CSConnects:

Connecting Students Across Computer Science

By: Shaina, Leman, Matthew, Maisa

Hunter College - CS499 - Major Capstone Fall 2020 - Final Demos

Product Definition

In order to connect the Computer Science community on our current **virtual** campus, CSConnects will solve students’ problem of feeling **disconnected** from our peers, **networking opportunities** and club **events** and programming by giving them easy access to our clubs **calendar** as well as **notifications** when there are events that will interest them.

Product Definition (cont.)

We know our product will work when:

- It displays a full **calendar** of events for all CS **clubs**
- Allows for users to filter their event **preferences** and **notifies** them about upcoming events
- Display helpful **resources** for students such as tips to get more **involved** on campus, links to group chats, links to internship opportunities, etc.

DEMO

Contributions

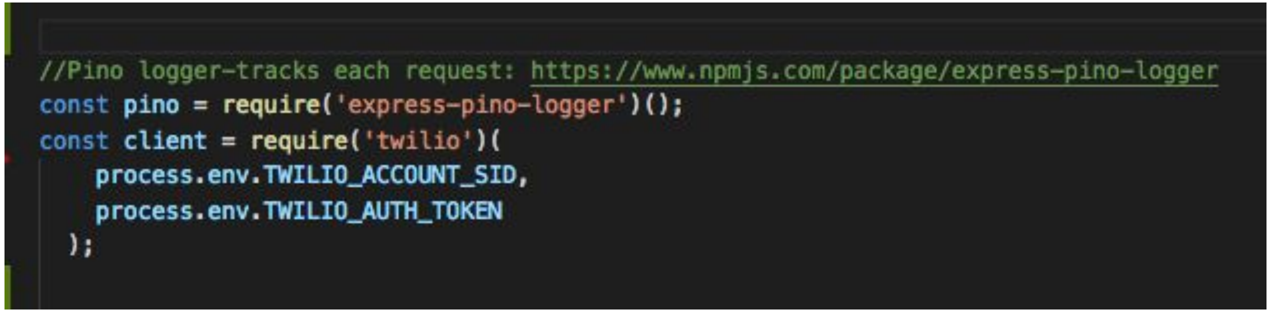
Shaina: - Added Twilio API feature and logic to send users SMS messages - Cron scheduler - UX of calendar page and accounts page - Delete event from calendar functionality - User signing up for event notifications functionality	Matt: - Created login, signup functionality. - Added login sessions and user roles. - Added calendar admin panel - Contributed to MySQL database design - Setup Heroku Mysql database - Setup API for filter feature of calendar - Setup API for accounts page
Leman: - Researched calendar APIs - Google Calendar, React Scheduler, FullCalendar - Set up events page, adding the calendar - Contributed to database design and backend APIs - Deployed project on Heroku - Setup filter feature for frontened	Maisa: - Created mockup for the project and identified reusable components - Created pages and components using ReactJS and SASS - Worked on responsive web design - Added react router for seamless user experience

Tools used

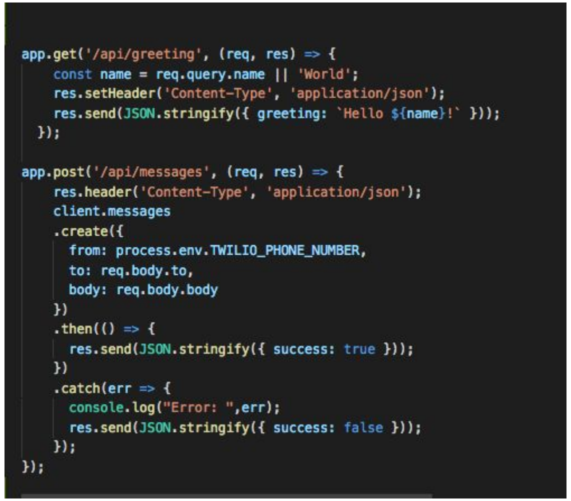
- MySQL
- Twilio
- React frontend
- Full Calendar
- Moment (for date and time)
- Cron (for scheduling our text messages)
- Axios (for API calls)
- Express backend
- Formspree API

Remaining Goals for CSConnects

Technical Presentation #1: Twilio

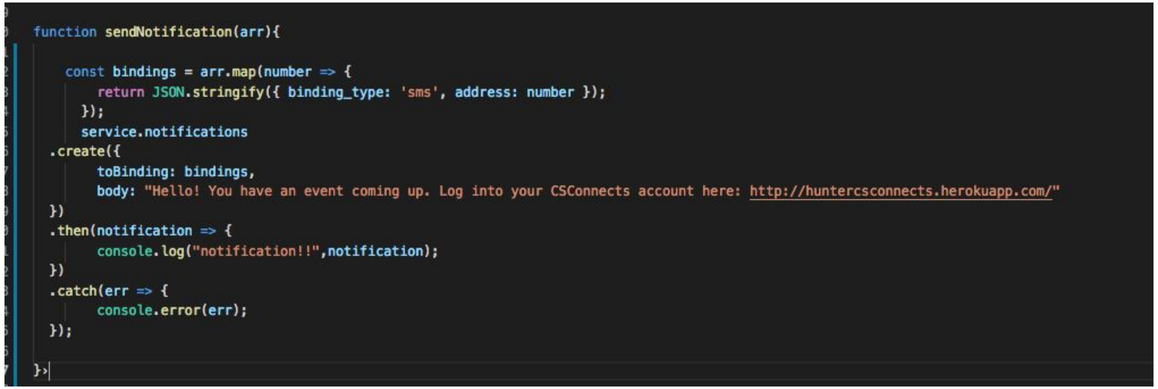


```
//Pino logger-tracks each request: https://www.npmjs.com/package/express-pino-logger
const pino = require('express-pino-logger')();
const client = require('twilio')(
    process.env.TWILIO_ACCOUNT_SID,
    process.env.TWILIO_AUTH_TOKEN
  );
```



```
app.get('/api/greeting', (req, res) => {
    const name = req.query.name || 'World';
    res.setHeader('Content-Type', 'application/json');
    res.send(JSON.stringify({ greeting: `Hello ${name}!` }));
  });

app.post('/api/messages', (req, res) => {
    res.header('Content-Type', 'application/json');
    client.messages
    .create({
      from: process.env.TWILIO_PHONE_NUMBER,
      to: req.body.to,
      body: req.body.body
    })
    .then(() => {
      res.send(JSON.stringify({ success: true }));
    })
    .catch(err => {
      console.log("Error: ",err);
      res.send(JSON.stringify({ success: false }));
    });
});
```

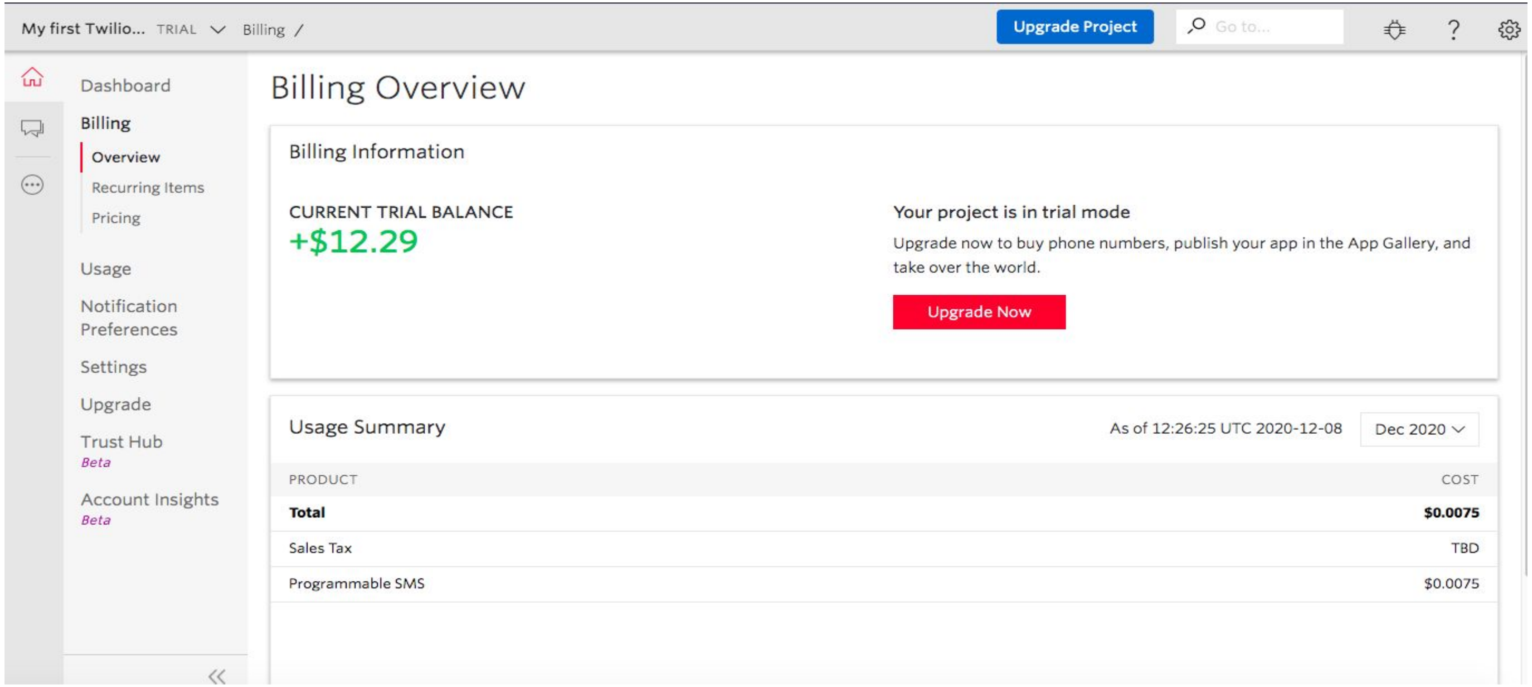


```
function sendNotification(arr){

    const bindings = arr.map(number => {
        return JSON.stringify({ binding_type: 'sms', address: number });
      });
      service.notifications
  .create({
        toBinding: bindings,
        body: "Hello! You have an event coming up. Log into your CSConnects account here: http://huntercsconnects.herokuapp.com/"
  })
  .then(notification => {
        console.log("notification!!",notification);
  })
  .catch(err => {
        console.error(err);
  });

}
```

Twilio cont.


My first Twilio... TRIAL / Billing /

Upgrade Project

Go to...

Dashboard
Billing
Overview
Recurring Items
Pricing
Usage
Notification Preferences
Settings
Upgrade
Trust Hub *Beta*
Account Insights *Beta*

Billing Overview

Billing Information

CURRENT TRIAL BALANCE
+$12.29

Your project is in trial mode

Upgrade now to buy phone numbers, publish your app in the App Gallery, and take over the world.

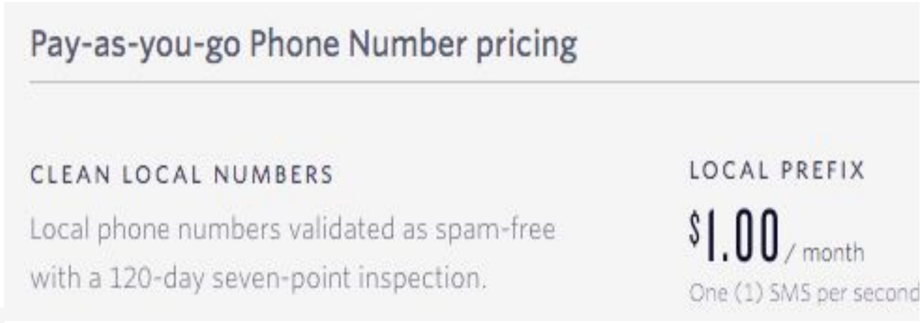
Upgrade Now

Usage Summary

As of 12:26:25 UTC 2020-12-08 Dec 2020

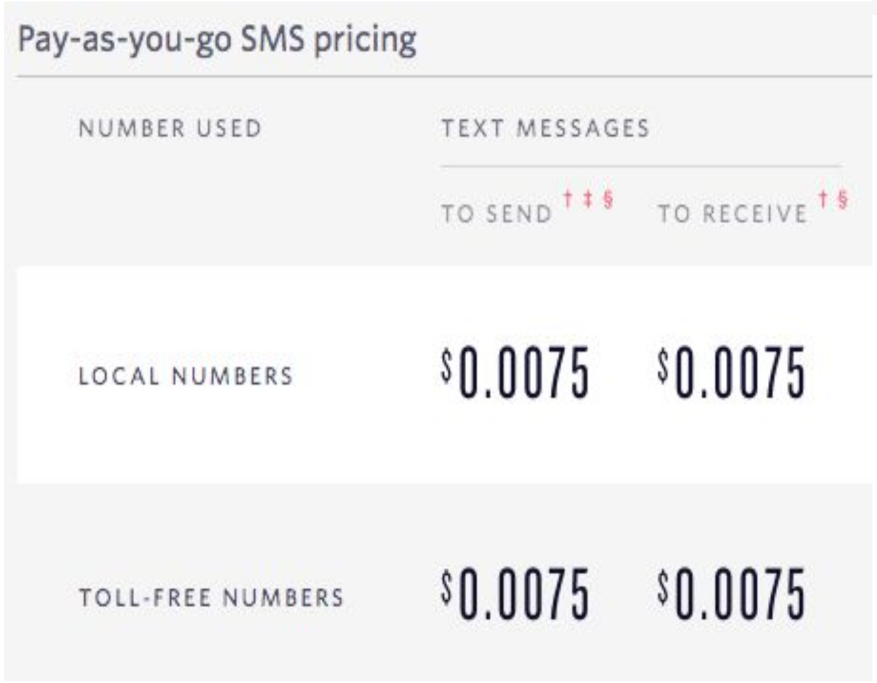
PRODUCT	COST
Total	**$0.0075**
Sales Tax	TBD
Programmable SMS	$0.0075

Twilio cont.



Pay-as-you-go Phone Number pricing

CLEAN LOCAL NUMBERS	LOCAL PREFIX
Local phone numbers validated as spam-free with a 120-day seven-point inspection.	$1.00 / month One (1) SMS per second



Pay-as-you-go SMS pricing

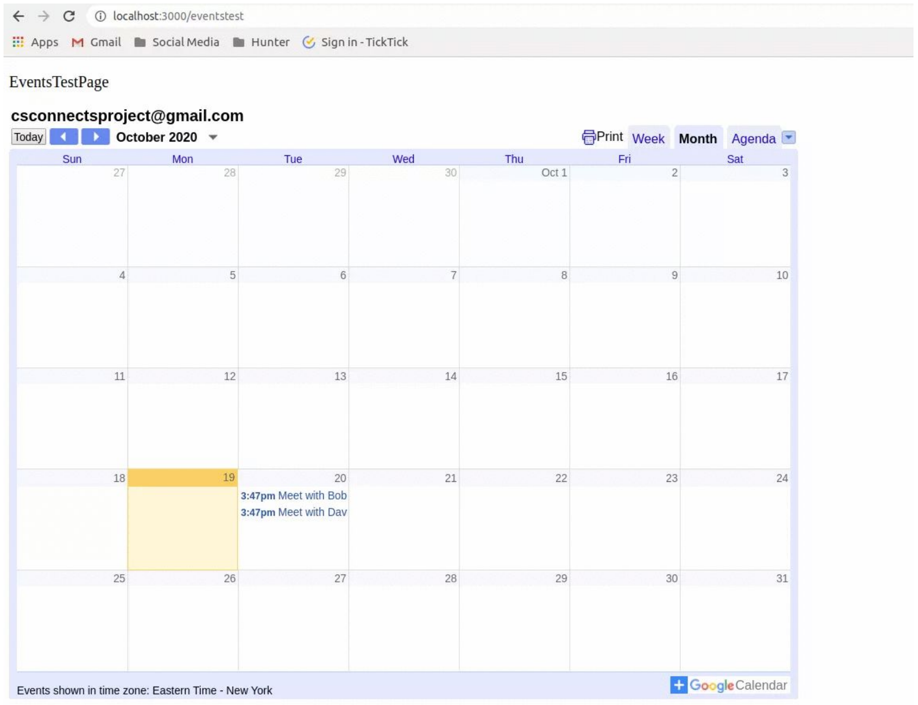
NUMBER USED	TEXT MESSAGES	
	TO SEND † ‡ §	TO RECEIVE † §
LOCAL NUMBERS	$0.0075	$0.0075
TOLL-FREE NUMBERS	$0.0075	$0.0075

Technical Presentation #2: Calendar APIs

Options:

1. Google Calendar API
2. DevExtreme REACTIVE React Scheduler
3. FullCalendar

Google Calendar API



localhost:3000/eventstest

Apps Gmail Social Media Hunter Sign in - TickTick

EventsTestPage

csconnectsproject@gmail.com

Today October 2020 Print Week Month Agenda

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	Oct 1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20 **3:47pm** Meet with Bob **3:47pm** Meet with Dav	21	22	23	24
25	26	27	28	29	30	31

Events shown in time zone: Eastern Time - New York

Google Calendar

Pros:

- Many people use it and has good documentation

Cons:

- Unable to filter the calendar w/o filtering another non-user's calendar.
- No dedicated React component

DevExtreme REACTIVE React Scheduler

Pros:

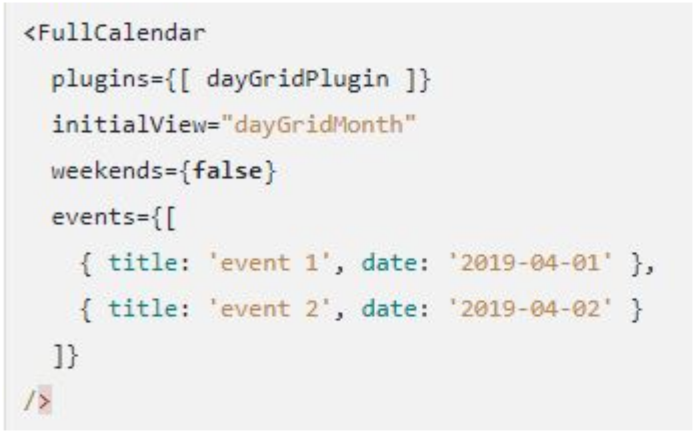
- Dedicated React calendar component

Cons:

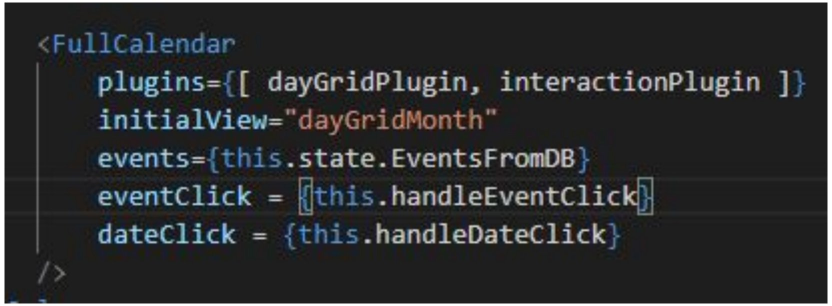
- Not many people use this so finding answers would be hard
- Documentation is also lackluster
- We also were unable to get it to work

FullCalendar

https://fullcalendar.io/docs/react

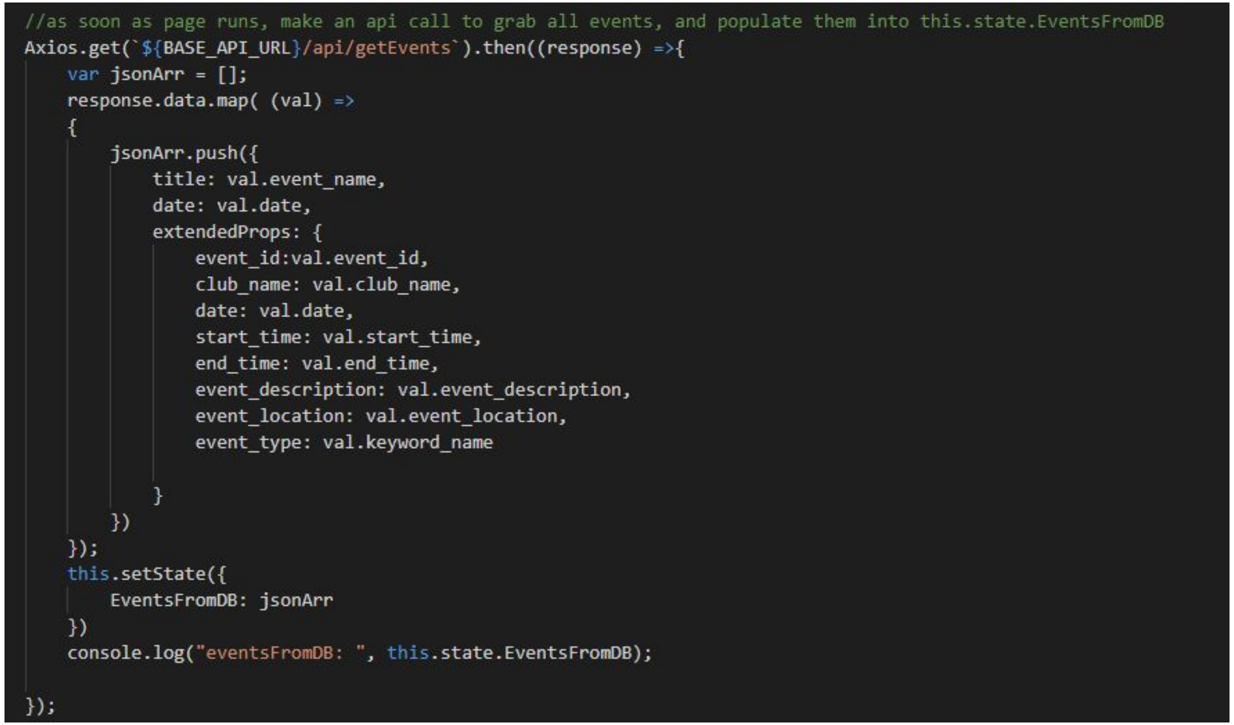


```
<FullCalendar
  plugins={[ dayGridPlugin ]}
  initialView="dayGridMonth"
  weekends={false}
  events={[
    { title: 'event 1', date: '2019-04-01' },
    { title: 'event 2', date: '2019-04-02' }
  ]}
/>
```



```
<FullCalendar
    plugins={[ dayGridPlugin, interactionPlugin ]}
    initialView="dayGridMonth"
    events={this.state.EventsFromDB}
    eventClick = {this.handleEventClick}
    dateClick = {this.handleDateClick}
/>
```

FullCalendar



```
//as soon as page runs, make an api call to grab all events, and populate them into this.state.EventsFromDB
Axios.get(`${BASE_API_URL}/api/getEvents`).then((response) =>{
    var jsonArr = [];
    response.data.map( (val) =>
    {
        jsonArr.push({
            title: val.event_name,
            date: val.date,
            extendedProps: {
                event_id:val.event_id,
                club_name: val.club_name,
                date: val.date,
                start_time: val.start_time,
                end_time: val.end_time,
                event_description: val.event_description,
                event_location: val.event_location,
                event_type: val.keyword_name

            }
        })
    });
    this.setState({
        EventsFromDB: jsonArr
    })
    console.log("eventsFromDB: ", this.state.EventsFromDB);

});
```

Thank you!

Any questions?