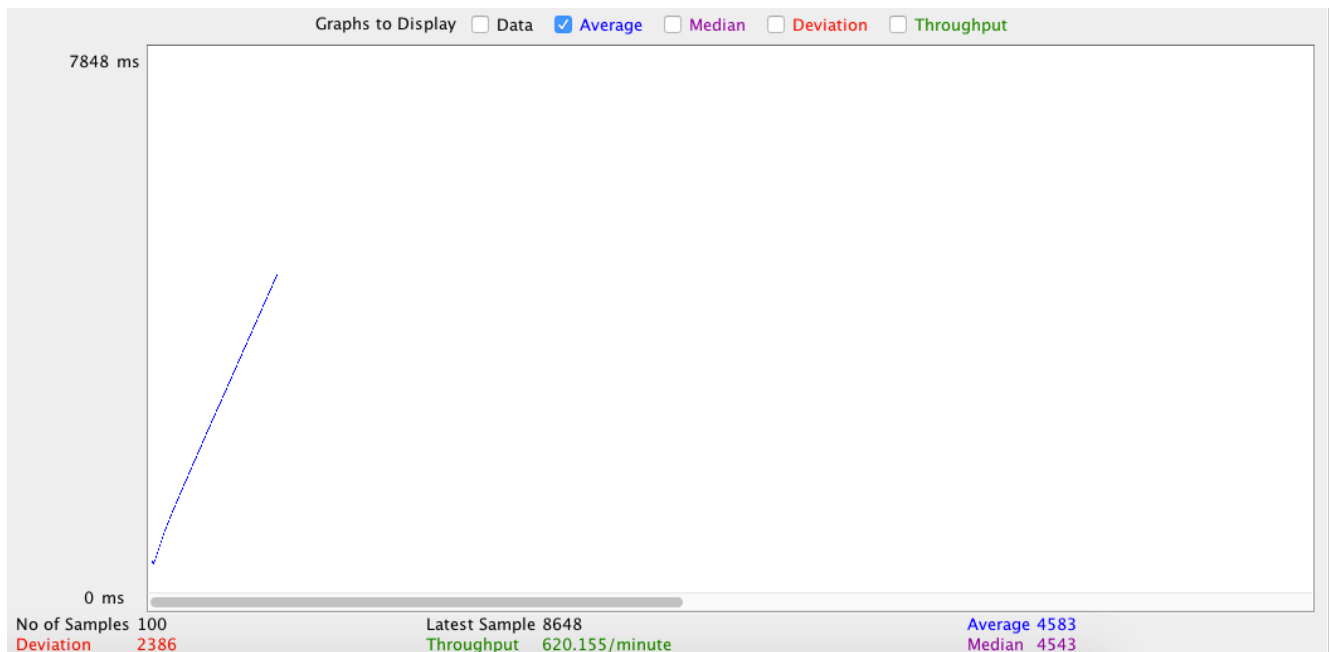
### 500 users:





## Performance tests with Rabbitmq:

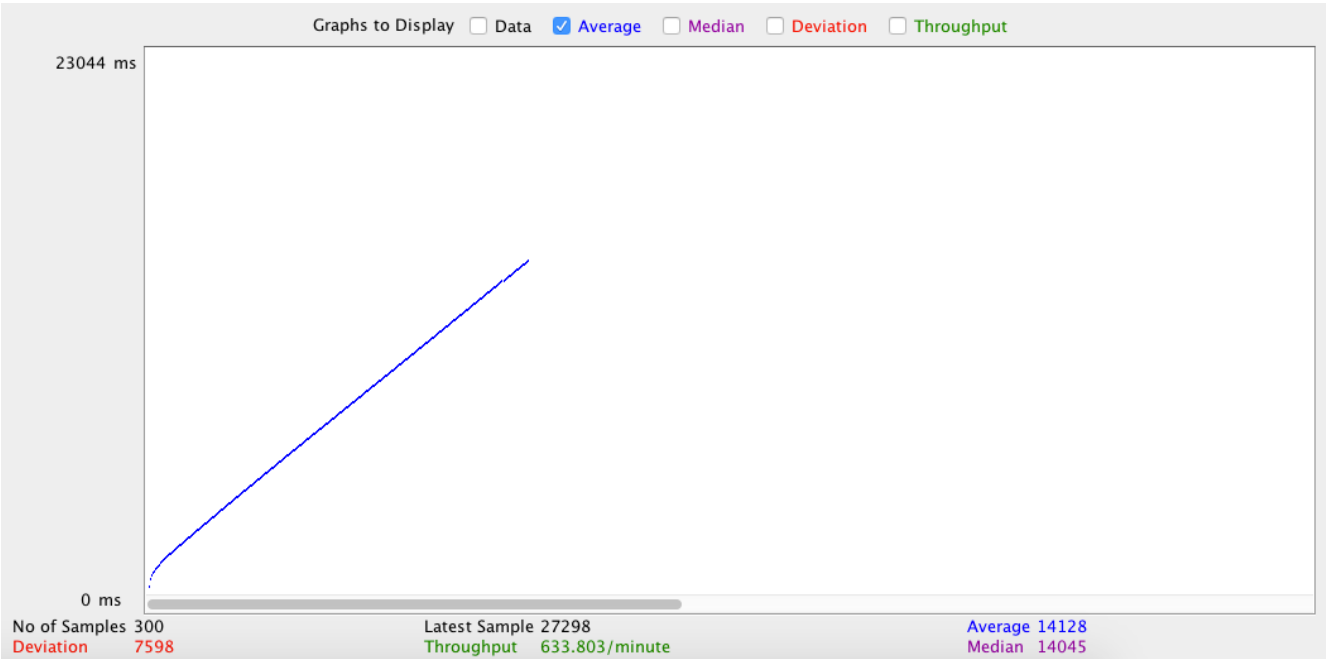
100 users:



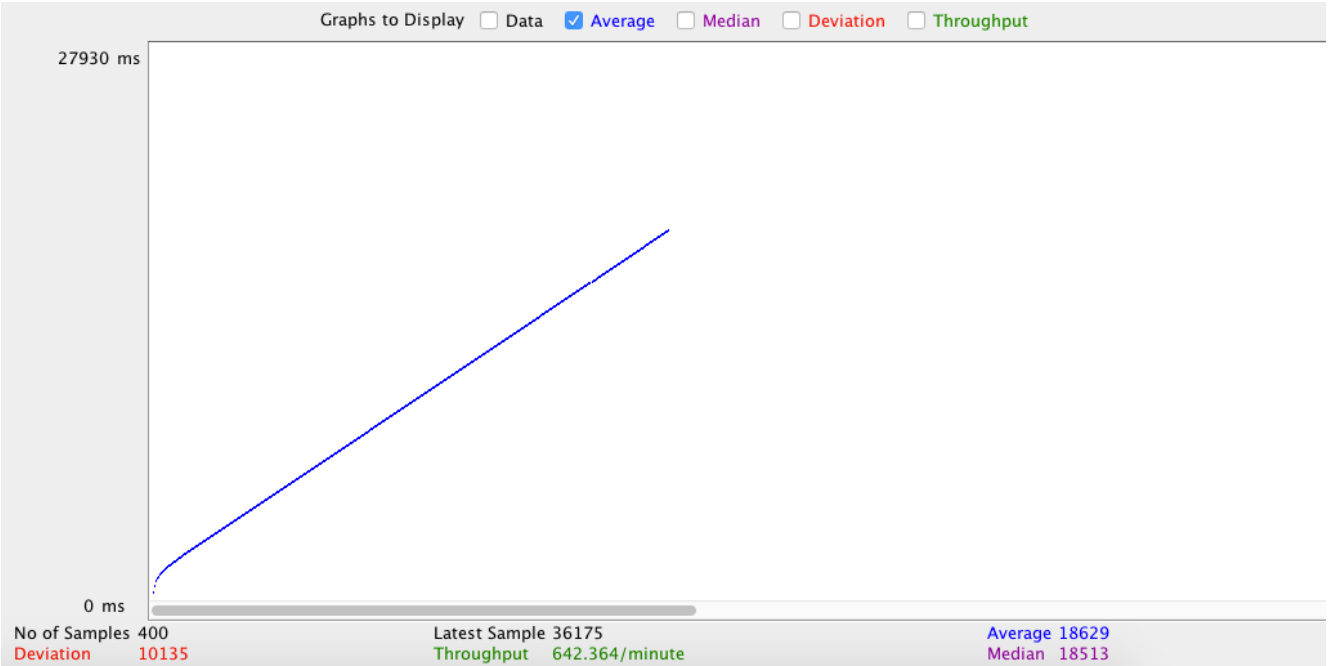
200 users:



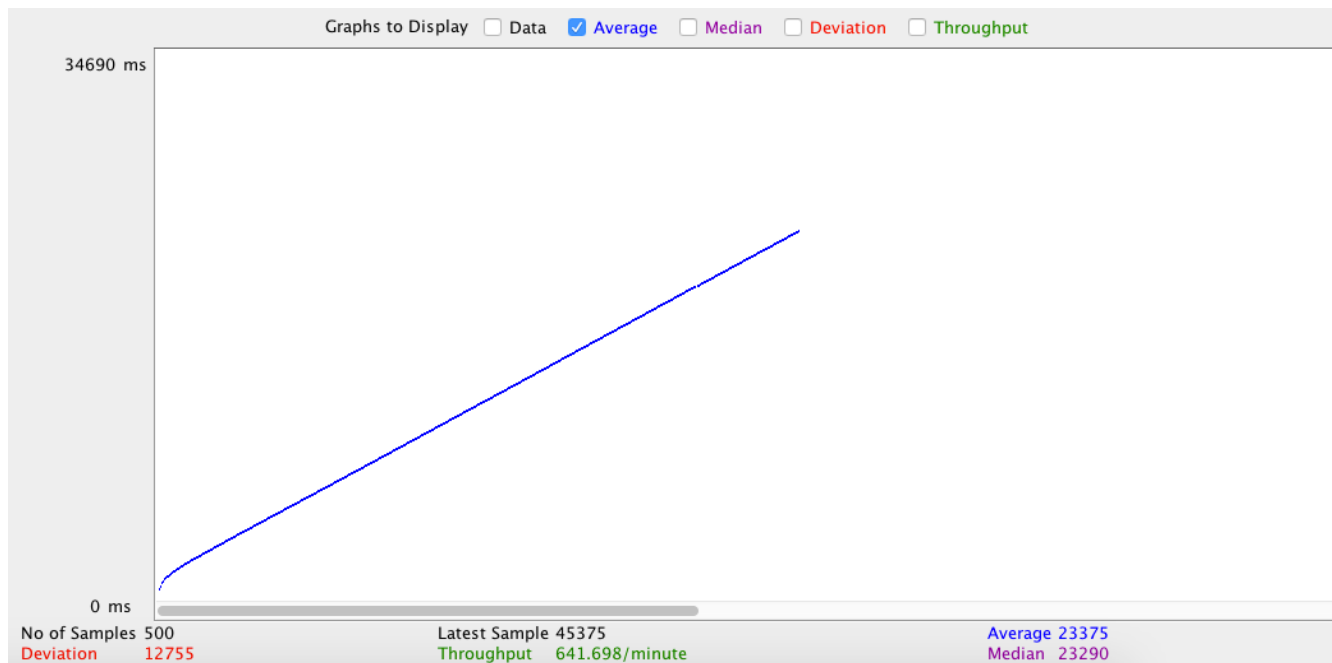
300 users:



400 users:



500 users:



Mocha tests:

```
Madhuris-MBP:TwitterClient-Lab2 Madhu$ node node_modules/.bin/mocha test/mocha_test.js
```

#### http tests

- ✓ signup page error if user signups with already existing username (41ms)
- ✓ signup page error if user signups with already existing email
- ✓ login page error if user logins with unexisting email
- ✓ following page should be displayed after successfull login (99ms)
- ✓ followers page should be displayed after successfull login (100ms)
- ✓ follow page of given user using dynamic url should be displayed if user exists (142ms)

6 passing (426ms)

---

## PART 2:

### Questions:

**Explain what performance change RabbitMQ provides? Elaborate on the results of throughput with and without using RabbitMQ. If you find any increase/decrease in the throughput, explain the reason for the same.**

-

	Without Rabbitmq- throughput	With Rabbitmq- throughput
100 users	616.5/min	620/min
200 users	641.9/min	639/min
300 users	635/min	633.8/min
400 users	649.0/min	642.36/min
500 users	646/min	641.69/min

Rabbitmq increases scalability as multiple requests are handled using multiple queues from single server. The client requests which are made to server are handled by rabbitmq using queues. There are different queues to serve different requests from the server. Rabbitmq server handles requests by asynchronous message transfer. The scalability will increase as different queues can be added to Rabbitmq to handle different requests.

When rabbitmq is introduced the throughput is increased. This increase is obtained as multiple queues are used to handle various requests from the user. The request which is obtained from user is handled to a queue by publisher in rabbitmq based on the message type it obtained. The request is then handled by the respective queue.

In the above table, the throughput is decreased when I used rabbitmq. The decrease is because using rabbitmq creates more responses. As more responses are handled the

---

throughput decreased. The actual goal cannot be achieved if we are using less number of users. But if the number of users are increased rabbitmq provides increases in throughput and decrease in response time.

**Explain the strategy used in implementing Sessions in this Lab. Compare your Sessions strategy with default express Sessions. Describe which Strategy is better.**

In this lab to implement sessions , I used mongodb to store the user sessions. With express framework cookie parser module the system creates cookie using default express session. System then stores these sessions in mongo-store. Connect-mongo is used as the mongo store, which stores the session values in the database .

The different between the default express sessions and this session strategy is that with default session strategy, the availability of the session is just limited to the application but when we use session state provider, the sessions which are stored using connect-mongo in backend can be used independent of application. The scope of sessions using this strategy has been increased. The scalability will also increase as system doesn't have to use short scope application level session.

Therefore, sessions that we store in the backend using connect-mongo is better.

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

I use mongodb in my application to store unstructured data like tweets. The videos images that are to be used can be stored in mongodb. When application needs to grow big , MongoDB will be a better option rather than to use mysql as mysql table performance decreases when the data crosses 5-10gb per table. In present twitter application I created , the tweets accepts just the tweet text but what if there is a requirement to change tweets table schema ? In scenarios like this I prefer to use mongodb than mysql.

---

As the data increases , and if the number of joins required to get results are more, than it is better to use mongodb than mysql .Because in mongodb, we can use embedded documents inside a collection which in turn helps to reduce the overhead of joins.

For example, in order to get following list , the application joins two tables users and following table. Use of join is required. If user is following millions of other users the overhead of joins will be high and this can be avoided using embedded documents inside the user collection to store the id's of the following people.