Bethe OPS Project

Developer Instructions: Emails

May 16, 2019

Table of Contents

1 Overview

This document describes how the email notification system works for the Bethe OPS system, and outlines instructions on how to create and send emails as well as how to modify the templates currently set up for Bethe OPS email notifications.

All email-related functionalities are encapsulated within **./exports/email.js** and **./app.js**. **email.js** contains the functions that create and send the emails. **app.js** calls the functions in **email.js** to send emails at certain endpoints and schedules email reminders.

2 Required Technologies

The required technologies for the email notification system are:

- NodeJS
- Nodemailer
- Node Schedule

3 How It Works

Functions specifically for formulating and sending emails are all contained in **email.js**. These functions are then called in certain routes or scheduled as to send at certain times in **app.js** (refer section 4 for which emails are sent in which routes) where the module, **email.js**, is imported and set to a variable called *email*. Each function requests the recipient's or recipients' net IDs and event information such as the name, date, time, and location; all this information is queried within the route or scheduled job. There are two general categories of email notifications: confirmations and reminders.

3.1 Confirmations

Confirmations are emails that confirm certain actions taken in regards to a user's participation in an event. The appropriate function from *email* is called in the appropriate route where an email needs to be sent out. Confirmations are sent by Bethe OPS for the following cases:

- User signs up for an event
- User is added to an event roster
- User is added to the waitlist of an event
- User is removed from an event roster
- User is removed from the waitlist of an event

For each confirmation email function, the general implementation is:

1. Create a transporter object using the default SMTP transport with Nodemailer.

```
const transporter = nodemailer.createTransport({
    service: 'Gmail',
    auth: {
        user: 'betheops@gmail.com',
        pass: 'Cornell2019'
    },
    tls: {
        rejectUnauthorized: false
    }
});
```

2. Set up the recipient's email address using the net ID passed to the function.
3. Set up the HTML string that will be the email content body using the event information passed to the function.

```
/**
 * Sends an email confirming a user's sign-up for an event given the user's net ID, [netID],
 * event name, [eventName], event date, [eventDate], event time, [eventStartTime] and
 * [eventEndTime], and event location, [eventLocation].
 * Requires:
 *  - [netID] is a valid String representing a valid Cornell net ID.
 *  - [eventName] is a valid String represending the name of the event signed up for.
 *  - [eventDate] is a valid String representing the date of the event signed up for.
 *  - [eventStartTime] is a valid String represending the start time of the event signed up for.
 *  - [eventEndTime] is a valid String representing the end time of the event signed up for.
 *  - [eventLocation] is a valid String representing the location of the event signed up for.
 */
function sendSignUp(netID, eventName, eventDate, eventStartTime, eventEndTime, eventLocation) {
    var recipient = netID + '@cornell.edu';

    // create HTML for email
    var html =
        '<p>Hello,<br /><br />This is a confirmation that you have signed up for the Bethe House event, ' +
        eventName +
        ', happening on ' +
        eventDate +
        ' at ' +
        eventStartTime +
        ' - ' +
        eventEndTime +
        ', ' +
        eventLocation +
        '.<br /><br />--<br />Thanks,<br />Bethe House Office Staff<br />607-255-7210<br />bethehouseoffice@gmail.com</p>';
```

4. Create a mail options object with the following email fields and set their values:
    a. from
        i. The value of "from" is set to 'Bethe OPS <betheops@gmail.com>', which is the email address used by the Bethe OPS system to send emails.
    b. to
    c. subject
    d. html

```
const mailOptions = {
    from: 'Bethe OPS <betheops@gmail.com>',
    to: recipient,
    subject: 'Sign-up confirmation - ' + eventName,
    html: html
};
```

5. Call sendMail on the transporter object to send the email to the recipient with the mail options object. Close the mail options object.

```
transporter.sendMail(mailOptions, function(err, info) {
    if (err) {
        console.log('Sending to ' + recipient + ' failed: ' + err);
    } else {
        console.log('Sent to ' + recipient);
    }

    mailOptions.transport.close();
});
```

3.2 Reminders

Reminders are emails that remind a user of an event (s)he signed up for that is occurring in the near future. Email reminders are scheduled as jobs in **app.js** using Node Schedule, which is imported and set to a variable called *schedule*. Reminders are sent by Bethe OPS at the following times:

- One day before the event happens
- Two hours before the event happens

To set up these reminders, they need to be scheduled as jobs first. One-day reminders are scheduled to send at 9AM the day before the events if there are any events occurring the next day.

```
// one day reminder
schedule.scheduleJob({ hour: 9, minute: 0 }, function() {
```

As for two-hour reminders, the system checks every fifteen minutes for events occurring in two hours and sends reminders for any events found.

```
// two hour reminder
schedule.scheduleJob('*/15 * * * *', function() {
```

The general implementation of the functions for email reminders is similar to that of the functions for email confirmations. However, for reminders, there may be multiple recipients. In addition, the email content is dependent on the size of the event roster and the size of the event waitlist; the email content depends on which of the three conditions are true for the event:

- Event roster is not full
- Event roster is full with no one on the event waitlist
- Event roster is full with people on the waitlist

Otherwise, for each reminder type, the general implementation is:

1. Create a transporter object using the default SMTP transport with Nodemailer (same as step 1 of section 3.1).
2. Set up the recipients' email addresses using the net IDs passed to the function.

```
function twoHourReminder(events) {
    for (var e in events) {
        // get eventName, eventDate, eventStartTime, eventEndTime, eventLocation, eventLeaderNetID
        let eventName = e;
        let eventDate = events[e].date;
        let eventStartTime = events[e].startTime;
        let eventEndTime = events[e].endTime;
        let eventLocation = events[e].location;
        let eventLeaderNetID = events[e].eventLeader.netID;

        // get list of attendees
        let attendees = events[e].attendees;
        let waitlist = events[e].waitlist;
        let recipients = [];
        attendees.forEach(function(p) {
            if (p.netID !== eventLeaderNetID) {
                let r = p.netID + '@cornell.edu';
                recipients.push(r);
            }
        });
        if (waitlist.length > 0) {
            waitlist.forEach(function(p) {
                let r = p.netID + '@cornell.edu';
                recipients.push(r);
            });
        }
```

3. Set up the HTML string that will be the email content body using the event information passed to the function and according to one of the three conditions aforementioned.

```
// send email reminder
var twoReminderHtml =
    '<p>Hello,<br /><br />This a friendly reminder that you signed up for the Bethe House event, ' +
    eventName +
    ', happening today, ' +
    eventDate +
    ' at ' +
    eventStartTime +
    ' - ' +
    eventEndTime +
    ', ' +
    eventLocation +
    '. <br /><br />';
// if event isn't full
if (attendees.length < events[e].maxCapacity) {
    twoReminderHtml += 'If you have a friend who would like to join there is currently at least one open space on the sign-up sheet. We would be happy to h
}
// if event is full with no one on the waitlist
if (attendees.length == events[e].maxCapacity && waitlist.length == 0) {
    twoReminderHtml += 'This event is currently full. If you can no longer attend please remove your name from the event roster online.<br /><br />';
}
// if event is full with people on the waitlist
if (attendees.length == events[e].maxCapacity && waitlist.length > 0) {
    twoReminderHtml += 'This event is currently full with a waitlist. We have included folks on the waitlist on this email.<br /><br />';
    twoReminderHtml += 'If you can no longer attend please remove your name from the event roster online. If you are on the waitlist your name is included
    // get waitlist for event
    // for each person on waitlist
    waitlist.forEach(function(p) {
        twoReminderHtml += p.first_name + ' ' + p.last_name;
        twoReminderHtml += '<br />';
    });
}
twoReminderHtml += '--<br />Thanks,<br />Bethe House Office Staff<br />607-255-7210<br />bethehouseoffice@gmail.com</p>';
```

4. Create a mail options object with the following email fields and set their values:
    a. from
        i. The value of "from" is set to 'Bethe OPS <betheops@gmail.com>', which is the email address used by the Bethe OPS system to send emails.
    b. to
    c. cc
        i. The value of "cc" is set to the event leader's email address.
    d. subject
    e. html

```
// create mail options
const twoReminderMailOptions = {
    from: 'Bethe OPS <betheops@gmail.com>',
    to: recipients.join(', '),
    cc: eventLeaderNetID + '@cornell.edu',
    subject: 'Event reminder two hours - ' + eventName,
    html: twoReminderHtml
}
```

5. Call sendMail on the transporter object to send the email to the recipient with the mail options object. Close the mail options object (same as step 5 of section 3.1).

4 Routes to Send Confirmation Emails

Confirmation emails are sent in routes in **app.js**. Outlined below are the routes in which each confirmation email function is or should be called.

Route(s)	Functions
/students/signup /eventleader/signup	sendSignUp( )
/eventleader/addattendee /admin/addattendee	sendAddAttendee( )
/students/removeattendee /eventleader/removeattendee /admin/removeattendee	sendRemoveAttendee( ) sendWaitlistAttendee( )

5 How to Modify Email Templates

To modify the current email templates, simply go into the function in **email.js** corresponding to the type of email you want to modify and change the value of the variable whose value is the HTML used for the email content.

```
// create HTML for email
var html =
    '<p>Hello,<br /><br />This is a confirmation that you have signed up for the Bethe House event, ' +
    eventName +
    ', happening on ' +
    eventDate +
    ' at ' +
    eventStartTime +
    ' - ' +
    eventEndTime +
    ', ' +
    eventLocation +
    '.<br /><br />--<br />Thanks,<br />Bethe House Office Staff<br />607-255-7210<br />bethehouseoffice@gmail.com</p>';
```

6 Bethe OPS Email Credentials

As seen in section 3, Bethe OPS has its own email it uses to send out emails to its users. The account information for the email used is as follows:

Username: betheops@gmail.com
Password: Cornell2019

Currently, the password of the email is hardcoded in and should be tokenized or hidden for security.