A Scalable Web Application Framework for Monitoring Energy Usage on Campus

Group 57: Daniel Schroeder, Aubrey Thenell, Parker Bruni

Parker

- Overview of progress report
- Completion since last progress report
- Progress on UI
- Cool CSS/Bootstrap tricks we've implemented
- OSU marketing team collaboration (fonts, colors, logo)
- Google Maps API

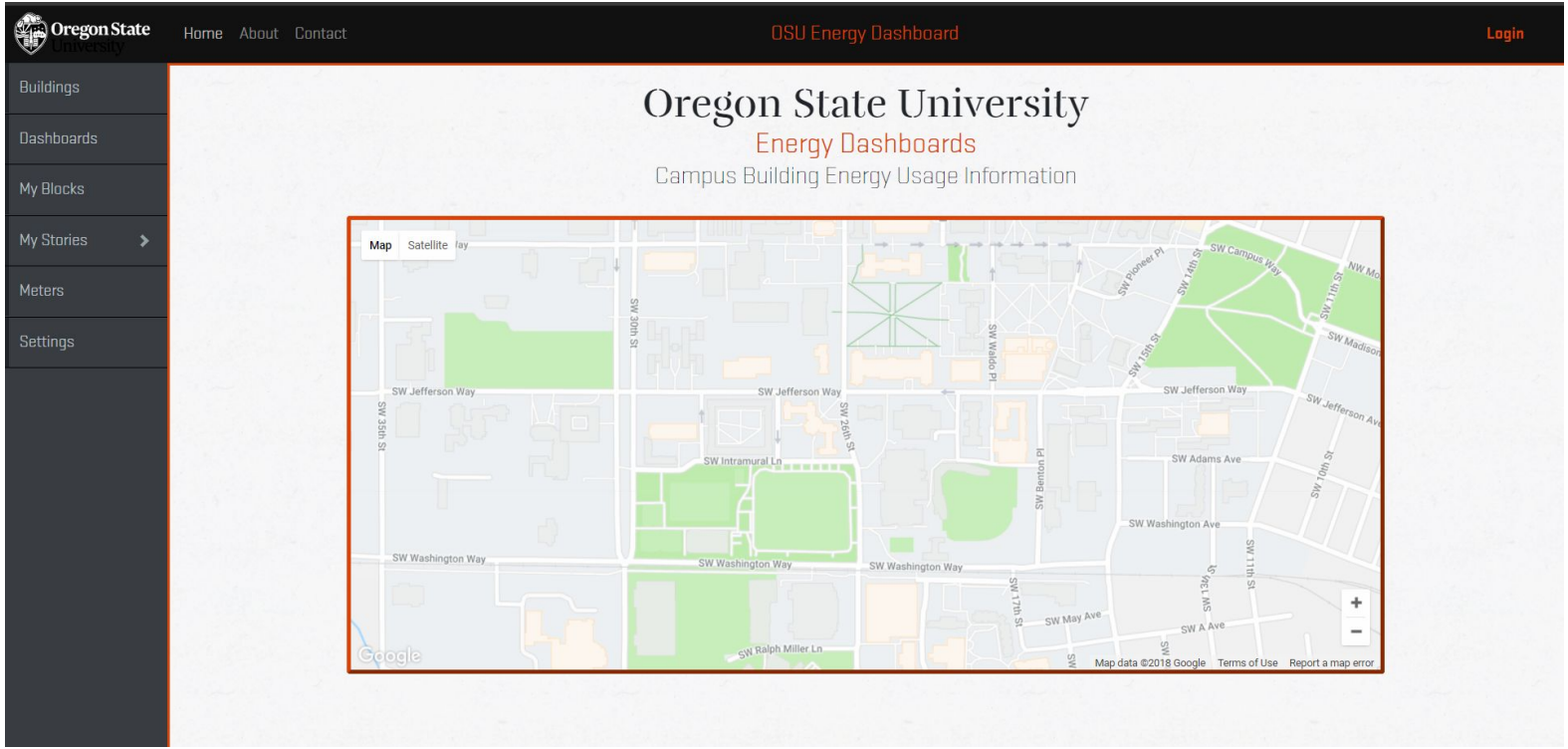
Overview of Progress Report

- UI/UX

- Core Application Functionality

- Backend Framework

Completion Since Last Progress Report



Progress on UI

- Navigation Bars

- Color Scheme

- Styling and Design

Progress on UI: Navigation Bars

- Top nav nearly finished, need to adjust our center title and get new logo

- Side nav UI is complete until further functionality is added

- May add features once bulk of functionality is added to nav bars

Progress on UI: Color Scheme

Progress on UI: Styling

- Clean, Modern

- No drastic colors or flashy animations

- Focus on utility over an aesthetic impression

Bootstrap/CSS Tricks

- Using Classes as “arguments” to modify element CSS

- Animations to give a more modern feel to our application

Bootstrap/CSS Tricks: Class as Arguments

We use use Bootstrap and custom CSS classes as “arguments” to modify our HTML elements

```
.rufina-stencil-regular {
    font-family: 'rufina-stencil-regular';
}
.Stratum2-Light {
    font-family: 'Stratum2-Light';
}
.Stratum2-Thin {
    font-family: 'Stratum2-Thin';
}
.Stratum2-Bold {
    font-family: 'Stratum2-Bold';
}
.text-orange {
    color: #DC4405;
}
```

```
<div class="container-fluid p-0">
<div class="row w-100 m-0" >
    <div class="col-sm-6 mx-auto text-center">
    <h1 class="rufina-stencil-regular m-0">Oregon State University</h1>
    <h3 class="Stratum2-Light text-orange m-0" >Energy Dashboards</h3>
    <h4 class="Stratum2-Thin" >Campus Building Energy Usage Information</h4>
    </div>
</div>
```

Bootstrap/CSS Tricks: Animations

Animations are utilized in the side navigation bar


```
.custom-nav-item div{
    border-left: 4px solid;

}
.active-nav-item div{
    border-left: 4px solid #DC4405;
    -webkit-transition: all 0.3s ease-in-out 0s;
    transition: all 0.3s ease-in-out 0s;
}
.active-nav-item .custom-nav-link{
    padding-left: 6px;
    -webkit-transition: all 0.2s ease-in-out 0s;
    transition: all 0.2s ease-in-out 0s;
}
```

```
.hideDash{
    -ms-transform: translate(0px, 50px); /* IE 9 */
    -webkit-transform: translate(0px, 50px); /* Safari */
    transform: translate(0px, 50px);
    transition-duration: 0.5s;
}
.showDash{
    -ms-transform: translate(80px, 50px); /* IE 9 */
    -webkit-transform: translate(80px, 50px); /* Safari */
    transform: translate(80px, 50px);
    transition-duration: 0.5s;
}
#dashboardSelector{
    width: 80px;
    z-index: 10;
}
.dash-item{
    border-left: 1px solid black !important;
    background-color: #1c1c1c !important;
}
```

Collaboration with OSU Marketing

- Fonts used as designated by OSU marketing

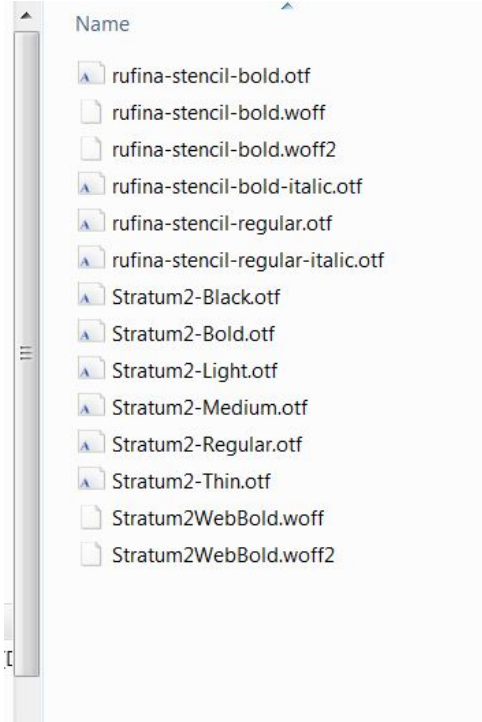
- Official OSU logo to be used on application, with potential for custom logo

- Required colors and page styling

Collaboration with OSU Marketing: Fonts

OSU marketing has has provided us with various font files

These files allow our application to fit the OSU marketing guidelines




```
229 @font-face {
230   font-family: 'rufina-stencil-regular';
231   src: url('assets/fonts/rufina-stencil-regular.otf') format('opentype'),
232        url('assets/fonts/rufina-stencil-regular.woff') format('opentype'),
233        url('assets/fonts/rufina-stencil-regular.woff2') format('opentype');
234 }
235 @font-face {
236   font-family: 'Stratum2-Thin';
237   src: url('assets/fonts/Stratum2-Thin.otf') format('opentype');
238 }
239 @font-face {
240   font-family: 'Stratum2-Light';
241   src: url('assets/fonts/Stratum2-Light.otf') format('opentype');
242 }
243 @font-face {
244   font-family: 'Stratum2-Bold';
245   src: url('assets/fonts/Stratum2-Bold.otf') format('opentype');
246 }
```

```
194 .rufina-stencil-regular {
195     font-family: 'rufina-stencil-regular';
196 }
197 .Stratum2-Light {
198     font-family: 'Stratum2-Light';
199 }
200 .Stratum2-Thin {
201     font-family: 'Stratum2-Thin';
202 }
203 .Stratum2-Bold {
204     font-family: 'Stratum2-Bold';
205 }
206 .text-orange {
207     color: #DC4405;
208 }
```

Collaboration with OSU Marketing: Logo

Current implementation of logo

Color scheme not ideal for our color scheme

Potential for OSU marketing to create custom logo for our application

Google Maps API

Trouble with creating map using basic method.

Needed to create controller for the map.

```
angular.module('mapController', [])
    .controller('mapController', function($scope, $timeout){

        $timeout(function(){
            $scope.map;
            $scope.markers = [];
            $scope.markerId = 1;
            //Map initialization
            var latlng = new google.maps.LatLng(44.563780557193354, -123.27947616577148);
            var myStyles = [{featureType: "poi",elementType: "labels", stylers: [{ visibility: "off" }]}];
            var myOptions = {
                zoom: 16,
                center: latlng,
                mapTypeId: google.maps.MapTypeId.ROADMAP,
                streetViewControl: false,
                fullscreenControl: false,
                Icons: false,
                mapTypeControl: true,
                mapTypeControlOptions: {mapTypeIds: [google.maps.MapTypeId.ROADMAP,google.maps.MapTypeId.SATELLITE]},
                styles: myStyles
            };
            $scope.map = new google.maps.Map(document.getElementById("map"), myOptions);
            $scope.overlay = new google.maps.OverlayView();
            $scope.overlay.draw = function() {}; // empty function required
            $scope.overlay.setMap($scope.map);
            $scope.element = document.getElementById('map');

        },100);

    });
```

Presenter Transition

Daniel Schroeder

Core Functionality

Daniel

- Block retrieval and display
- Dashboard retrieval and display
- Building retrieval and display
- Building images and file name regEx
- Stories
- Future plans of work (D3.js, view/edit/delete better, user sessions, limited access to content, public vs. private)

Component: Building

Bloss Hall

Type: Residence Hall/Dormitory

Meter ID: 25709

View

Cauthorn Hall

Type: Residence Hall/Dormitory

Meter ID: 21777

View

Finley Hall

Type: Residence Hall/Dormitory

Meter ID: 17097

View

Callahan Hall

Type: Residence Hall/Dormitory

Meter ID: 22763

View

Component: Building

Cauthorn Hall

Type:Residence Hall/Dormitory

Meter ID:21777

Entry	Date	KW/hr Reading
1	2018-01-24T00:05:05.134Z	4312
2	2018-01-24T00:05:05.134Z	4495
3	2018-01-24T00:05:05.134Z	4256
4	2018-01-24T00:05:05.134Z	1013

Component: Block

User Blocks

Create Block

Test Block 1

Edit Delete

Buildings:

- Cauthorn Hall
- Bloss Hall
- Sackett Hall

High

XXX,XXXmW/h

Median

XXX,XXXmW/h

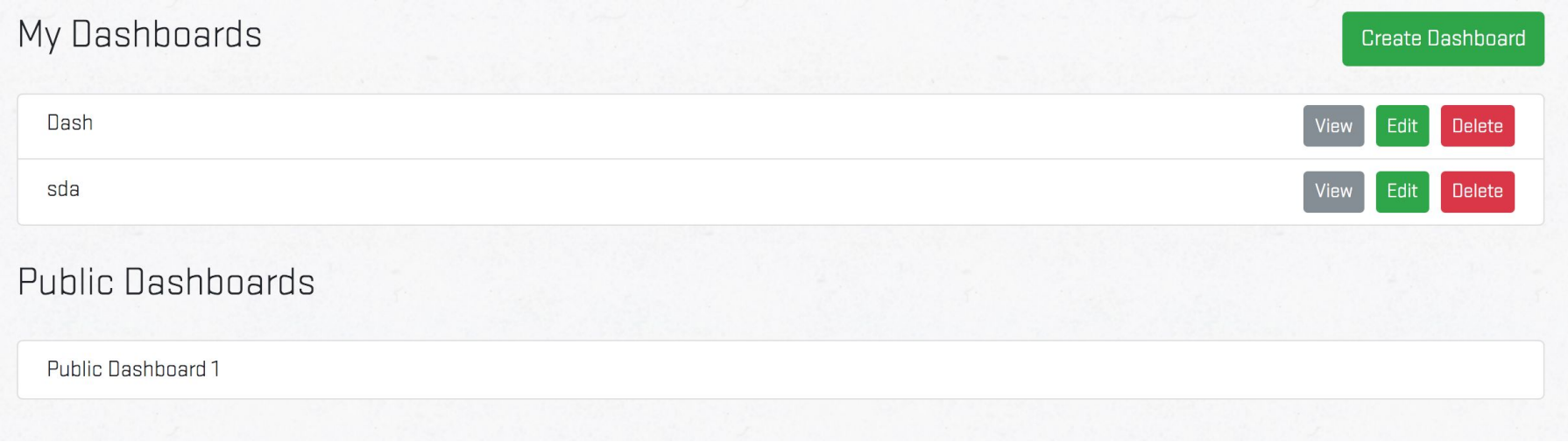
Low

XXX,XXXmW/h

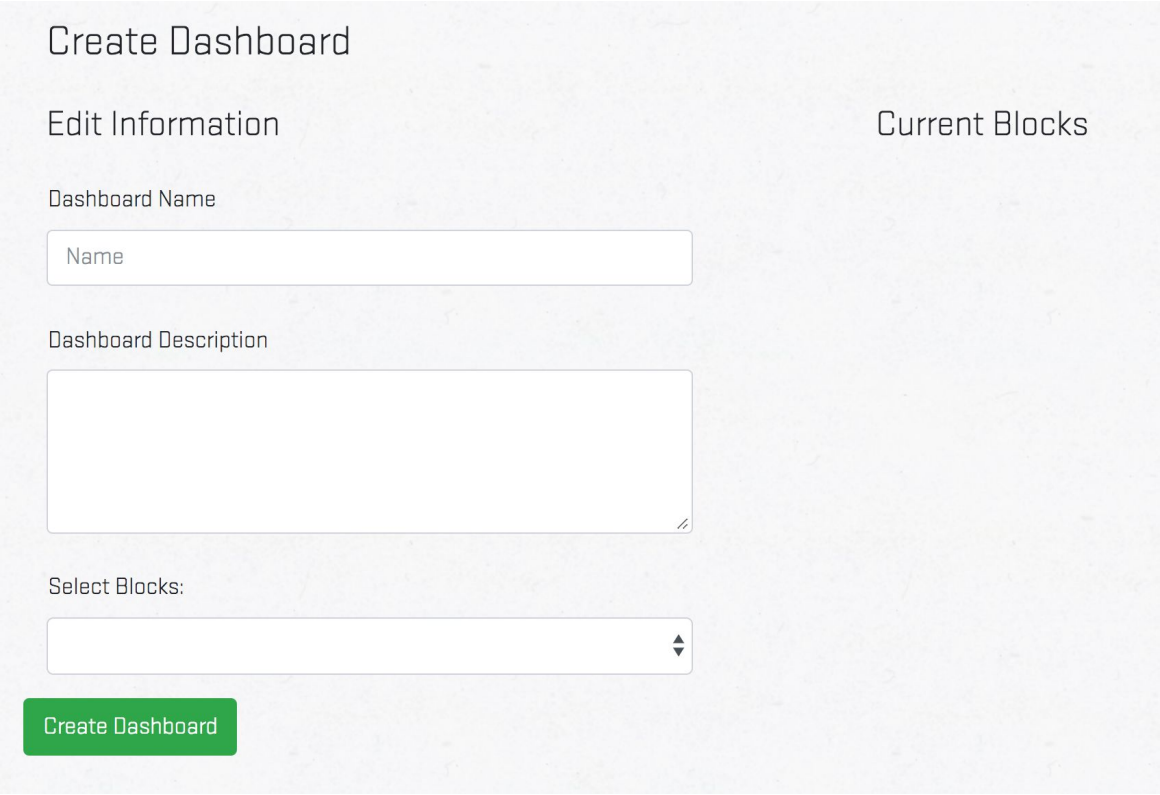
Insert Chart Here

Filter 1 Filter 2

Component: Dashboard



Component: Dashboard



Feature: Authentication

What is left to do?

Implement story functionality.

Create ``edit'' pages for block, dashboard, and story components.

Filter settings on buildings page.

Limit viewable content based on user roles and authentication.

Make public vs. private flag on user items.

Create public facing application distinction.

Generate D3.js templates for different graph types.

Aubrey

- Acquisuite data retrieval / XML Parsing
- Schemas
- Buildings Component
- Other Components
- Database Management
- Mongoose bugs

AcquiSuite Data Retrieval / XML Parsing

```
app.post('/receiveXML', xmlparser({ trim: false, explicitArray: false }), function (req, res) {
```

Schemas

Building:

```
var mongoose = require('mongoose');
var buildingSchema = mongoose.Schema({
    name: String,
    building_type: String,
    serial: String,
    data_entry: [{
        entry:      { type: mongoose.Schema.ObjectId, ref: 'DataEntry' },
        timestamp: { type: Date, required: true }
    }]
});

// create the model for users and expose it to our app
module.exports = mongoose.model('Building', buildingSchema);
```

DataEntry:

```
var mongoose = require('mongoose');
var dataEntrySchema = mongoose.Schema({
    meter_serial: String,

    timestamp: { type: Date, required: true },
    point: [{
        number: Number,
        name: String,
        units: String,
        value: Number
    }]
});

// create the model for users and expose it to our app
module.exports = mongoose.model('DataEntry', dataEntrySchema);
```

Buildings

```
Building.findOne({serial: req.body.das.serial}, function (err, doc) {
    if(doc === null){
        var entry = {
            name: req.body.das.devices.device.name,
            building_type: 'Academic',
            serial: req.body.das.serial
        }
        addBuildingToDatabase(entry);
    }
```

Buildings Cont.

```
function addBuildingToDatabase(entry) {
    Building.findOne({name: entry.name}, function (err, docs) {
          if(docs === null){ // ensure building doesn't exist
              var build = new Building();
               // set all of the relevant information
              build.name = entry.name
              build.building_type = entry.building_type;
              // serial can be used as identifier when adding data (data has s
              build.serial = entry.serial;
              // save the building
              build.save()
                   .catch( err => {res.status(400)
                   .send("unable to save to database");})
              console.log("The building '" + entry.name + "' has been added.")
          }
          else
              console.log('Nothing was added');
        });
  };
```

Buildings Cont.

```
    data = new DataEntry();
    data.meter_serial = req.body.das.serial;
    data.timestamp = new Date(pathShortener.time._);
    pathShortener.point.forEach((e,i) => {data.point[i] = e.$;});
    data.save(function(err, savedBlock) {
        if (err)
            throw err;
        else {
            Building.findOneAndUpdate({serial: req.body.das.serial},
            {$push:{data_entry: data, timestamp: data.timestamp}},
            {safe: true, upsert: true, new: true},
            (err) =>{if (err) throw(err);})
        }
    });
    res.send(req.body);
);
```

Other

```
else
    User.findByIdAndUpdate(
        { _id: user._id},
        { $push:{blocks: savedBlock}},
        {safe: true, upsert: true, new: true},
        (err) =>{if (err) throw(err);});
```

What’s Left?

- Backend
- Frontend
- Building Entries
- Building sort
- Data sanitation