PROJECT REPORT ON

OPEN SOURCE PROGRAMMING- ITE314

TWEET-IT – USING METEOR

MOHIT CHAUBEY – 14BIT0097

KRISHNA TEJA REDDY – 14BIT0116

YASH ARORA – 14BIT0181

Submitted to -
Faculty: Prof. Pradeep Reddy C.H.

ABOUT TWEET-IT

This application is a twitter-like platform where one can sign up for an account, log-in and add friends and post messages for each other. This application has a user friendly and an attractive user interface. The UI is build using materialize CSS. Tweet-it is build using Meteor which is a JavaScript platform used to develop web application using packages from Node.js.

This application has the following features-

- Sign up- User can sign up to create account by adding his details like username and password. All the details are saved in Collection which is a group of MongoDB documents.
- Log in, log out- User can log in or log out to start or end current session.
- Add Friends to your profile- User can add friends to his profile, friends can view tweets posted by the user.
- Post a tweet- User can post tweets which are visible to all friends added in his profile.
- View tweets posted by friends- User can view posts posted by friends.
- Delete your tweets- User can delete comment whenever they want.
- Deleting friends- User can delete friends from his friend list if he no longer wants any added friend to see his posts.
- Displaying count- The user profile also displays the number of friends added to a profile as well as the number of tweets which are posted by an application user from on its wall.

WHAT IS METEOR?

Meteor is a full-stack JavaScript platform for developing modern web and mobile applications. Meteor includes a key set of technologies for building connected-client reactive applications, a build tool, and a curated set of packages from the Node.js and general JavaScript community. It integrates with MongoDB and uses the Distributed Data Protocol and a publish–subscribe pattern to automatically propagate data changes to clients without requiring the developer to write any synchronization code.

- Meteor allows you to develop in **one language**, JavaScript, in all environments: application server, web browser, and mobile device.
- Meteor uses **data on the wire**, meaning the server sends data, not HTML, and the client renders it.
- Meteor **embraces the ecosystem**, bringing the best parts of the extremely active JavaScript community to you in a careful and considered way.
- Meteor provides **full stack reactivity**, allowing your UI to seamlessly reflect the true state of the world with minimal development effort.

Out of the box, Meteor comes with an amazing ability called 'hot code push'. Hot code push lets you update all your connected clients, browsers and mobile apps when the server is updated, Meteor rebuilds (refreshes client) the app every time a change is made. This module along with a companion meteor package allow you to hot-push the code into your meteor app without refreshing the page.

Installing Meteor

Meteor supports OS X, Windows, and Linux, and is simple to install. The command line installer supports Mac OS X 10.7 (Lion) and above, and Linux on x86 and x86_64 architectures. The Windows installer supports Windows 7, Windows 8.1, Windows 10, Windows Server 2008, and Windows Server 2012. iOS development requires the latest XCode.

Install Meteor:

curl https://install.meteor.com | /bin/sh

Create a project:

meteor create try-meteor

Run it:

cd try-meteor meteor

Deploy it

meteor deploy try-meteor.meteor.com

MongoDB

MongoDB (from humongous) is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas. MongoDB is developed by MongoDB Inc. and is free and open-source, published under a combination of the GNU Affero General Public License and the Apache License. MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability.

MongoDB works on concept of collection and document. Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose. A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

These are its main features-

- **General purpose database**, almost as fast as the key: value NoSQL type.
- **High availability**
- **Scalability** (from a standalone server to distributed architectures of huge clusters). This allows us to shard our database transparently across all our shards. This increases the performance of our data processing.
- **Aggregation**: batch data processing and aggregate calculations using native MongoDB operations.
- **Load Balancing**: automatic data movement across different shards for load balancing. The balancer decides when to migrate the data and the destination Shard, so they are evenly distributed among all servers in the cluster. Each shard stores the data for a selected range of our collection according to a partition key.
- **Native Replication**: syncing data across all the servers at the replica set.
- **Security**: authentication, authorization, etc.
- **Automatic failover**: automatic election of a new primary when it has gone down.

CODE

- **SERVER**

main.js

```
import '../imports/api/tasks.js';

Meteor.startup(() => {
  // code to run on server at startup
});
```

- **IMPORTS -> API**

tasks.js

```
import { Mongo } from 'meteor/mongo';


export const Tweet = new Mongo.Collection('tweet');
export const Friend = new Mongo.Collection('friend');
export const User = new Mongo.Collection('user');


Meteor.methods({
  'tweet.remove'(id, title) {
    Tweet.remove({uid:id, t_title: title});
  },
  'friend.remove'(uid, fid) {
    Friend.remove({id:uid, u_friend: fid});
  },
});
```

- **CLIENT**

main.js

```
import { Session } from 'meteor/session';

import '../imports/ui/body.js';

import '../imports/ui/js/jquery-2.1.1.min.js';

import '../imports/ui/js/materialize.min.js';
```

main.html

```
<head>
        <title>
                TweetIt | Home
        </title>
</head>
```

main.css

```
@import url("http://fonts.googleapis.com/icon?family=Material+Icons");

@import <'../import/ui/css/materialize.min.css'>

body {
   font-family: sans-serif;
  background-color: #315481;
  background-image: url('a.jpg');
  background-attachment: fixed;
```

```
position: absolute;
top: 0;
bottom: 0;
left: 0;
right: 0;

padding: 0;
margin: 0;

font-size: 14px;
}
```

- **IMPORTS -> UI**

body.js

```
import { Template } from 'meteor/templating';

import './body.html';
import './login.js';
import './home.js';



Session.setDefault('login', 'loggedout');
```

```
Session.setDefault('id', '');
Session.setDefault('error', '');
Session.setDefault('lerror', '');
```

body.html

```
<body>
        {{> login}}
        {{> home}}
</body>
```

login.js

```
import { Template } from 'meteor/templating';
import './login.html';
import { User } from '../api/tasks.js';
Template.login.helpers({
  loggedout() {
        return Session.get("login") == 'loggedout';
  },
  error() {
        return Session.get("error");
  },
  lerror() {
        return Session.get("lerror");
```

```
  },
});

Template.login.events({
  'submit .reg'(event){

  event.preventDefault();

  const target = event.target;
  const uname = target.uname.value;
  const pass = target.pass.value;
  var error="";

  var exist = User.find({id : uname}).count();
  if(exist!=0)
  {
  error= "Username Already Exists. Try another Name!!";
  Session.set('error', error);
  }
  else{
          Session.set('error', '');
          User.insert({
     id: uname,
```

```
        password: pass,
  });

        target.uname.value = '';
        target.pass.value = '';
        }
},

'submit .login'(event) {

event.preventDefault();

const target = event.target;
const uid = target.id.value;
const pwd = target.pwd.value;

var exist = User.find({id : uid, password : pwd}).count();
if(exist==1)
{
        Session.set('lerror', '');
        Session.set('id', uid);
        Session.set('login', 'loggedin');
        target.id.value = '';
```

```
        target.pwd.value = '';
        }
        else{
        var lerror = "Invalid Login. Try Again."
        Session.set('lerror', lerror);
        }
  },
});
```

login.html

```
<template name="login">
{{#if loggedout}}
<div class="row">
<p class="flow-text" style="color:white;text-align:center"><B>Tweet - It </B></p>
</div>

<div class="container">



<div class="row">
   <div>
     <ul class="tabs black">
       <li class="tab col s6"><a href="#test1" class="white-text">Login</a></li>
```

```
    <li class="tab col s6"><a href="#test2" class="white-text">Register</a></li>
  </ul>
</div>

        <script>
          $(document).ready(function(){
                  $('ul.tabs').tabs();
          });
          </script>

        <div id="test1" class="col s12 black">

                                              <div class="row">
  <div class="col m6 offset-m3">
    <div class="card-panel" style="margin-top:20%">

                            <div class="row">
<form class="login col s12">

                  <p class="red-text">{{lerror}}</p>
    <div class="input-field col s12">
      <input id="first_name" name="id" type="text" class="validate">
      <label for="first_name">User Name</label>
```

```
      </div>

      <div class="input-field col s12">
        <input id="password" name="pwd" type="password" class="validate">
        <label for="password">Password</label>
      </div>

    <div class="row">
  <center><button type="submit" class="waves-effect waves-light btn"><i class="material-icons right">done</i>Login</button></center>
  </div>

   </form>
  </div>
      </div>
    </div>
   </div>
         </div>
         <div id="test2" class="col s12 black">

          <div class="row">
     <div class="col m6 offset-m3">
      <div class="card-panel" style="margin-top:20%">
```

```
                        <div class="row">
  <form class="reg col s12">

    <div class="input-field col s12">
      <input id="user_name" name="uname" type="text" class="validate">
      <label for="user_name">Enter Unique User Name</label>
    </div>
            <p class="red-text">{{error}}</p>
    <div class="input-field col s12">
      <input id="password" name="pass" type="password" class="validate">
      <label for="password">Password</label>
    </div>

  <div class="row">
  <center><button type="submit" class="waves-effect waves-light btn"><i class="material-icons
right">done</i>SUBMIT</button></center>
  </div>
  </form>
  </div>
    </div>
   </div>
  </div>

      </div>
```

```
  </div>
</div>
{{/if}}
</template>
```

home.js

```
import { Template } from 'meteor/templating';

import './home.html';

import { Tweet } from '../api/tasks.js';
import { Friend } from '../api/tasks.js';
import { User } from '../api/tasks.js';

import './ftcard.js';



Template.home.helpers({
  uid() {
        return Session.get("id");
  },
  friend_count() {
```

```
        return Friend.find({id: Session.get("id")}).count();
  },
  post_count() {
        return Tweet.find({uid: Session.get("id")}).count();
  },
  loggedin() {
        return Session.get("login") == 'loggedin';
  },
});


Template.home.events({
  'submit .add_post'(event){

  event.preventDefault();

  const target = event.target;
  const title = target.title.value;
  const message = target.message.value;

        Tweet.insert({
    uid: Session.get("id"),
          t_title: title,
```

```
            t_message: message,
    createdAt: new Date(),
  });

        target.title.value = '';
        target.message.value = '';
},

'submit .add_friend'(event) {

event.preventDefault();

const target = event.target;
const f_id = target.fsearch.value;

var exist = User.find({id : f_id}).count();
var f_exist = Friend.find({id:Session.get("id"), u_friend: f_id}).count();
if(exist!=0 && f_exist==0)
{
Friend.insert({
                    id: Session.get("id"),
                    u_friend: f_id,
                  });
```

```
        }

        target.fsearch.value = '';

  },


  'click .logout'(event) {

        Session.set('id', '');

        Session.set('login', 'loggedout');

  },

});
```

home.html

```
<template name="home">

{{#if loggedin}}

<br><br>

<div class="container">

<div class="row">

        <div class="col s12">

        <center><p class="flow-text" style="color:white"><b>{{uid}}</b></p></center>

        <center><p style="color:white"><i class="tiny material-
icons">perm_identity</i> Friends : {{friend_count}}
     <i class="tiny material-icons">mode_edit</i>  Posts :
{{post_count}}      <a class="logout"><i class="tiny material-
icons">power_settings_new</i>  Logout</a></p></center>

        </div>

        <hr>

    <div class="">
```

```
    <ul class="tabs black">
      <li class="tab col s4"><a href="#test1" class="white-text">Tweet</a></li>
      <li class="tab col s4"><a href="#test2" class="white-text">Wall</a></li>
                <li class="tab col s4"><a href="#test3" class="white-text">Friends</a></li>
    </ul>
  </div>

        <div id="test1" class="col s12 black">

                <!-- Modal Trigger -->
                <br><br>
                <a class=" center-align waves-effect waves-white btn-flat col s4 offset-s4 cyan white-text" href="#modal1" >Add a new post</a>
                <br><br><br>
                <hr class="col s12">
          <!-- Modal Structure -->
          <div id="modal1" class="modal">
                <form class="add_post">
                <div class="modal-content">
                  <h4>Submit Your Post</h4>

                  <div class="input-field col s12">
        <input id="title" name="title" type="text">
```

```
<label for="title">Title</label>

</div>


                <div class="input-field col s12">

                <textarea id="message" name="message" class="materialize-
textarea"></textarea>

                <label for="message">Message</label>

                </div>


              </div>

              <div class="modal-footer">

                <button type="submit" class="modal-action modal-close waves-effect waves-
green btn-flat">Submit</button>

              </div>

          </form>

          </div>


          <script>

          $(document).ready(function(){

                $('.modal').modal();

                $('ul.tabs').tabs();

          });

          </script>
```

```
<div class="row" style="margin-top:2%">
 {{> utcard}}
</div>


</div>



<div id="test2" class="col s12 black">
<br>
        <div class="row">
                {{> ftcard}}
        </div>
</div>



<div id="test3" class="col s12 black">

<div class="row">
<p class="flow-text" style="color:white;text-align:center">Add Friends</p>
</div>

<div class="row">
```

```
                <div class="col s10 offset-s1">
                        <form class="add_friend">
                                <div class="input-field" style="background-color:white">
                                  <input id="search" placeholder="SEARCH FRIENDS HERE" type="search" name="fsearch" required>
                                  <label class="label-icon" for="search"></label>
                                  <i class="material-icons">close</i>
                                </div>

                                <button type="submit" class=" center-align modal-action modal-close waves-effect waves-white btn-flat col s2 offset-s5 cyan white-text">Add</button>

                        </form>
            </div>
  </div>
            <hr>
            <div class="row">
           <p class="flow-text" style="color:white;text-align:center">Friend List</p>
           </div>

           {{> friend_list}}
```

```
        </div>


  </div>
</div>
 {{/if}}
</template>
```

utcards.js

```
import { Template } from 'meteor/templating';

import { Tweet } from '../api/tasks.js';

import './utcard.html';
import './friend.js';

Template.utcard.helpers({
  u_posts() {
        return Tweet.find({uid : Session.get("id")});
  },
});

Template.utcard.events({
```

```
  'submit .remtitle'(event){


  event.preventDefault();


  var target = event.target;
  var title = target.rem.value;


  Meteor.call('tweet.remove', Session.get("id"), title);
  },
});
```

utcards.html

```
<template name="utcard">
        {{#each u_posts}}
                <div class="col s10 offset-s1">
                  <div class="card white">
                        <div class="card-content">
                          <h5 align="center"><b>Title :</b> {{t_title}} <a
href="#remove{{t_title}}"><i class="material-icons right black-text">close</i></a></h5>
                          <hr>
                          <p align="center">{{t_message}}</p>
                        </div>
                  </div>
                </div>
```

```
        <div id="remove{{t_title}}" class="modal">
            <div class="modal-content">
            <form class="remtitle">
              <h5 align="center">Are you sure you want to remove tweet?</h5>
            <hr><br>
              <button type="submit" name="rem" value="{{t_title}}" class="center-align modal-action modal-close waves-effect waves-white btn-flat col s2 offset-s5 cyan white-text">Remove</button>
              <br><br>
              </form>
            </div>
        </div>

            <script>
        $(document).ready(function(){
            $('.modal').modal();
        });
        </script>
      {{/each}}
</template>
```

ftcards.js

```
import { Template } from 'meteor/templating';
```

```
import { Tweet } from '../api/tasks.js';
import { Friend } from '../api/tasks.js';

import './ftcard.html';
import './utcard.js';

Template.ftcard.helpers({
  friend_post() {
  var fid = _.uniq(Friend.find({id:Session.get("id")}, {
                                                    sort: {u_friend: 1}, fields: {u_friend: true}
                                                    }).fetch().map(function(x) {
                                                            return x.u_friend;
                                                    }), true);
    return Tweet.find({uid : {$in : fid} });
  },
});
```

ftcards.html

```
<template name="ftcard">
        {{#each friend_post}}
                <div class="col s10 offset-s1">
                  <div class="card white">
                        <div class="card-content">
                          <h5 align="center"><b>{{uid}}</b></h5>
```

```
                        <hr>
                        <p align="center">{{t_message}} </p>
                      </div>
                    </div>
                  </div>
        {{/each}}
</template>
```

friend.js

```
import { Template } from 'meteor/templating';

import { Friend } from '../api/tasks.js';

import './friend.html';

Template.friend_list.helpers({
  friends() {
        return Friend.find({id: Session.get("id")});
  },
});

Template.friend_list.events({
  'submit .remfriend'(event){
```

```
  event.preventDefault();


  var target = event.target;
  var fid = target.remf.value;


  Meteor.call('friend.remove', Session.get("id"), fid);
  },
});
```

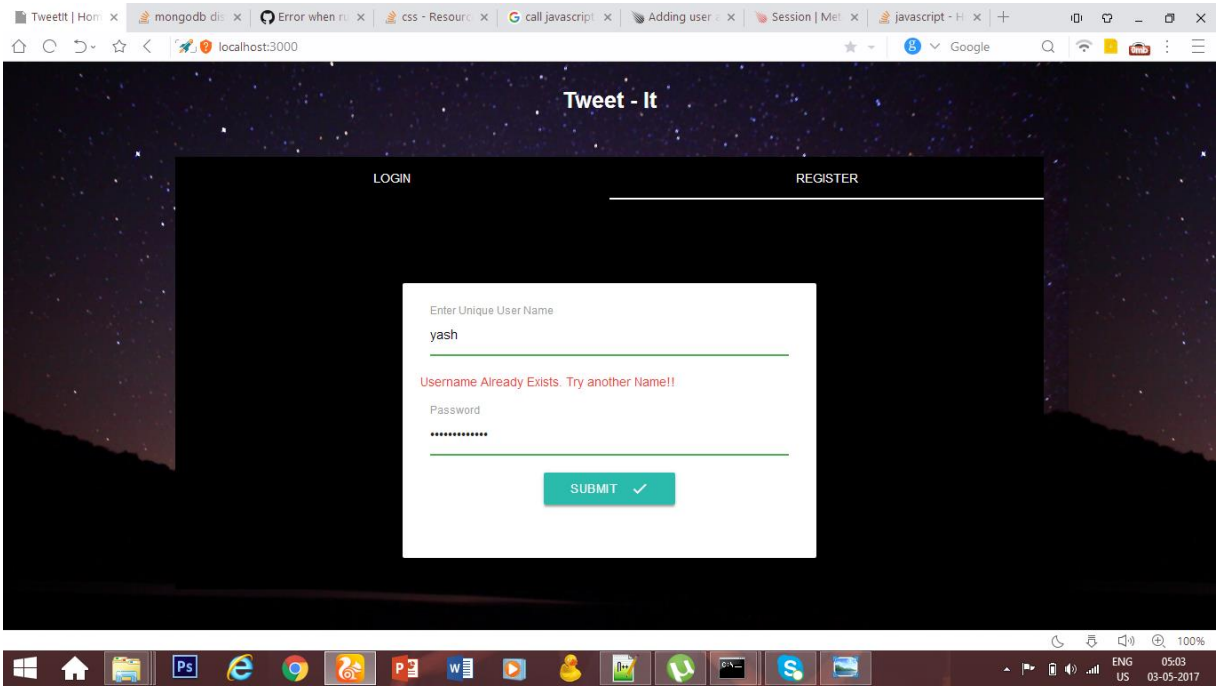
friend.html

```
<template name="friend_list">
<ul class="collection col s10 offset-s1 white">
        {{#each friends}}
                <li class="collection-item avatar col s12"><br>
                  <p class="flow-text">{{u_friend}} </p>
                   <a href="#remove{{u_friend}}" class="secondary-content"><i class="material-icons right black-text">close</i></a>
                </li>
                <div id="remove{{u_friend}}" class="modal">
                        <div class="modal-content">
                                <form class="remfriend">
                                  <h5 align="center">Are you sure you want to remove {{u_friend}} as your friend?</h5>
                                  <hr><br>
```

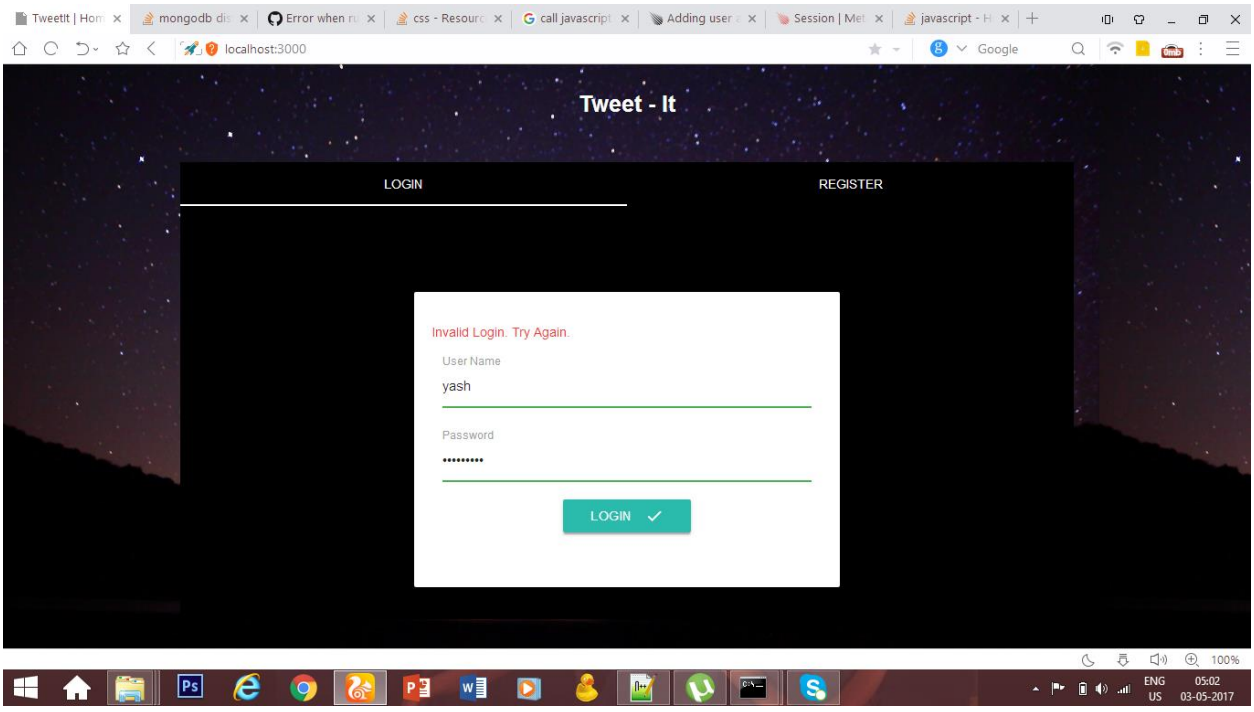
```
                                    <button type="submit" name="remf" value="{{u_friend}}" class="center-align modal-action modal-close waves-effect waves-white btn-flat col s2 offset-s5 cyan white-text">Remove</button>

                                    <br><br>

                                </form>

                            </div>

        </div>


                <script>

        $(document).ready(function(){

                $('.modal').modal();

        });

        </script>

       {{/each}}

</ul>

</template>
```

SCREENSHOTS

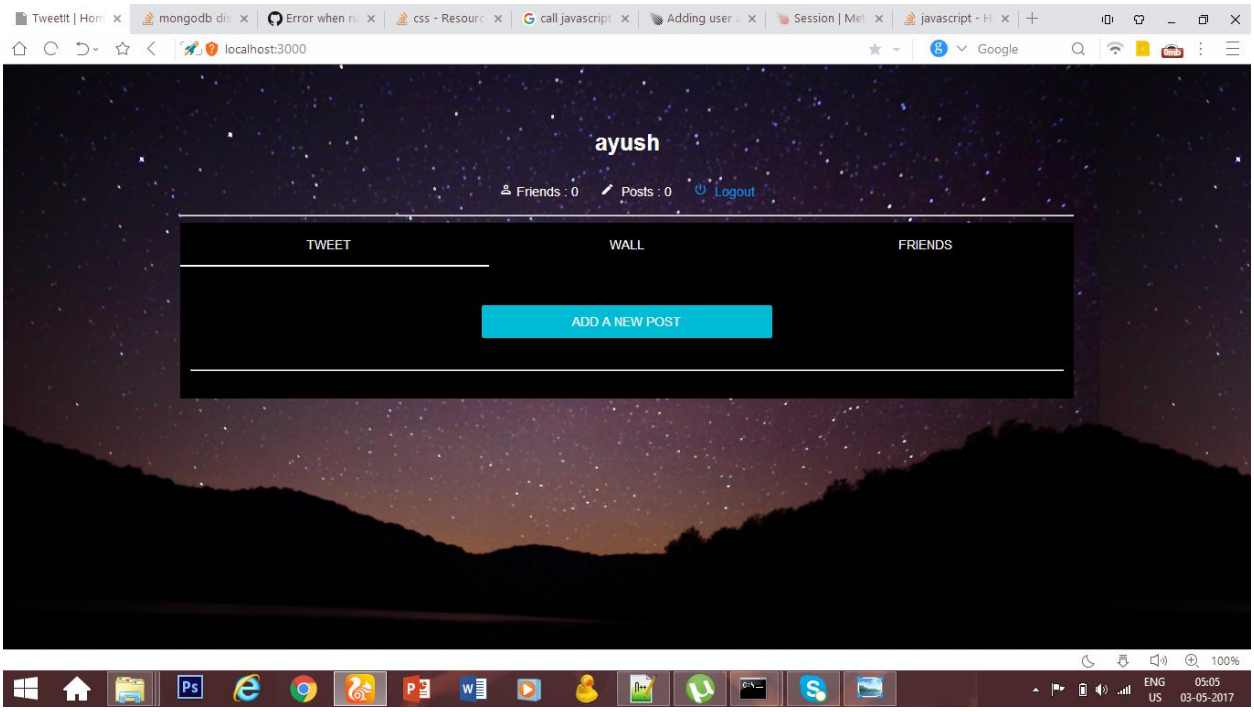
Registration- Enter unique name and password to register, if name entered isn't unique, it shows error.



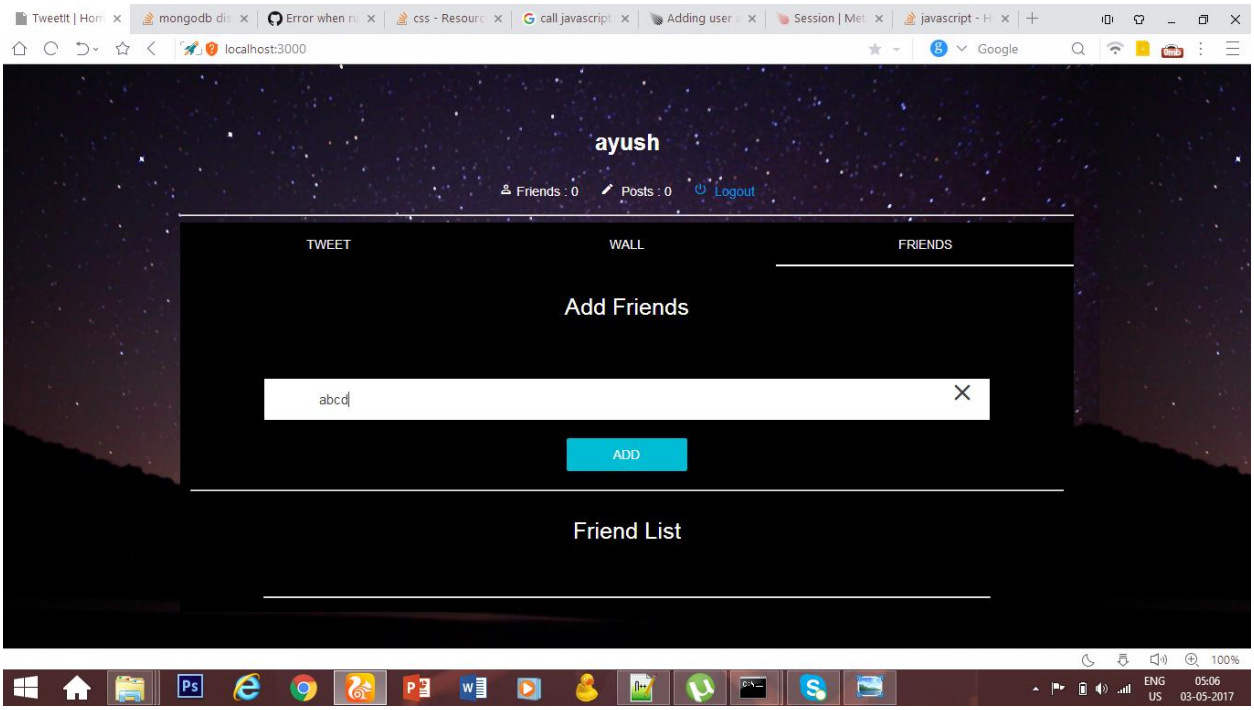
Login- enter name and password to login, if details aren't correct it will show error.



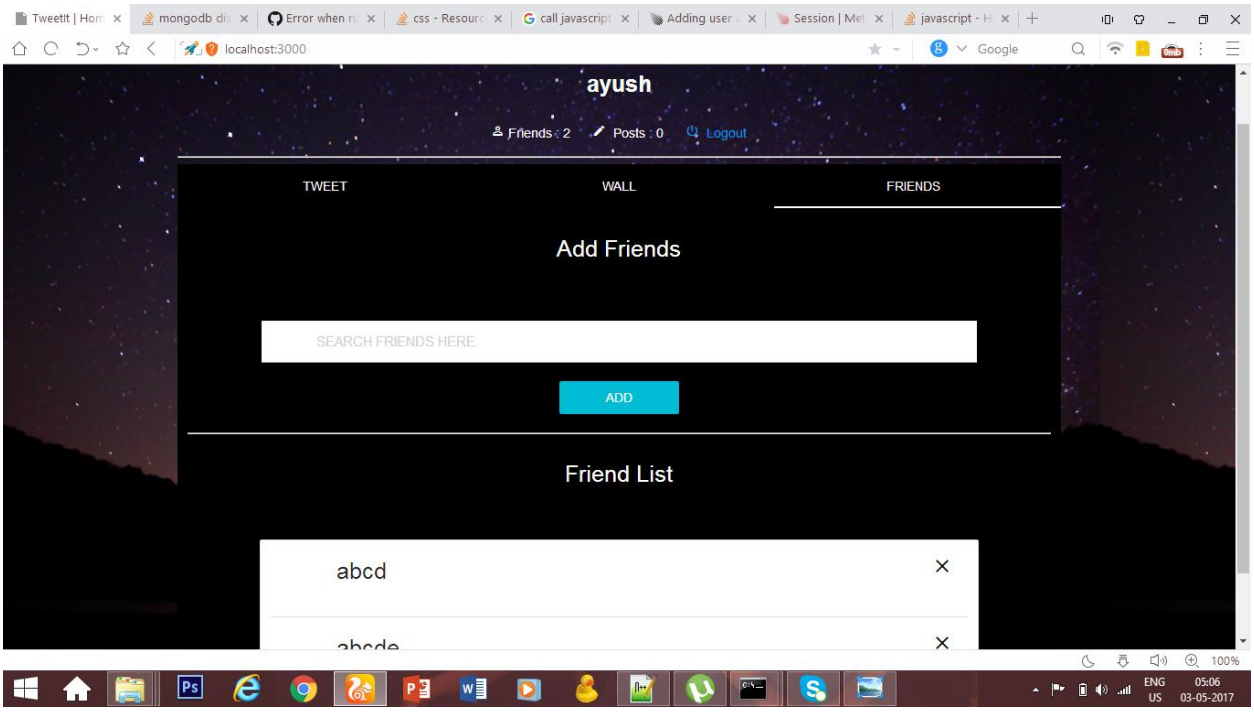
Home Page- correct login details will take you to home page



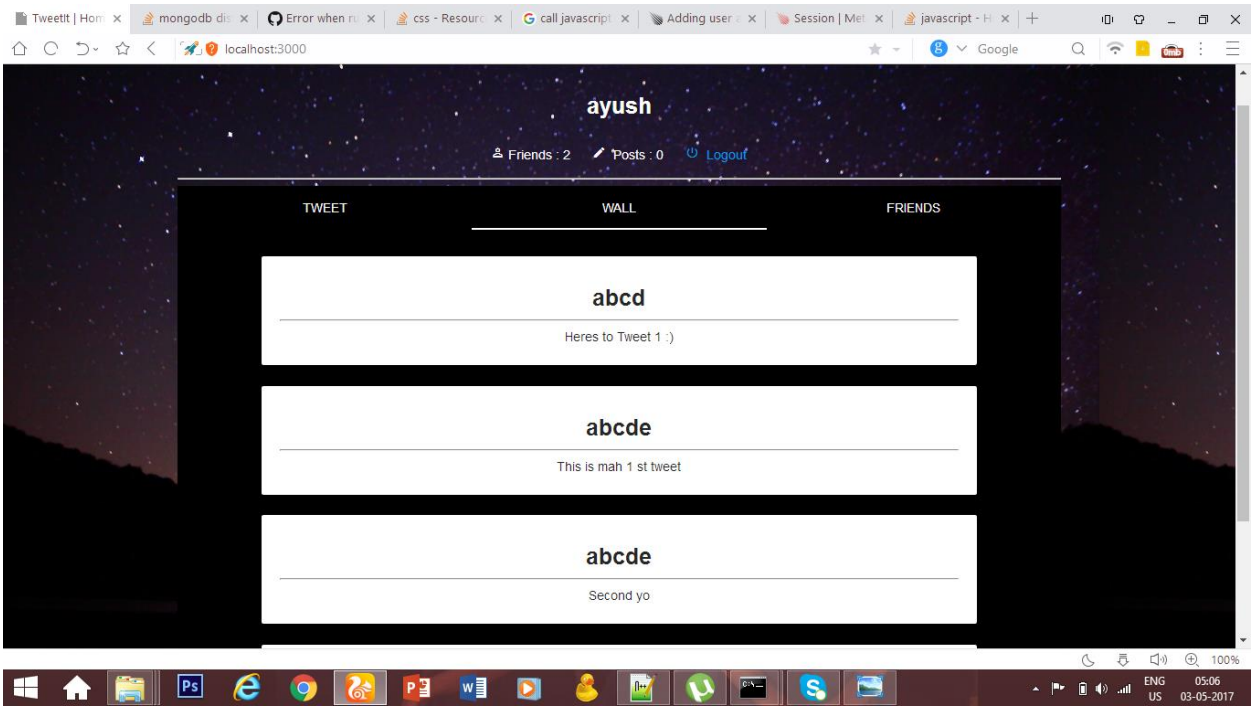
Friends- clicking on friends tab will take you to a page with search bar where we can search friends and add them



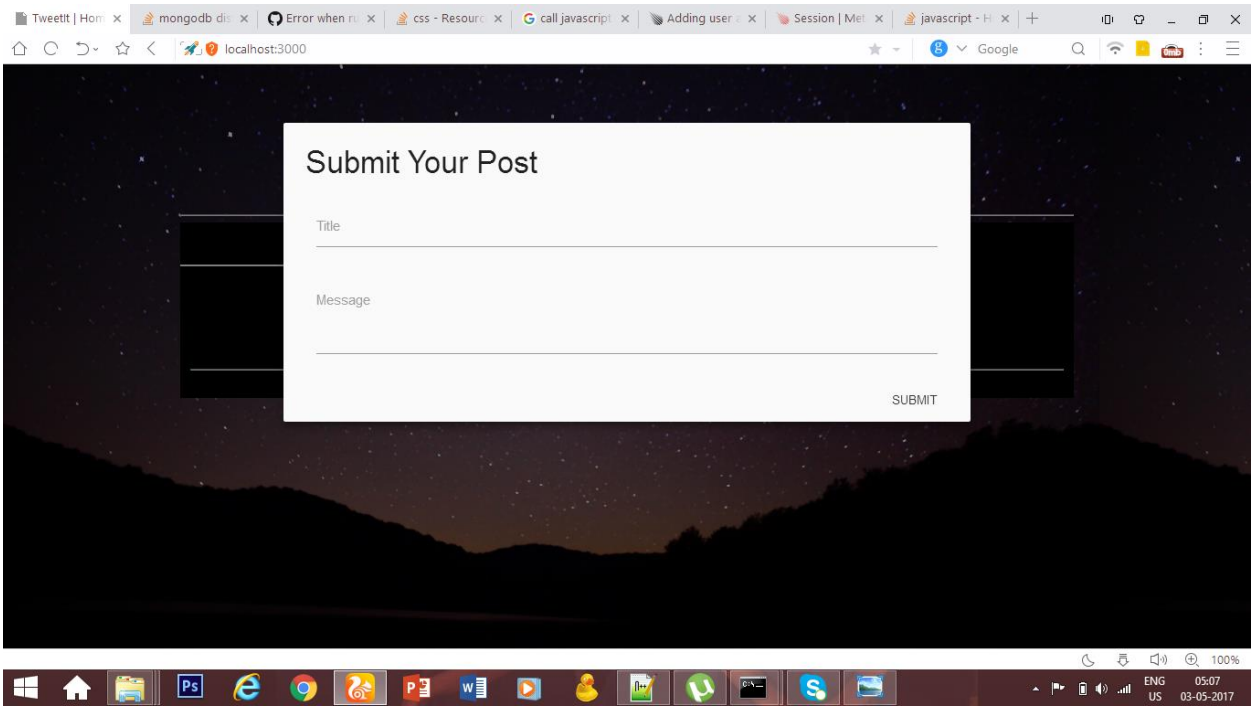
Added Friends- the list of all added friends is visible

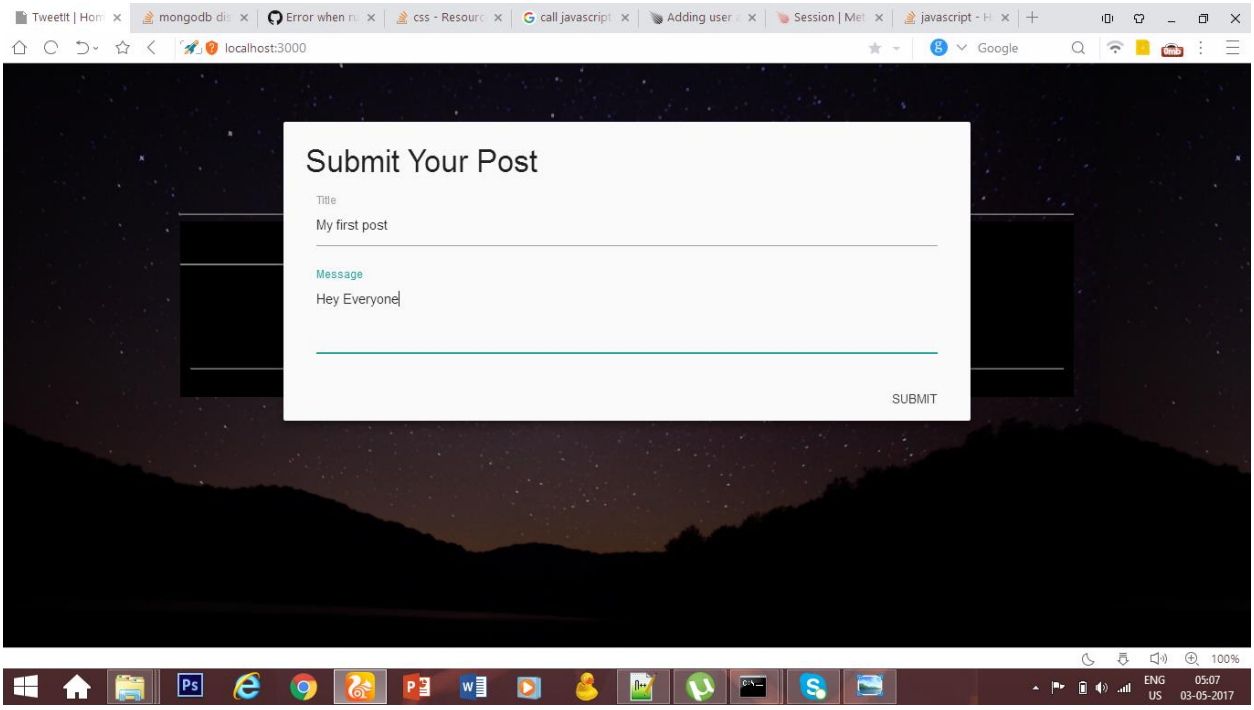


Wall- the wall shows the tweets of all added friends. Also the count of friends is shown on the top.

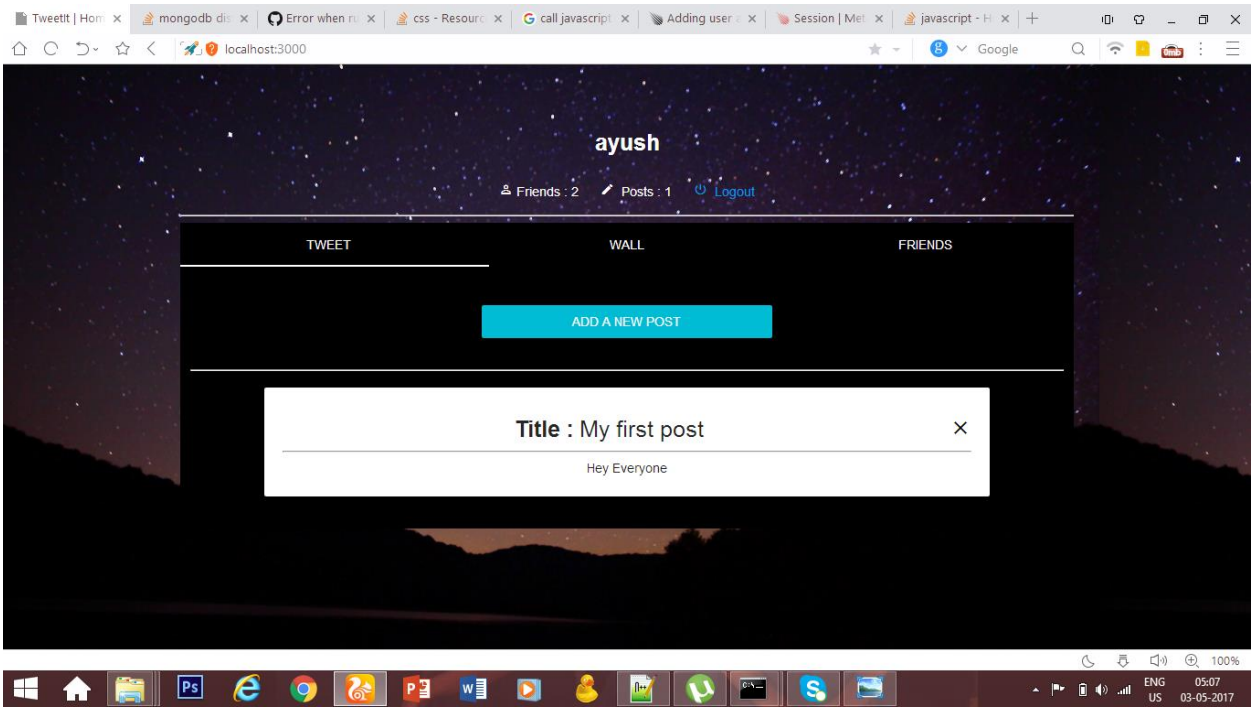


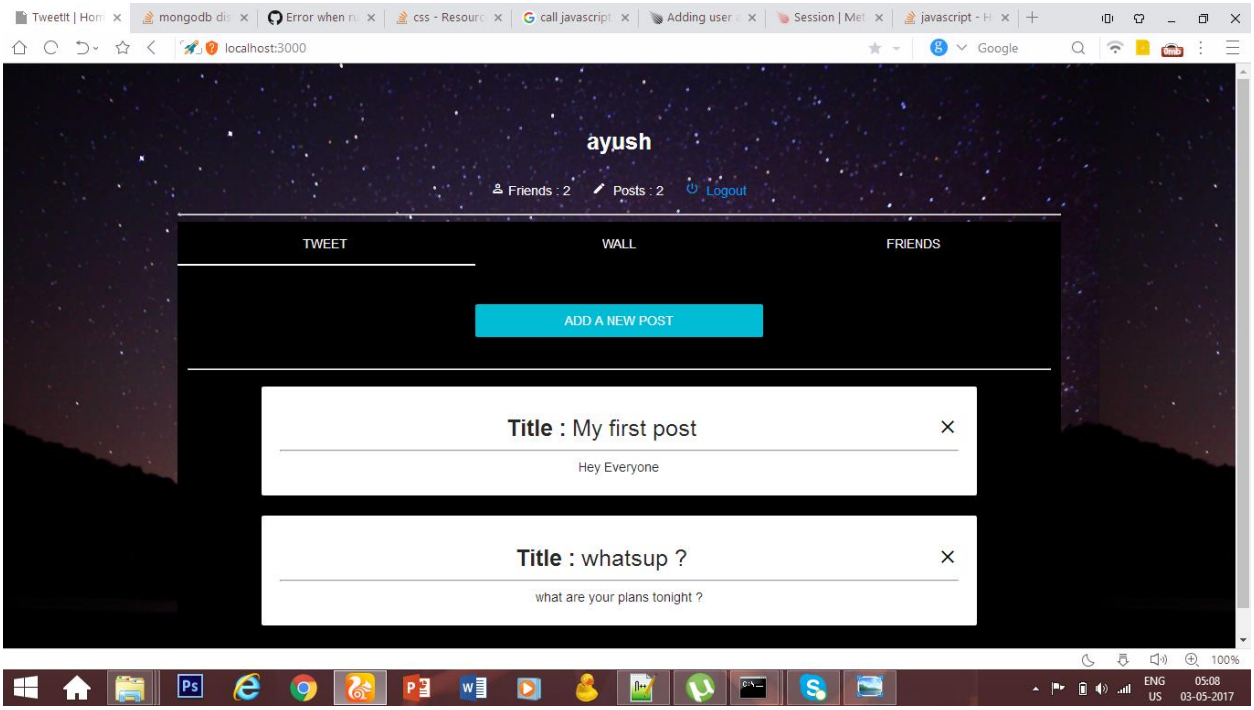
Add a new post- Clicking the tweet tab followed by ‘add a new post’ will open a dialogue box where user can add the title along with the message which will be posted on his wall.



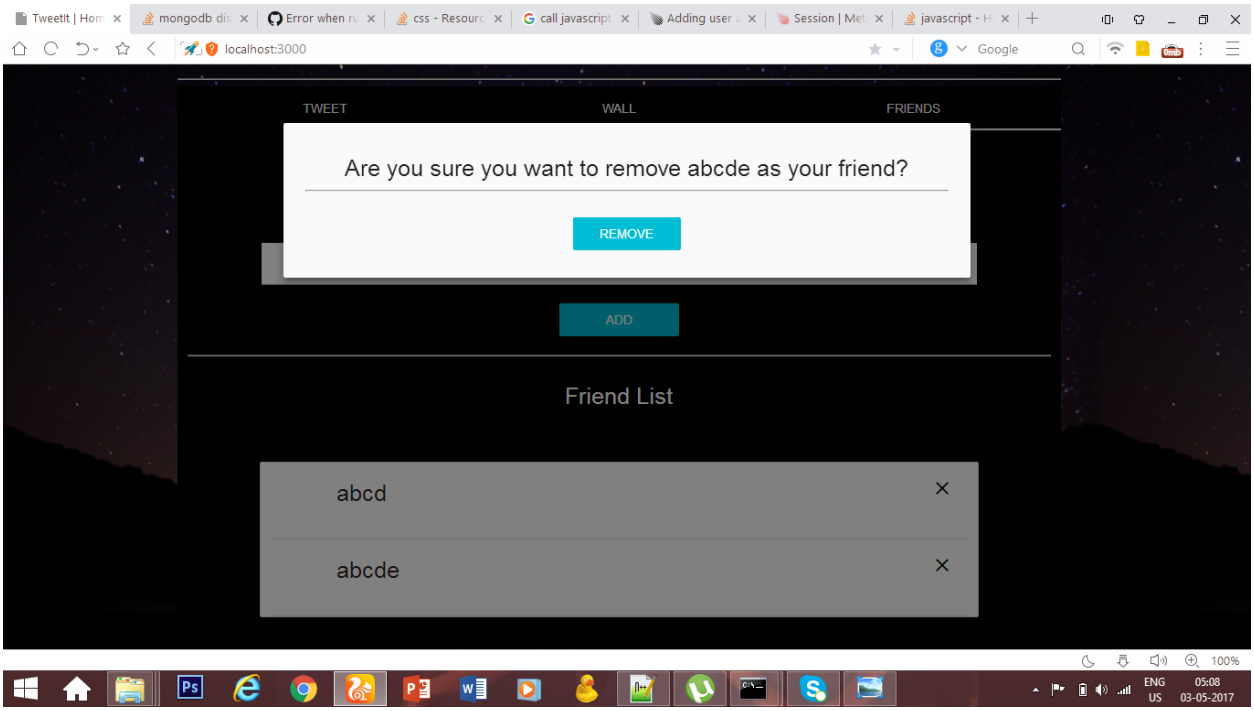


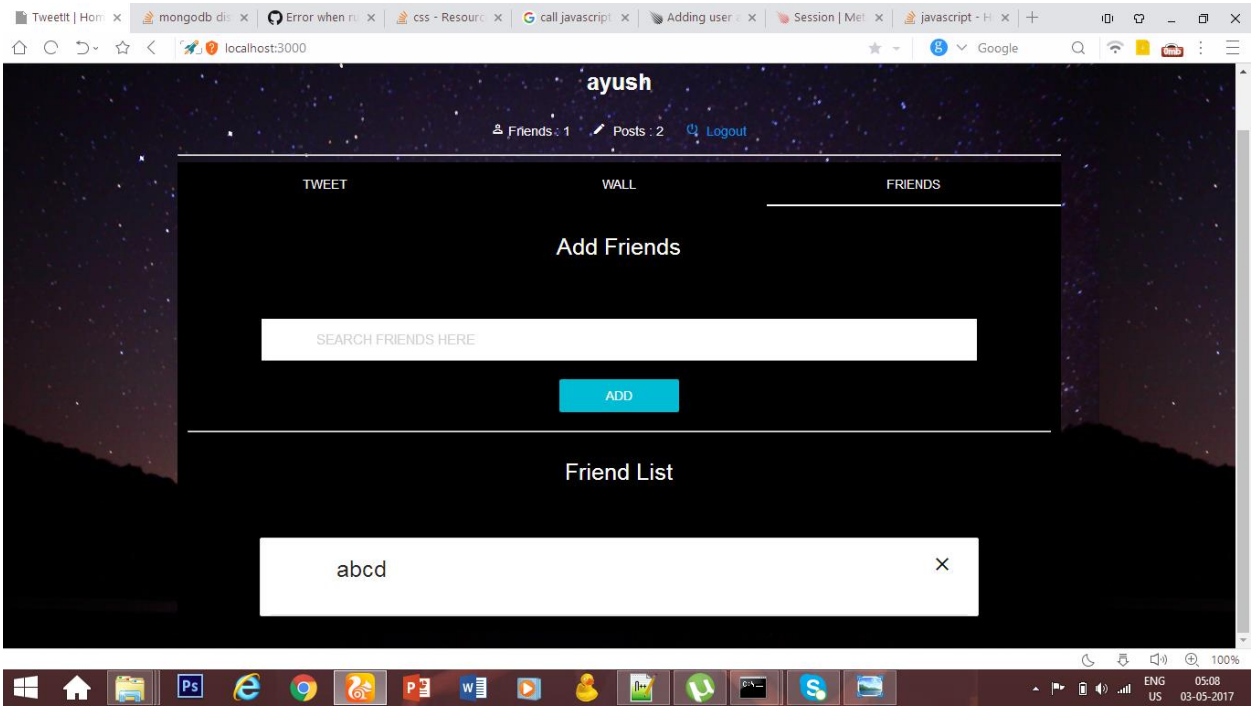
Tweet- The tweet wall displays all the tweets posted by the user. Also the count of tweets posted is visible at the top.



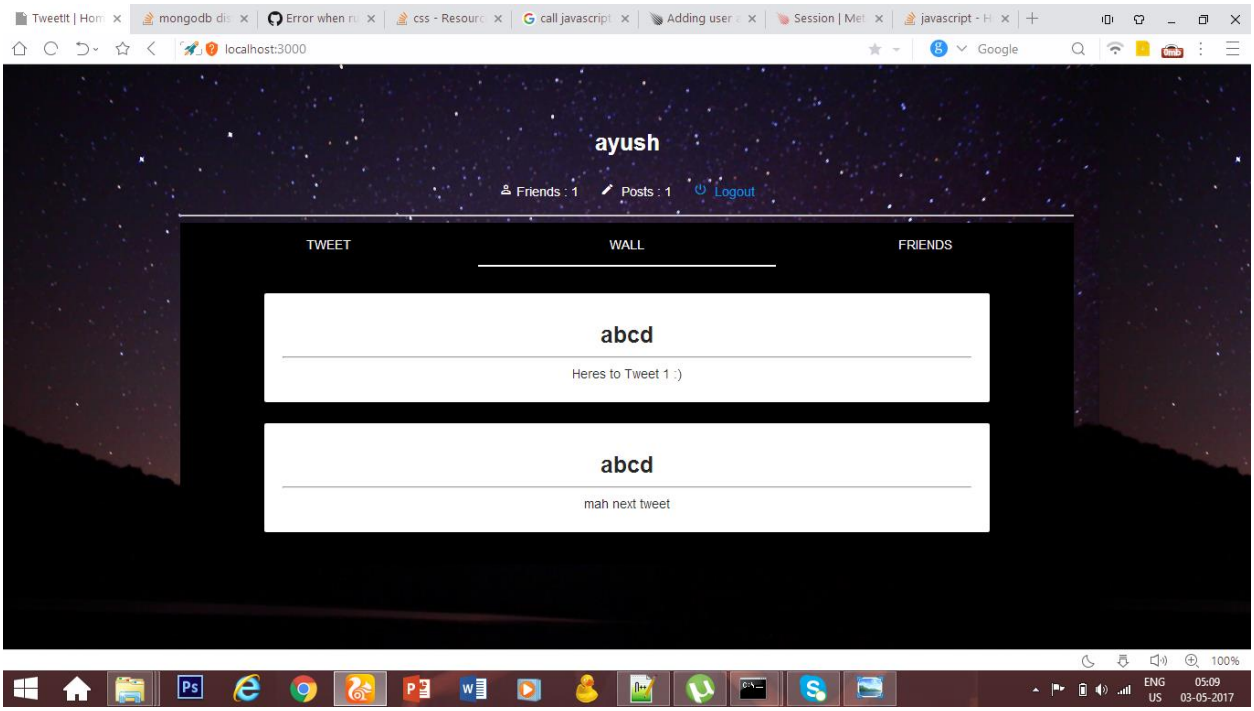


Remove Friends- Clicking on the cross button allows user to delete friends from his friend-list, deleted friends won't be able to view user's tweets.

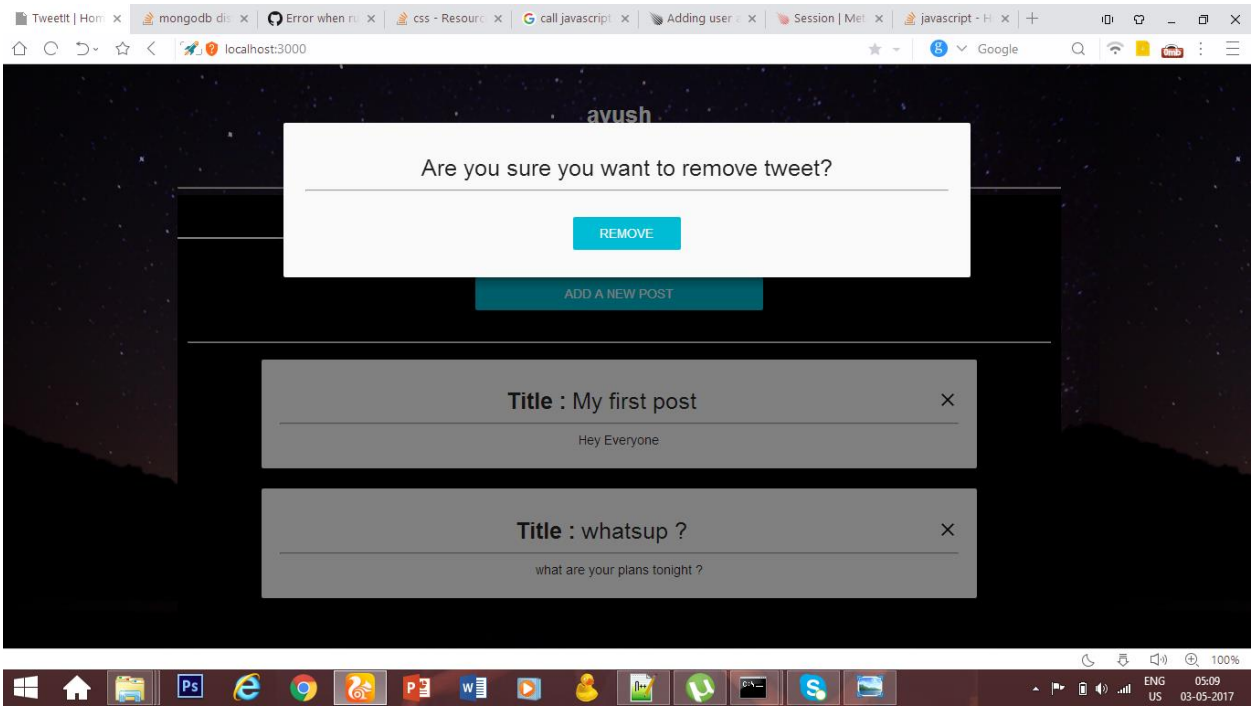




Updated Wall- The wall now doesn't show tweets of friends which were removed from the list.



Remove Tweet- Similarly, user can delete his tweets from his wall if he wants to.



Updated wall- Now the wall shows the remaining tweets of the user. User can now Logout using the Logout option on the top.