Scalable Web Application Framework for Monitoring Energy Usage on Campus

Group 57

Daniel Schroeder

Abstract

Data visualization is an important technique for analyzing data trends and identifying problems that may not have been apparent. This document outlines our plan to create a web application which will gather energy data from data aquisition servers and create information management dashboards. Our application will allow users to generate charts and graphs to see energy use over time, compare energy usage across multiple buildings, and create collections of different visualizations.

Contents

1 Introduction

2 Visualization Frameworks

Our web application will provide near-real time data visualizations for energy consumption on campus buildings. This application will need to dynamically create charts and graphs based on energy data from the database. A key to choosing a visualization library will be to find one that can be dynmaically created and changed as new data is received from the data acquisition servers, and the ability to create chart templates that can be reused on multiple pages with different input parameters.

2.1 D3.js

Repository Commits: 4,104

Contributors: 120

D3.js is a clunky, massive library that makes use of SVG elements in webpages to append charts and graphics to the DOM. It is extremely well documented and widely used. D3 has a high learning curve for beginners to take head-on, but has a wide array of different visualizations and customization.

Pros

- A lightwight, versatile javascript library that creates SVG elements within web pages and appends them to DOM elements.
- Makes use of javascript functions and DOM controlling functionality to dynamically change the content of the page.
- Provides a lot of variety and ability to customize graphics.
- Widely used and there is a lot of documentation and resources available to assist the learning and development processes.

Cons

- D3 is essentially an API to to manipulate SVG, it is not a charting library in of itself.
- You cannot easily pass a dataset into a specified chart type like other libraries.
- Considered to be “code-heavy” and difficult to jump right into as a novice user.
- Angular and D3 both attempt to control the DOM and so you have to find a way to make the two work together which is counterintuitive to both framework’s APIs.

2.2 Vis.js

Repository Commits: 3,165

Contributors: 137

Vis.js is a lightweight charting library that allows users to create clean charts from dynamic datasets. It is responsive and allows for interaction with the data on the page. Vis.js is praised for it’s network chart capabilities but is limited in the number of different modules you can create.

Pros

- Easy to use and less of a learning curve than D3.
- Allows for interaction and minpulation of data on the chart.

- Able to handle large amounts of dynamic data.
- Really clean and nice looking graphics.

Cons

- Limited amount of possible chart types.
- Does not have built in heat map.

2.3 Chart.js

Repository Commits: 2,465

Contributors: 236

Chart js is a very lightweight library that provides 6 chart types and fully responsive designs. ChartJS is well documented and easy to use, but lacks in variety.

Pros

- Uses HTML5 canvas element.
- Allows for easy creating based on chart type specification.
- Library provides Line Charts, Bar Charts, Radar Charts, Pie Charts, Polar Area Charts, and Doughnut Charts.
- Very responsive charts based on screen width.
- Simple API, easy to use.

Cons

- Limited amount of possible chart types.
- Does not have built in heat map.

2.4 Conclusion

In conclusion, despite the steep learning curve associated with D3.js we think it will be the best option for our web application. It has the widest range of available graphs to accomodate all the client's requirments and desired visualizations. There are also a number of wrapper libraries available for D3.js like DC.js and dimple.js to help create charts from D3. This is a great way to get around the clunkiness and downsides to D3.js and reap the benefits of all the other charting libraries. Another benefit to using D3 is the extensive amount of templates, examples, and documentation that exists to help guide the process and implmentation of our application.

3 Means of Incorporating Authentication

Our web application will have an authentication layer which will allow users to register for our application and design their own "stories." Our application will also authenticate user roles so that administrative users will have access to exclusive parts of the application. We want a simple way of authenticating users, while keeping personal information safe.

3.1 Building Our Own Authentication Layer

Node.js has a lot of helpful modules and packages that allow you to create your own password hashing functions and generate a custom authentication layer. There is a "crypto" module that is included in Node.js that "provides cryptographic functionality that includes a set of wrappers for OpenSSL's hash, HMAC, cipher, decipher, sign and verify functions"[1]. In addition to the crypto module, there are numerous Node.js extension modules that perform different hashing functions and provide the same functionality as the native crypto module. There are modules like *bcrypt* and *scrypt* that use different hashing algorithms to salt and hash password into a fixed length string to be stored into the database.

```
'use strict';
var crypto = require('crypto');
/**
 * generates random string of characters i.e salt
 */
var genRandomString = function(length){
    return crypto.randomBytes(Math.ceil(length/2))
            .toString('hex') /** convert to hexadecimal format */
            .slice(0,length);   /** return required number of characters */
};
/** hash password with sha512.
 * @function
 * @param {string} password - List of required fields.
 * @param {string} salt - Data to be validated.
 */
var sha512 = function(password, salt){
    var hash = crypto.createHmac('sha512', salt); /** Hashing algorithm sha512 */
    hash.update(password);
    var value = hash.digest('hex');
    return {
        salt:salt,
        passwordHash:value
    };
};
function saltHashPassword(userpassword) {
    var salt = genRandomString(16); /** Gives us salt of length 16 */
    var passwordData = sha512(userpassword, salt);
    console.log('UserPassword = '+userpassword);
    console.log('Passwordhash = '+passwordData.passwordHash);
    console.log('nSalt = '+passwordData.salt);
}
```

Listing 1. An example of how to salt hash passwords using the Node.js crypto module (Taken from *Rahil Shaikh's example*)[2]

Pros

- Creating our own authentication layer would provide us full control over how passwords are hashed and stored into the database.
- Provide understanding of every component taht goes into our application's authentication.
- Do not have to rely on another API to authenticate users.

Cons

- Laborious work and very time consuming.
- A lot of room for error and the possibility of data being compromised.

3.2 Outsourcing Authentication to Google

Google oAuth 2.0 is a Google API that authenticates users by signing in with their google accounts. There is a lot of documentation about how to integrate Google's authentication API with Node.js. *Passport* is authentication middleware for Node.js that provides simple, integratable authentication using "A comprehensive set of strategies support authentication using a username and password, Facebook, Twitter," and Google [3].

```
var passport = require('passport');
var GoogleStrategy = require('passport-google-oauth').OAuth2Strategy;

// Use the GoogleStrategy within Passport.
//   Strategies in Passport require a `verify` function, which accept
//   credentials (in this case, an accessToken, refreshToken, and Google
//   profile), and invoke a callback with a user object.
passport.use(new GoogleStrategy({
    clientID: GOOGLE_CLIENT_ID,
    clientSecret: GOOGLE_CLIENT_SECRET,
    callbackURL: "http://www.example.com/auth/google/callback"
  },
  function(accessToken, refreshToken, profile, done) {
       User.findOrCreate({ googleId: profile.id }, function (err, user) {
         return done(err, user);
       });
  }
));
```

Listing 2. (Taken from *Passport Documentation*)[4]

Pros

- Takes all the security risks out of creating our own authentication.
- Saves a lot of development time.
- Widely used with a lot of documentation.
- Relieves the need to store user passwords in the database.
- Can use the passport authentication middleware to simplify authentication even further.

Cons

- Limits users to having a google account.
- Relies on Google API to be working and running.
- Adds dependencies to the project.

3.3 Use CAS (Central Authentication Service)

CAS is an authentication process that redirects users to a CAS login page like the one that Oregon State University uses for most things like Canvas, Box, and MyDegrees.

CAS uses sessions and a CAS Client server to authenticate users throughout a web application:

> *When CAS redirects the authenticated user back to your application, it will append a {ticket} parameter to your url.*
>
> *The ticket returned to your application is opaque, meaning that it includes no useful information to anyone other than the CAS Server. The only thing that your application can do is send this ticket back to CAS for validation.*
>
> *CAS will then either respond that this ticket does not represent a valid user for this service, or will acknowledge that this ticket proves authentication. In the later case, CAS will also supply the user's NetID so that you know the identity of the user [5].*

Pros

- Can use the OSU official CAS to provide a nice theme.
- Keeps the same centralized CAS server as the rest of Oregon State University's application.

Cons

- Relies on Oregon State University's CAS server to be operational.
- Adds exteraneous code and implementation.
- Not as modular as the Node.js/Passport implementations of oAuth 2.0.

3.4 Conclusion

After reviewing the different means of including authentication for our web application, we think that using the Passport middleware to implement Google's oAuth 2.0 API would be the best option. It is easily integratable into Node.js applications, everyone at Oregon State University has a gmail account, and it removes the need to store any passwords into our database. The Google oAuth 2.0 authentication process uses a token that is received from the Google API that we will store along with the user ID and name into our database.

Another large benefit to using the Passport.js and Google oAuth 2.0 authentication process is that passport handles most of the overhead involved with user sessions and per-page aithentication. Using a small cookie set that contains the user id, "Passport will serialize and deserialize user instances to and from the session" to handle login sessions throughout the application [6].

```
1   passport.serializeUser(function(user, done) {
2       done(null, user.id);
3     });
4
5     passport.deserializeUser(function(id, done) {
6       User.findById(id, function(err, user) {
7         done(err, user);
8       });
9     });
```

Listing 3. (Taken from *Passport Documentation*)[6]

4 Front-end Framework

Our web application will be a series of dashboards to display energy data based on different buildings and subsets of buildings on Oregon State University's campus. A key part to designing a clean dashboard is having a well-spaced

grid-like layout with different charts and graphs to display a multitude of different datasets and trends. Rather than customizing classic html elements and using div containers to space our dashboard, we wanted to look into bootstrapped dashboard templates that allow easy customization and clean-looking results.

4.1 CSS Bootstrap

Repository Commits: 17,255

Contributors: 953

CSS Bootstrap is the most widely used CSS framework available. It has the most expansive component list and helps developers create clean and beautiful UX/UI in minimal time frames. Bootstrap allows developers to get their paplications up and running quickly without having to worry about front-end styling. A major issue with Bootstrap is that it tends to look very similar across applications and leaves little room for simple customization.

Pros

- Extensive component list, responsive design, and built-in Javascript functions.[7]
- EFully responsive design.
- Huge developer contrbution and maintainence.
- Used by major companies like Lyft.com, Vogue.com, Vevo.com, and Newsweek.com. [7]

Cons

- Unsuitable for small scale projects. [7]
- Not good if you want to have large control over UI.

4.2 Pure CSS

Repository Commits: 541

Contributors: 51

Pure CSS is known for its lightness and simplicity. Because it's small, it is very fast loading and makes for a lightweight web application. It is also unique in that it allows modules/components to be downloaded individually, reducing its size even more. Pure CSS is good for small projects that need to get up and running quickly and easily.

Pros

- Meant for small project to get up and running quickly.
- Responsive design by default.
- Pure CSS is modular so you can download only the components you need.
- The complete module is very small so it is quick loading.
- Able to be used complimentary with other frameworks.

Cons

- Not as extensive component list as Bootstrap.

4.3 Foundation

Repository Commits: 15,094

Contributors: 959

Framework is the second most popular css framework on the market behind bootstrap and tends to perform just as well. Unlike bootstrap's very prominent theme, Foundation allows for more customization with the look and feel of web pages [8]. Foundation also has a very good grid system to make the layout of components clean and responsive [9].

Pros

- Responsive design by default.
- Easier to customize than bootstrap.
- CSS classes are built in. [10]
- More unique look than the more-popular bootstrap.
- Good grid implementations with customizable grid layouts.

Cons

- Less maintained than bootstrap.
- Lack of support.
- Higher learning curve than bootstrap.

4.4 Conclusion

A lot of online resources acknowledged the fact that it is hard to make a website not look like bootstrap when using bootstrap css. Despite this, similar to our reasoning behind choosing D3.js, we would like to use a framework that is heavily supported and well-documented as it will be easier to answer questions during development. We think that the time spent restyling Bootstrap to make it look unique will not compare to the time saved by utilizing a well-documented framework. There is also a project called UI Bootstrap that works with AngularJS to create directives for each of the bootstrap components. [11]

REFERENCES

[1] Crypto — node.js v9.2.0 documentation. [Online]. Available: https://nodejs.org/api/crypto.html#crypto_crypto

[2] R. Shaikh. (2016, Jan.) Salt hash passwords using nodejs crypto. [Online]. Available: https://ciphertrick.com/2016/01/18/salt-hash-passwords-using-nodejs-crypto/

[3] Passport. Passport. [Online]. Available: http://www.passportjs.org/

[4] ——. Passport. [Online]. Available: http://www.passportjs.org/docs/username-password

[5] CAS. (2015, Jun.) How cas works. [Online]. Available: https://idms.rutgers.edu/cas/how_does_it_work.shtml

[6] Passport. Passport. [Online]. Available: http://www.passportjs.org/docs

[7] puranjay. (2015, Jul.) What css frameworks should you use? [Online]. Available: https://www.csshero.org/css-frameworks-use-comparing-5-popular-css-frameworks/

[8] N. Pettit. (2016, May) Should you use bootstrap or foundation? [Online]. Available: http://blog.teamtreehouse.com/use-bootstrap-or-foundation

[9] W. Blankenship. (2017, Oct.) Bootstrap vs. foundation: Which framework is right for you? [Online]. Available: https://www.upwork.com/hiring/development/bootstrap-vs-foundation-which-framework-is-right-for-you/

[10] L. Bradford. (2015, Jun.) Bootstrap 3 vs. foundation 5: Which front-end framework should you use? [Online]. Available: https://www.codementor.io/codementorteam/bootstrap-3-vs-foundation-5-which-front-end-framework-should-you-use-8s90mhsgn

[11] C. Sevilayha. (2015, Jan.) How to correctly use bootstrapjs and angularjs together. [Online]. Available: https://scotch.io/tutorials/how-to-correctly-use-bootstrapjs-and-angularjs-together