SFEVENTS

By:

Team 2

CSC648-848 Spring 2019

Tejal Ghadge (Team Lead/Backend-End)

Yujin Li (Front-End Lead)

David Lopez Martinez (Back-End Lead)

Benjamin Louie (GitHub Master/Back-End)

Wenying Chen (Front-End)

Milestone 4

May 14, 2019

Milestone	***Date submitted for review/ feedback integrated***
*M*ilestone 4	May 14, 2019
Milestone 3	*April 26, 2019*
Milestone 2 version 2	*April 20, 2019*
Milestone 2 version 1	*April 03, 2019*
Milestone 1 version 2	*April 02, 2019*
Milestone 1 version 1	*March 14, 2019*

1. Product Summary

Name of Product - **SFEvents**

1. All unregistered users are able to register to a new account and other users can Log into an existing account (and logout afterwards).
2. User can search for other event and check the details of the event without registering to SFEvents.
3. User can book the event after registering to account
4. All registered users can login to the account and create an event
5. Register user can see all created events by him in his account
6. Register user can send the invitations by mail to recipients listed in the event
7. Register user is able to update his own event
8. Register user can navigate and search the event
9. Registered user can report any fake event to admin
10. Register user can log out from the account
11. User with admin role can see all reported users and events
12. Admin can verify the reported event manually
13. Admin can block the event if found as invalid / fake

URL

http://ec2-18-222-238-235.us-east-2.compute.amazonaws.com/

Greetings and say goodbye to your lonely late night reheatable dinner for one. We are SFEvents and we provide only the best event service around to get you out of the house on those lonely Saturday nights. Now you might think that there are other sites out there such as EventBrite, why bother making another account for a different event page. Our sites offer more ways for you to search events without having to even bother making an account until you find an event you're interested in. Once you're registered, you'll be taken to our beautiful home page where you can search for fun things to do and even host one yourself if you're feeling up to it. Either way SFEvents is the way to go to get yourself out of the house and find something fun to do tonight.

2.Usability Test Plan

Test objectives

Purpose:

- To verify that SFEVENTS is a platform to create events and invite people with ease.
- To verify that SFEvents is easy to use for the average user and that the user would be very likely to use SFEvents to find an event they would want to attend. User can check the event and report the event to admin if he find it fake.

Problem Statement:

Is the main function of SFEvents(finding an event) easy to use for the average user and easy to report event to admin if found as invalid.

Test description:

System setup:

The website is hosted on an amazon web service running on a Windows Machine

Starting Point:

Users will start on the default home page of SFEvents(logged out) on a Mozilla Firefox browser of Windows 10. We have 9 events in our MySQL database.
URL: http://ec2-18-222-238-235.us-east-2.compute.amazonaws.com/#/

Test Monitor Role:

Provide access to SFEvents and watch as users try to complete the tasks silently.

Legal Issues:

The test is completely voluntary, and the user’s name will not be kept so that the user will not be identified later on. The only information that is kept is the user’s experience with the website and nothing else. Any personal information entered when the user registered is deleted after the test.

Report:

The report afterwards will contain information based on how easy users found it to navigate and operate SFEvents

Usability Task description

1. Users will attempt to do a search for an event in San Francisco and view that event's details.

2. User will next try to create and register for an account on the website and then register for an event in San Francisco.

Task 1: Search for any event in San Francisco and view the event's location

Task	Find the location of an event in San Francisco
Machine State(starting point)	Homepage of SFEvents http://ec2-18-222-238-235.us-east-2.compute.amazonaws.com
Success Criteria	Seeing the address of an event in San Francisco
Benchmark	8 seconds

Task 2: Create Event and send invitations to people

Task	Register to SFEvents, create an event and send invitation through mail
Machine State	Home page of SFEvents http://ec2-18-222-238-235.us-east-2.compute.amazonaws.com/#/
Success Criteria	Event invitation is sent to all intended recipients
Benchmark	4 minutes

3. Questionnaire
Please check the statements that correspond to your experience with the search function.

You had trouble using the search function
❑ Strongly Disagree
❑ Disagree
❑ Neutral
❑ Agree

❏ Strongly Agree

You had trouble searching for all the specified terms
❏ Strongly Disagree
❏ Disagree
❏ Neutral
❏ Agree
❏ Strongly Agree

The non-alphabetic characters affected your search results negatively
❏ Strongly Disagree
❏ Disagree
❏ Neutral
❏ Agree
❏ Strongly Agree

You had trouble retaining the search entries
❏ Strongly Disagree
❏ Disagree
❏ Neutral
❏ Agree
❏ Strongly Agree

The search results were relevant to the search's category
❏ Strongly Disagree
❏ Disagree
❏ Neutral
❏ Agree
❏ Strongly Agree

The register button was in an obvious place
❏ Strongly Disagree
❏ Disagree
❏ Neutral
❏ Agree
❏ Strongly Agree

The registration process was fast and straightforward
❏ Strongly Disagree
❏ Disagree
❏ Neutral
❏ Agree

❑ Strongly Agree

The event creation captures all required inputs
❑ Strongly Disagree
❑ Disagree
❑ Neutral
❑ Agree
❑ Strongly Agree

The invitation with correct event was send to recepient
❑ Strongly Disagree
❑ Disagree
❑ Neutral
❑ Agree
❑ Strongly Agree

3. QA Test Plan

Test objectives

Purpose:
The purpose of Quality Assurance testing is to validate that all system functionalities are meeting the requirements. Testing is vital in order to meet specifications mentioned in previous milestones.Testing is performed on priority one functionalities.

Problem Statement:
To verify that SFEvents functionalities are at a point where they are performing according to the specifications.

Test plan

We will have users follow test cases that we will list below to find and view for a specific event. Users are not required to log in or register to search for data.

Hardware Setup-
Website on Amazon Web Services running on a windows machine with 64 bit architecture.

Software Setup-
SFEvents homepage that is in its default state(logged out) on Google Chrome(74.0.3729.131) browser of Windows 10.

Features we are going to test: (search)
Events can be searched by entering the name of the event, location, or category in the search bar.

The search feature will be tested through the following 3 test cases below:

1. Search for an event in San Francisco using the search bar
2. Search for an event with the words “euge” using the search bar
3. Search something with a number in its string
4. Use search without putting anything in text box

Test Cases

Number	Description	Text Input	Expected Result	Pass/Fail
1	Search for an event that takes place in San Francisco with the search bar	“San francisco”	:= 2 returns event matching the location with san francisco	PASS
2	Search for an event with the words “euge” using the search bar	“euge”	:= 3 returns event which contains euge substring in name, description or location	PASS
3	Search for an event using a number	“1”	:= returns event containing 1 in name, description or location	PASS
4	Click search without typing any text in the search bar	“”	:=displays list of events from closest date	PASS
5	Filter events based on date	Start date or End date	:=return event matching to Start date or End date	ON TRACK

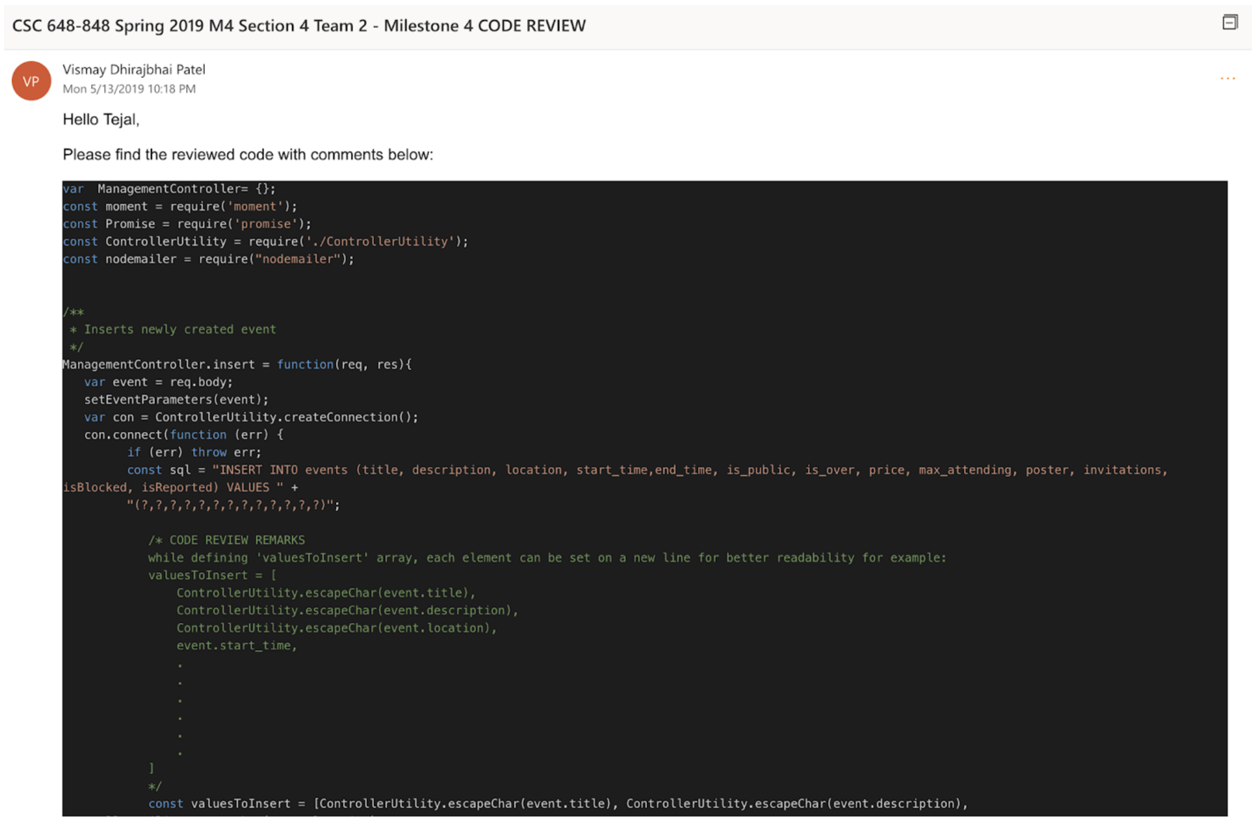
4.Code Review

Coding Style

We try to write self-explanatory variables and try to make it so that almost readable to the degree that we do not need documentation. We only add code to the dev branch and each frontend and backend team has their directories as to not conflict with one another. Once anyone finishes a task, they immediately notify the group. After a task is finished, other group members will also test the code on their own devices to confirm that it is working. If the code is not working, the person will be informed and will continue to work on said task. The github master will fix any merge conflicts that happen by managing pull requests and ensuring each branch is backed up before fixing the merge conflicts in case anything unexpected happens. The master branch always has the working version and is only touched during milestone due dates.

Code Example and Review – Mail

Review - 1



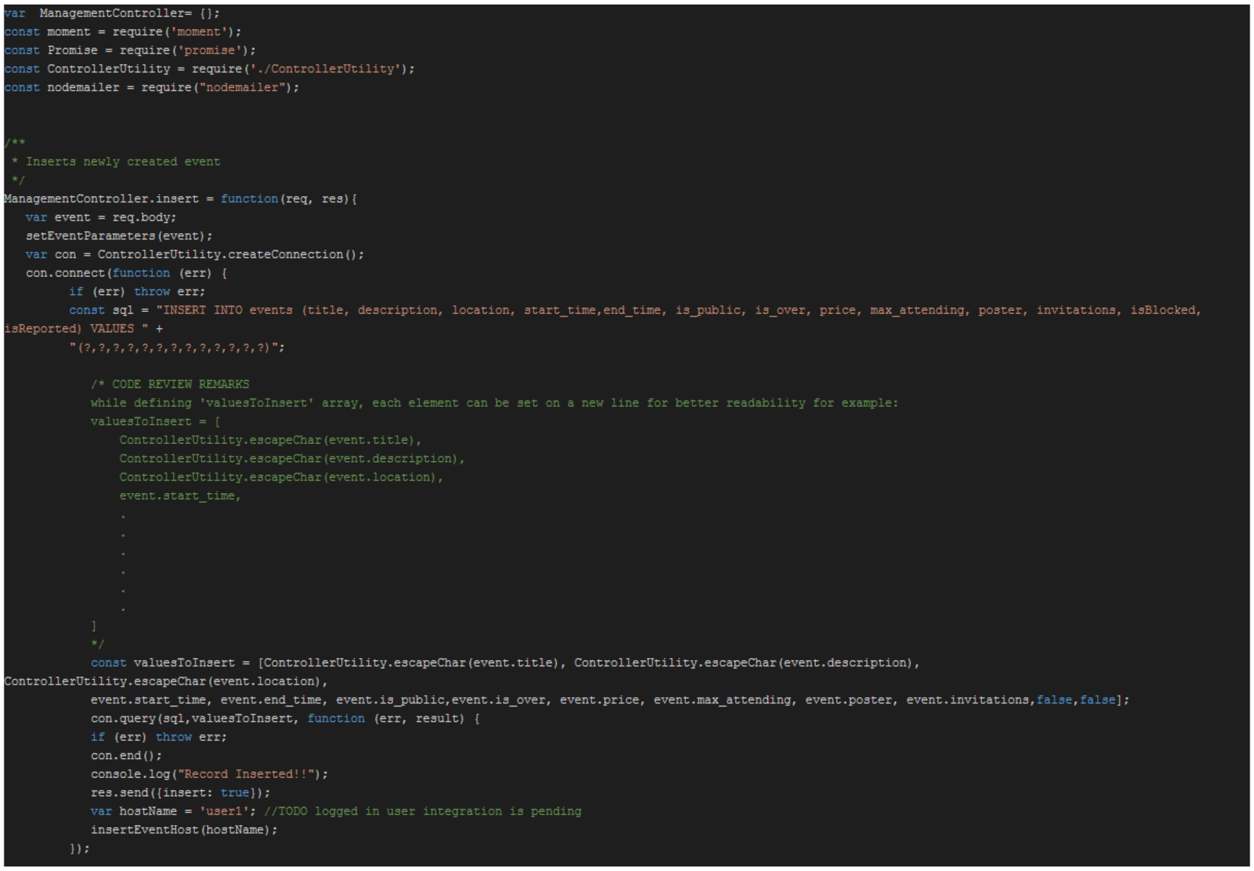
CSC 648-848 Spring 2019 M4 Section 4 Team 2 - Milestone 4 CODE REVIEW

Vismay Dhirajbhai Patel
Mon 5/13/2019 10:18 PM

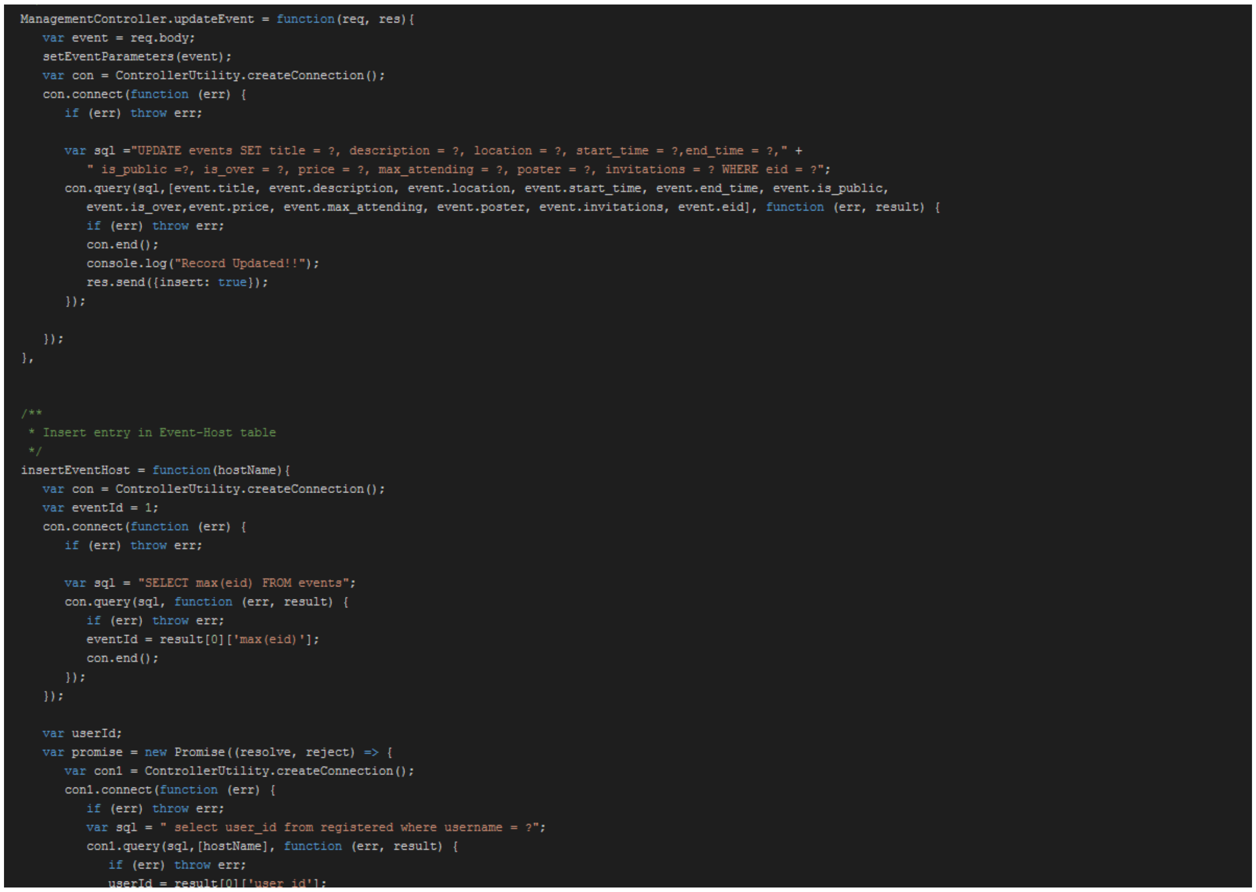
Hello Tejal,

Please find the reviewed code with comments below:

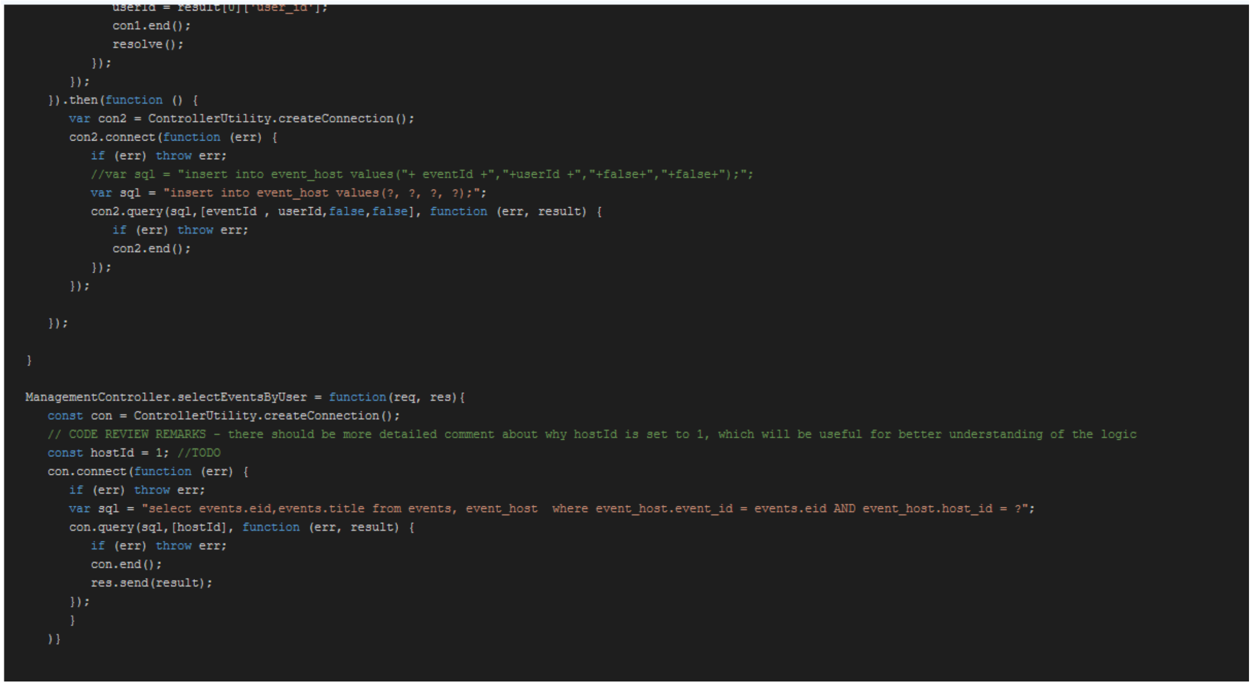
```
var  ManagementController= {};
const moment = require('moment');
const Promise = require('promise');
const ControllerUtility = require('./ControllerUtility');
const nodemailer = require("nodemailer");


/**
 * Inserts newly created event
 */
ManagementController.insert = function(req, res){
   var event = req.body;
   setEventParameters(event);
   var con = ControllerUtility.createConnection();
   con.connect(function (err) {
         if (err) throw err;
         const sql = "INSERT INTO events (title, description, location, start_time,end_time, is_public, is_over, price, max_attending, poster, invitations,
isBlocked, isReported) VALUES " +
         "(?,?,?,?,?,?,?,?,?,?,?,?,?)";

            /* CODE REVIEW REMARKS
            while defining 'valuesToInsert' array, each element can be set on a new line for better readability for example:
            valuesToInsert = [
                ControllerUtility.escapeChar(event.title),
                ControllerUtility.escapeChar(event.description),
                ControllerUtility.escapeChar(event.location),
                event.start_time,
                .
                .
                .
                .
                .
                .
            ]
            */
            const valuesToInsert = [ControllerUtility.escapeChar(event.title), ControllerUtility.escapeChar(event.description),
```



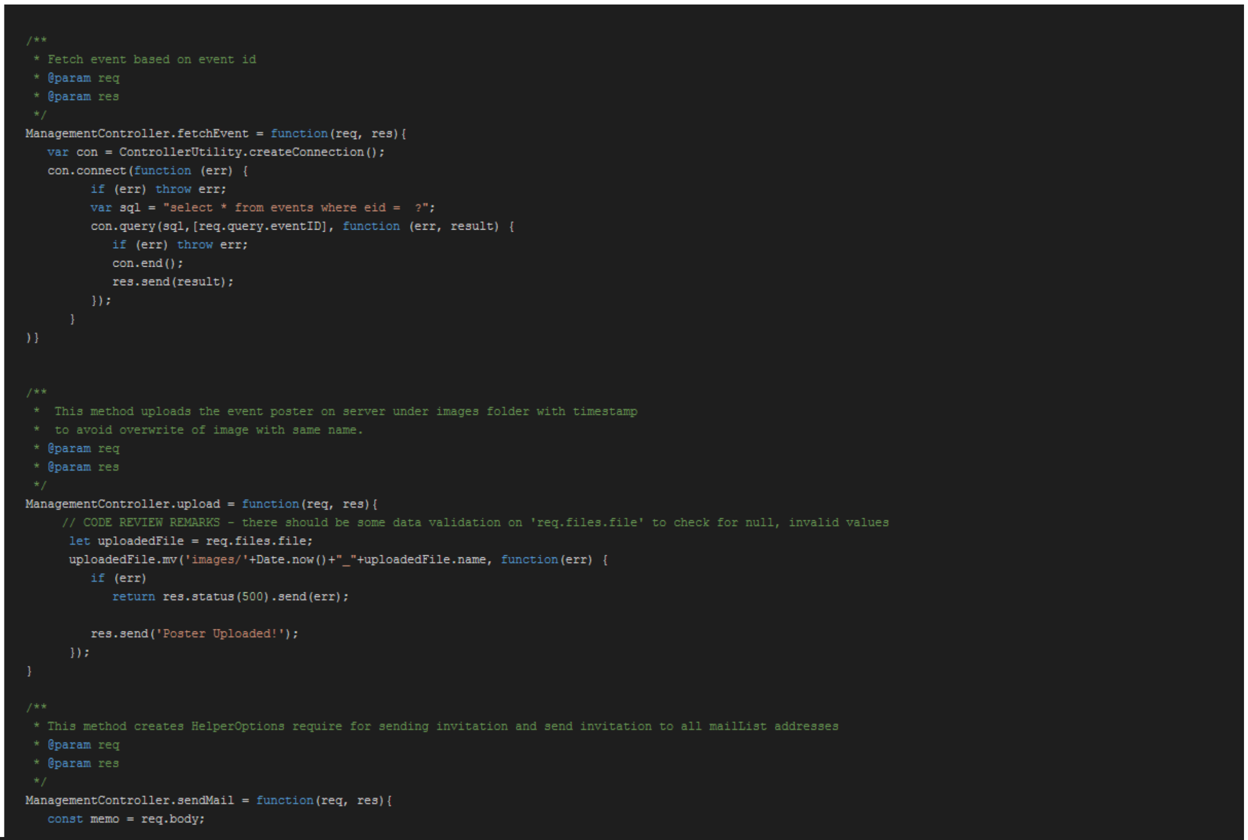
```
var  ManagementController= {};
const moment = require('moment');
const Promise = require('promise');
const ControllerUtility = require('./ControllerUtility');
const nodemailer = require("nodemailer");


/**
 * Inserts newly created event
 */
ManagementController.insert = function(req, res){
   var event = req.body;
   setEventParameters(event);
   var con = ControllerUtility.createConnection();
   con.connect(function (err) {
         if (err) throw err;
         const sql = "INSERT INTO events (title, description, location, start_time,end_time, is_public, is_over, price, max_attending, poster, invitations, isBlocked,
isReported) VALUES " +
         "(?,?,?,?,?,?,?,?,?,?,?,?,?)";

            /* CODE REVIEW REMARKS
            while defining 'valuesToInsert' array, each element can be set on a new line for better readability for example:
            valuesToInsert = [
                ControllerUtility.escapeChar(event.title),
                ControllerUtility.escapeChar(event.description),
                ControllerUtility.escapeChar(event.location),
                event.start_time,
                .
                .
                .
                .
                .
                .
            ]
            */
            const valuesToInsert = [ControllerUtility.escapeChar(event.title), ControllerUtility.escapeChar(event.description),
ControllerUtility.escapeChar(event.location),
            event.start_time, event.end_time, event.is_public,event.is_over, event.price, event.max_attending, event.poster, event.invitations,false,false];
            con.query(sql,valuesToInsert, function (err, result) {
            if (err) throw err;
            con.end();
            console.log("Record Inserted!!");
            res.send({insert: true});
            var hostName = 'user1'; //TODO logged in user integration is pending
            insertEventHost(hostName);
         });
```



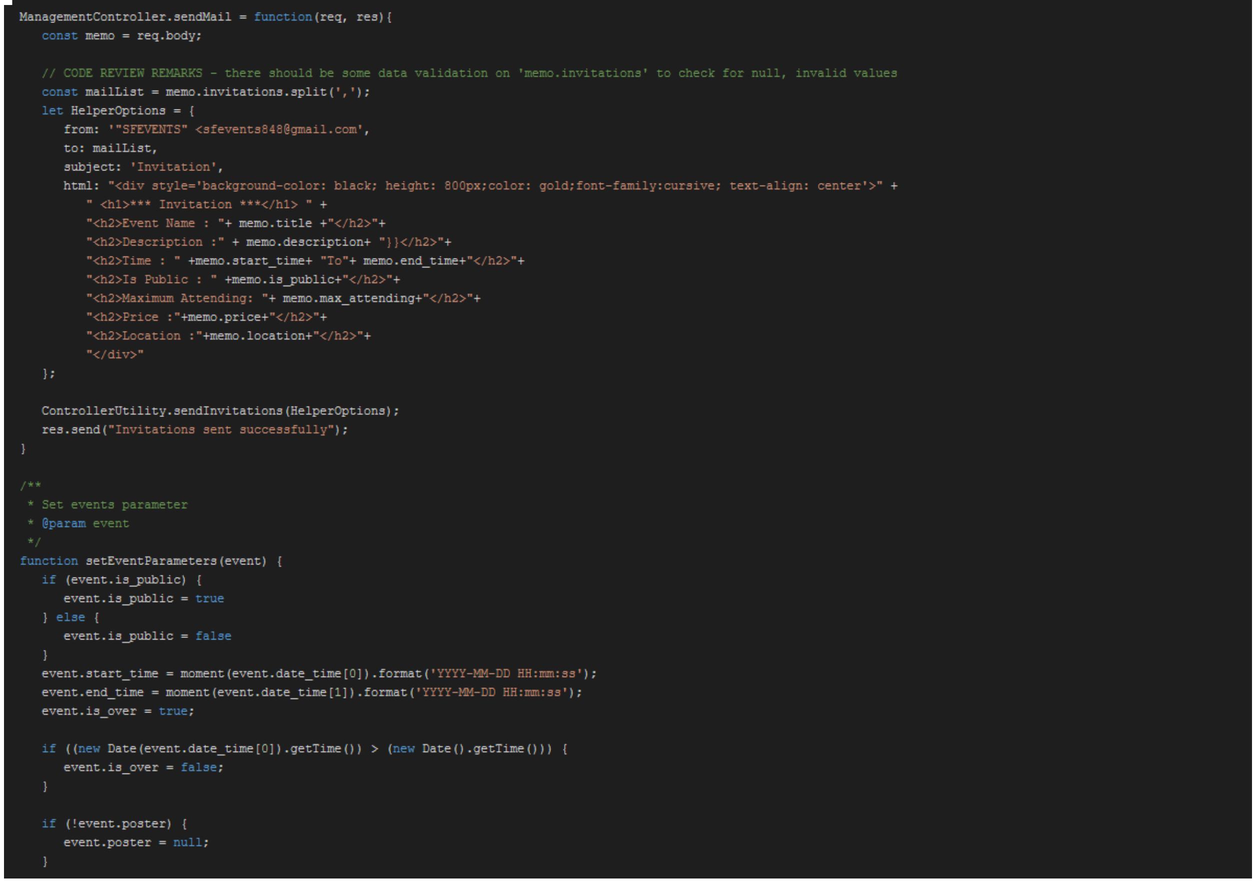
```javascript
ManagementController.updateEvent = function(req, res){
   var event = req.body;
   setEventParameters(event);
   var con = ControllerUtility.createConnection();
   con.connect(function (err) {
      if (err) throw err;

      var sql ="UPDATE events SET title = ?, description = ?, location = ?, start_time = ?,end_time = ?," +
         " is_public =?, is_over = ?, price = ?, max_attending = ?, poster = ?, invitations = ? WHERE eid = ?";
      con.query(sql,[event.title, event.description, event.location, event.start_time, event.end_time, event.is_public,
         event.is_over,event.price, event.max_attending, event.poster, event.invitations, event.eid], function (err, result) {
         if (err) throw err;
         con.end();
         console.log("Record Updated!!");
         res.send({insert: true});
      });

   });
},


/**
 * Insert entry in Event-Host table
 */
insertEventHost = function(hostName){
   var con = ControllerUtility.createConnection();
   var eventId = 1;
   con.connect(function (err) {
      if (err) throw err;

      var sql = "SELECT max(eid) FROM events";
      con.query(sql, function (err, result) {
         if (err) throw err;
         eventId = result[0]['max(eid)'];
         con.end();
      });
   });

   var userId;
   var promise = new Promise((resolve, reject) => {
      var con1 = ControllerUtility.createConnection();
      con1.connect(function (err) {
         if (err) throw err;
         var sql = " select user_id from registered where username = ?";
         con1.query(sql,[hostName], function (err, result) {
            if (err) throw err;
            userId = result[0]['user_id'];
```



```javascript
            con1.end();
            resolve();
         });
      });
   }).then(function () {
      var con2 = ControllerUtility.createConnection();
      con2.connect(function (err) {
         if (err) throw err;
         //var sql = "insert into event_host values("+ eventId +","+userId +","+false+","+false+");";
         var sql = "insert into event_host values(?, ?, ?, ?);";
         con2.query(sql,[eventId , userId,false,false], function (err, result) {
            if (err) throw err;
            con2.end();
         });
      });

   });

}

ManagementController.selectEventsByUser = function(req, res){
   const con = ControllerUtility.createConnection();
   // CODE REVIEW REMARKS - there should be more detailed comment about why hostId is set to 1, which will be useful for better understanding of the logic
   const hostId = 1; //TODO
   con.connect(function (err) {
      if (err) throw err;
      var sql = "select events.eid,events.title from events, event_host  where event_host.event_id = events.eid AND event_host.host_id = ?";
      con.query(sql,[hostId], function (err, result) {
         if (err) throw err;
         con.end();
         res.send(result);
      });
      }
   )}
```

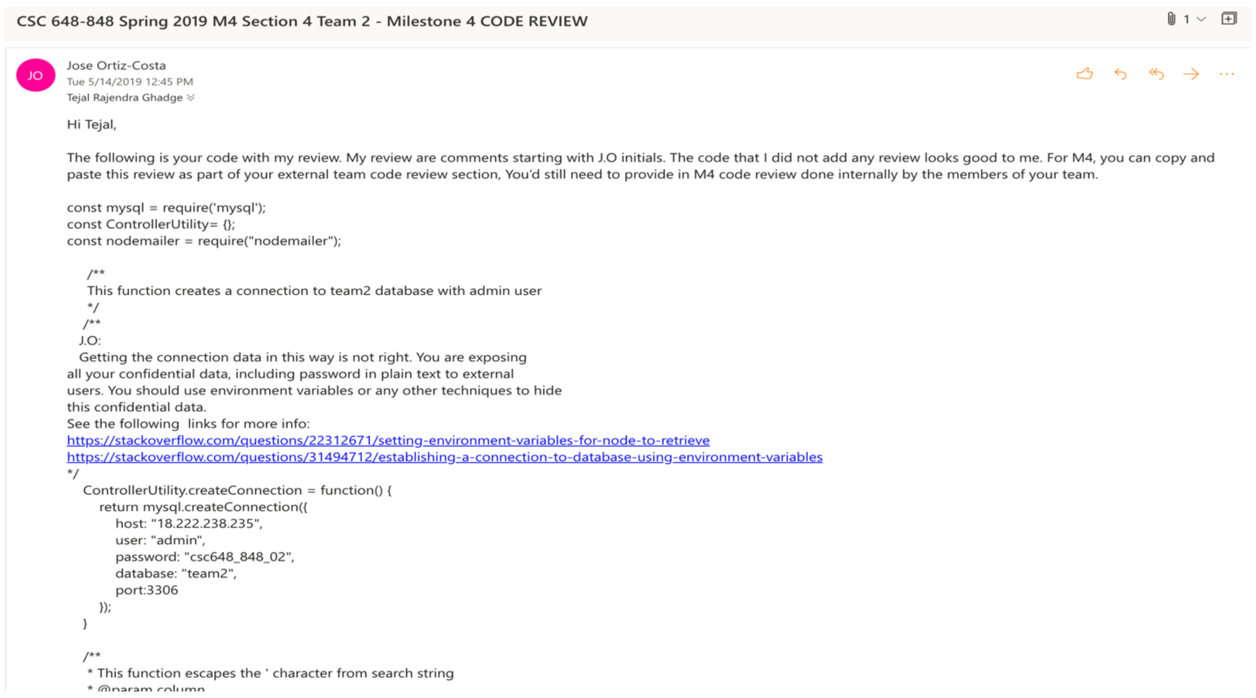


```
/**
 * Fetch event based on event id
 * @param req
 * @param res
 */
ManagementController.fetchEvent = function(req, res){
   var con = ControllerUtility.createConnection();
   con.connect(function (err) {
         if (err) throw err;
         var sql = "select * from events where eid =  ?";
         con.query(sql,[req.query.eventID], function (err, result) {
            if (err) throw err;
            con.end();
            res.send(result);
         });
      }
)}


/**
 *  This method uploads the event poster on server under images folder with timestamp
 *  to avoid overwrite of image with same name.
 * @param req
 * @param res
 */
ManagementController.upload = function(req, res){
     // CODE REVIEW REMARKS - there should be some data validation on 'req.files.file' to check for null, invalid values
      let uploadedFile = req.files.file;
      uploadedFile.mv('images/'+Date.now()+"_"+uploadedFile.name, function(err) {
         if (err)
            return res.status(500).send(err);

         res.send('Poster Uploaded!');
      });
}

/**
 * This method creates HelperOptions require for sending invitation and send invitation to all mailList addresses
 * @param req
 * @param res
 */
ManagementController.sendMail = function(req, res){
   const memo = req.body;
```



```
ManagementController.sendMail = function(req, res){
   const memo = req.body;

   // CODE REVIEW REMARKS - there should be some data validation on 'memo.invitations' to check for null, invalid values
   const mailList = memo.invitations.split(',');
   let HelperOptions = {
      from: '"SFEVENTS" <sfevents848@gmail.com',
      to: mailList,
      subject: 'Invitation',
      html: "<div style='background-color: black; height: 800px;color: gold;font-family:cursive; text-align: center'>" +
         " <h1>*** Invitation ***</h1> " +
         "<h2>Event Name : "+ memo.title +"</h2>"+
         "<h2>Description :" + memo.description+ "}}</h2>"+
         "<h2>Time : " +memo.start_time+ "To"+ memo.end_time+"</h2>"+
         "<h2>Is Public : " +memo.is_public+"</h2>"+
         "<h2>Maximum Attending: "+ memo.max_attending+"</h2>"+
         "<h2>Price :"+memo.price+"</h2>"+
         "<h2>Location :"+memo.location+"</h2>"+
         "</div>"
   };

   ControllerUtility.sendInvitations(HelperOptions);
   res.send("Invitations sent successfully");
}

/**
 * Set events parameter
 * @param event
 */
function setEventParameters(event) {
   if (event.is_public) {
      event.is_public = true
   } else {
      event.is_public = false
   }
   event.start_time = moment(event.date_time[0]).format('YYYY-MM-DD HH:mm:ss');
   event.end_time = moment(event.date_time[1]).format('YYYY-MM-DD HH:mm:ss');
   event.is_over = true;

   if ((new Date(event.date_time[0]).getTime()) > (new Date().getTime())) {
      event.is_over = false;
   }

   if (!event.poster) {
      event.poster = null;
   }
```

Review 2:



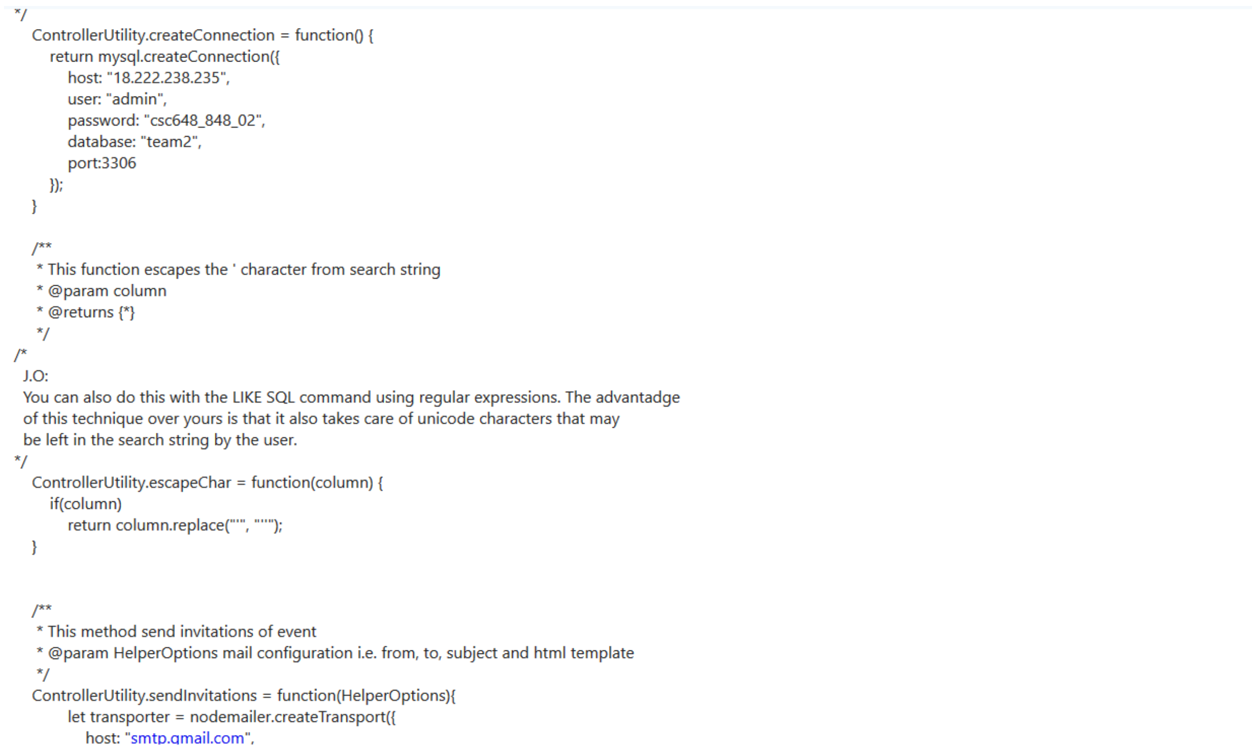
CSC 648-848 Spring 2019 M4 Section 4 Team 2 - Milestone 4 CODE REVIEW

Jose Ortiz-Costa
Tue 5/14/2019 12:45 PM
Tejal Rajendra Ghadge

Hi Tejal,

The following is your code with my review. My review are comments starting with J.O initials. The code that I did not add any review looks good to me. For M4, you can copy and paste this review as part of your external team code review section, You'd still need to provide in M4 code review done internally by the members of your team.

```
const mysql = require('mysql');
const ControllerUtility= {};
const nodemailer = require("nodemailer");

    /**
    This function creates a connection to team2 database with admin user
    */
   /**
  J.O:
  Getting the connection data in this way is not right. You are exposing
all your confidential data, including password in plain text to external
users. You should use environment variables or any other techniques to hide
this confidential data.
See the following  links for more info:
https://stackoverflow.com/questions/22312671/setting-environment-variables-for-node-to-retrieve
https://stackoverflow.com/questions/31494712/establishing-a-connection-to-database-using-environment-variables
*/
   ControllerUtility.createConnection = function() {
       return mysql.createConnection({
           host: "18.222.238.235",
           user: "admin",
           password: "csc648_848_02",
           database: "team2",
           port:3306
       });
   }

   /**
    * This function escapes the ' character from search string
    * @param column
```



```
*/
   ControllerUtility.createConnection = function() {
       return mysql.createConnection({
           host: "18.222.238.235",
           user: "admin",
           password: "csc648_848_02",
           database: "team2",
           port:3306
       });
   }

   /**
    * This function escapes the ' character from search string
    * @param column
```

```
    * @returns {*}
    */
/*
 J.O:
 You can also do this with the LIKE SQL command using regular expressions. The advantadge
 of this technique over yours is that it also takes care of unicode characters that may
 be left in the search string by the user.
*/
   ControllerUtility.escapeChar = function(column) {
       if(column)
           return column.replace("'", "''");
   }


   /**
    * This method send invitations of event
    * @param HelperOptions mail configuration i.e. from, to, subject and html template
    */
   ControllerUtility.sendInvitations = function(HelperOptions){
           let transporter = nodemailer.createTransport({
               host: "smtp.gmail.com",
// J.O: Try to hide this info with environment variables,
// that way if you need to change servers, you don't need to
// modify this code again
user: 'sfevents848@gmail.com',
          pass: 'sfevents@12345'
        }
      });

      transporter.sendMail(HelperOptions, (error, info) => {
        if (error) {
          return console.log(error);
        }
      });

  }

module.exports = ControllerUtility;
```

Events component:

```
import React,{Component} from 'react';
import {Layout,Menu,Icon,Row, Col} from "antd";
import {Link,HashRouter,Route,Switch} from 'react-router-dom';
import axios from 'axios';
import CreateEvent from './CreateEvent'
import MyEvent from './MyEvent'
const {Header,Content,Footer,Sider} = Layout;
const SubMenu = Menu.SubMenu;

/**
 * This class creates new event react component
 */
export default class Event extends Component{
  constructor(props) {
    super(props);
    this.state={
      myEvents : [],
      editEventData: {}
    };
    this.populateEventsForUser();
  }
```

```
/**
 * This function populates events for the user
 */
populateEventsForUser = () =>{
    axios.get('/api/selectEventsByUser').then((result)=>{  //TODO integration with logged in user
        let events =[];
        Object.keys(result.data).forEach(function(key) {
            events.push(result.data[key]);
        });
        this.setState({myEvents: events});
    }).catch((err)=>{
        console.log(err)
    })
};

/**
 * This function allows to update the already created event,
 * The result is updated on component's state.
 * @param e
 */
editEvent = (e) =>{
    let eventID = e.target.closest("div").id;
    if(eventID == "")
        eventID =  e.target.closest("i").id;

    axios.get('/api/fetchEvent?eventID='+eventID).then((result)=>{
        this.setState({editEventData: result.data[0]});
    }).catch((err)=>{
        console.log(err)
    })
};


/**
 * renders Event component
 * @returns {*} component layout
 */
render(){
     return (
          <HashRouter>
```

```
  @returns { } component layout
 */
render(){
  return (
      <HashRouter>
        <Layout style={{minHeight: '100vh'}}>
          <Sider
            style={{zIndex: 10000}}
            breakpoint="lg"
            collapsedWidth="0"
            onBreakpoint={(broken) => {
              console.log(broken);
            }}
            onCollapse={(collapsed, type) => {
              console.log(collapsed, type);
            }}
          >
            <div className="logo">
            </div>
            <Menu onClick={() => {
            }} mode="inline" theme={'dark'} defaultOpenKeys={['sub1']} defaultSelectedKeys={['menu1']}>
              <Menu.Item key="menu1">
                <Link to={'/management/createevent'}>
                  <Icon type="plus"/>
                  <span>New Event</span>
                </Link>
              </Menu.Item>

              <SubMenu key="sub1" title={<span><Icon type="switcher"/><span>My Events</span></span>}>
                {this.state.myEvents.map((myevent, index) =>
                  <Menu.Item key={index} >
                    <Row >
                      <Col sm={20} id={myevent.eid} onClick={this.editEvent}>
                        <Link to={'/management/myevent'}  style={{display : 'block'}}>
                          <span title={myevent.title} style={{color: 'white', 'text-overflow': 'ellipsis', display : 'block', overflow : 'hidden'}} >{myevent.title}</span>
                        </Link>
                      </Col>
                      <Col sm={4}>
                        <Link to={'/management/updateevent'}>
                          <span><Icon id={myevent.eid} type="edit" style={{color: 'white'}} onClick={this.editEvent}/></span>
                        </Link>
                    </Row>
                  </Menu.Item>
                )}
              </SubMenu>


            </Menu>
          </Sider>

          <Layout>
            <Header style={{background: '#fff', padding: 0}}>
            </Header>
            <Content>
              <div>
                <Switch>
                  <Route exact path={'/management/createevent'} component={() => <CreateEvent populateEvents = {this.populateEventsForUser}/>}/>
                  <Route exact path={'/management/updateevent'} component={() => <CreateEvent data={this.state.editEventData}/>}/>
                  <Route exact path={'/management/myevent'} component={() => <MyEvent data={this.state.editEventData} />} />
                </Switch>

              </div>
            </Content>
            <Footer style={{textAlign: 'center'}}>
              @2019 SFEvents.com <a href="/privacy">privacy Policy</a>
            </Footer>
          </Layout>
        </Layout>
      </HashRouter>

    )
}
```

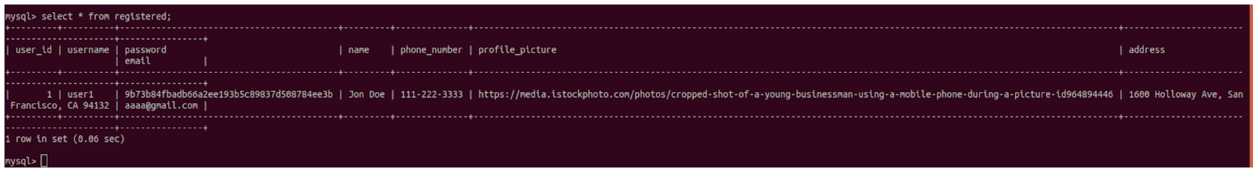
5.Self-check on best practices for security

Major Protected assets

ASSET	VALUE	EXPOSURE
Node,js server	HIGH	High/Server has to be up in order for website to be up and for the application to do what it does.
Database	HIGH	High. Database contains the events that is the main function of the web app. If database is not valued the data would be compromised and would need to be restored
User's personal information(passwords)	HIGH	High. Would affect users privacy and information
Event Images	Low	High. Makes site displeasing and not very visually pleasing.

Threat	Probability	Control	Feasibility
Unauthorized user gains access to other user's data	Low	Authentication with username and password of each user	Basic encryption
Unauthorized User gains access to database	Low	High security in order to protect the server and backup	Secure keys are distibuted only as needed.

PW encryption show below: passwords are encrypted as show below:



```
mysql> select * from registered;
+---------+----------+------------------------------------------+---------+--------------+--------------------------------------------------------------------------------------------------------------+------------------------
---------------------+-----------------+
| user_id | username | password                                 | name    | phone_number | profile_picture                                                                                              | address
                     | email           |
+---------+----------+------------------------------------------+---------+--------------+--------------------------------------------------------------------------------------------------------------+------------------------
---------------------+-----------------+
|       1 | user1    | 9b73b84fbadb66a2ee193b5c89837d508784ee3b | Jon Doe | 111-222-3333 | https://media.istockphoto.com/photos/cropped-shot-of-a-young-businessman-using-a-mobile-phone-during-a-picture-id964894446 | 1600 Holloway Ave, San
 Francisco, CA 94132 | aaaa@gmail.com |
+---------+----------+------------------------------------------+---------+--------------+--------------------------------------------------------------------------------------------------------------+------------------------
---------------------+-----------------+
1 row in set (0.06 sec)

mysql> 
```

6.Self-check: Adherence to Original Non-functional specs:

Security

1. Login and password are required admin and host permissions. DONE
2. Username will be the registered user's email address. DONE

Performance

1. The site should load quickly and instantly across all pages. (DONE)
2. Searches within the site should take no longer than 1 second. (DONE)

Capacity

1. The site shall support future update features. (DONE)

Compatibility

1. This site will be compatible with the latest version of Firefox. (DONE)
2. This site will be compatible with the latest version of Microsoft edge. (DONE)
3. This site will be compatible with the latest version of Chrome Browser. (DONE)

Look and Feel

1. Layout and design of the website shall look easy to use. (DONE)
2. Design of the website will be simple for users of any level to use. (DONE)

Internationalization / Localization Requirements

1. Default language of the website will be English. (DONE)

Data Integrity

1. Database tables shall be backed up. (ON TRACK)
2. Image sizes shall be limited up to 1 megabyte. (ON TRACK)
3. Images shall be uploaded in the correct format(jpeg). (ON TRACK)

Recovery:

1. In a total failure case, the whole site should be put down to revision. (ON TRACK)
2. If broken, the mean time to recovery shall not excess of one day. (ON TRACK)