
CMPE 273 - Lab 1 Report

Madhuri Motoori - March 19, 2016

Introduction:

Goals:

The goal of this project is to develop REST services for Calculator and Twitter application using Node.js , Angular JS, HTML5 handling proper validations and exceptions. Twitter application is aimed at developing a fully functional front end application using Bootstrap, Angular JS and back end application using Node.js and Mysql.

Purpose of this project:

- To gain practical experience with Node.js , Angular JS, HTML5, Bootstrap and understanding the concepts related to each and focusing on the performance.
- To develop server which demonstrates stateless web services and client to include all those functionalities implemented by web services.

Calculator:

- Server includes simple calculator operations like Addition, Subtraction, Multiplication and Division and client includes all those functionalities provided by web service.

Components:

- Server is implemented using NodeJS
- Client is implemented using HTML5 and Bootstrap.

Twitter:

- Server includes various modules related to basic user functionalities like signup, login , logout , profile update, showing tweets and retweets of user and his followers, able to follow other users, able to show tweets based on hashtags and implementing connection pooling.
- Client includes all functionalities provided by web service.

Components used in twitter application:

- In order to develop twitter application front end UI framework Bootstrap is used which is a collection of ready to use HTML, CSS and javascript templates of UI components.
- Angular JS which is client side javascript framework is used
- NodeJS is used in the backend .
- Mysql is used for data storage.

Calculator:

System Design:

- To implement the above functionalities addition, subtraction, multiplication and division , I used stateless web services to call asynchronous method to give result for the above functionalities.

Initial request on the server opens a view to the user which displays a form to provide input. This form contains fields to input first number , second number , operation that needs to be performed on provided two number and Calculate button.

When calculate button on view is clicked it calls a post request “/afterSubmit” on the server which calls “afterSubmit” in “cal.js”. This method calls “calculatorResult” method in calculator.js module which is an asynchronous method. The user input is passed to this asynchronous method. Exceptions and errors are being handled in this function which returns error if any error occurs else the error is set to null and result is passed.

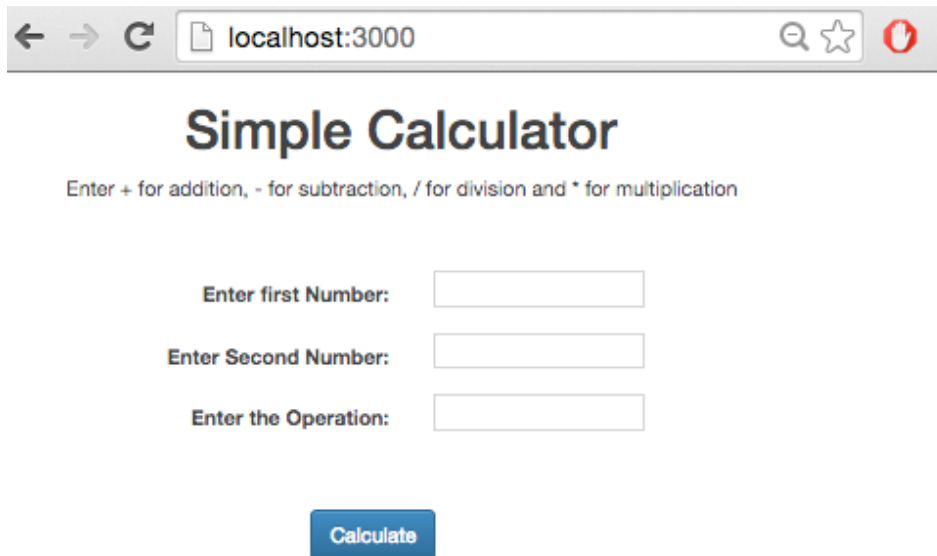
Whenever the event occurs with “calculatorResult” module inside afterSubmit method the view is displayed with appropriate message.

Exceptions and errors being handled:

- If operations other than +, -, * and / specified by user.
- If input for all the fields are not provided by user
- If input for first, second number fields are not numbers.
- If operator is division and second number provided by user is zero.

Screenshots:

Initial request to the server:



← → ↻ localhost:3000 🔍 ☆ 🛑

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

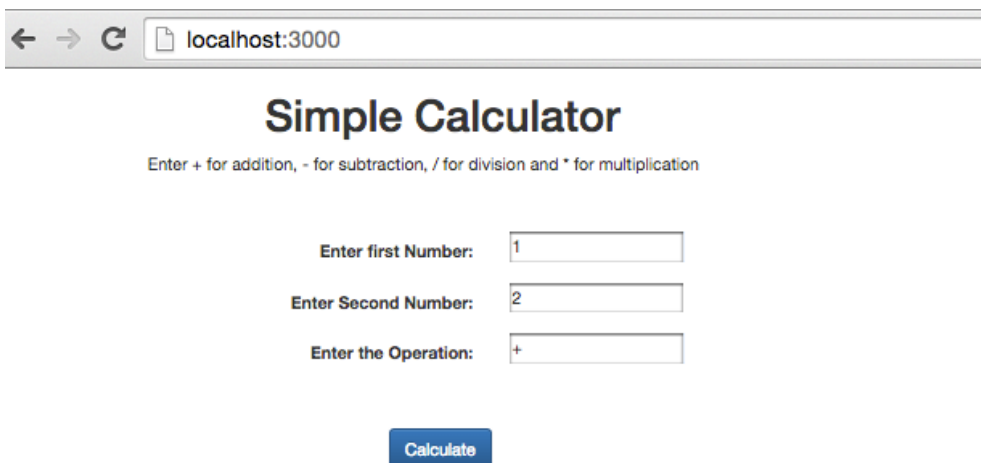
Enter Second Number:

Enter the Operation:

Calculate

Addition:

Input two numbers 1, 2 and operation as +



← → ↻ localhost:3000

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

Calculate

Result: Click calculate button

 localhost:3000/afterSubmit

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:


Enter the Operation:

The result of $1 + 2 = 3$

Calculate

Subtraction:

Input two numbers 1000 , 100 and operation as -.

 localhost:3000

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

1000

Enter Second Number:

100

Enter the Operation:

-

Calculate

Result: Click Calculate button

localhost:3000/afterSubmit

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

The result of $1000 - 100 = 900$

Calculate

Multiplication:

Input two numbers 99 , 234 and operation as *

localhost:3000

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

Calculate

Result: Click Calculate button

localhost:3000/afterSubmit

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

The result of $99 * 234 = 23166$

Calculate

Division:

Input two numbers 99 , 11 and operation as /

localhost:3000

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

Calculate

Result: Click Calculate button

localhost:3000/afterSubmit

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

The result of 99 / 11 = 9

Calculate

Error conditions:

1. Input two numbers 11, 99 and any other operation other than specified four example &

localhost:3000

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

Calculate

Result: Click the Calculate button

localhost:3000/afterSubmit

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Error: Please enter the given operations

Enter first Number:


Enter Second Number:

Enter the Operation:

Calculate

Error shown if user does not input value for the fields:

Input one numbers 11 and operation as “-” and don't provide second number

 localhost:3000

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

Calculate

Result: Click the Calculate button

localhost:3000/afterSubmit

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Error: Please enter value for numbers


Enter first Number:

Enter Second Number:

Enter the Operation:

Calculate

Error shown if user does not input numbers for first Number and second Number fields
Input first number as “and” and second number as 2, operation as “-”

 localhost:3000

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

Calculate

Result: Click the Calculate button

localhost:3000/afterSubmit

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Error: Please enter a number for first and second numbers

Enter first Number:

Enter Second Number:

Enter the Operation:

Calculate

Error is shown if user provides second number as 0 and operation as /
Input first number as 100 and second number as 0, operation as /

localhost:3000

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Enter first Number:

Enter Second Number:

Enter the Operation:

Calculate

Result: Click the Calculate button

localhost:3000/afterSubmit

Simple Calculator

Enter + for addition, - for subtraction, / for division and * for multiplication

Error: Second number cannot be zero to do division

Enter first Number:

Enter Second Number:

Enter the Operation:

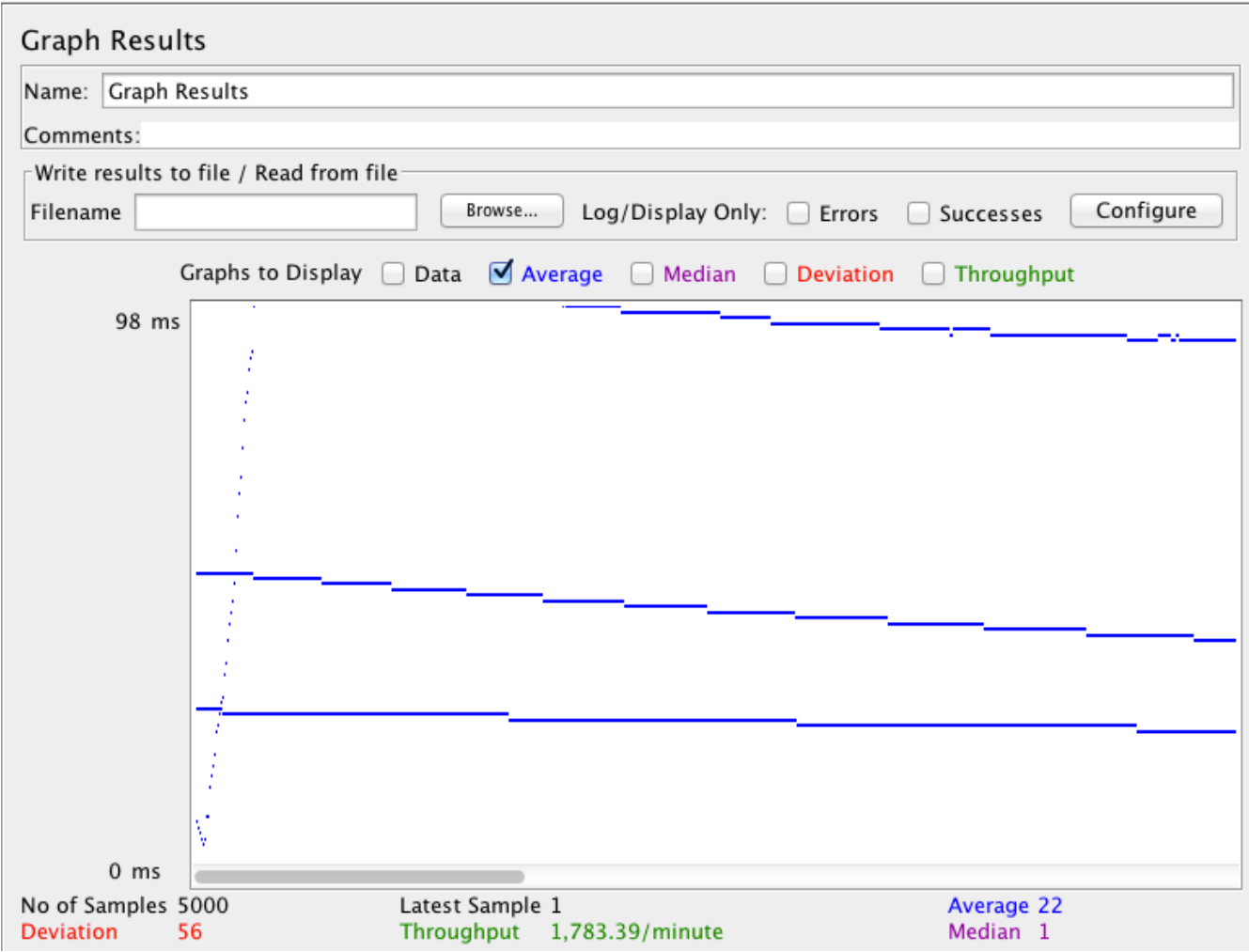
Calculate

Performance graphs:

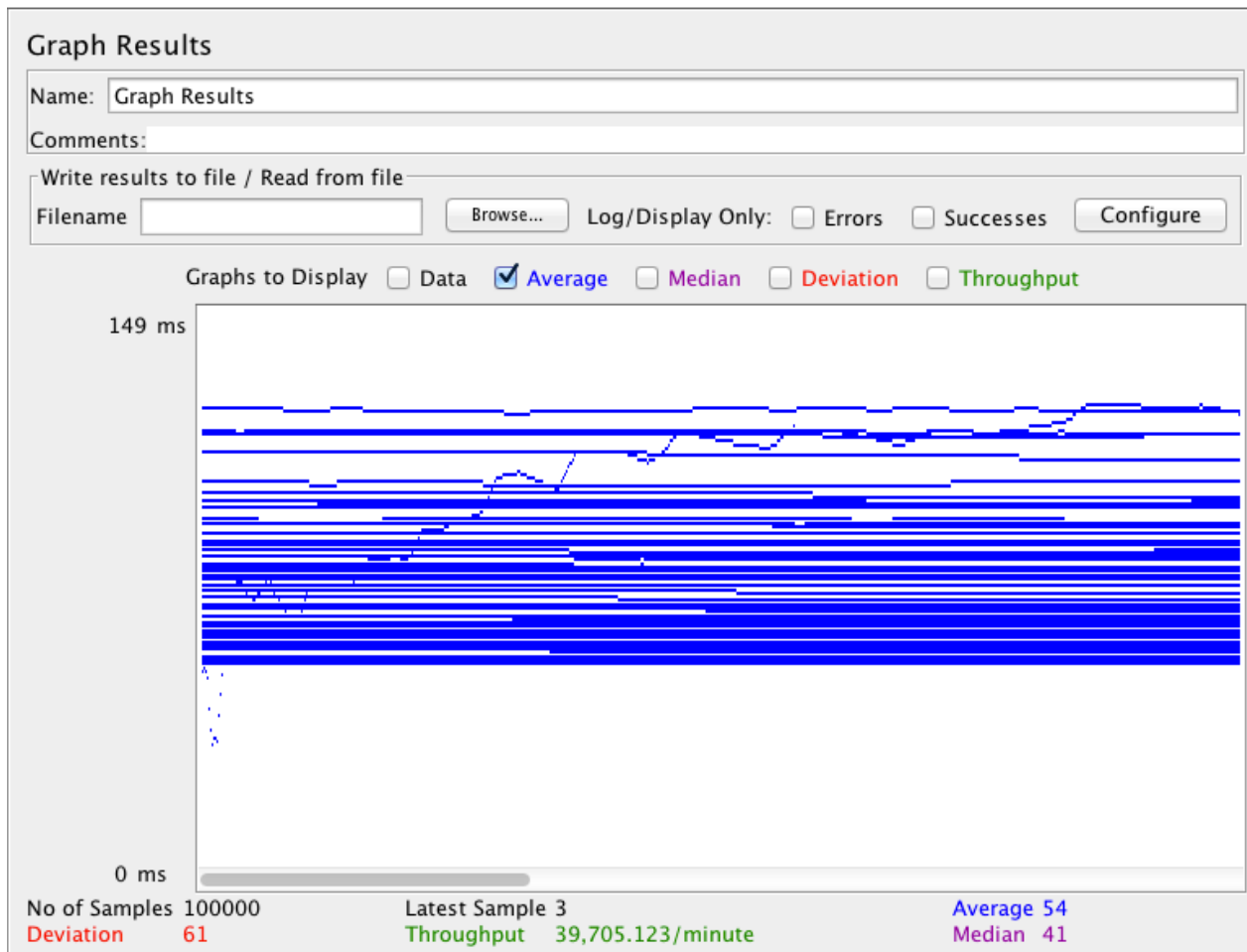
1000 calls:



5000 calls:



100 concurrent users with 1000 calls:



Compare performance:

Based on the above average values , as the number of concurrent calls increased the average time is also increased as the load on the server is increased the time to respond request might be decreased.

Twitter Application:

System Design:

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

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.

The GET and POST requests are called from angular js controller using \$http module, nodejs processes the requests and will send appropriate response based on the information provided either success, failure and error. The status codes and corresponding messages are passed as Json responses.

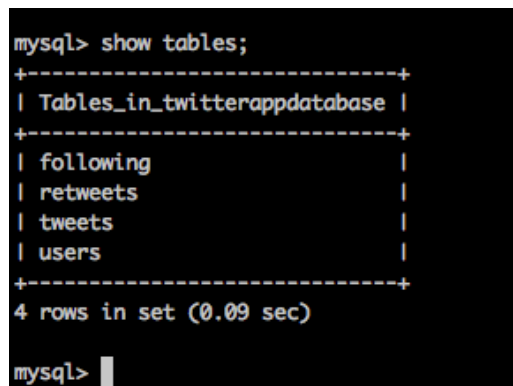
NodeJS interacts with the database in asynchronous manner to get the data required based on the request. System design of twitter application is explained below in detail using the screenshots.

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.

Databases:

The following tables are stored in “twitterappdatabase” which are used in the application.



```
mysql> show tables;
+-----+
| Tables_in_twitterappdatabase |
+-----+
| following                      |
| retweets                      |
| tweets                       |
| users                        |
+-----+
4 rows in set (0.09 sec)

mysql>
```

Users table is created to store information about user like id, username(twitter handle), firstname, lastname, email, password, gender, location, dob, phone number etc.

tweets table contains information about the tweets like id of tweet, userid who created this tweet and tweet value.

following table contains information about the following users . Using this table the information about followers and following can be found. userid and followinguserid columns are created to store the details of following user.

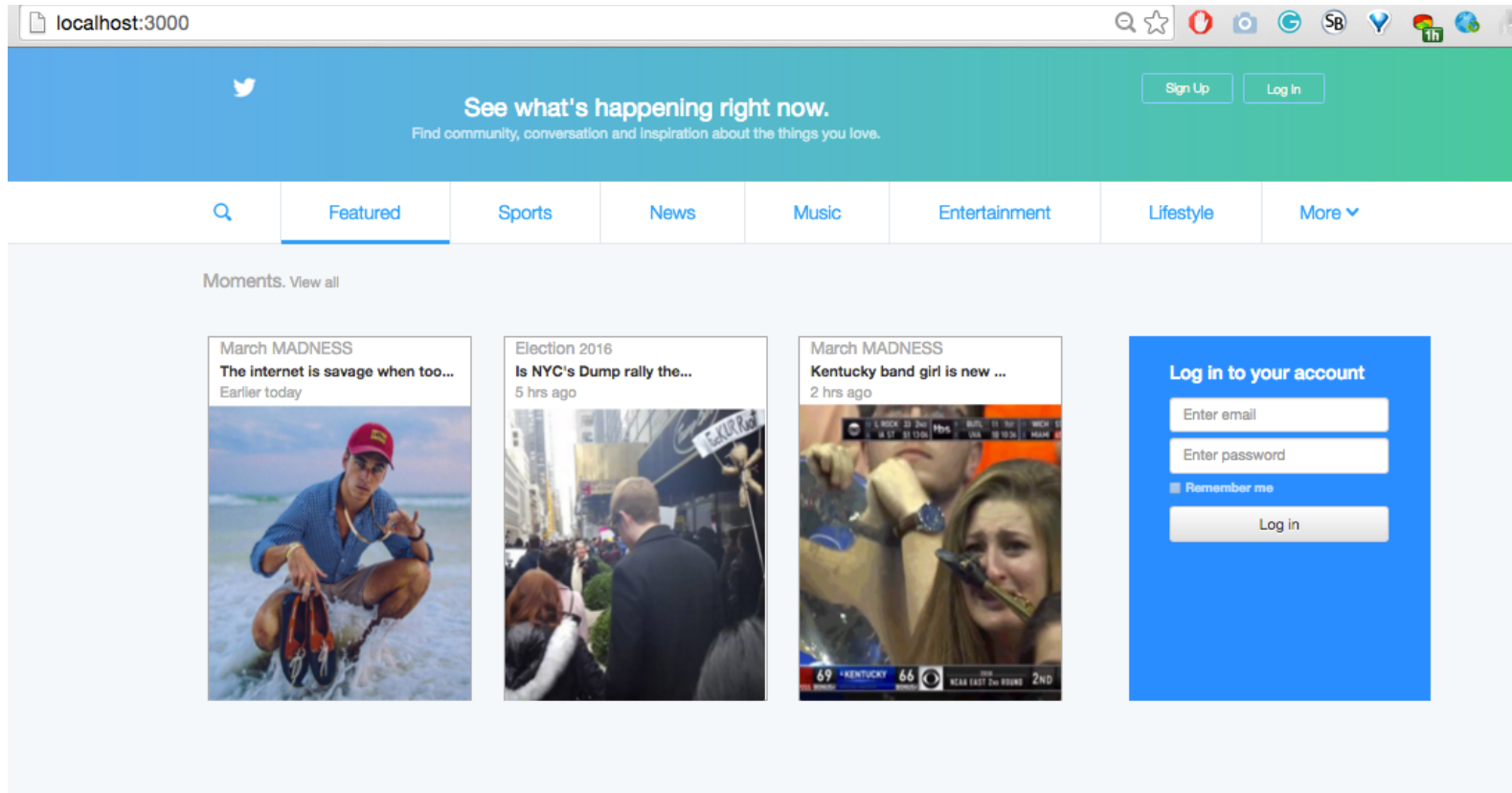
retweets table contains information about the retweets like userid of user who retweeted, id of tweet which is retweeted, retweet comment .

Date is also added to retweets and tweets table to show the tweets including retweets in order based on time.

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

0/signupPage



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.

 Please fill out this field.

☒ Male ☐ Female

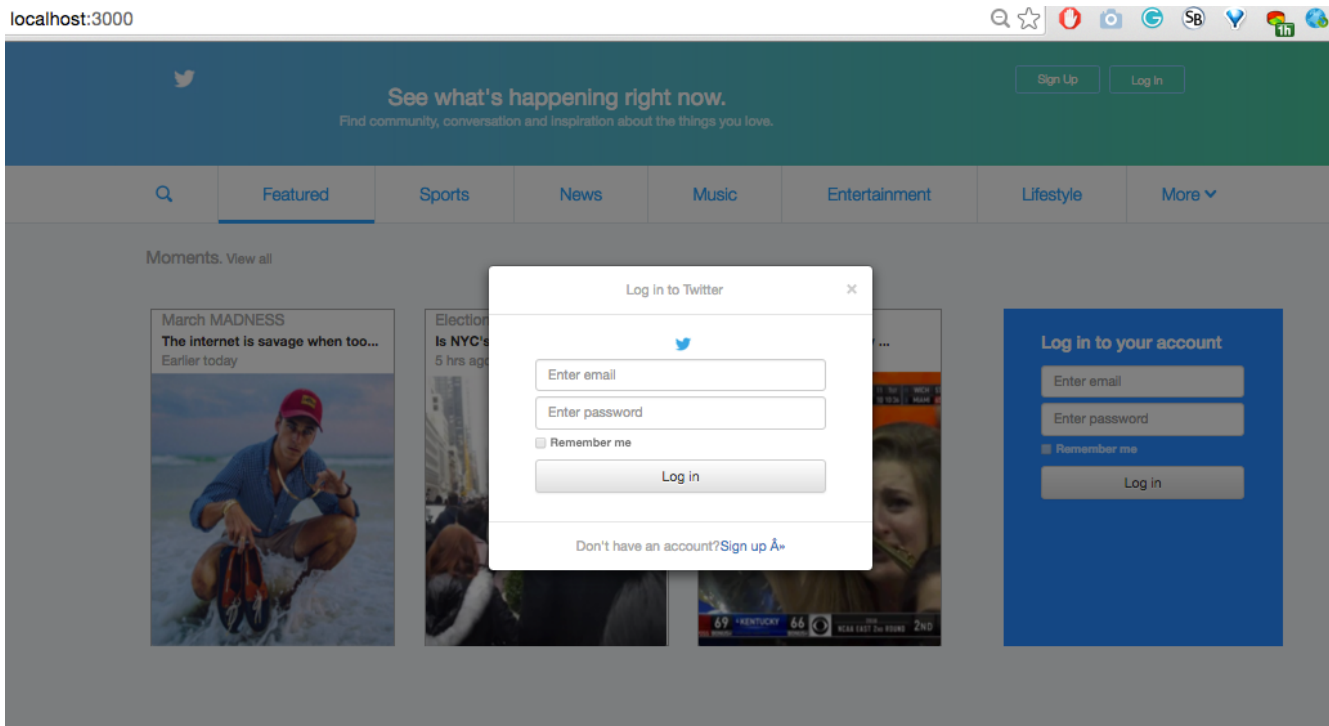
☐ Remember me

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.

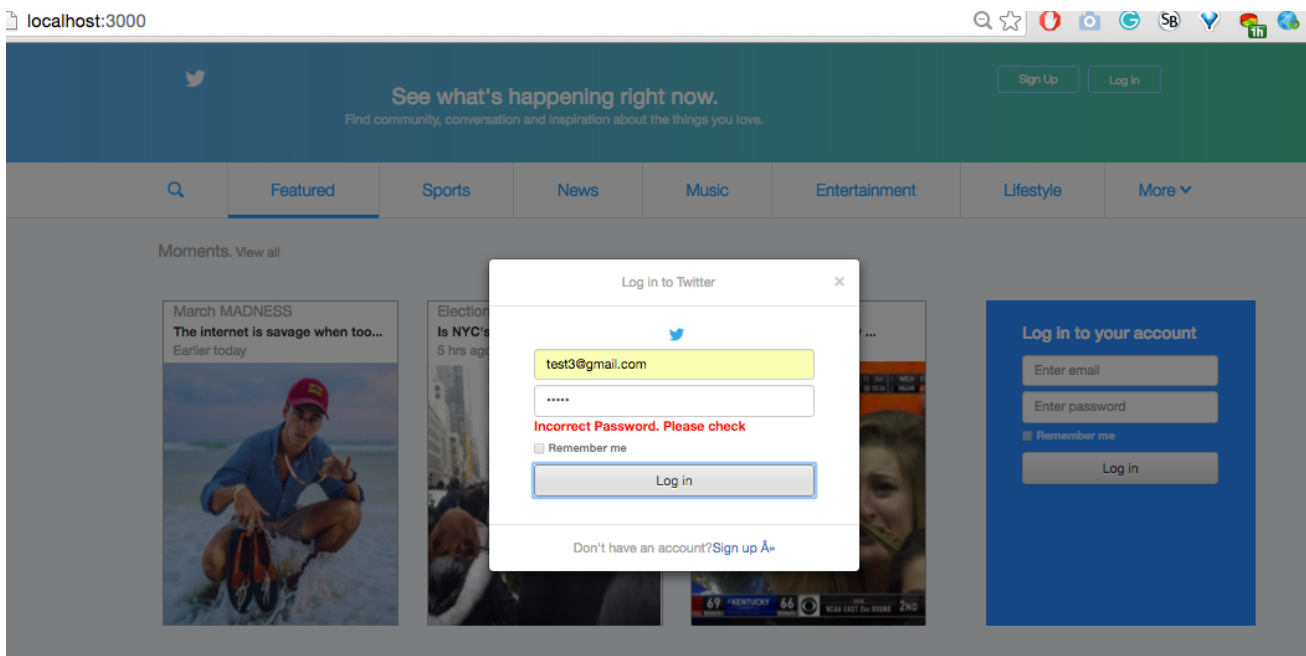
LogIn:

Login modal from main site page:

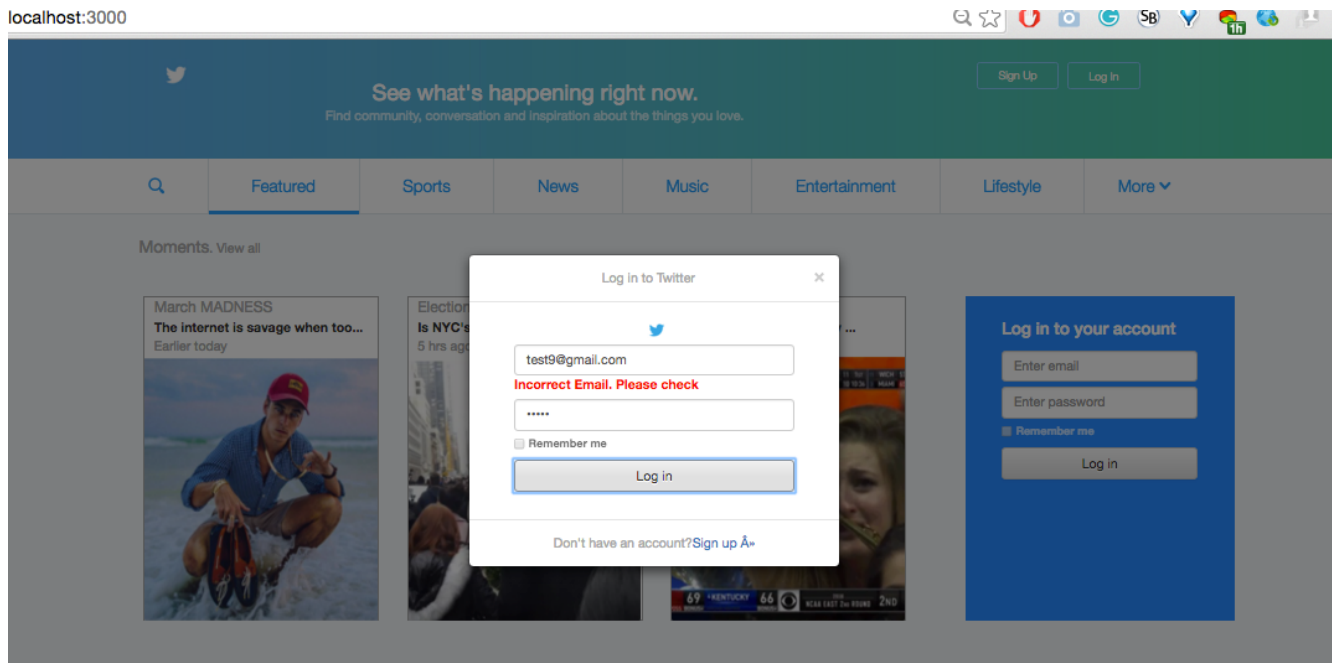
If we click the login button from main page , the following modal appears



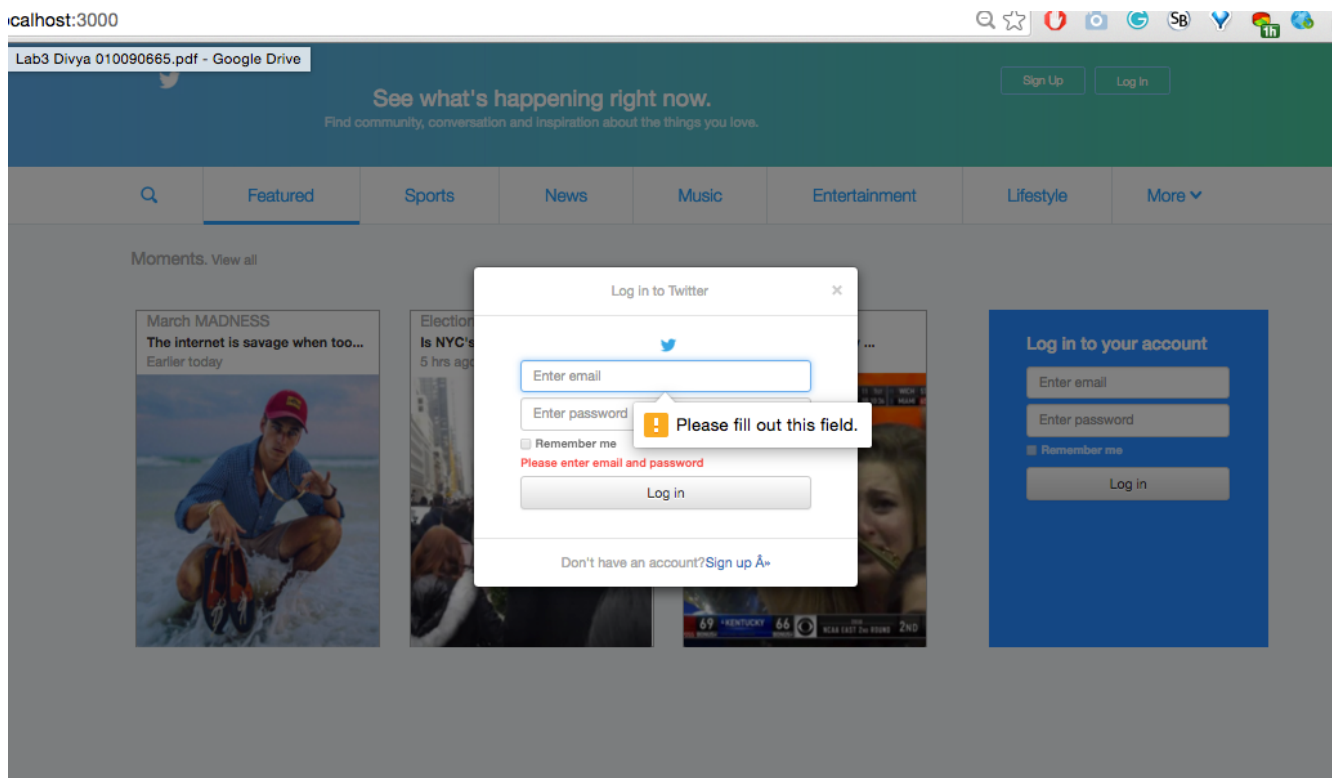
Error cases: If user provides incorrect password, appropriate error message is shown



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

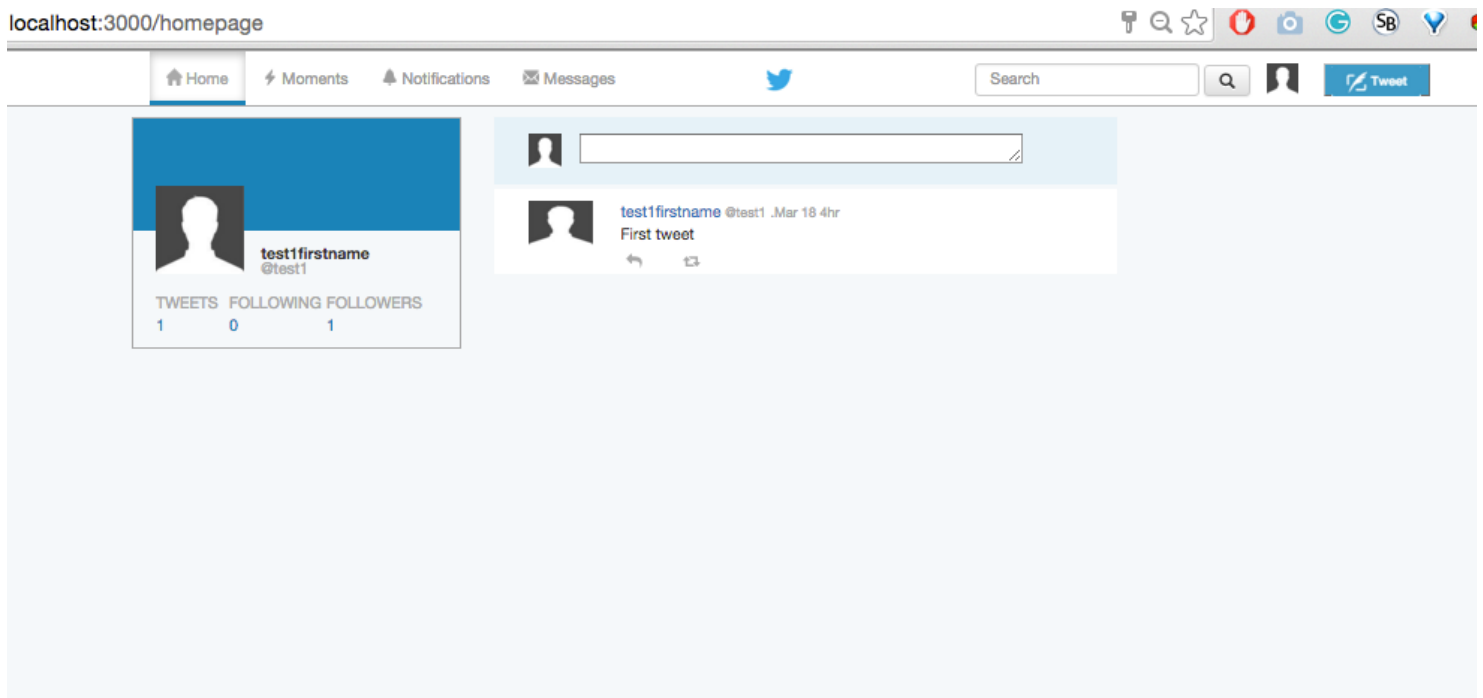


If user does not provide all the information, appropriate error message is shown



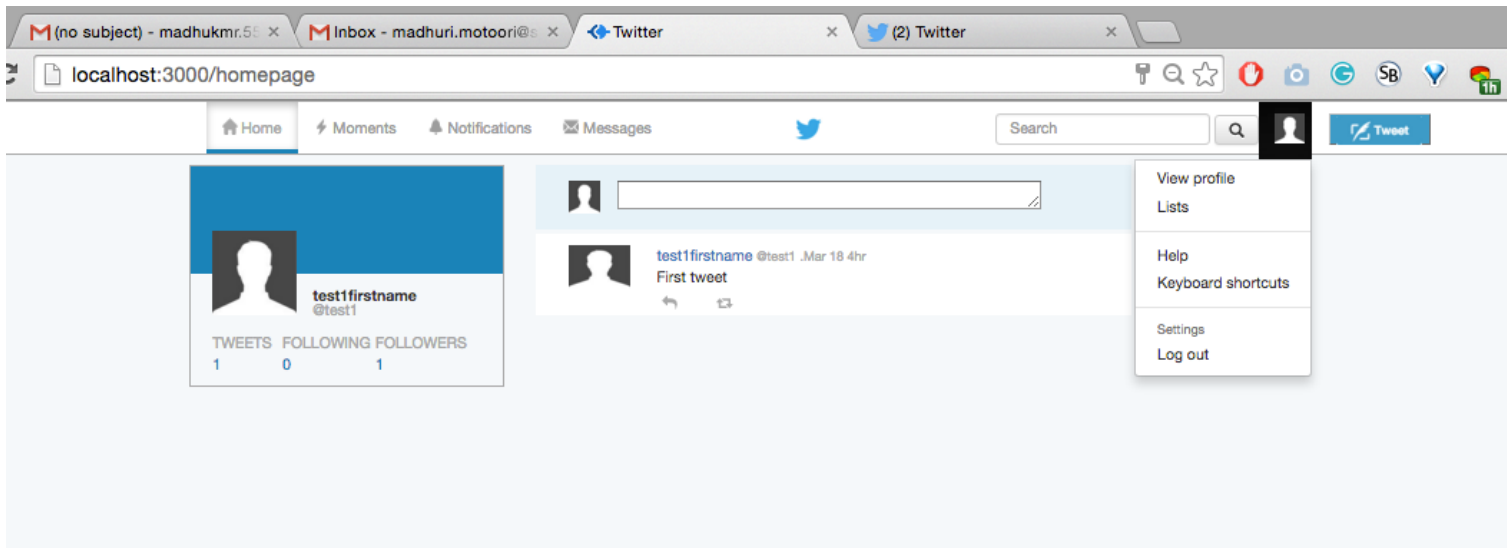
After user provides the input and clicks the Login button , using angularJS it calls post method on /login . The server redirects to loginModule.login where it checks email , password are not undefined and not empty. If not it tries to fetch data from mysql asynchronously based on email. If email does not exists it sends Json response status code and appropriate error message which will be shown by angularJS and views like above error that email does not exists . If email exists the password is hashed and checked with user password, if does not match invalid password error is shown using angularJS and views. On checking these conditions if the provided information is valid then user session is created and redirected to homepage.

HomePage:

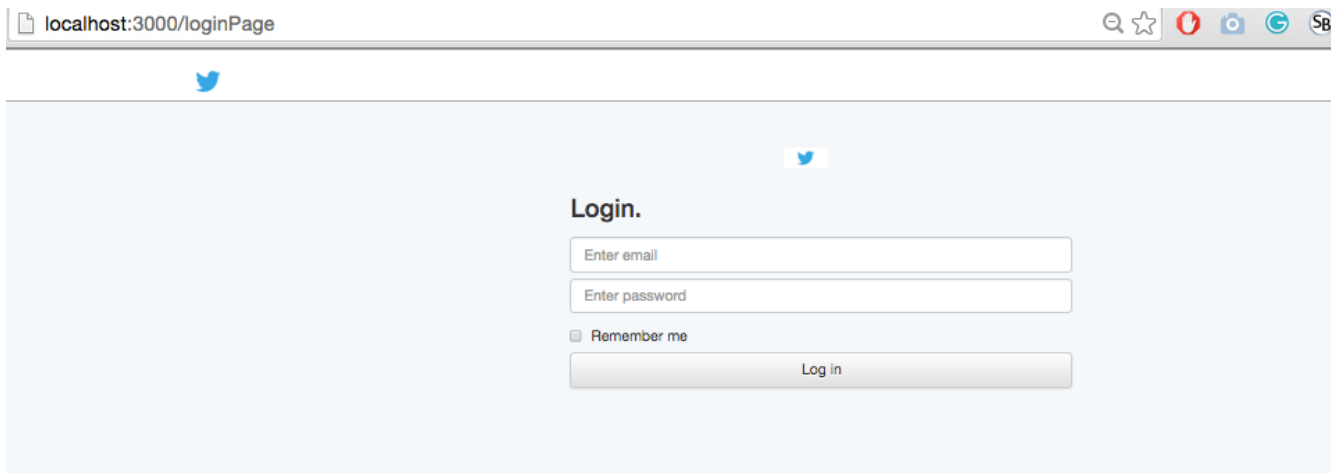


Signout:

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.

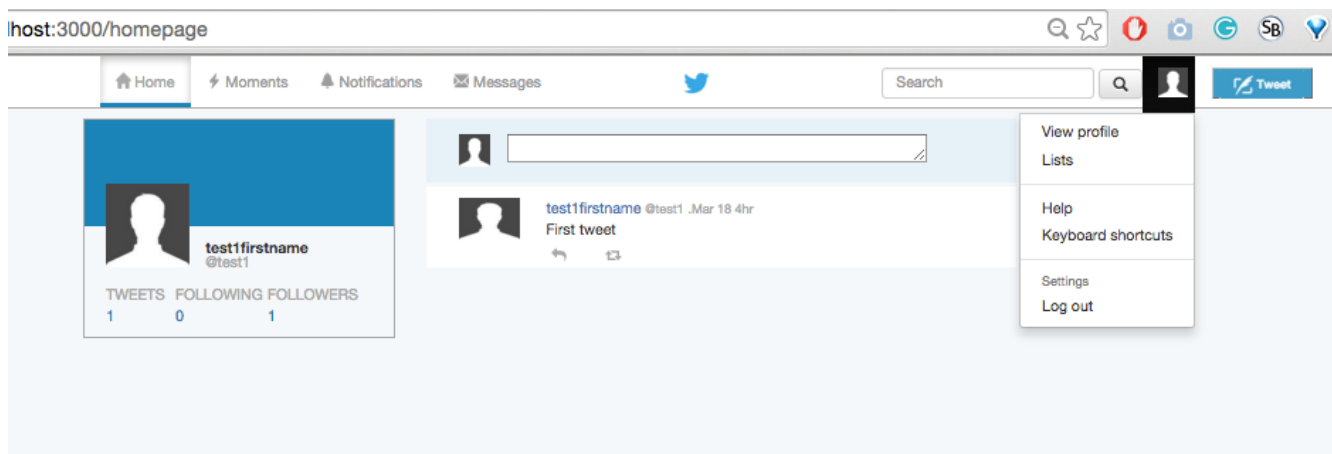


Profile functionalities:

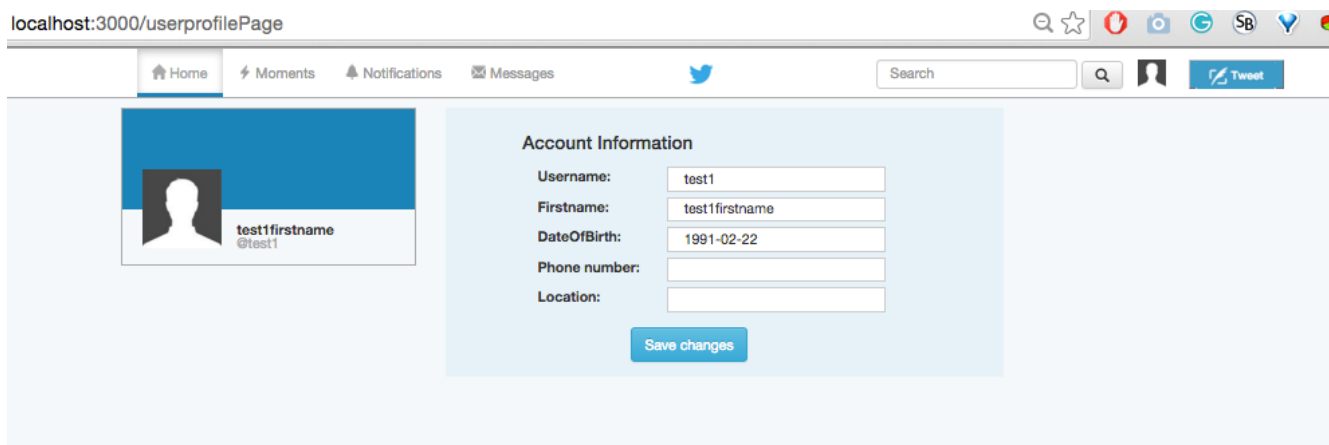
a. About:

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 invalid 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.

```
mysql> select * from users where username= "test1"\G;
***** 1. row *****
      id: 1
    email: test1@gmail.com
  username: test1
 password: $2a$10$iImRrGgej3bvhat09ZiCj.6sMDzje71S4X9AhSpoP0uQckvCq5Ju
firstname: test1firstname
  lastname: test1lastname
    gender: male
      date: 1991-02-22
   joindate: 2016-03-18 16:00:01
    location: USA
phonenumber: 123456
1 row in set (0.00 sec)
```

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

localhost:3000/userprofilePage

Home Moments Notifications Messages

Search

test1firstname @test1

Account Information

Username: test1

Firstname: test1firstname

DateOfBirth: 1991-02-22

Phone number: 123456

Location: USA

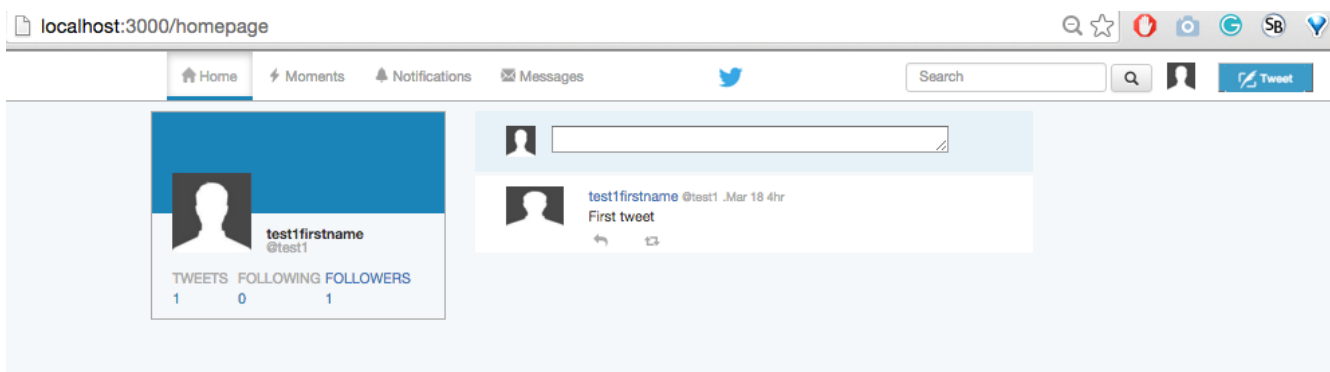
Save changes

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

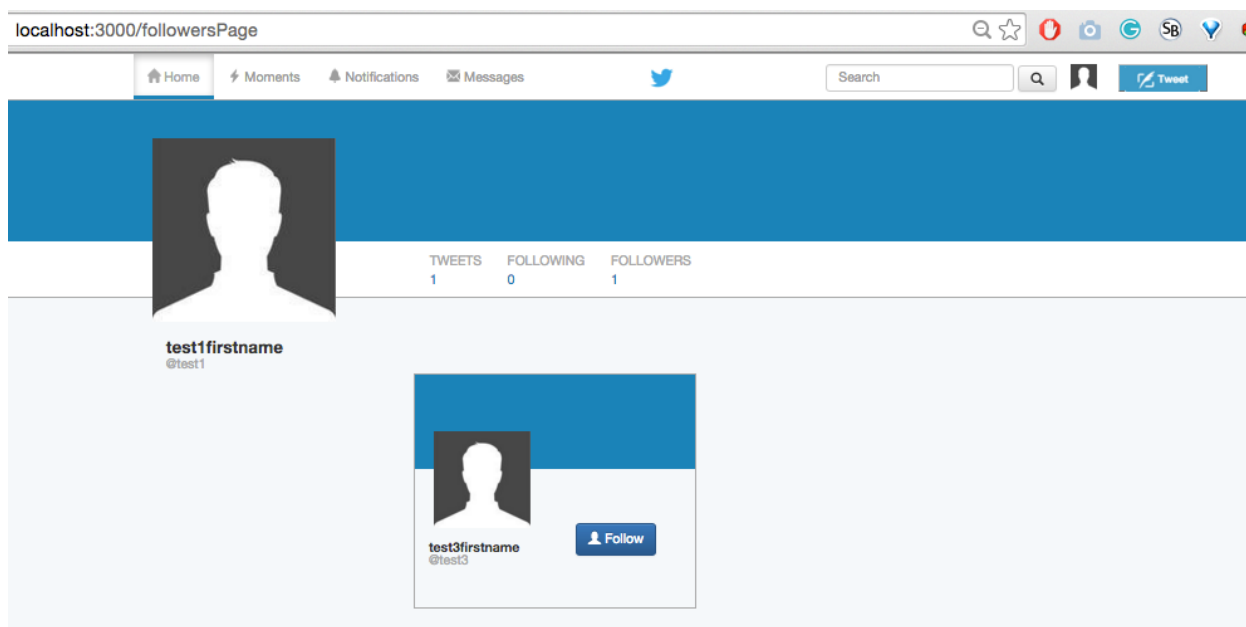
Followers list:

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.

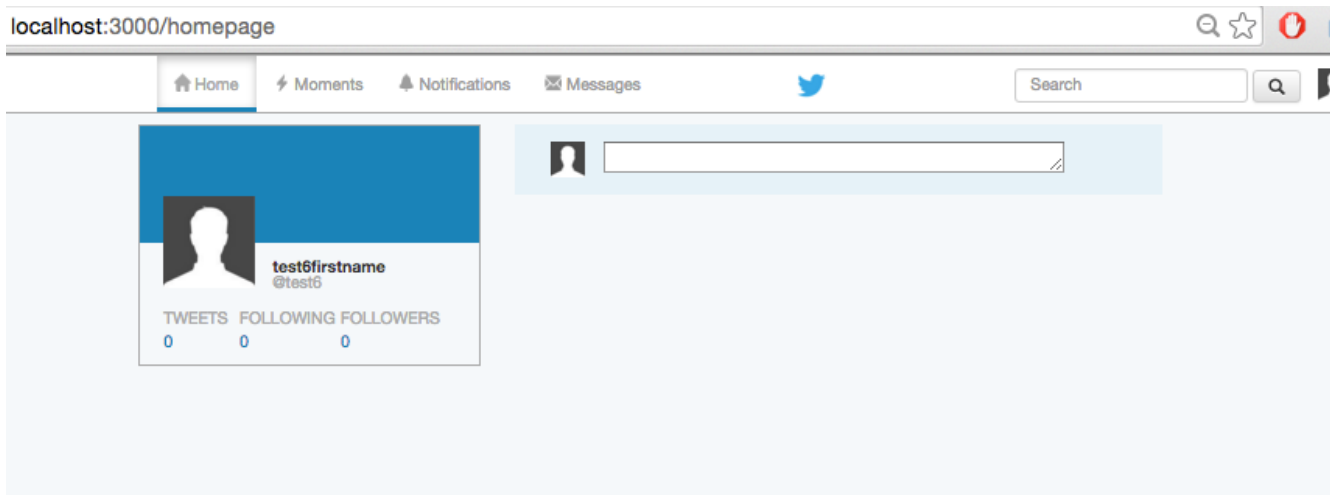


When user clicks followers it redirects the user to /followerspage which provides information about the followers like below.

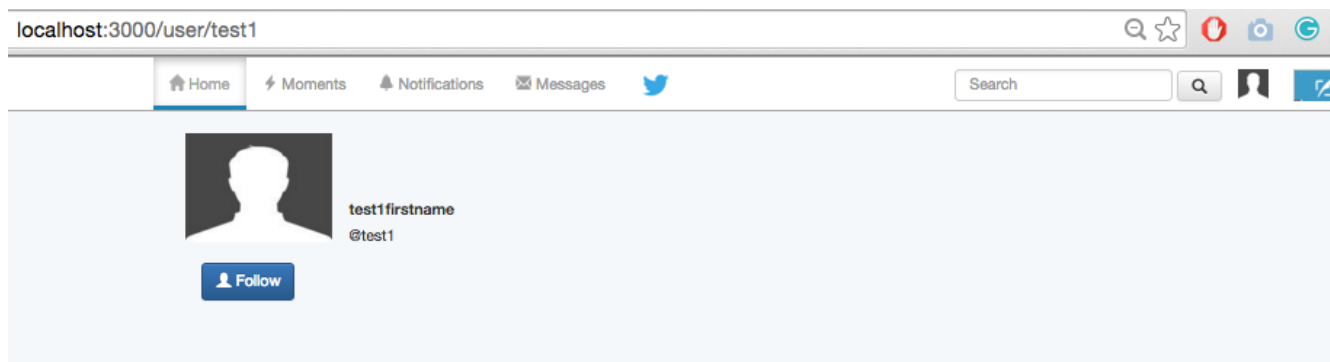


Lets add test6 and test7 users to follow test1 like below. For this scenario am logging out of test1 account and logging in from test6 account.

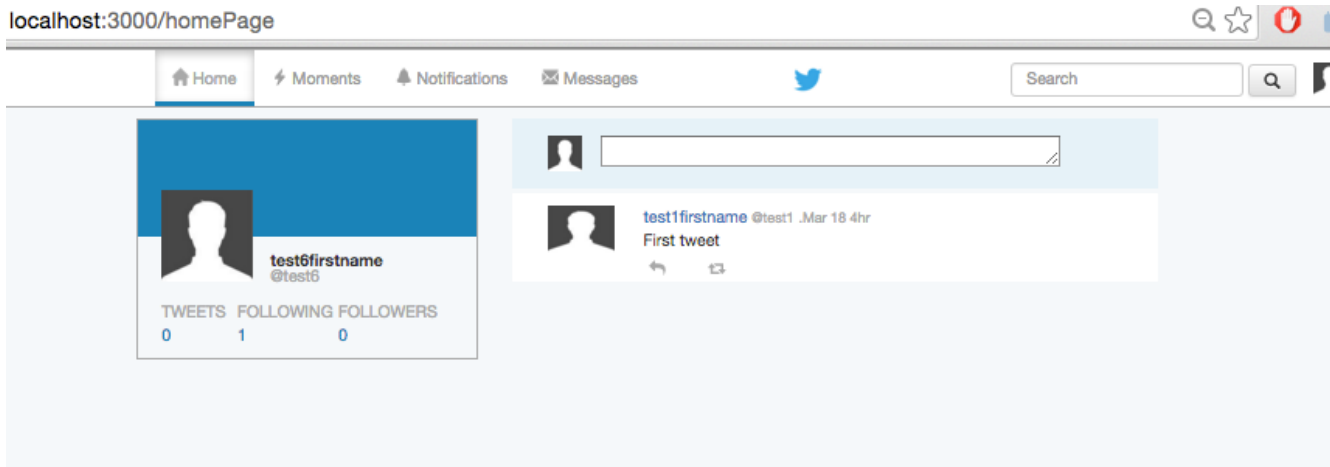
The homepage of test6 is like below he is not following anyone for now.



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

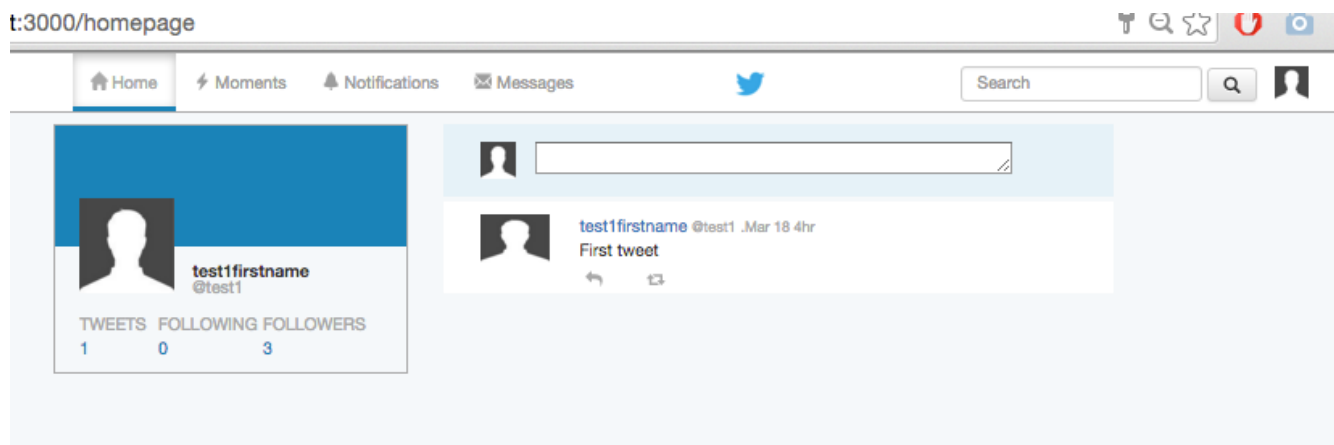


When follow button is clicked , the number of following list is increased by one in the user page like below.

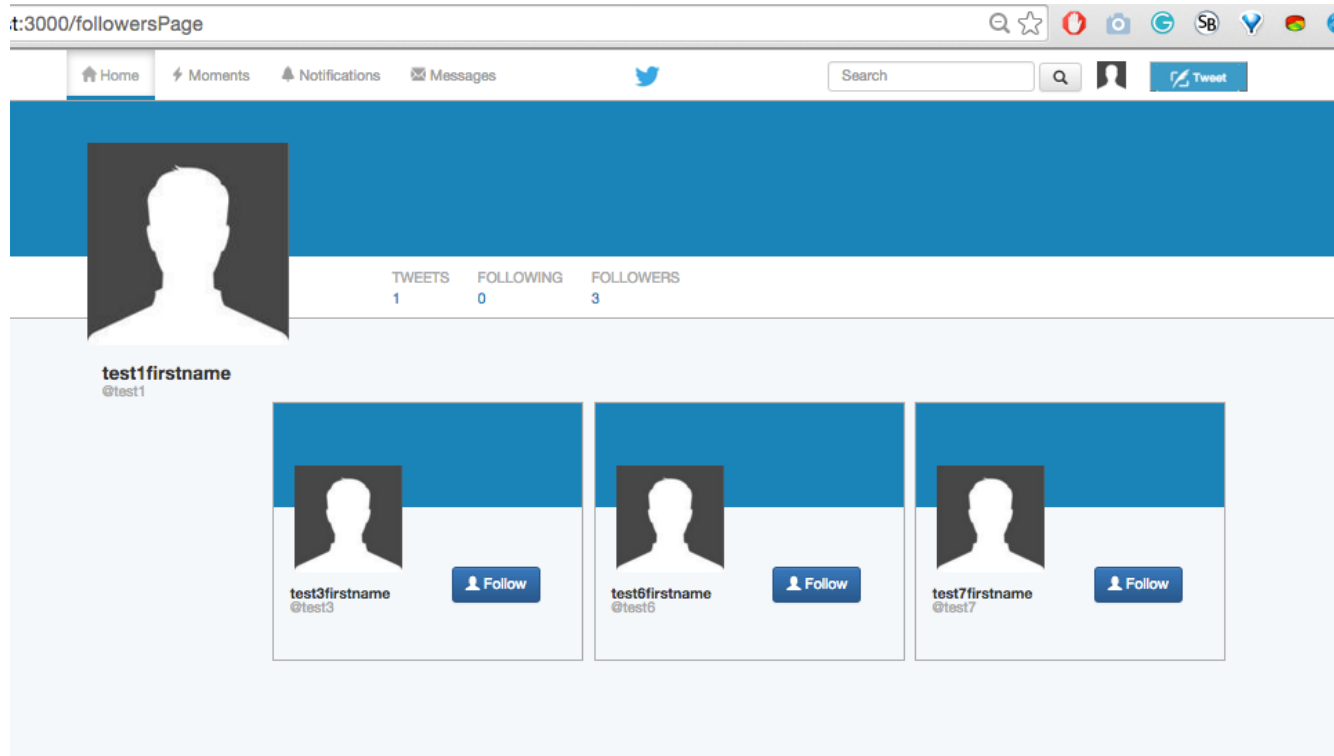


Similar process is carried out in case of test7 user to follow test1.

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.



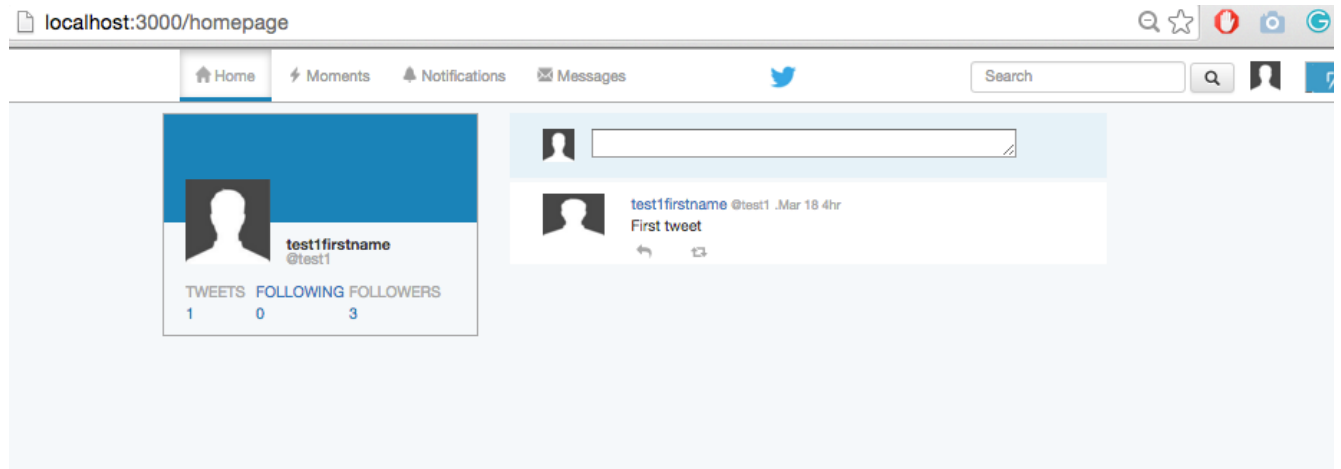
The followers list contains details of test6 and test7 too like below.



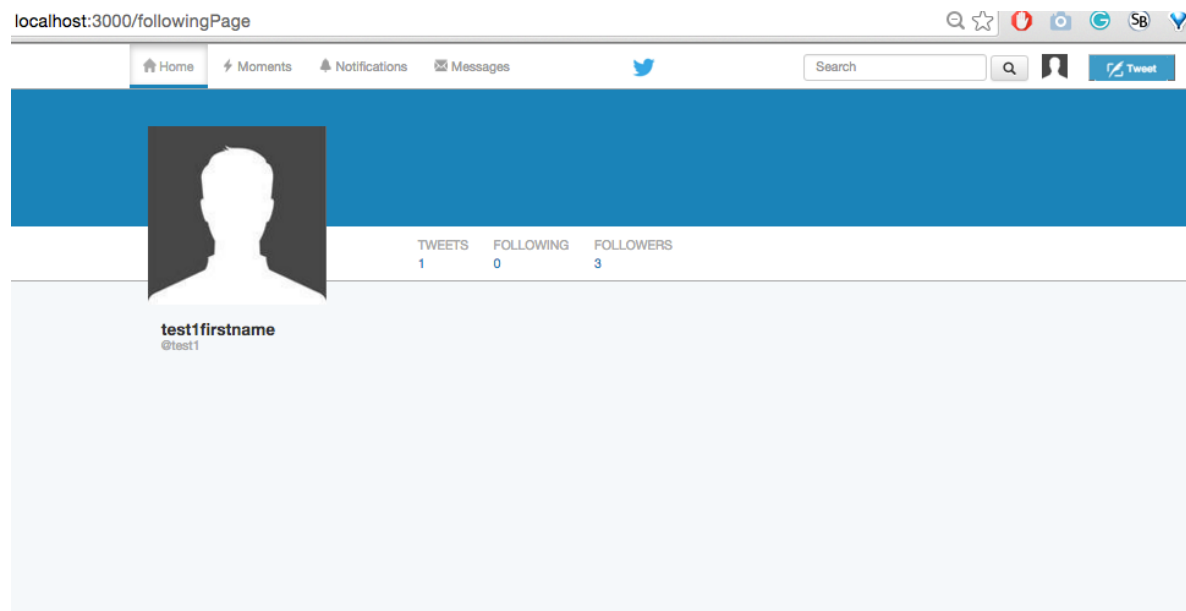
FollowingList:

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.

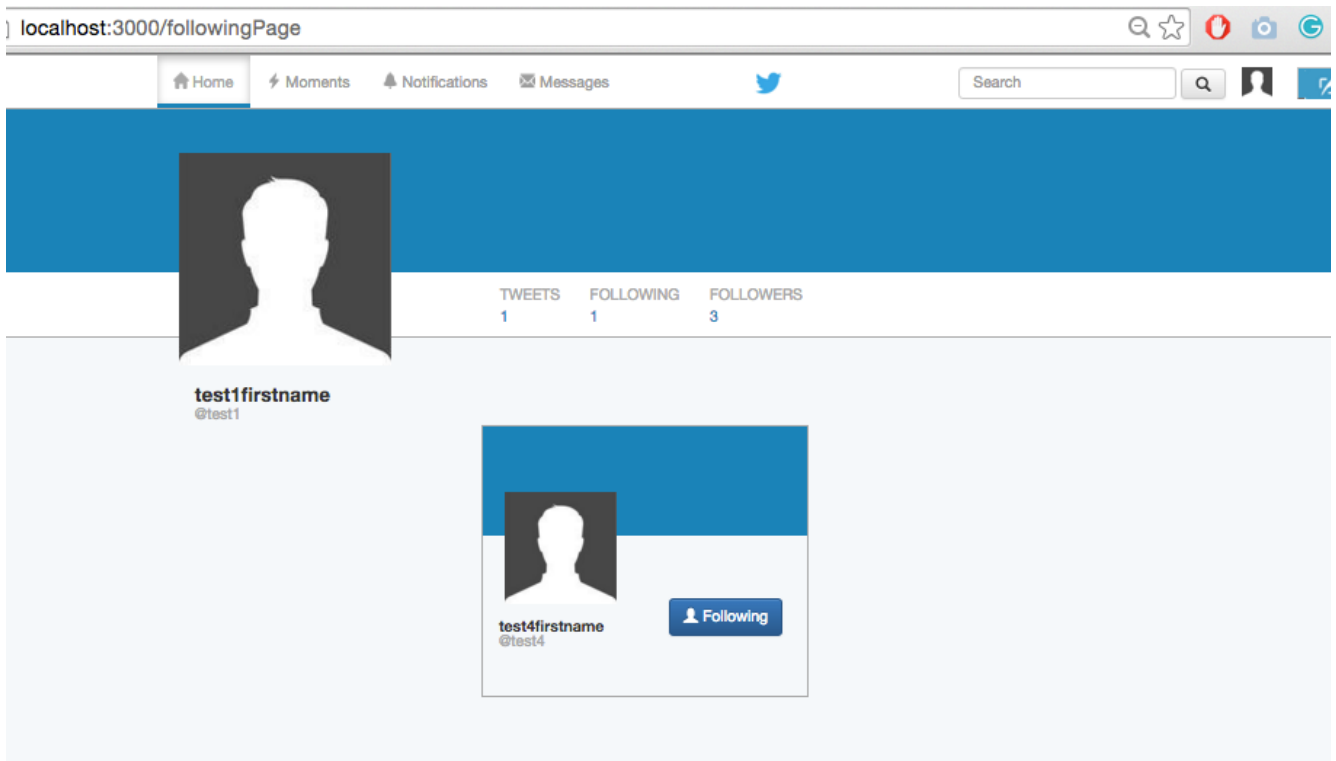
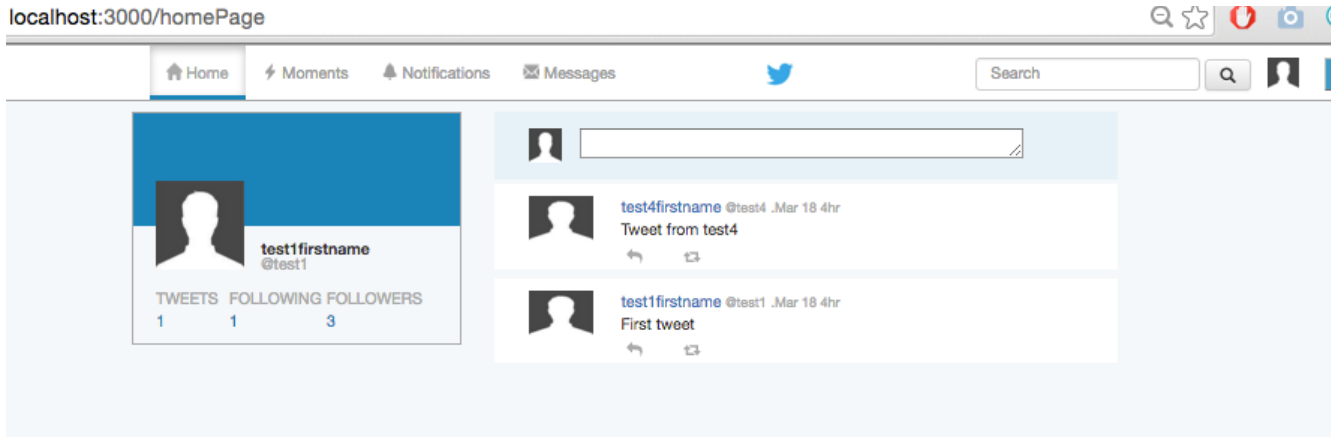
Incase of test1 user , at present situation the number of users he is following are 0 like below.



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.



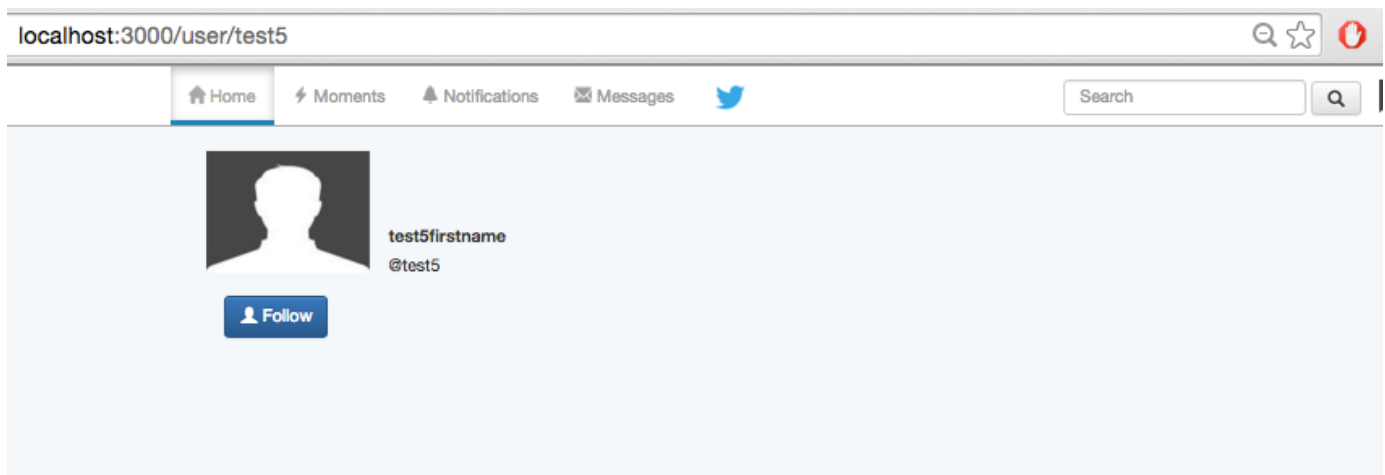
Let test1 follow test4 , then following count will be increased to one and followingList page contains information about test4.



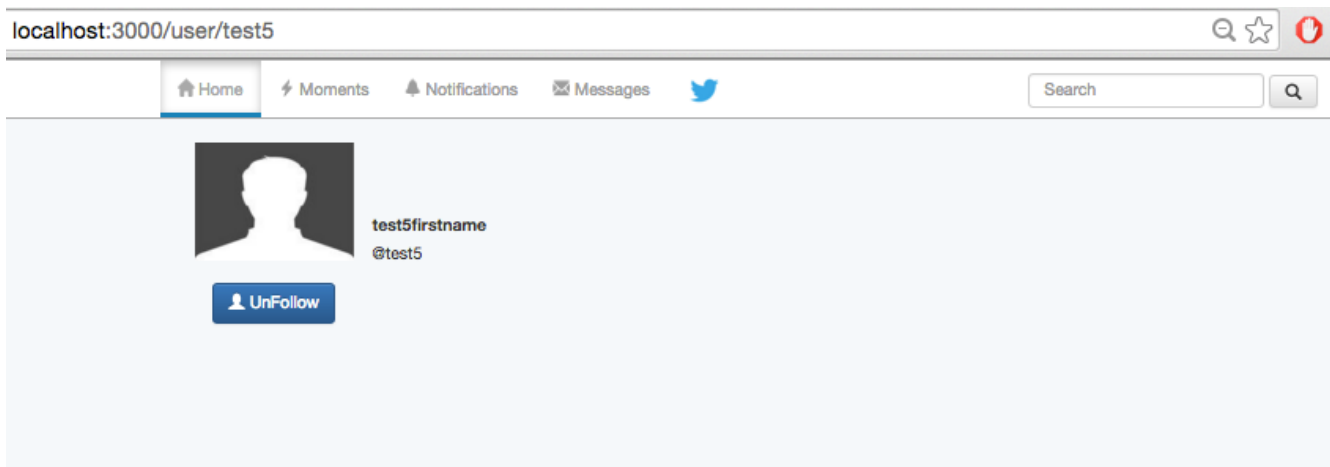
Follow:

In this application user will be able to follow other users using dynamic url .
In dynamic urls the username is passed. Using this information nodejs 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.

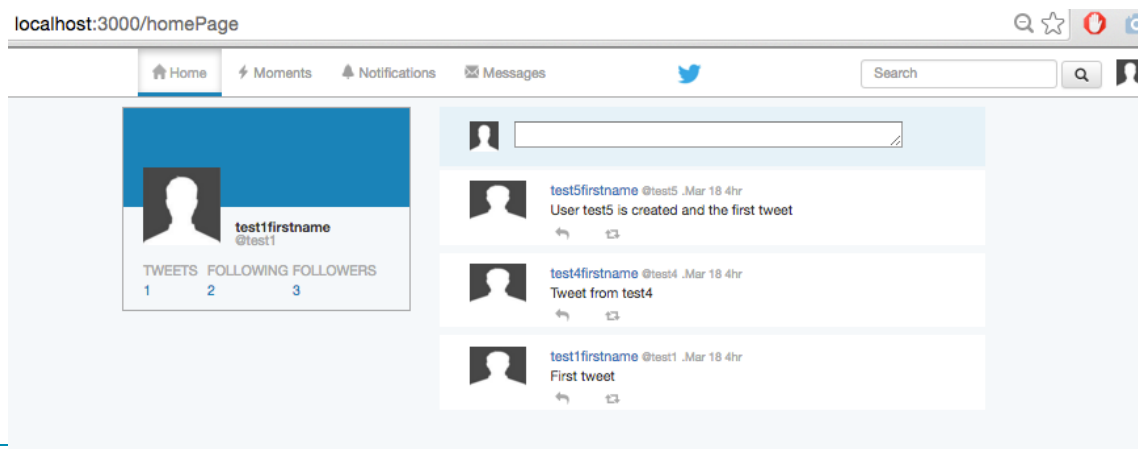


If test1 tries to follow user test9 who is not existing , then he will be redirected to homepage.

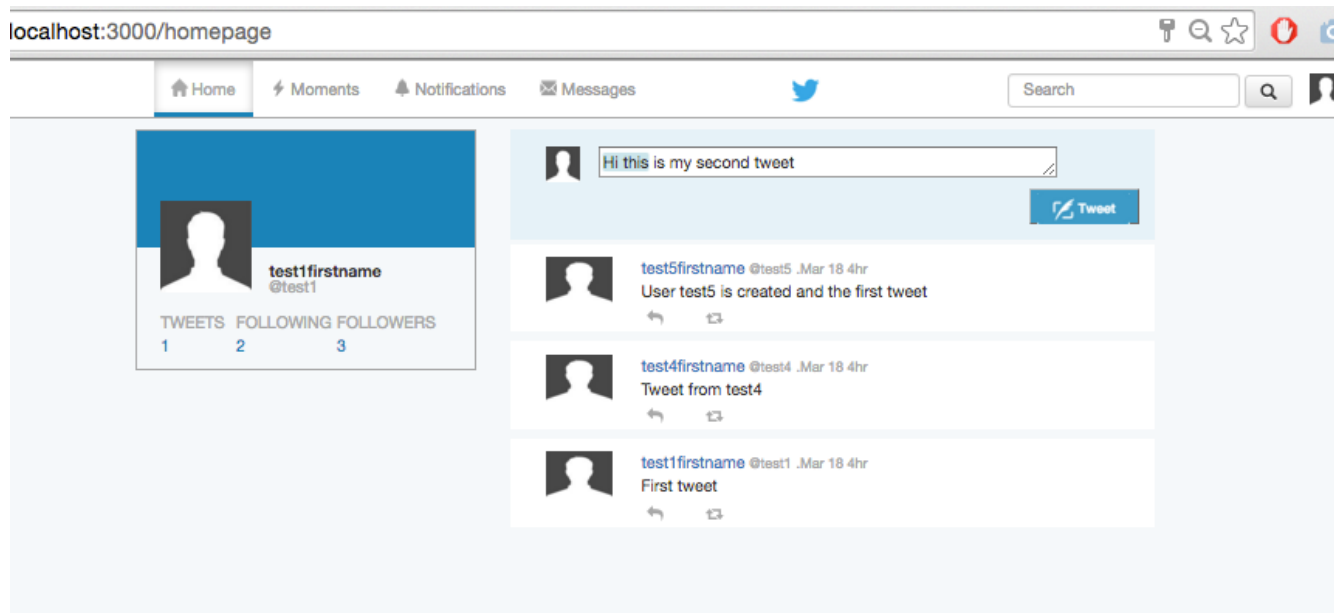
Show user tweets:

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.

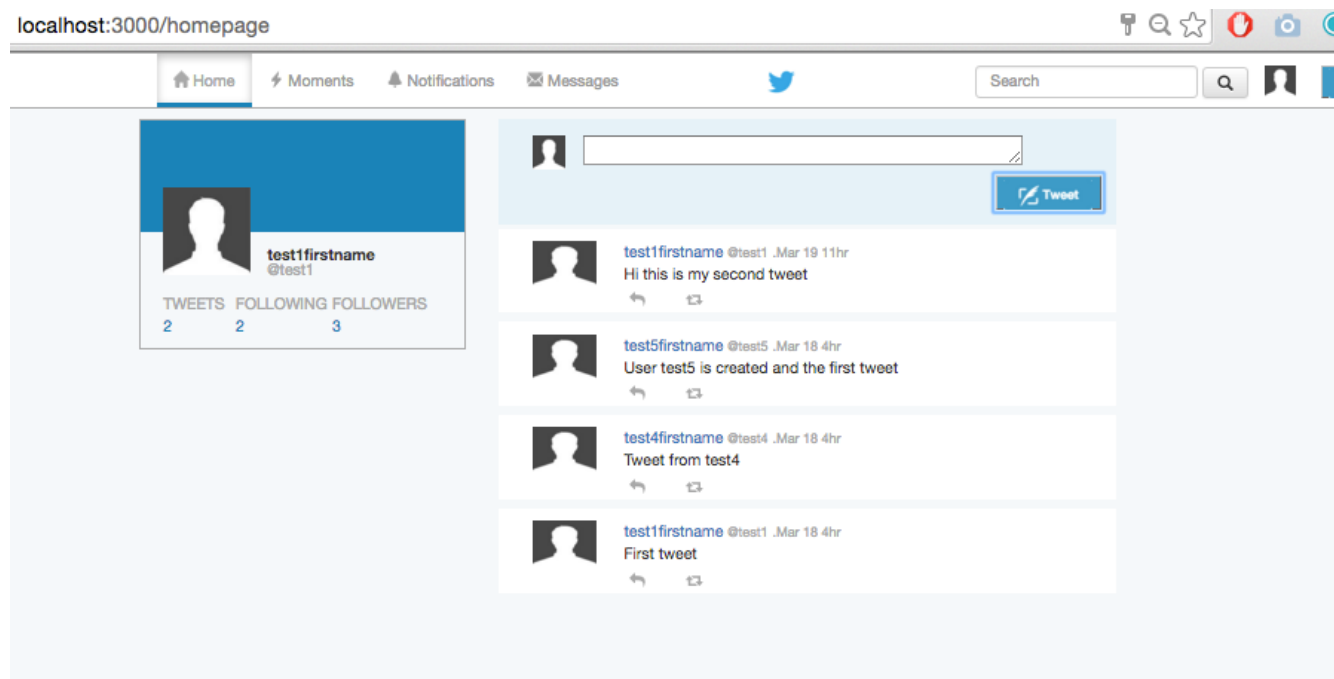
Example of test1, in present state the user tweets and followers tweets are displayed like below.



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



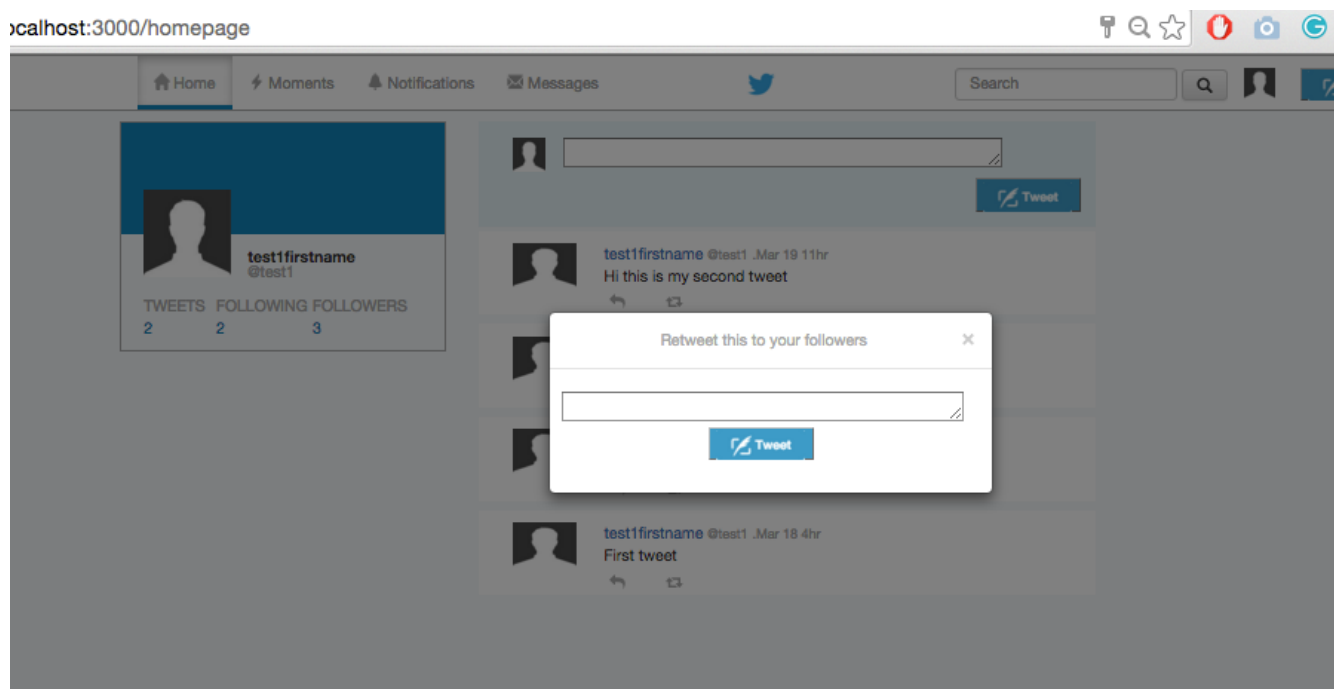
When user clicks the tweet button, the required information is stored and tweets of the user are shown in the homepage like below and tweets count is increased by one.



Show user's retweets:

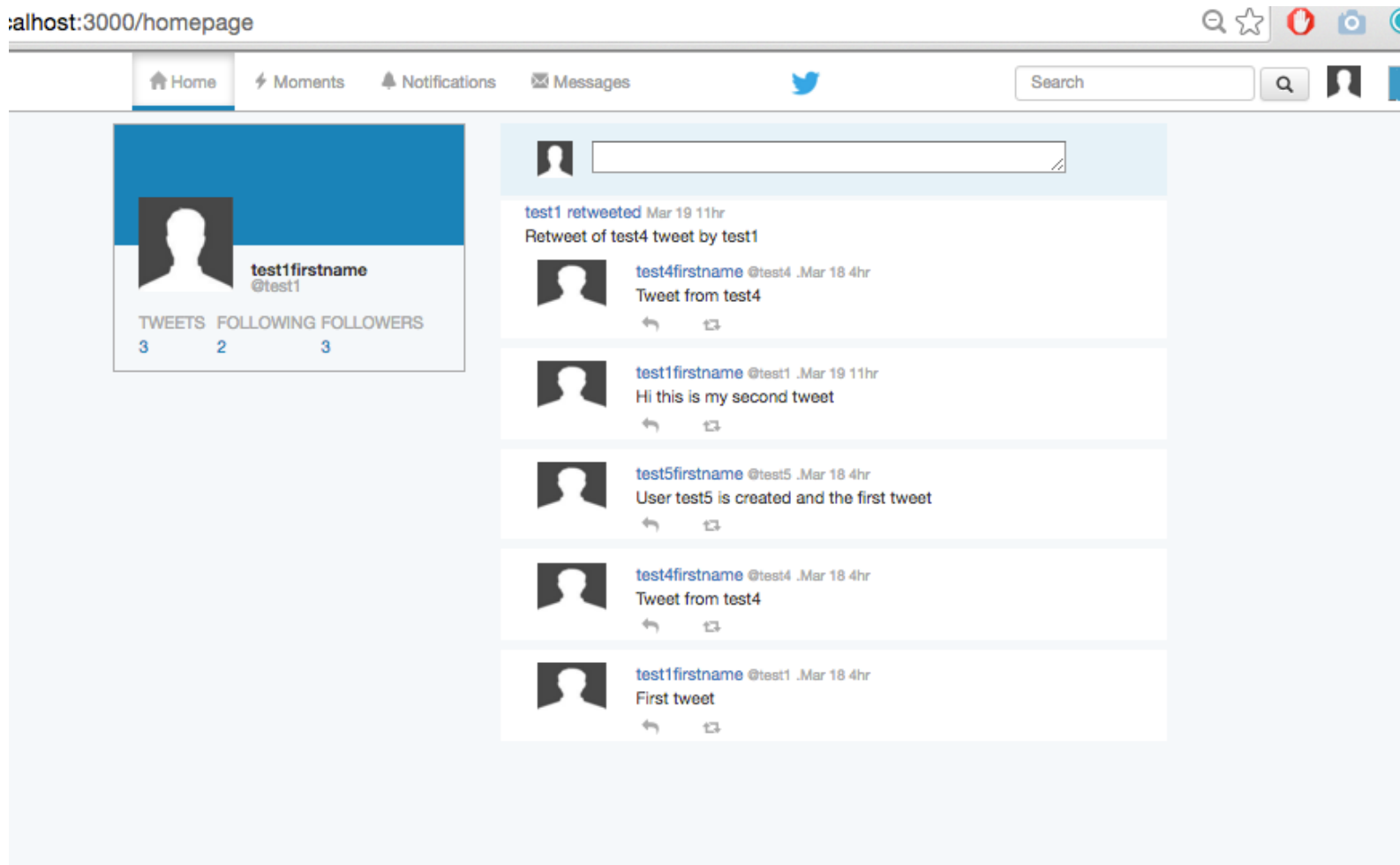
User is able to retweet the tweets using the retweet font icon displayed in the bottom of each tweet. When it is clicked , a modal is shown where the user can give retweet comment and retweet the tweet like below.

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



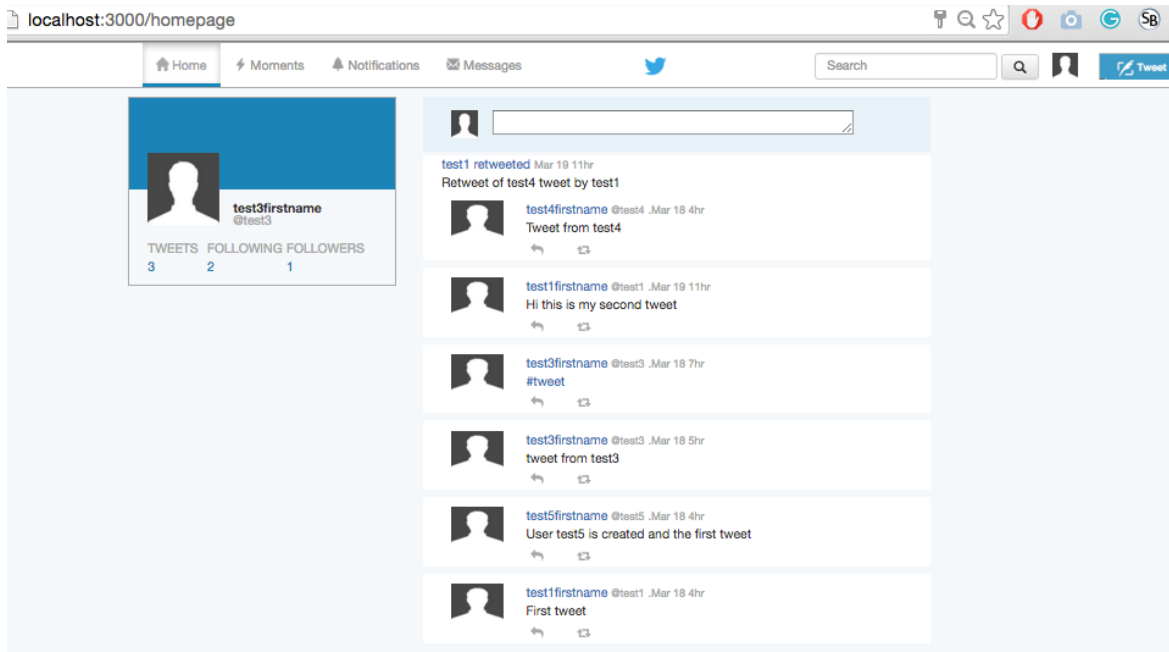
test1 can provide retweet comment and click tweet button like below. User can also not provide retweet comment. When it is retweeted the retweet will appear in user home page and his followers like below according to the time.

test1mainpage:

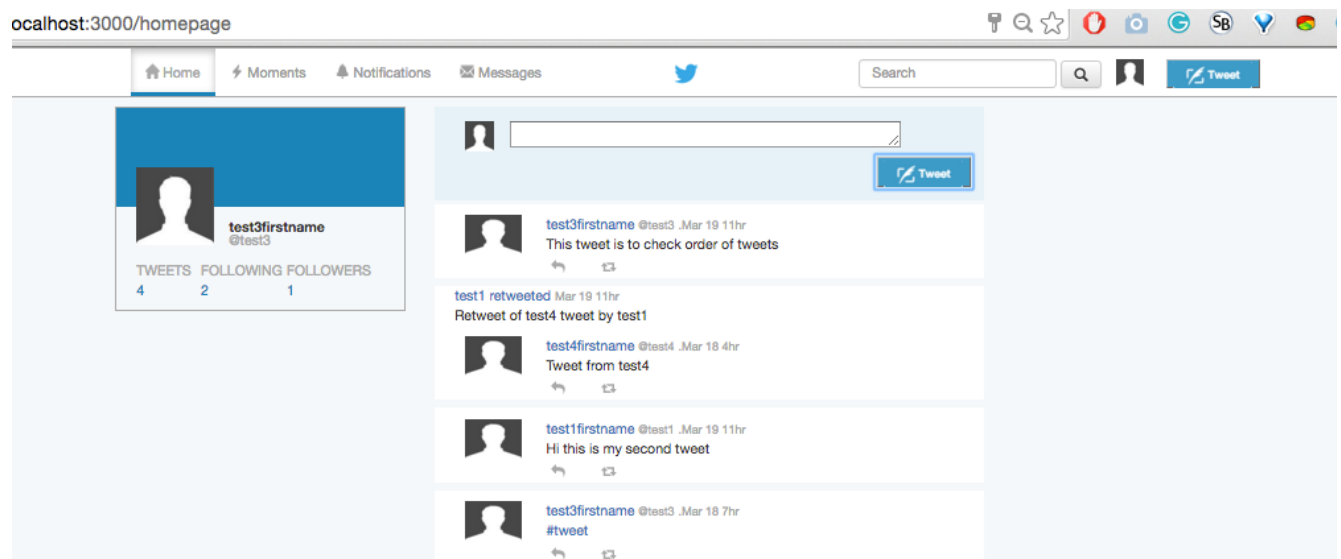


Retweets also appear in the followers page of this user like below,
From the followers list test1 follows test3, test6 and test7.

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



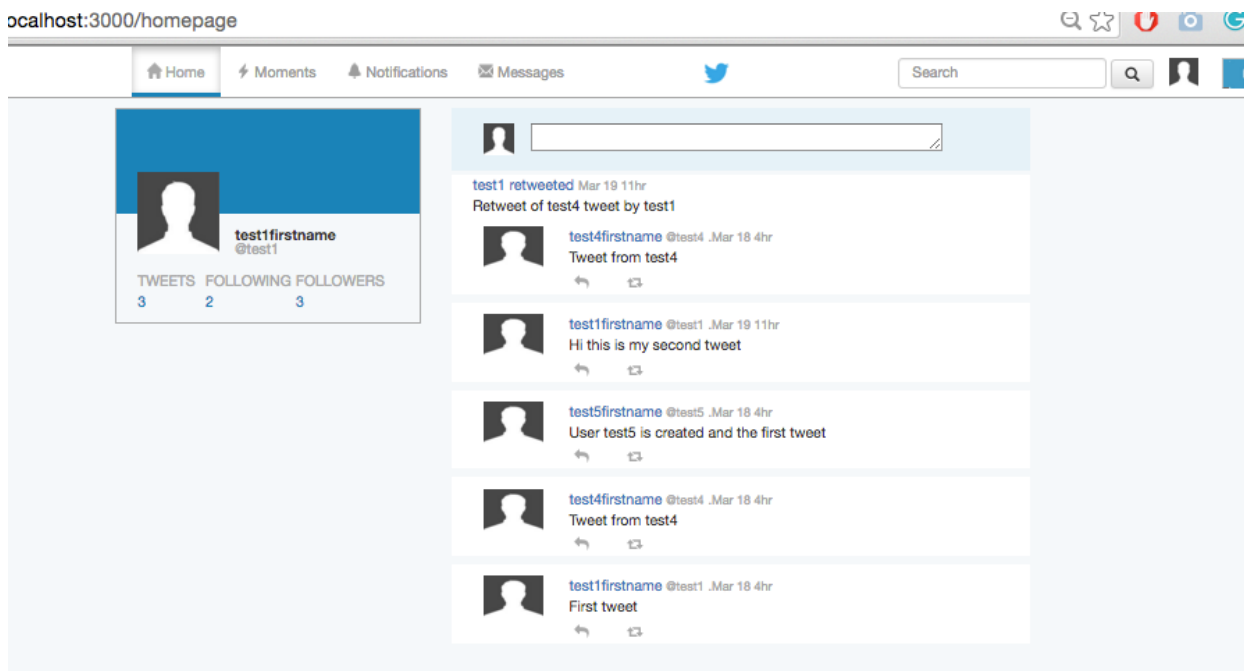
The tweet is displayed in order according to time like below, when test3 tweets another tweet, the retweet by test1 goes below the new tweet.



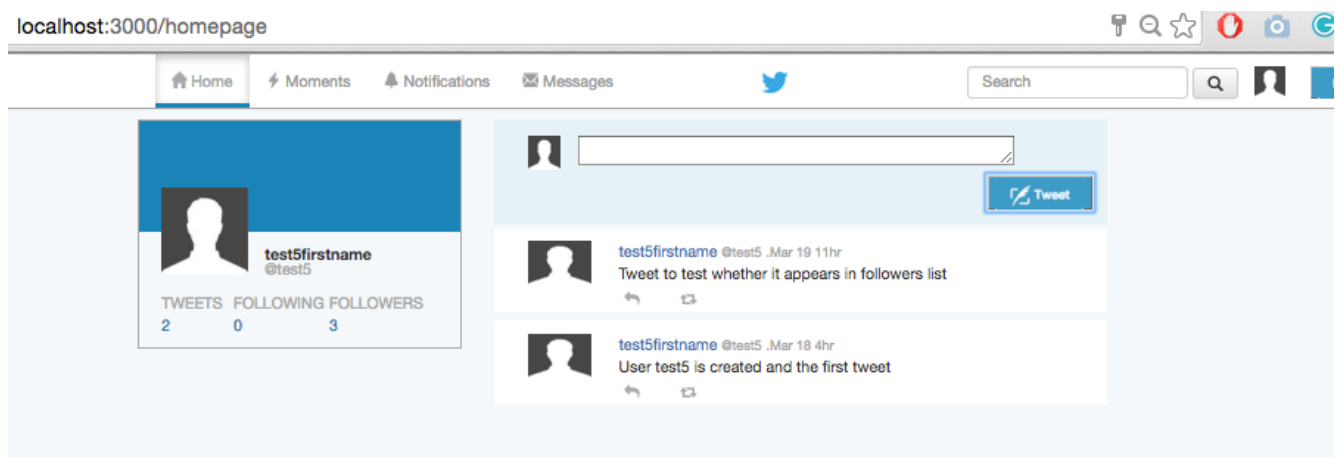
3. Twitter feed functionality:

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 .

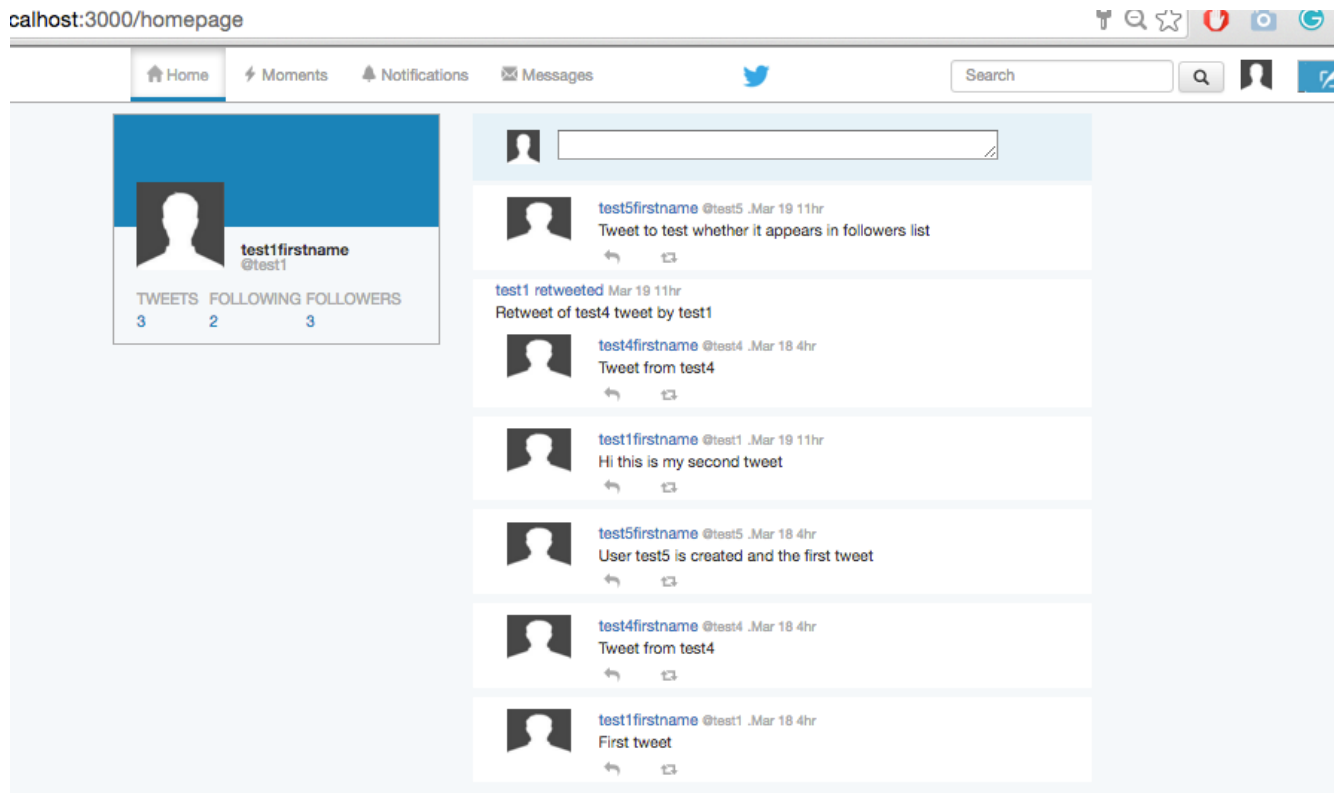
test1 is following test4, test5. The tweets related to them will be displayed in test1 page like below.



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

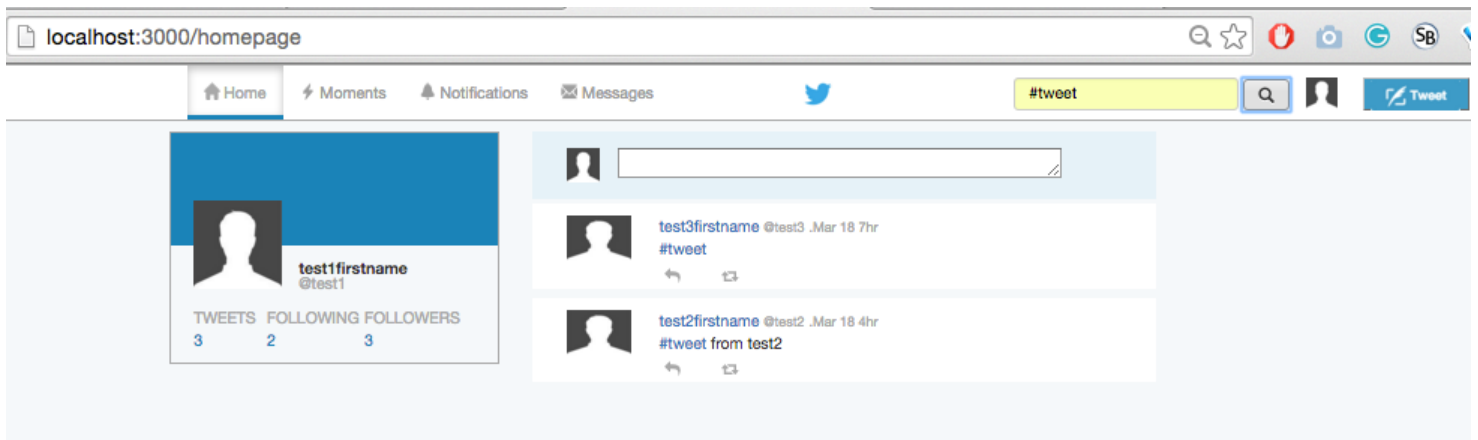


4. Hashtag functionality

In search:

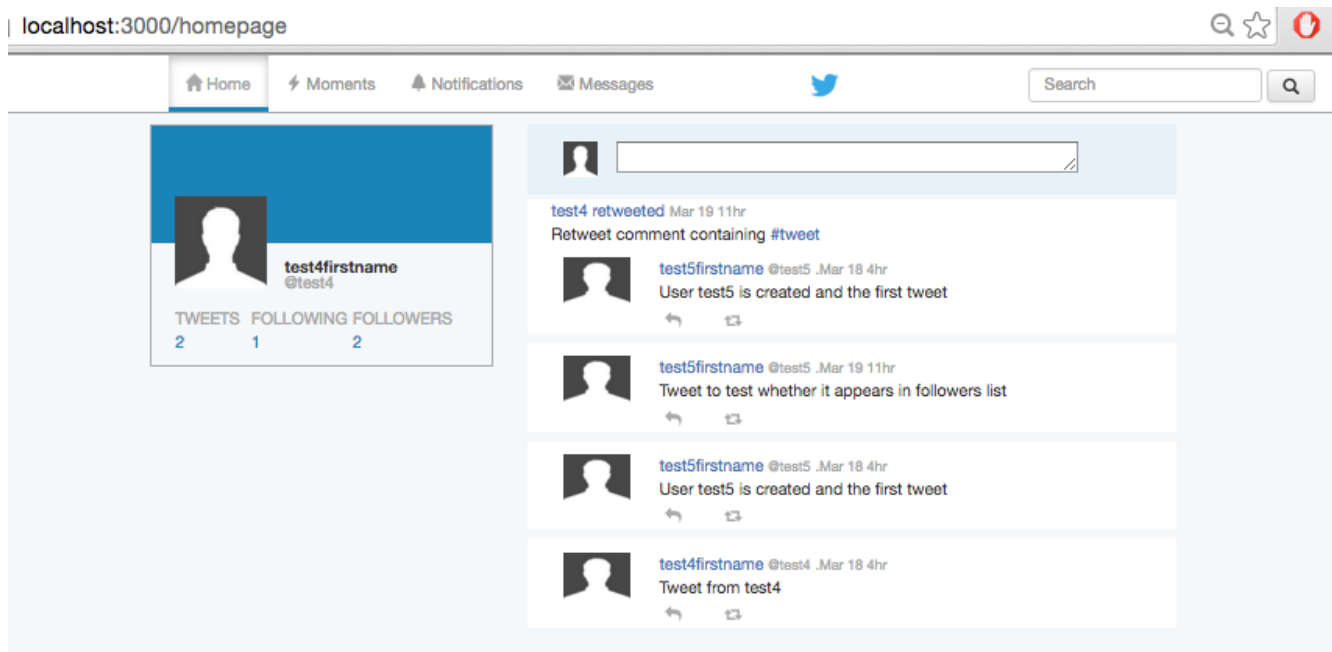
If user inputs the “#<text>” in the search box displayed in the navigation bar , all the tweets and retweets which contains the given keyword will be displayed on the homepage of user.

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

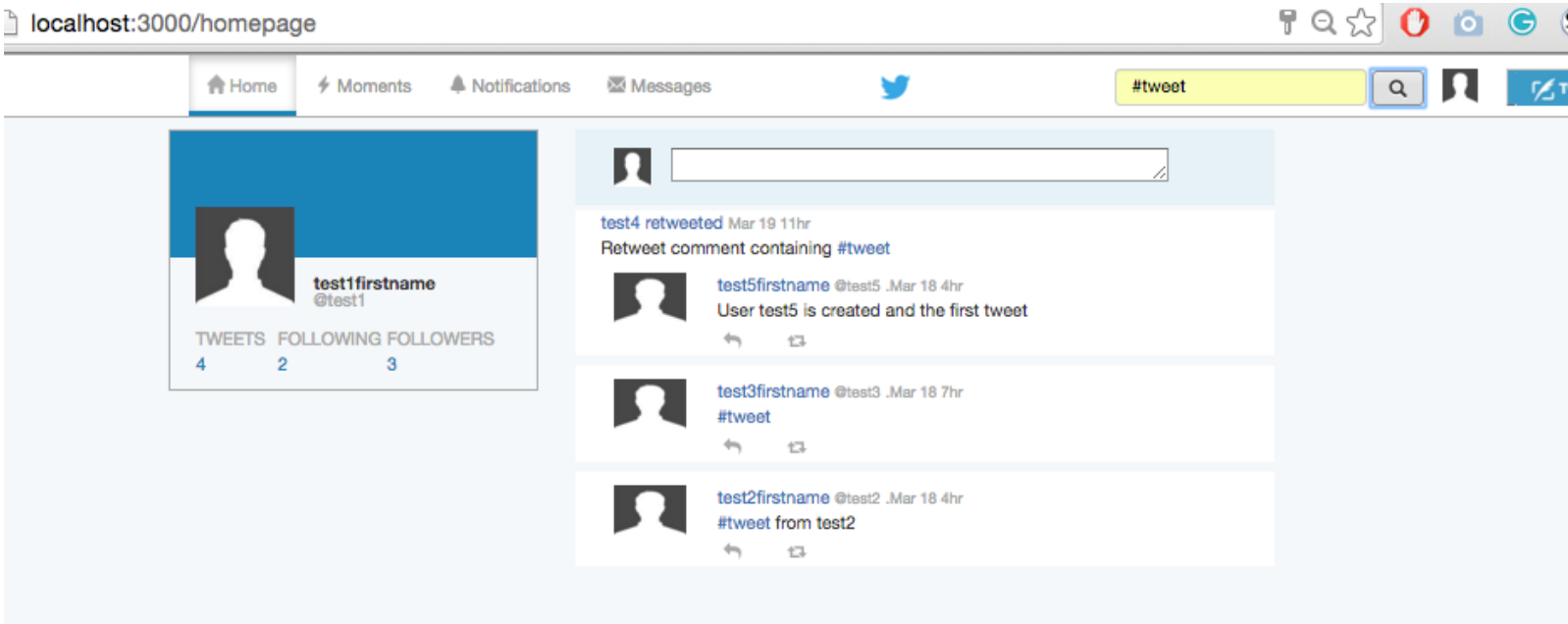


Though the test1 does not follow test2 but as it contains the #tweet in one of its tweet it is displayed in test1 tweet feed.

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

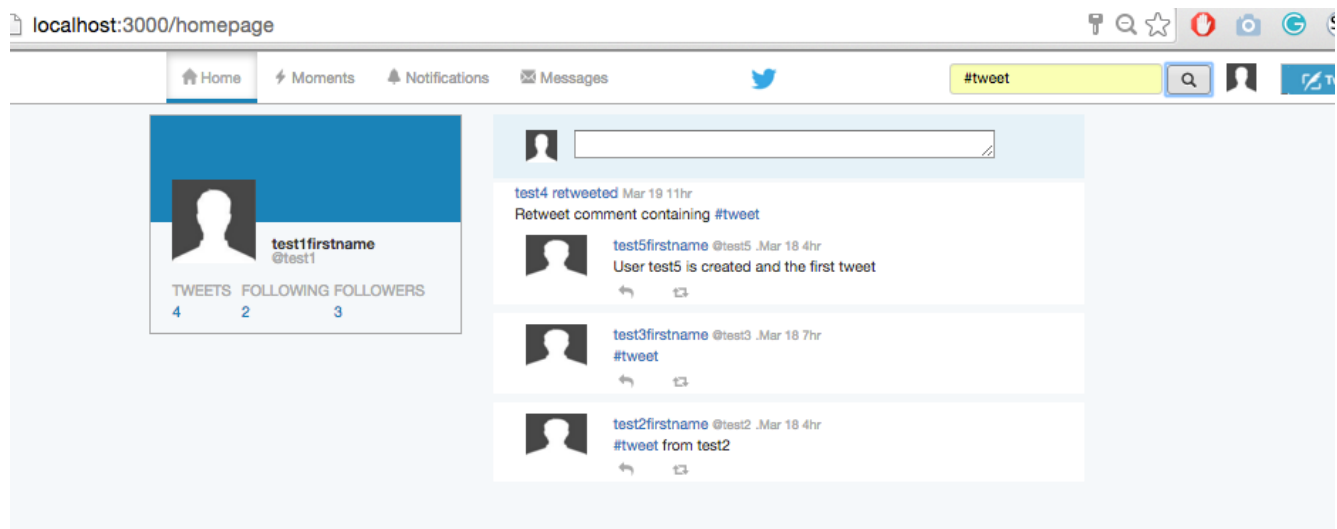


If we search for #tweet , it displays all the tweets including retweets containing hashkeyword like below.



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:

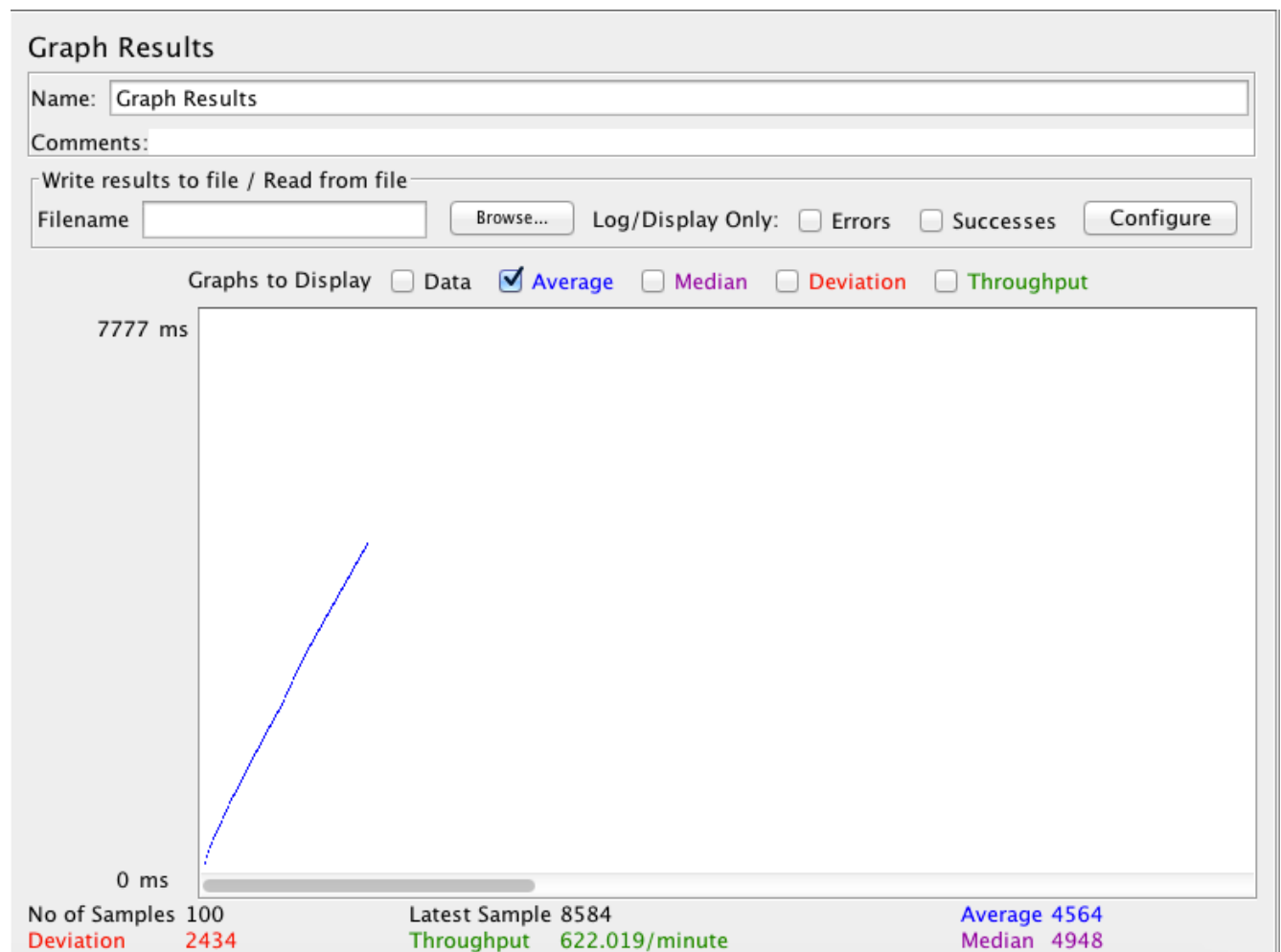
ConnectionPooling code is created in mysql.js of routes folder.

(The algorithm used to create connection pooling is explained at the questions part of this report below).

Performance testing using JMeter :

Login feature test results:

100 concurrent users:



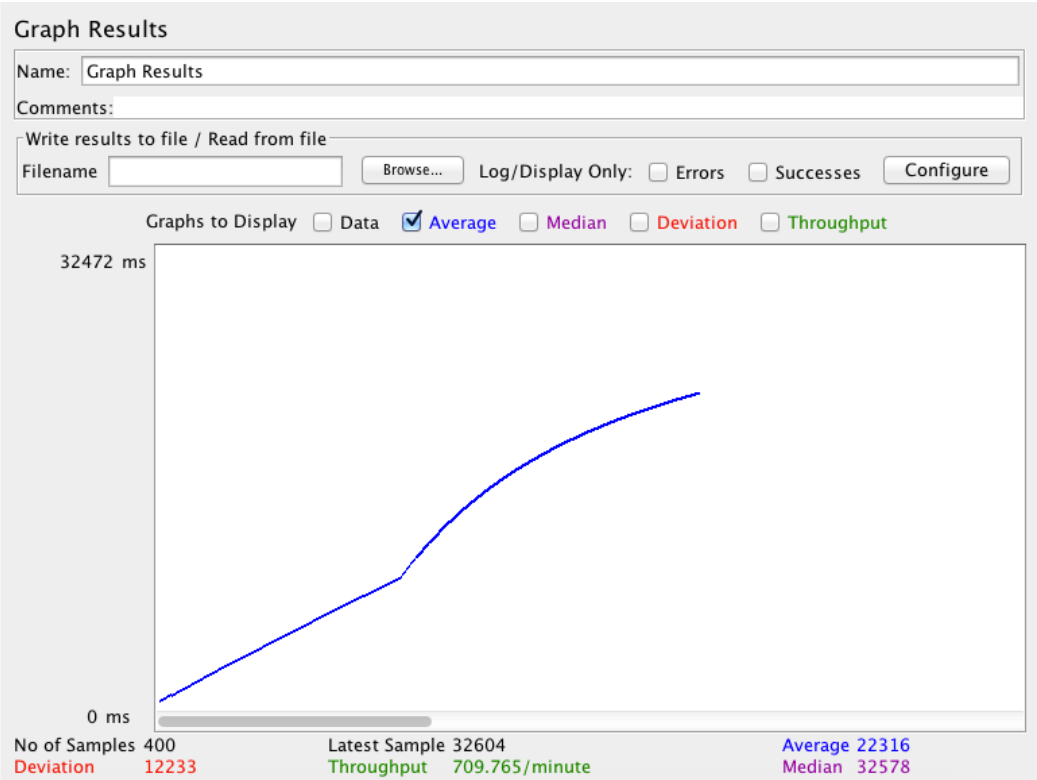
200 concurrent users:



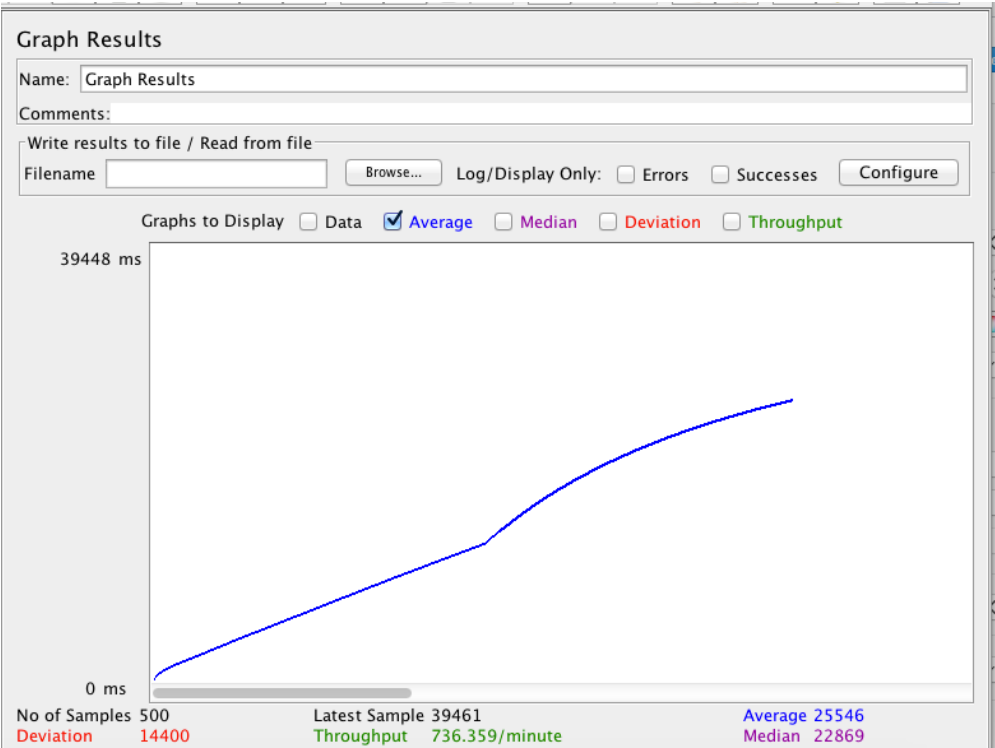
300 concurrent users:



400 concurrent users:



500 concurrent users:



Mocha tests:

Result:

```
Madhuris-MBP:TwitterApp Madhu$ mocha

http tests
  ✓ should return the twitter site main page if the url is correct
  ✓ should not return the twitter site main page if the url is wrong
  ✓ should be able to login with correct details (134ms)
  ✓ should not be able to login with incorrect details (99ms)
  ✓ should not be able to show user following page without logging in
  ✓ should be able to logout successfully after logging in (95ms)

6 passing (376ms)

Madhuris-MBP:TwitterApp Madhu$
```

Questions :

1. Explain the encryption algorithm used in your application. Mention different encryption algorithms available and the reason for your selection of the algorithm used?

The encryption algorithm used in my application is bcrypt called as blowfish encryption algorithm. It uses a 128-bit salt and encrypts a 192-bit magic value.

For using this

npm module bcrypt is installed.

In single line of code the salt plus hash is generated , while inserting the passwords into database, the following line of code is used.

```
var passwordhash = bcrypt.hashSync("password", 10);
```

While logging into the application, inorder to check whether the given password

is the correct one or not , I used bcrypt compareSync method which returns true if password given and password hash are same otherwise false.

```
bcrypt.compareSync("password", hash);
```

Different types of algorithms present:

There are different types of encryption algorithms like md5, sha1, sha512, sha256, sha-3 but these are all general purpose hash functions which are used to ensure the integrity of the data but not for storing passwords.

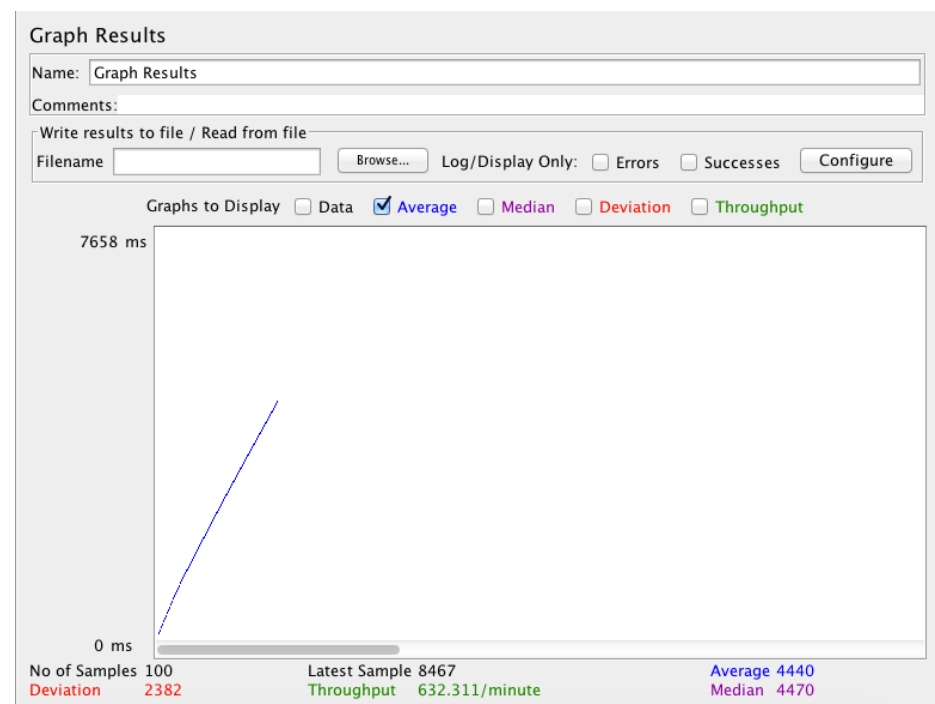
If we simply store the hashing of password then the third person or hacker who gets hold of the database, can still use rainbow tables to decrypt the password. In order to avoid this , developers add salt to passwords which makes rainbow attacks not feasible.

Reason to use bcrypt :

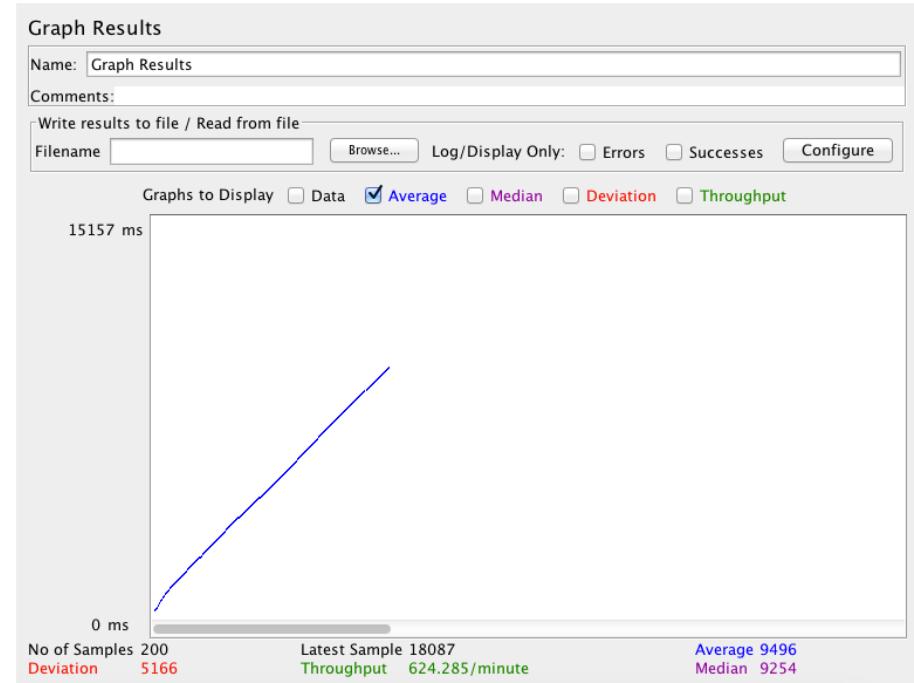
bcrypt is a password hashing function which is very slow. It is slow because it uses a modified key setup algorithm . This slow process makes password more secure against dictionary attacks because the attacker needs lot of time to test each and every possible key.

2. Compare the results of the graphs with and without connection pooling. Explain the results in detail. Describe the algorithm of connection pooling used in your application?

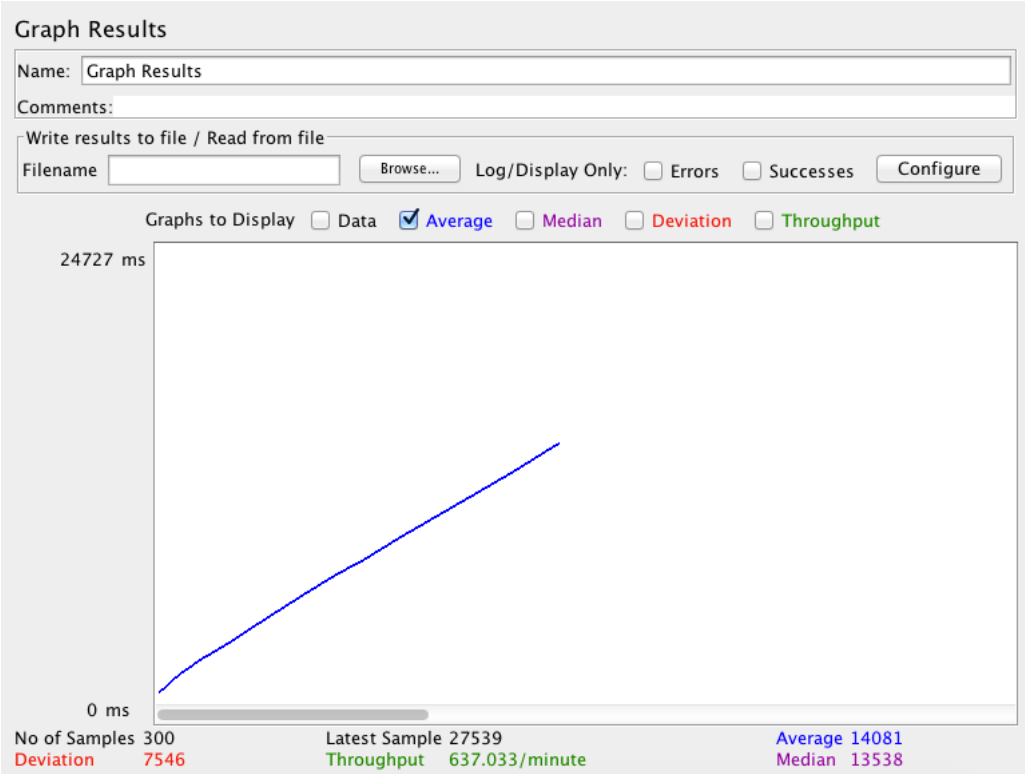
With connection pooling:
100 concurrent users:



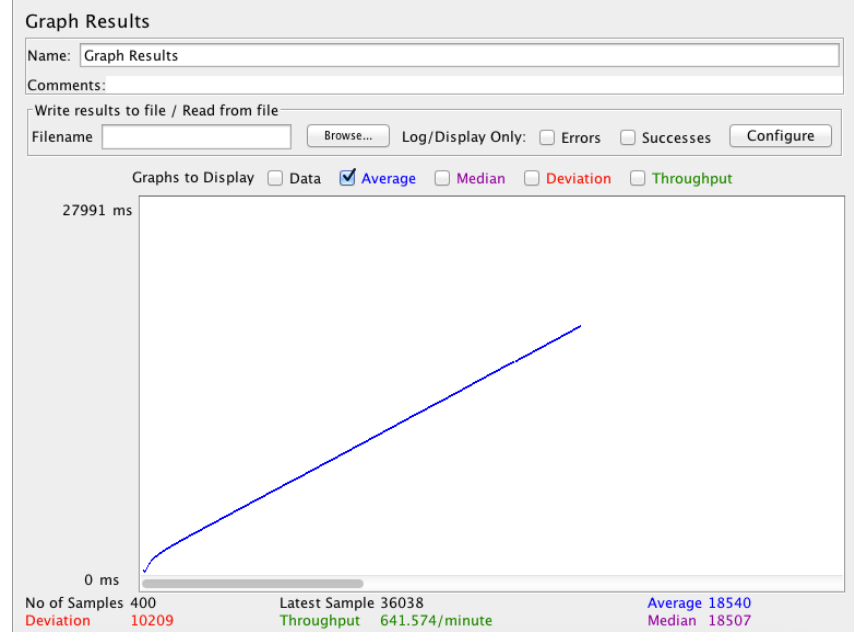
200 concurrent users:



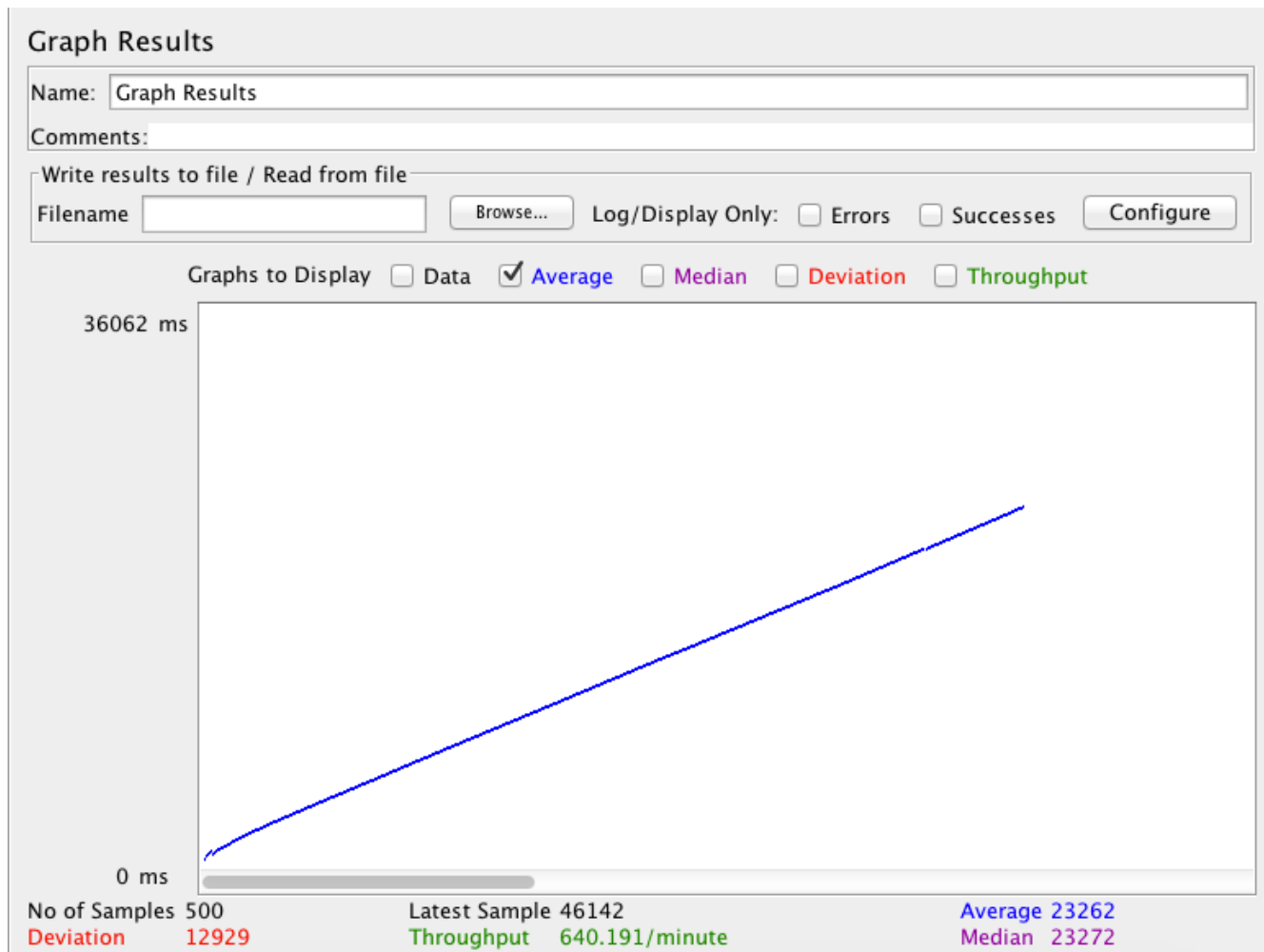
300 concurrent users:



400 concurrent users:



500 concurrent users:



Comparing the results:

The average time taken with connection pooling is less compared to without connection pooling. 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. After the use of connection object it is pushed back to connection pool for later use but not released incase of connection pooling.

Algorithm used in application for connection pooling:

- To implement connection pooling, initially connection pool is created with 10 connection objects. I used a queue data structure as connection pool to store connection objects.

If the connection pool is empty I tried to update the size of connection pool by adding more connection objects.

Every time when connection object is required from, it taken from pool using the below code.

```
function getPoolConnection() {  
  if(getPoolSize() <= 0) {  
    updateConnectionPool(100);  
  }  
  connectionPool.reverse();  
  connection = connectionPool.pop();  
  connectionPool.reverse();  
  return connection;  
}
```

after the use it is pushed back to queue using,

```
function returnConnection(connectionObject){  
  if(connectionPool != null) {  
    connectionPool.push(connectionObject)  
  }  
}
```

3. How would you implement request caching? Explain in detail.

The purpose of caching is mainly to provide a mechanism to make applications scale better and perform faster. Depending on the application we need to choose a proper caching mechanism so that it is not needed for every request to contact server each and every time.

Inorder to implement request caching I want to use application data caching, a do it yourself approach using memcached or similar API's.

<http://blog.sqlauthority.com/2014/03/18/sql-server-performance-do-it-yourself-caching-with-memcached-vs-automated-caching-with-safepeak/>

SQL servers has its own cache, it can cache query plans , pages from the database files but does not cache results from a query.

For example, for this query how many different countries we have in our customer database: `SELECT DISTINCT names from nametable GROUP BY names`). SQL Server will scan the WHOLE name table, but the result-set will only be a few entries long. When we reissue this query, SQL Server will reuse the query plan and will rescan the name table, and if we are lucky the pages are in memory and can retrieve it. Inorder to remove luck condition we can also choose memcache to store results.

When we use application cache, we store the result-sets in Memcached RAM. Then reuse them over and over again without connecting to the database server, thus offloading workloads from your database server.

Pseudo code:

Memcache is a hash table that lives entirely in the RAM on multiple servers. It contains methods `put(k)` to put value of a key `k` into memcache and use `get()` function to get the results.

Wrapping a database query using memcache:

example: define a query and use it as a key for the memcache

```
query = "select * from users where user_id= " + userid;
```

Query acts as a key for the memcached.

```
key = 'SQL:' + userid . ':' .md5sum(query)
```

if the value is defined in the cache we try to return the corresponding results if not available in cache will get query result set from database server in the application and convert it to result set array.

```
result = memcli(get(key))
if(result) {
    return result
} else {
    //run sql query on database server
    //get results

    //cache the results for certain time
    memcli(set(key, result, 5*60)
    return result;
}
```

Using this approach whenever user makes a change, it can take up to five minutes for users to see the correct new data. This issue can be resolved

either

- by updating the item in the cache or
- delete the old item.

pseudo code for deletion:

```
query = "select * from users where user_id= " + userid;
```

Query acts as a key for the memcached.

```
key = 'SQL:' + userid . ':' .md5sum(query)
memcli:delete(key)
```