Team LibNav
Technology Review Document
CS Senior Capstone

CS461 (Fall 2016)

Authored by
Nathan Healea, Stephen Krueger, Matthew Zakrevsky

ABSTRACT

Currently, the Oregon State University Valley Library's flat, un-interactive map fails to provide visitors the information needed to navigate the library effectively. Visitors struggle to navigate to marked locations and find places with desired attributes. In this technology review, different technologies are researched and analyzed to determine there effectiveness towards developing the LibNav application.

CONTENTS

1 Introduction

Thoughout the project, the web applicaton has been broken down into 9 main pieces with each team member taking on three pars. Below is the roles that each team member will take on to ensure that the project is completed on time.

In the LibNav project Nathan Healea is tasked with user experience, user Interface design and development and user information. His role here is make sure that users can login into the Librarian Administration Dashboard and session information is saved so that they can interact with the application. He is also responsible for making sure that the Interactive Map is properly displayed across multiple platforms and device and to make sure the the user interface is accessible according to the Oregon State University Web Accessibility.

Stephen Krueger is tasked to oversee database management, user location, and navigation. His role is to make sure the users can find locations and destinations. User location and navigation are closely tied together. He will also be in charge of database management and operations, specifically the storage and retrieval of data and to also oversee the creation and development of the database schema.

In the LibNav project Matthew Zakrevsky is in charge of making sure that the web application has a web framework to handle routing, communication between databases and navigation of web pages through correct routing. He is also tasked in developing the map display and data rendering on to the map interface. His third and final task in this develop the templates and layout for the adminstrative dashboard.

2 Nathan Healea

2.1 User Interface

The User Interface for this project is present in both the front-end and back-end. In the back-end librarians will use the Librarian Administration Dashboard to manage the application by filling out forms, and assigning roles to users to manage the application. In the front-end visitors of the Valley Library will interact with the application through the Interactive Map. There a user will be able to find Known and Unknown locations and acquire more information if needed. This User Interfaces will need to scale to fit mobile device to 27 computer monitors and adhere to Oregon State University Web Accessibility.

2.1.1 Bootstrap

Bootstrap, created by a designer and developer at Twitter, is a front-end framework for web development. Bootstrap is built on two CSS preprocessors, Less and Sass, that can be modified, and also contains vanilla CSS that can be quickly added. This framework also supports responsive design allow for quick scalability for any device without much work. The Featured pack provides many HTML and CSS compounds combined with jQuery to make for functional and interactive user interface. [**?**]

2.1.2 Foundation

Foundation is another front-end framework just like Bootstrap. It semantic has a clean semantic with the sacrifice of utility and speed it provides. Foundation was created using mobile first design, designing for the smallest device first, then adding layers for larger device. This a great feature because a lot of users will be using phones and other devices to interact with the application. Foundation is fully customizable allowing developers and designs add or remove elements, as well of changing column size, color, fonts, and much more. Being a professional framework, Foundation is used by millions of designers and developers. [**?**]

2.1.3 Semantic-UI

Semantic-UI is development framework for creating clean, responsive layout using human-friendly HTML. Semantic-UI, while offering many of the same features as Bootstrap and Foundation, has a different focus. This framework uses concise HTML, a natural language noun/modifier relations for its syntax classes. It also uses Intuitive Javascript, trigger functionality and behaviors that are simple phrases. Simplified Debugging is also another feature focused on in Samantic-UI. This allows designers and developers to log performance to detect bottlenecks in user interfaces. [**?**]

2.1.4 Analysis

For our User Interface the LibNav Project is focusing on three criteria, scalability/responsiveness, displaying of information, and accessibility. In the table a below you will see key features that is relative to three criteria, and features that make each framework unique from one another.

Bootstrap	**Foundataion**	**Semantic-UI**
Responsiveness	Responsiveness	Responsiveness
HTML Components	HTML Components	HTML Components
User friendly	Highly Developed (version 6)	Built in Debugging
Offers Preprocessing CSS	Clean Markup	Comprehensive Javascript Functionality
	Mobile First Design	Built in User Interface Animations and Effects
	Customizable	

After consider the functionality offered by each framework, the project will primary use Foundation. Because Foundation is highly used, maintenance update and new features are continually being integrated into the framework. Also after spending sometime going through the showcase for each framework, check responsive and functionality, Foundation had the smoothest transition from large screen device to smaller screen device. Sites that used Foundation also did not show web element positioning issues, that were present in Bootstrap and Semantic-UI

2.2 Information Sanitation

LibNav Librarian Administration Dashboard will consist of forms to allow an authorized user to create, delete, edit Known and Unknown location. Same with the Interactive Map, it to will have a form where an user will enter in their local inorder to get direction to navigate to their desired location. To help prevent a user from entering undesirable information into our system, data entry sanitation will accrue. This will insure that when information is rendered back out to the interactive map, incorrect information will not be displayed.

2.2.1 Validate.js

Validate.js is a JavaScript library containing a validating JavaScript Object. Validate uses Json to share validation constraints between client and server. This allows the developer to make changes to the default settings without having to recode functions. For example Null, Undefined and whitespace are considered to be empty values. If the team wanted to add /n to that list the modification to the validate.isEmpty is simply editing the values in the constraints. This is all possible because nothing is private or inaccessible in the library.[**?**]

2.2.2 Validator.js

Just like Validate.js, Validator.js is a JavaScript library that provides object and string validation. With Validator.js a developer can define an object that will contain information and then define constraints for attributes with in that JavaScript Object. This ensures that each attributes within the object, representing some sort of data type or database table, has the correct information contained in it. [**?**]

2.2.3 Validator.js by Github Chriso

Validator.Js, created by Github user Chriso, is a JavaScript library that provides validation and sanitation functionality for frontend web sites. This project is being considered because of its on going development and community. With 104 releases and 120 contributors new features and functionality updates are continuously happening. Validator.js work similar like Validate.js where function are call for each individual piece of information that need to be sanitise or validated. [**?**]

2.2.4 Analysis

To make sure information is sanitize before entering our system two criteria are being looked at: customization, and functionality. Even though many of the data types that will be used are simple object containing string and numbers, the exact formatting of that information needs to be checked. The table below will show key features and functionality desired in each library that will help accomplish the sanitation process.

Validate.js	**Validator.js**	**Validator.js by Chriso**
Provides validating functionality	Provides validating functionality	Provides validating functionality
Utilities for string manipulation	Utilities for string manipulation	Utilities for string manipulation
Editable constraints	Constraints declared as like object	Large community
Shared constraints between Client and Server	Group validation for object attributes in single declarations	Continuously updated

After considering the different options and the functionality that is provided by each javascript library provide the project will primarily use Validator.js for its validation and sanitization. The reason Validator.js is going to be use do to the way validation is declared and use. Creating a JavaScript object with desired attributes and creating a constraint object that is mapped to the original JavaScript object make for a clean flow of validating.

2.3 User Authorization and Session

Since LibNav is going to have user entering data into the system through the Librarian Administration Dashboard, User Authorization and Session are going to need to be handled. Many conversations have already happened on the base framework that is going be used for this project, Node.js. So the following section, user authorization and session technologies consider will be focused on using that framework. Also if the follow section will look at one technology for User Authorization and two technologies for user sessions. Reason for this will be described below.

2.3.1 CAS

LibNav Librarian Administration Dashboard is going to consist of a user login. Instead of developing a new system for a user to login, the project team has already decided to use CAS Authentication do to the fact that many application at Oregon State University using CAS including other Valley Library application. CAS is a Central Access Service that provides a single sign-in for multiple user. In the Oregon State Ecosystem user login into CAS using an ONID and password. Then in turns sends session information back to the application to be stored. This session information can be sorted in two location, the front-end as cookies and the back-end through files and databases.

2.3.2 Node-client-sessions

node-client-sessions is a Node.js packages for client side session. This client side module used encrypted cookies to secure store the session data. This module uses a cryptography, encryption and signature key, passed into a function to encrypt session data and provides many options to configure the algorithms and keys used. Node-client-session also offers the ability to use multiple sessions concurrently with each other and to generate keys if needed. [**?**]

2.3.3 Express-session

Express-session is a node.js module that used a combination of both front-end and back-end functionality to store a session. In the front end the session id is stored on the client side using a cookie. That session id is related to session that is stored in a database. When the session need to be checked a call to database is made querying in the client side session id. [**?**]

2.3.4 Analysis

After taking in consideration the technologies used by Oregon State University and the Valley Library, CAS will be the primary module for allowing users to login, because verification and authentication is already handled and it is one less password to remember. As for storing the session, the project will use express-sessions. Express is one of the frameworks being condenser for the server side framework, and the ideas of splitting session information between two places take some of the load off the client and allows for Interactive Map function to happen on the client side.

3 Stephen Krueger

3.1 Navigation/Routing

The user will need to not only find information about the different rooms and spaces on the interactive map, but also be guided from their current location to their specified destination. To accomplish this, the team will need to implement some form of navigation or routing system. The navigation system should, at the very least, give a guiding line from the users location to their destination. The user location will be given by a form input and the destination can be selected on the map or passed to the application in the URL. The navigation system should present the user with the shortest path from point A to point B. Also the system shouldnt lead students through areas that they dont have access to. The system is going to be built in AngularJS and HTML. The system will need to be find the shortest path between two points. There will be a grid that overlays the map. On this grid, the admin can mark the grid to designate navigable areas and mark off spaces of the grid that shouldnt be traversed. Then, the team will use a shortest path algorithm to find the shortest distance between navigable nodes from start to destination and display the path on the map.

3.1.1 PathfindingJS

There are a few great path finding libraries in javascript to be considered. The first is PathFinding.js. Pathfinding.js is a library that was developed for gird based games but can be tweaked to work for our map. There are a few really nice things about this package. To start, its easily installable as a node.js package. Pathfinding.jss ease of use is high. To create a grid you simply call a function and specify the size. When creating the grid, the function can be sent a matrix that specifies which nodes are and are not walkable. This means that after the map is marked off by the admin, the only thing that needs to be stored is the matrix array. Then, if any edits need to be made for navigable areas, the system just needs to edit the matrix. Pathfinder.js has some helpful parameters that can be specified, such as allowing diagonal traversal and whether or not you can cross corners. The package also comes with multiple path finding algorithms implemented, including A*, breadth first, best first, and Dijkstra as well as bi-directional versions of all those algorithms. Testing will have to be done on which algorithm is the fastest for our system, but the team will most likely be using the A*. [**?**]

3.1.2 l1 Pathfinder

The second technology that our application could use l1 path finder. This technology is really cool, but would be harder to implement as the documentation isnt as helpful and its a more complicated package. Instead of searching through the grid one cell at a time, l1 pathfinder pre-process the grid and creates a visibility graph. It then performs an A* search on the visibility graph which results in very fast performance compared to other js path finding implementations. A benchmark was done to find a path on a map for a game. l1 took 22 ms while pathfinding.js A* search took 3925ms. l1 path finder relies on ndarrays, which is a modular multidimensional array. This is a separate package that would need to be installed, but it is very small so wouldnt be an issue. It is simply a different way to format a multidimensional array.[**?**]

3.1.3 EasyStarJS

The third technology the application can use is easystar.js. This package is the easiest to use and has a good, simple API. Its an asynchronous A* path finding API that can easily be installed as a node package. Its requires no other packages and can be used with any framework. The biggest problem with this package is its performance. In the same benchmark test that was used to test the I1 and pathfinding.js algorithms, Easystar was by far the worst. Easystar came in around 42177 ms compared to I1 which took 22ms. This test was done using a map from a video game which will be much more complex than our grid, but the difference in the benchmark is definitely worrisome. [**?**]

3.1.4 Analysis

Pathfinder.js is the technology the team will go with. Its A* algorithm is fast, its easy to install, and the API is simple to use. Pathfinder is also not framework dependent and requires no other technology. Its not as complicated as the I1 search and because our grid will be fairly simple compared to a video game map, pathfinder.js should be more than fast enough. Easystar is just too slow in comparison.

3.2 User Location

The application will need to know the users starting location in order to provide navigation. Determining the users staring location can be done either manually or automatically. The user can manually provide their starting location or select their location on the map. The other option is to have the users device give us a GPS location. There are many positives and negatives to both methods.

3.2.1 HTML and Javascript

Determining the users location manually is much easier to implement. The user would be given an HTML form to fill out their starting location. They could also select their starting location on the map that would auto fill the form. The location they select would have a corresponding coordinate pair on the grid that would then be sent to another function to handle navigation. The grid system is talked about in the navigation section. Bootsrap will be used to to style it. This would be very simple to set up. QR codes could also be set up in the main areas of the building that would pass the user location via URL and auto fill their start location. There are some obvious downsides to having the user manually give the application their location. For instance, the user might not know where they are, and if they deviate from the path there is no automatic redirection that you would find on Google maps.

3.2.2 GPSjs

To try and account for the disadvantages of the user manually giving us their location, the use of GPS was considered. There are some awesome GPS libraries for javascript like GPS.js. GPS.js is rather simple to install and implement. It has a great library that will give you information like how many satellites youre communicating with, your heading, distance between locations. The problem is that its just not accurate enough. The GPSm on a phone doesnt have the precision to guide a user inside a building, especially one that doesnt have a glass ceiling. [**?**]

3.2.3 Accuware IPS

Indoor positioning and navigation systems overcome the accuracy problem with gps. There is a company called Accuware, which used routers and iBeacons to give real time positioning. They provide an SDK and an API and also have a cloud based platform. Since the team will be building a web app will be using their API. They claim that their accuracy is within 1 meter. Unfortunately, they need to have a map in either PNG, Gif or JPG format, and the application will be using SVG images. It is also not a free service, and pricing is determined based on the size of the building. [**?**]

3.2.4 Analysis

The application is not going to include any indoor or real time positioning systems. GPS isnt accurate enough and the indoor positioning systems would not only add extra cost but also be time consuming to implement. The best course of action is to have a user manually enter their location. By installing QR codes throughout the building, even if the user is lost theyll be able to find out where they are. This method is simple, accurate, and the most cost effective.

3.3 Database

The database will be a crucial part of the application and its critical that it is set up properly. Due to what is currently on the servers the application will be running on, a MySQL database will most likely be used for storing the applications data. Since are application will be using AngularJS, data will be passed back and forth as JSON objects. In the database the team will be storing all typical data types and Blobs.

3.3.1 MongoDB

Instead of using a relational database such as MySQL, something like MongoDB could suit the needs of the application. Mongo stores data as collections of JSON documents which is convenient as the application will already be using JSON objects. With Mongo, the database manager can store data without having to first define the structure. This is called a dynamic schema and it allows for the storage of complex data structures. It also makes it easier to change the schema as the program is developed. Mongo also has a much more intuitive query language for programmers coming from object orient languages compared to SQL. Performance compared to MySQL for the applications needs is a non-factor due to our applications relatively small size.[**?**]

3.3.2 MySQL

MySQL has a number of advantages over a non-relational database like MongoDB. The biggest advantage is that everyone on the project has at least some cursory experience with MySQL. That means that the team is not going to be in the dark when discussing database operations and the team does not have to learn a new technology. Interacting with the database should be easier for everyone as well, as all of the team members have at some point built an application that interacts with a SQL database. Another benefit is that much of the data that the application will pull from the library is sitting in MySQL databases. Having the application use MySQL keeps it standardized.[**?**]

MySQL is open source, which makes it an easy sell. It is also both fast and scalable. MySQL uses a few methods to increase its performance. Replication, for instance, which allows for the rapid creation of multiple instances of the database. It also has rapid connection handling so that connecting and disconnecting does not have a performance penalty. MySQL also has great security features such as being able to block specific machines and SSH and SSL support. [**?**]

3.3.3 PostgreSQL

PostgreSQL is another open source SQL solution. The biggest advantage of Postgre for our project is its extendibility. Due to the good third party support, Posgres extendibility allows for custom data types such as JSON, XML, multidimensional arrays, etc. Posgre also has a full text search, and a compression layer called Toast that will automatically zip large data. The disadvantage of using Postgre is that there will be a learning curve, as none of the team has used it before. It is also not as widely used, so support is not as easy to find if you run into a problem. Since the database for the application will be relatively simple, many of the features of Postgre will simply go unused. It might just be overkill. [**?**]

3.3.4 Analysis

Due to the ubiquity of MySQL, the application will use MySQL for its database needs. Keeping it standard with the library servers means that the app should have no problem running when it is passed over to them. They will also be able to maintain the database without having to learn a new tech. Its easier for our team from a development standpoint as well as everyone has experience with MySQL.

4 Matthew Zakrevsky

4.1 Frameworks

In the LibNAv Application a web framework will be used to allow the front end web page built in HTML, CSS, and other web languages to communicate between the servers that will be hosting a backend SQL database. This framework will be used to handle webpage routing and the management of a file hierarchy to make navigation of the web page easier and more robust. For the LibNav project the frameworks that can be used must be available to the Library IT staff and must work with the existing servers that host will be hosting the final web application.

Choosing a framework to use in our application will be based upon the existing technologies the library is currently using. Currently the library can support web applications built upon PHP, Ruby, and Node.js. The majority of the websites are built upon PHP and HTML,CSS with a few web pages making use of Ruby On Rails. The server also has a deprecated version of Node.js that the Library IT team can update to a new version once the team is ready to begin development. Below are three options of frameworks that can be used with Ruby, PHP, and Node.js applications

4.1.1 Ruby On Rails

Ruby On Rails(RoR) is a web application framework that is written in Ruby. RoR provides all of the basic functionality that is desired for the LibNav project. RoR provides routing between different web pages while also providing built in functionality to manage web applications. These built in functionalities include form builders which help to create HTML forms in embedded in Ruby, models to handle the web pages, and migrations for moving data and creating database tables. RoR also provides a series of validators to validate that forms have the correct data, allowing for cleaner data going to and from the database. RoR is already installed on the Library Web Servers so there would be no additional work to get permission to add RoR to the server and is listed under the MIT license.[**?**]

4.1.2 Symfony

Like Ruby Symfony is a web application framework written for PHP. This framework provides routing options for web pages, debugging toolbar, and templates to build web pages. Symfony has built in validators to manage form validation, and request and response protocols to handle PHP request and response calls to the PHP pages, something that will make building the application easier. Symfony also presents other options such as bundles which provide additional libraries and add additional functionality. Symfony is not currently installed on the library servers but it is licensed under the MIT license, saving a headache due to licensing.[**?**]

4.1.3 Express

Express is a Node.js framework that handles much of the same functionality as Ruby On Rails and Symfony. Express has prebuilt options to help with the routing of the web application through JavaScript web pages. Error handling is built in making it easier to find errors in the application . Express also provides a middle ware layer to abstract away calls to databases and external systems, increasing the security. By using Express the LibNav application also gains access to the Node Package Manager, which opens up many options available to the LibNav application.[**?**]

4.1.4 Analysis

The LibNav web application will be using the Express framework because of the number of mapping libraries are written in JavaScript. This will allow the web application more options for implementing the most important functionality. Node.Js has already been installed on the Library servers which means less work in order to get a technology approved for use on the library servers. Express does have less built-in functionality but by using Node the application will be able to make up for the loss of functionality in the form of Node packages.

4.2 Map Software

In the LibNav application this technology is responsible for rendering the map to the user and allows the user to access the data stored. This should be dynamic, allowing the user to zoom, pan, and manipulate the map view. This should show all marked spaces and should respond when the space is clicked. These technologies come in many options, with the predominant libraries being used for geographical maps, much like google maps. Since the maps of the Library are not designed for use in Geospatial Information Systems(GIS) the unique challenge of displaying maps with non-mercator projections.

4.2.1 ESRI Leaflet

Esri Leaftlet is a JavaScript library that allows for the projection of geographic maps but also allows for creation of non-geographic maps. There are image overlay options which will allow for data to be created and stored to the web-page. This will allow for layer overlays which can render data over specific areas of the map. With this in mind the Leaftlet does not need a second library to handle the creation and rendering of objects on the map. This will cut down on overhead of building the web application. There is an Node package which will make adding the library to the LibNav simple and unobtrusive. Leaflet is licensed under the apache 2.0 license which allows for full modification and use.[**?**]

4.2.2 D3.js

D3 is an image manipulation javascript library to perform data visualization with HTML, SVG and CSS. D3 makes direct use of Document Object Models to add information to a document, which can be modified and manipulated in javascript, allowing for zooming, panning, clickable events. D3 does not need any additional libraries to perform modifications on the map, with a robust api to change map views and data projections. D3 is also available to use as a node package allowing for easy integration. D3.js is licensed under BSD-clause 3 which allows for full use of the source code so long as it retains the original copyright.[**?**]

4.2.3 OpenLayers

Much like Leaflet the OpenLayers Library is built for the modification of GIS maps but also can be used for static images to produce maps. OpenLayers provides Direct Object manipulation not unlike D3.js but also has the built in panning, zoom and clickable events that both D3 and Leaflet provide. OpenLayers supports static image rendering for maps but presents layer management not unlike Leaflet. OpenLayers is licensed under BSD-clause 2 allowing for use of the source code so long as the source code retains its original copyright.[**?**]

4.2.4 Analysis

The best option for the LibNav application would have to be the D3 JavaScript library. This is because D3 provides better modification of SVG files and data vectors. Data manipulation will be a large part of this and D3 is designed to manage data. There are more examples of using static images for map generation for OverLayer and D3 than there are for Leaflet. D3 has many examples of data manipulation and rendering that will be very important for developing the LibNav application. D3 is also bundled as a Node package, allowing for an easy integration to the application.

4.3 Administative Dashboard

The best option for the LibNav application would have to be the D3 JavaScript library. This is because D3 provides better modification of SVG files and data vectors. Data manipulation will be a large part of this and D3 is designed to manage data. There are more examples of using static images for map generation for OverLayer and D3 than there are for Leaflet. D3 has many examples of data manipulation and rendering that will be very important for developing the LibNav application. D3 is also bundled as a Node package, allowing for an easy integration to the application.

4.3.1 Gentelella

Gentelella is a bootstrap template licensed under the MIT license and packaged with Node.ss, making this library easier to integrate into the LibNav application should Node.js become a mainstay technology that can be used. Gentelella is a Bootstrap administration template theat can create panels for the user to make use of. This tool provides many options to our library team, with calendar view, text forms and notifications to name a few of the panel options available. Gentelella has also been built to other frameworks beyond node, two options being Ruby on Rails and Symfony. This will allow us to make use of this dashboard with whatever framework should be used.[**?**]

4.3.2 Twig

Twig is a PHP specific template engine that could be used of to handle building templates. It makes use of text based formatting like HTML but does allow for manageable formatting. This is a good choice for if the LibNav application is tied up in having to use PHP as the web language but if a cannot preexisting template cannot be used. This one more language that needs to be learned which can waste a lot of time.[**?**]

4.3.3 JavaScript

This is the last resort and is easily the most time consuming options. This would be for if ther ever was situation where the templates are failing or if licenses cannot be acquired. A lot of would have to be put into building the web page and dashboard from scratch and could a huge time sink, when there are other options available.

4.3.4 Analysis

Gentelella is the best option available to the LibNav application. It is a Bootstrap template which means that is usable across different technologies . This good as it will save time and energy to build a dashboard that will meet the requirements of the project. The ability to install with the node package manager will also save some frustration porting the application to the library server.The time that is saved is the primary decision maker for this technology, for the project will already have a lot of the time going to the map functionality.

5 BIBLIOGRAPHY